고속 시뮬레이션을 위한 모델합성 방법

Model Composition Methodology for High Speed Simulation

이완복
중부대학교 게임학과

Wan-Bok Lee(wblee@joongbu.ac.kr)

요약

DEVS 형식론은 이산사건 시스템의 구조를 계층적으로 나타낼 수 있기 때문에 복잡한 시스템을 모델링하기에 적합하며, 가독성이 좋기 때문에 유지 보수에 유리한 장점이 있다. 반면에, 계층적인 모델의 구조는 시뮬레이션 실행 시에 빈번한 메시지 전달을 야기 시켜 시뮬레이션 속도가 저하되는 단점이 있다. 본 논문에서는 계층적인 DEVS 모델들을 하나로 합성하여 빈번한 메시지 전달을 방지하고 시뮬레이션 실행 속도를 개선할 수 있는 모델 합성법을 제안한다. 제안한 방법은 시뮬레이터 실행코드를 생성하기 이전에 구성 모델들 간의 메시지 전달 관계를 해석하여 실행시간에는 모델 해석과정을 생략하도록 하는 기법으로 기존의 모델 수평화 기법보다 훨씬 뛰어난 속도 향상 결과를 보인다. 제안한 방법의 효용성을 보이기 위해 실험을 통하여 시뮬레이션 속도가 18배 정도 개선될 수 있음을 보인다.

■ 중심어 : □DEVS | 모델합성 | 시뮬레이션 속도개선 | 정형모델 |

Abstract

DEVS formalism is advantageous in modeling large-scale complex systems and it reveals good readability, because it can specify discrete event systems in a hierarchical manner. In contrast, it has drawback in that the simulation speed of DEVS models is comparably slow since it requires frequent message passing between the component models in run-time. This paper proposes a method, called model composition, for simulation speedup of DEVS models. The method is viewed as a compiled simulation technique which eliminates run-time interpretation of communication paths between component models. Experimental results show that the simulation speed of transformed DEVS models is about 18 times faster than original ones.

■ keyword : □DEVS | Model Composition | Simulation Speedup | Formalism |

Ⅰ. 서 론

컴퓨터 시뮬레이션은 사람들이 만든 시스템이나 규칙 등에 대하여 특성 검증, 성능 평가 등의 목적으로 많이 활용되고 있다. 시뮬레이션 모델로서는 시스템을 보는 관점(World View)에 따라 여러 가지가 존재한다. 이러한 모델 중에는 시스템의 상태 변화를 야기하는 이벤트를 중심으로 시스템을 모델링하는 이벤트 스케줄링

접수번호 : #061011-002
접수일자 : 2006년 10월 11일
심사완료일 : 2006년 11월 16일
교신저자 : 이완복, e-mail : wblee@joongbu.ac.kr

모델과, 상호 작용을 하는 프로세스의 집합으로 보는 프로세스 인터렉션, 그리고 액티비티 스캐닝 모델로 구분하는 것이 일반적이다[1]. 이벤트 스케줄링 관점에 기반한 모델들은 모델의 구조나 가독성이 좋지 않지만, 빠른 속도로 실행 가능한 장점이 있는 반면, 프로세스 인터렉션 관점에 기반한 모델은 가독성이나 모델 구조는 좋은 편이나, 이벤트 스케줄링 기반 모델보다 실행 속도가 느린 단점이 있다[1]. 모델의 구조가 잘 정립되어 있으면, 모델의 재사용이나 디버깅 등에 유리한 장점이 있지만, 실행 속도가 느린 단점이 있다. 이러한 측면은 객체지향 프로그래밍 언어로 작성한 프로그램이 유지 보수에 유리하지만, 실행 속도는 느린 점과 유사한 측면이 있다.

Zeigler 에 의해 제안된 DEVS(Discrete Event Systems Specification) 형식론은 이산사건 시스템 모델을 모듈러하고도 계층적으로 모델링할 수 있는 형식론이다. 그렇기 때문에 이 형식론으로 시스템을 모델링하면 모델의 재사용이 용이하며, 대규모의 복잡한 시스템 모델링에 유리한 장점이 있다[2]. 반면에 계층적인 모델 구조는 계층을 이루는 모델 사이에서 빈번한 상태 정보의 교환을 야기하기 때문에 시뮬레이션 실행 속도가 느려지게 된다. 그렇기 때문에 모델의 구조나 가독성은 좋은 반면 실행 속도가 이벤트 스케줄링 모델보다 느리다는 단점이 있다.

시뮬레이션 속도를 개선하기 위해서 여러 가지 방법들이 제안된 바 있다. 병렬 및 분산 시뮬레이션 기법은 계산 과정에 연관 관계가 없어서 동시에 계산할 수 있는 요소들을 다수의 CPU에서 계산하는 기법이다[3-5]. 이 기법은 상호 연관 관계가 적은 모델의 경우 많은 속도 향상이 가능하지만, 순차적으로 계산되어야만 하는 모델의 경우에는 속도 향상이 전혀 이루어지지 않을 수도 있다. 복합 시뮬레이션(Hybrid Simulation) 기법은 정적인 상태에 도달한 시스템의 요소들을 각종 통계학적인 분산 함수들로 대체함으로써 속도 향상을 꾀하지만 범용적으로 적용되기에는 한계가 있다[6]. 모델 구조를 변형하여 속도향상을 꾀하는 방법도 연구된 바 있다[7]. 모델 수평화(Flattening) 이라 불리는 이 방법에서는 여러 계층으로 구성된 모델의 구조를 더 적은 수의 계층으로 변형함으로써 구성 모델들간의 통신 경로를 줄이고자 하는 방법이다. 이 방법에서는 25% 정도의 속도 향상이 가능함이 보고된 바 있다. 또한, 캐드 분야의 로직 회로 시뮬레이션 기법에서는 모델의 해석 과정을 run time 이전에 계산하고, 시뮬레이터 코드를 생성하여 시뮬레이션 속도 향상을 꾀하는 Compilied Code 시뮬레이션 기법이 제안된 바 있다[8]. 그러나, 캐드 분야의 Compiled Code Simualtion 기법은 정형적인 모델을 기준으로 적용하는 것이 아니라 C언어와 같은 프로그래밍 언어에 대해 적용한 것이기 때문에 범용적인 시뮬레이션 모델에 적용되기 어려운 단점이 있다.

본 논문에서는 DEVS 형식론으로 표현되는 이산사건 시스템(Discrete Event System) 모델들을 빠르게 시뮬레이션 할 수 있는 새로운 방법을 제시한다. 이산사건 시스템 시뮬레이션 과정에는 구성요소 모델들 사이에 많은 이벤트들이 상호 인과 관계에 의하여 통신하게 된다. 본 논문에서는 연관성이 많은 구성요소 모델들을 하나의 모델로 합성함으로써 통신하는 메시지 수를 줄이며 전체 시뮬레이션 속도를 개선하게 되었다. 이를 위해 DEVS 구성요소 모델들 간의 통신의미(Communication Semantics)를 명확히 정하고, 모델합성 기법을 정의하였다.

본 논문은 다음과 같이 구성되었다. 2장에서는 본 논문에서 논의하는 이산사건 시스템 모델과 시뮬레이션 알고리즘에 대해 소개하고, 3장에서는 시뮬레이션 속도 개선을 위해 제안하는 모델합성 기법에 대하여 정의한다. 4장에서는 모델합성 기법을 적용하였을 시 시뮬레이션 속도가 개선될 수 있음을 실험을 통하여 확인하고, 5장에서 결론을 맺는다.

II. DEVS 형식론과 시뮬레이션

1. DEVS 형식론 개요

DEVS 형식론은 이산사건 시스템을 모듈러하고도 계층적으로 모델링 할 수 있는 방법을 제시한다.[2] 이 형식론에는 모델을 계층적으로 분해할 수 있도록 원자 모델(Atomic Model)과 결합 모델(Coupled Model)의 두

가지 클래스로 나누어서 모델을 표현하고 있다.

원자 모델은 더 이상 나눌 수 없는 모델로서 각 구성 요소의 동적 특성을 시간명세 상태전이시스템(timed state transition system) 레벨에서 기술한다. 원자모델에는 세 개의 집합과, 4개의 특성함수가 명세된다. 원자 모델의 수학적 정의는 다음과 같다.

입력사건집합
상태집합
출력사건집합
→ 내부상태전이함수
× → 외부상태전이함수
→ 출력함수
→ 시간전진함수
∈ □ □

결합 모델은 구성요소 모델들 간의 연결 관계를 표현하는 모델로서 구성요소 모델들을 조립하여 더 큰 단위의 모델을 만들 수 있게 한다.

입력사건집합
출력사건집합
콤포넌트이름집합
∈
콤포넌트모델집합
인플루언시집합
∈
→ 연결관계
→

DEVS 형식론의 모델링 예를 보이기 위해 컴퓨터 시스템이나 네트워크 시스템 해석에 자주 사용되는 단일 서버 큐 시스템을 모델링하고자 한다. 전체 시스템의 구조는 [그림 1]과 같다. EF는 결합모델 PEL에 입력을 생성하고 처리된 결과를 수집하여 통계치를 계산하는 모델이다. Buff는 입력을 보관하고 있다가 ready 입력이 들어오면 Proc에 전달하는 역할을 한다. Proc은 전달받은 것을 처리한 후 done 포트를 통해 완료되었음을 알린다. 처리가 완료된 것은 다시 EF에 전달되어진다. 이 모델에 대한 자세한 설명은 [2]를 참조할 수 있다.

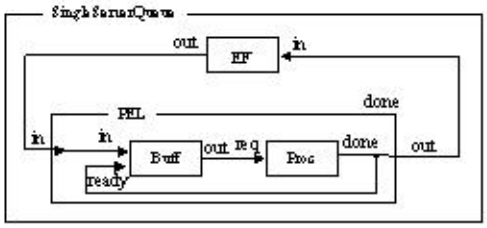


그림 1. 예제 모델, 단일서버 큐 시스템

이 모델을 DEVS 형식론으로 명세하면 다음과 같다.

∈ ∈

∈

–

결합모델 PEL은 두 원자 모델 Buff와 Proc이 결합되어 다음과 같이 명세 되어 진다.

마찬가지로, EF와 전체 시스템을 의미하는 SingleServerQueue 모델도 구성될 수 있는데, 지면 관계상 생략한다.

2. 추상화된 시뮬레이터

DEVS 모델을 실행하는 추상화된 시뮬레이터 알고리즘은 Zeigler에 의해 제안되었다[2]. 이 알고리즘에는 원자 모델을 실행하는 Simulator, 결합 모델을 실행하는 Coordinator라고 하는 프로세스가 있어서 정해진 알고리즘에 의해 해당 모델의 특성 함수들을 실행하여 시뮬레이션을 수행한다.

이 추상화 시뮬레이터 알고리즘을 C++로 구현한 것이 Devsim++ 시뮬레이션 환경이다[9]. Devsim++ 환경에서는 (*), (y), (x), (done) 이라는 네 가지의 메시지 전달을 통하여 시뮬레이션이 진행된다. (*)는 어떤 개체가 내부 상태 전이를 할 시간이 되었을 경우 시뮬레이션 엔진 내부의 스케줄러가 발생시키는 메시지이며, (y),(x)는 어떤 개체의 출력 이벤트가 다른 개체들의 입력 이벤트와 연결 관계가 있을 경우에 발생하는 메시지이다. (done) 메시지는 다음 스케줄 시간을 정하기 위해서 하위 계층의 시뮬레이터가 상위 계층의 시뮬레이터에게 보내주는 메시지이다.

시뮬레이션 과정에서는 위 네가지 메세지가 상호 인과관계에 의해서 연속적으로 발생하게 된다. 그러므로 구성관계가 복잡하고 상호 작용(interaction)이 많은 모델들 간에는 많은 메시지들이 시뮬레이션 도중에 발생하며, 그 만큼 많은 시간이 소요되게 된다.

다음 장에서 소개하는 모델합성 기법은 모델들 간의 상호작용을 상당히 줄이고 이러한 통신 메시지들을 줄일 수 있는 방법이다.

III. DEVS 모델 합성법

1. 통신의미 (Communication Semantics)

모델합성을 정의하려면, 먼저 전제되는 통신의미(Communication Semantics)를 분명히 해야 한다[10]. 전통적인 시스템 검증 방법론 분야에서 사용되어 지는 Automata, Process Algebra 등의 모델들은 공유이벤트(shared event)로써 통신의미를 규정하고 있다. 공유이벤트는 모든 구성요소에서 동시에 발생할 수 있는 경우에 한해서만 발생할 수 있다. 그러므로 어떤 하나의 구성요소 모델이 현재 상태에서 공유이벤트를 수용(Accept)할 수 없다면 그 공유이벤트는 발생할 수 없으며, 경우에 따라서는 교착상태(dead lock)가 발생하기도 한다.

공유이벤트를 이용한 통신의미는 일반적으로 DEVS 모델과 같은 모듈러한 IO 시스템의 통신의미에 부합되지 않는다[11]. 예를 들어, 두 개의 구성요소 모델에서 같은 라벨(label)의 출력 이벤트를 가지고 있을 경우, 이들 이벤트들은 동기화되어 동시에 발생함을 의미하는데, 이것은 물리 세계에서 수용되기 어려운 현상이다.

이러한 문제점 때문에 Lynch는 IO 시스템 모델의 일종인, IO Automata를 제안할 때 기존 공유이벤트를 이용하여 시스템을 모델링하되, Compatability 라는 별도의 조건을 두어, 출력 이벤트는 항상 자율적으로 발생할 수 있도록 허용하고 있다[12].

여기서는 DEVS 모델의 통신 의미에 대해서 논의하려 한다. DEVS 모델은 IO 시스템 모델이기 때문에, IO 시스템의 의미와 부합하는 통신의미가 필요하다. 이러한 필요성 때문에 약한 동기화 라는 통신 의미를 제안한다. 약한 동기화는 Heymann이 제안한 Prioritized Synchronization의 의미와 일맥 상통하는 바가 많다[10].

정의1) 약한 동기화 (Weak Synchronization)

시스템을 구성하는 이벤트가 세 종류의 형태, 입력, 출력, 내부 이벤트로 나뉘어 지며 다음의 특성들을 만족할 때 시스템의 통신 의미는 약한 동기화라 한다.

1) 출력 이벤트와 내부 이벤트는 블록되지 않고 자체적으로 언제든지 발생시킬 수 있다. (spontaneous output)
2) 입력 이벤트는 항상 같은 라벨을 가지는 출력 이벤트와 동기화 되어서 발생되며 외부 환경에 의해서만 트리거 될 수 있다. (passive input).

본 논문에서 다루는 모델 합성법은 약한 동기화 가정하에서 정의 되었다.

2. 시뮬레이션 계산 시간 고찰

DEVS 모델의 시뮬레이션 과정 중에 발생하는 주요 연산들을 살펴보기 위해, [9]에서 개발한 시뮬레이션 환경을 GNU의 gprof[13] 도구를 사용하여 [그림 2]에서 보이는 바와 같이 원자모델과 결합모델 별로 분석하였다.

원자 모델의 경우에는 크기 상태 전이 및 출력을 생성하기 위한 특성함수들을 호출한 후에 다음 실행할 이벤트와 그 발생 시간을 결정하기 위해서 시간전진함수인 ta()를 실행하고 (done) 메시지를 상위의 시뮬레이터에게 전달한다. 반면에, 결합모델의 경우에는 모델들간의 입출력 연결관계를 따져서, 영향을 받는 구성 모델들로 이벤트를 라우팅해주고, 다음 실행할 이벤트를 스케줄링하는 과정으로 이루어져 있다. 결론적으로 시뮬레이션 수행 과정에는 1) 상태전이, 2) 이벤트 라우팅, 3) 스케줄링이라는 세 가지 세부 작업들이 연속적으로 수행되는 것을 알 수 있다.

이 세부 작업에 소요되는 시간이 어느 정도 인지 측정하기 위해 GNU gprof 도구를 이용하여 측정하면 [표 1]과 같다. 사용한 예제 모델은 앞에서 소개한 단일서버 큐 시스템이다.

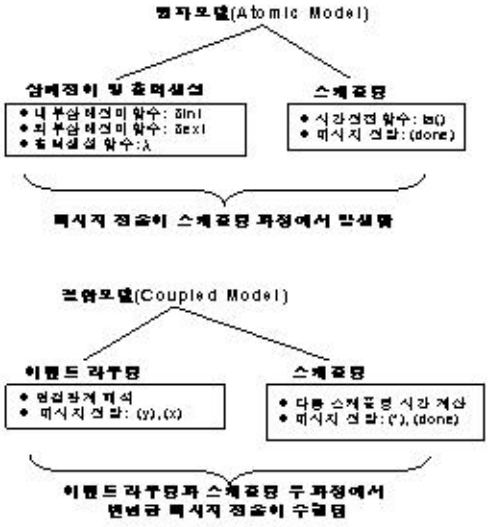


그림 2. 모델별 시뮬레이션 계산 시간 분석

[표 1]을 살표 보면 전체 시뮬레이션 시간 중 90% 이상이 상태 전이 이외의 작업에 소요된다는 것을 알 수 있다. DEVS 시뮬레이션 알고리즘을 살펴보면, 이벤트 라우팅과 스케줄링 과정에서 구성 요소 모델 사이에서 빈번한 메시지 전달이 발생하게 되는데, 이 메시지 전달 과정을 생략하여 시뮬레이션 속도 개선을 이룰 수 있다.

표 1. 시뮬레이션 세부 작업 소요 시간

세부 작업 종류	소요시간(백분율)
1) 상태 전이	1.6초(6%)
2) 이벤트 라우팅	8.9초(32%)
3) 스케줄링	17.6초(63%)

3. 모델합성 기법

앞에서 소개한 세 가지 주요 연산 중 이벤트 라우팅, 스케줄링 연산은 모델의 구조를 변경함으로써 그 계산 부담을 줄일 수 있다. 더욱 이상적으로는 여러 구성요소 모델들을 하나의 원자 모델로 합성하게 되면, 메시지 전달 과정이 더 이상 발생되지 않게 된다. 본 논문에서는 이 과정을 모델 합성이라하며, 다음과 같이 정의한다.

정의2) 모델 합성

주어진 결합모델

과 그것을 구성하는 원자 모델들

가 결합되면 원자 모델 이 생성되어 진다. 의 각 튜플들은 다음과 같이 정의된다.

× ∈ ×

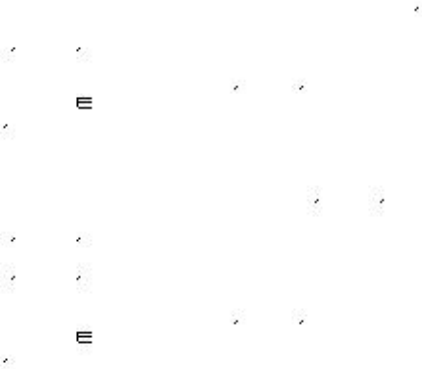
∈ ≤ ≤

그리고 ∈ 은 다음 영향자 이다

∈

∈

∈



시스템 검증 분야에서는 다수 개의 FSMP이 Deadlock과 같이 원하지 않는 상태에 도달하는지 살펴 보거나, 특정 특성들이 전체 State Space 상에서 만족되는지 살펴보고자 Reachability Tree를 생성하는 경우가 있다. 이러한 경우에 두 개 이상의 FSMP에서 도달 가능한 모든 상태들을 도출하게 되는데 이것을 모델합성(Composition)이라 부른다. 그러나, 본 논문에서 제안하는 모델 합성은 이와 같이 전체 State Space 를 생성하는 것이 아니라 단지 State 를 변화시키는 규칙들만을 합치는 과정이기 때문에 상태 폭주 (State Explosion)와 같은 문제가 발생하지 않으며, 합성을 하는 과정도 매우 간단하다.

정의 2)의 모델합성 과정에서는 몇몇 이벤트들이 내부화(internalization) 과정을 거칠 수 있다. 그림 1에서 Buff의 "out" 출력 이벤트와 Proc의 "req" 입력 이벤트는 결합모델 내부에서만 연결 관계가 있기 때문에 내부화되어진다.

합성된 모델의 상태집합은 단순히 각 구성요소 모델들의 상태집합의 조합(Cartesian Product)이 아니라, 각 구성요소 모델의 머문 시간(elapsed time)이 별도로 들어간다. 이것은 각 이벤트들의 다음 스케줄 시간이 머문 시간에 의존할 수 있기 때문이다.

결합모델을 구성하는 여러 원자모델 중 다음 실행될 이벤트를 제공하는 다음 영향자 는 각 원자모델 중 가장 빠른 시간내에 내부 상태 전이를 하는 것으로 정해진다. 그러나, 합성된 모델에서는 더 이상 시뮬레이터의 도움 없이 스스로 다음 영향자 를 알고 있어야 하므로, 상태 변수에 를 가지고 있게 된다. 는 상태 변이가 있을 시마다 SELECT 함수에 의해 결정되어 진다.

합성된 모델은 주어진 구성요소 모델들보다 복잡해지기 때문에 디버깅이 어려워질 수 있다. 반면에, 합성 모델은 메시지 통신이 적고 내부화(Internalization) 과정에 의해 특정 함수들이 결합하기 때문에 그만큼 빨리 실행될 수 있다. 다음 장은 모델합성 과정을 통해 시뮬레이션 속도가 개선됨을 실험을 통해 보인다.

Ⅳ. 모델합성 기법을 이용한 시뮬레이션 속도 개선

합성된 모델에서는 동일한 이벤트에 의해 영향을 받는 원자 모델들의 특성함수들이 상호 결합되기 때문에, 이벤트 라우팅이 더 이상 발생하지 않게 된다. 그러므로 (y), (g)와 같은 메시지 전달이 발생하지 않게 되고 시뮬레이션 속도가 향상될 수 있다. 합성 모델에서 생각해 보아야 할 또 다른 점은 스케줄링 과정에서 (*),(done) 메시지의 전달 없이 단지 지역변수들을 참조함으로써 이루어진다는 점이다. 이러한 이유로 인해 합성 모델은 훨씬 빠른 속도로 실행이 가능할 것이다.

여기서는 합성된 모델의 시뮬레이션 속도 개선 정도를 살펴보기 위해 단일서버 큐 예제와 간단한 통신 프로토콜인 Alternating Bit Protocol (ABP)에 대해 실험을 한다.

1. 실험 모델

단일서버큐 모델에 대한 설명은 2장에서 하였기 때문에 자세한 설명은 생략한다.

두 번째 모델인 ABP는 송신지로부터 목적지까지 통신 메시지를 보내는 프로토콜이다. 이 프로토콜에서는 통신 미디어가 불완전하여서 메시지의 손실이나 중복이 발생할 수 있다고 가정한다. 0 또는 1의 태그를 통신하는 메시지마다 붙여서 안전한 통신이 될 수 있는 프로토콜이다.

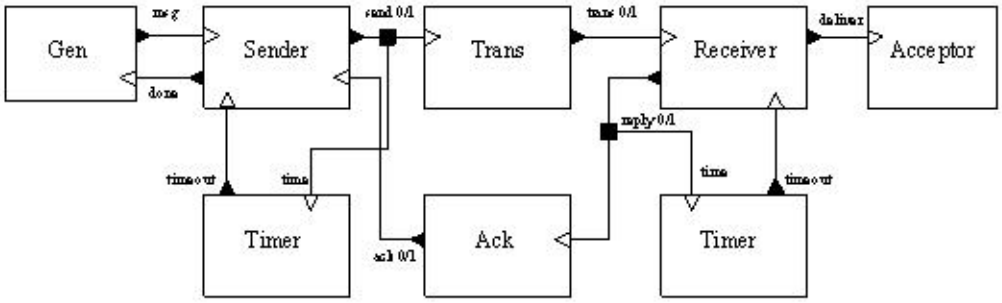


그림 3. ABP 모델의 구성

이 프로토콜을 DEVS 형식론으로 모델링하기 위해 [그림 3]과 같이 8개의 원자모델로 구성하였다. Sender는 메시지를 Trans라 불리는 통신 미디어를 통해서 전달하고, Receiver는 Ack라는 통신 미디어를 통하여 송신자 측에 그 결과를 통보한다. Ack 미디어에서 메시지 손실이 발생할 수 있는데, 이것을 판단하기 위해 Sender와 Receiver는 일정시간이 지나면 timeout 이벤트를 발생시킨다. 만일 timeout이 발생하면 메시지의 재전송 메카니즘이 발생하게 된다. 이러한 과정에서 모든 메시지에는 0 또는 1 비트의 태그가 순차적으로 붙어져서 발송된다. ABP 프로토콜에 대한 자세한 설명은 [14][15]을 참조할 수 있다.

2. 결과 분석

위 두 모델에 대해 Devsim++[9] 시뮬레이션 환경에서 시뮬레이션 하였다. 사용한 컴퓨터 시스템은 펜티엄 4 1.8GHz 모델이었으며, GNU의 gprof을 이용하여 시뮬레이션 실행시간을 측정하면 다음 [표 2]와 같다. 두 예제 모델 모두 모델합성을 적용하였을 시 시뮬레이션 속도가 개선되었는데, 평균적으로 17.8 ((18.5+17) /2) 배 빨라 질 수 있었다. 특히, 합성된 모델에서는 이벤트 라우팅과 관련된 함수 호출이 전혀 없었으며, 스케줄링에 소요되는 시간도 93% (1-(1.2/17.8 + 1.44/20.1)/2) 가량 줄었다.

결론적으로 합성된 모델에서는 이벤트 라우팅과 스케줄링에 소요되는 시간이 대폭 줄어들었음을 알 수 있다.

표 2. 실험 결과

실험모델 / 시뮬레이션 소요시간		Bul-Proc		ABP	
		합성전	합성후	합성전	합성후
세부 소요시간 (초)	상태전이	1.6	0.3	2.2	0.42
	이벤트 라우팅	8.9	0.0	9.3	0
	스케줄링	17.8	1.2	20.1	1.44
전체 소요시간(초)		28.3	1.5	31.6	1.86
속도증가비		1.0	18.5	1.0	17.0

V. 결 론

본 논문에서는 DEVS 모델의 시뮬레이션 속도를 개선할 수 있는 모델합성법에 대하여 제시하였다. 제안한 방법은 전체 시스템을 구성하는 원자모델들을 하나로 합치는 방법으로 원자모델들 간의 연결관계를 사전에 해석하기 때문에 합성된 모델에서는 이벤트 라우팅 과정이 더 이상 발생하지 않게 된다. 또한, 스케줄링 과정에서도 모델들 간의 메시지 전달이 발생하지 않기 때문에 그 계산 속도가 매우 향상될 수 있다.

두 개의 예제 모델에 대해 모델합성법을 적용할 시 평균적으로 18배 정도 속도 향상이 가능함을 실험을 통해서 확인할 수 있었다. 특히, 제안한 방법은 정형화되어 있기 때문에, 모든 DEVS 모델들은 기계적인 과정을 통하여 합성된 후 빠른 속도로 시뮬레이션 될 수 있다. DEVS 모델과 같이 시스템을 계층적으로 모델링할 수

있는 형식론은 시스템을 구조적으로 잘 표현할 수 있으나, 시뮬레이션 속도가 느려지는 단점이 있다. 제안한 방법은 이러한 단점을 보완하여 DEVS 모델의 빠른 시뮬레이션 속도를 보장할 수 있다. 특히 제안한 합성 방법은 정형화되었기 때문에 향후 자동화된 툴로서 쉽게 구현될 수 있다.

참 고 문 헌

[1] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*, Irwin, Boston, 1993.

[2] B. P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Orlando, FL, Academic PRess, 1984.

[3] R. M. Fujimoto, "Optimistic Approaches to Parallel Discrete Event Simulation," Trans. of The Society for Computer Simulation, Vol.7, No.2, pp.153-191, 1990.

[4] B. P. Zeigler and G. Zhang, "Mapping Hierarchical Discrete Event Models to Multiprocessor Systems : Concepts, Algorithm, and Simulation," J. Parallel and Distributed Computing, Vol.10, No.3, pp.271-281, 1990.

[5] Y. H. Wang and B. P. Zeigler, "Implementing the DEVS formalism on a parallel SIMD architecture," International J. in Computer Simulation, Vol.5, No.3, pp.229-245, 1995.

[6] M. S. Ahn and T. G. Kim, "A Framework for Hybrid Modeling/Simulation of Discrete Event Systems," Proceedings of AI Simulation and Planning in High Authomony Systems, pp.199-205, Dec. 1994, Gainesville, FL.

[7] Y. G. Kim and T. G. Kim, "Optimization of Model Execution Time in the DEVSim++ Environment," Proc. of 1997 European Simulation Symposium, pp.215-219, Oct. 1997, Passau, Germany.

[8] D. Lewis, "A Hierarchical Compiled Code Event-Driven Logic Simulator," IEEE Trans. CADICS, Vol.10, pp.726-737, 1991.

[9] ftp://smslab.kaist.ac.kr/pub/DEVSim++-1.0/PC/DEVSim.zip

[10] M. Heymann, "Concurrency and Discrete Event Control," IEEE Control Systems Magazine, Vol.10, No.4, pp.103-112, 1990.

[11] F. W. Vaandrager, "On the Relationship Between Process Algebra and Input/Output Automata," LICS, pp.27-38, 1991.

[12] N. A. Lynch and M. R. Tuttle, "An introduction to Input/Output automata," CWI-Quarterly, Vol.2, No.3, pp.219-246, Sept. 1989.

[13] http://www.cs. utah.edu/dept/old/texinfo/as/gprof.html

[14] K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson, "A Note on Reliable Full-duplex Transmission Over Half-duplex Lines," Comm. of the ACM, Vol.12, No.5, pp.260-265, 1969.

[15] R. Milner, A Calculus of Communicating Systems, *Volume 92 of Lecture Notes in Computer Science*, Springer-Verlag, 1980.

저 자 소 개

이 완 복(Wan-Bok Lee) 종신회원

- 1993년 2월 : KAIST 전기및전자공학과 (공학사)
- 1995년 2월 : KAIST 전기및전자공학과 (공학석사)
- 2004년 2월 : KAIST 전자전산학과 (공학박사)
- 2005년 7월 ~ 현재 : 중부대학교 게임학과 교수

<관심분야> : 시뮬레이션, 컴퓨터 게임, 정보보호, 이산 사건 시스템