한국시뮬레이션학회 논문지
Vol. 15, No. 2, pp. 1-12 (2006. 6)

계층적 계획을 이용한 이산 사건 시뮬레이션 모델링: HRG-DEVS

이미라[1†]

DEVS Modeling with Hierarchical Planning: HRG-DEVS

Mira Yi

ABSTRACT

As the needs of intelligent systems increase, there have been diverse approaches that combine artificial intelligence (AI) and simulation in the last decade. RG-DEVS, which is the basis for this paper, embedded AI planning techniques into the simulation modeling methodology of DEVS, in order to specify dynamically a simulation model. However, a hierarchy concept, which is used for various types of problem solving systems, is not included in the planning of RG-DEVS. The hierarchy concept reduces the computational cost of planning by reducing the search space, and also makes it easy to apply the hierarchical process flow of a target system to planning. This paper proposes Hierarchical RG-DEVS (HRG-DEVS) in an attempt to insert hierarchical planning capability into RG-DEVS. For the verification of the proposed modeling methodology, HRG-DEVS is applied to model the block's world problem of ABSTRIPS, which is a classical planning problem.

Key words : AI and simulation, Simulation modeling methodology, RG-DEVS, Hierarchical planning, ABSTRIPS

요 약

지능형 시스템에 대한 요구가 지속적으로 증가하면서, 최근에는 인공지능과 시뮬레이션 기술을 연동하기 위한 다양한 접근이 이루어지고 있다. 본 논문의 기반이 되는 RG-DEVS는 이산 사건 시뮬레이션 모델링 방법론인 DEVS에 인공지능의 계획(planning) 기술을 반영함으로써 동적으로 시뮬레이션 모델이 정의될 수 있는 인공지능과 시뮬레이션의 연동 기술이다. 그러나, 오늘날 많은 문제 해결 시스템들에 반영되고 있는 계층성(hierarchy)이 계획에 반영되어 있지 않다. 계층성은 탐색 공간을 작게 하여 계획의 계산 비용을 줄일 수 있을 뿐 아니라, 모델링 대상 시스템의 계층적 작업 흐름을 반영하기에도 유용하다. 본 논문은 RG-DEVS에 계층적 계획 능력을 추가하여 확장한 모델링 방법론인 HRG-DEVS를 제안하고, 이를 검증하기 위하여 고전적인 계획 문제로 알려진 계층적인 블록 쌓기 문제인 ABSTRIPS에 적용하였다.

주요어 : 인공지능과 시뮬레이션, 시뮬레이션 모델링 방법론, RG-DEVS, 계층적 계획, ABSTRIPS

1. 서 론

최근 인공지능분야 연구들은 컴퓨터 계산 능력의 급속한 증가에 힘입어 여러 분야의 지능적인 시스템에 대한 요구들을 현실화시키기 위한 방법들이 나오고 있다. 또, 시스템이 복잡해지면서 시스템의 미래 상태를 미리 예측하여 분석해보는 시뮬레이션 분야의 연구도 지능 시스템을 구성하는 중요한 요소가 된다. 그래서, 지능 시스템을 위한 이러한 두 요소 기술을 연동하기 위한 연구가 지난 십여 년에 걸쳐 지속적으로 이루어지고 있다[1-4]. 이러한 인공지능과 시뮬레이션이 연동 방법은 두 시스템간의 상호작용 및 외부와의 상호작용 방식에 따라 여러 형태가 있으며[5,6], 연동하는 기술 또한 다양하다.

시뮬레이션과 주로 연동되는 인공지능 이론들에는 전문가시스템(Expert System), 신경망(Artificial Neural Network), 퍼지(Fuzzy), 계획(planning) 등이 있다. 이 중 계획(planning)[1]은, 대상 시스템의 주어진 제약 조건 하에서 특정 목표를 달성하기 위한 행동을 설계하는데 이용된다. 예를 들어, 로봇의 동작 계획, 제조업에서의 생산 계획, 멀

2005년 12월 7일 접수, 2006년 4월 26일 채택
[1] 목포해양대학교 해양전자통신공학부
주 저 자 : 이미라
교신저자 : 이미라
E-mail; yimira@mmu.ac.kr

1) 일반적으로는 scheduling 보다 좀 더 긴 시간 단위의 작업 계획을 의미하나, 인공지능에서는 표현하는 시스템의 초기상태에서 목표 상태에 이르기까지의 액션(action) 순서를 생성하는 것을 의미한다.

티 에이전트 시스템 상에서의 협력 계획, 네트워크 제어, 객체 지향 모듈 개발 등의 분야들에 적용이 된다.

한편, 이산 사건 시스템에 대한 시뮬레이션 모델링에 대한 체계적인 접근방법으로 DEVS(Discrete EVent system Specification) 형식론이 있다. DEVS에서는 독립적인 행동 단위가 되는 모델과, 모델들을 연동하여 더 큰 단위의 모델을 정의함으로써, 시스템을 체계적으로 조립하듯 표현 할 수 있도록 한다[7]. 이러한 체계적인 모델링 접근은 DEVS를 기반으로 다른 모델링 기술들을 연동하기 위한 연구들을 시도하게 하는 특징이 되며, 실제로 DEVS를 확장한 다양한 연구 결과들이 발표되고 있다[8-12]. 그 중 RG-DEVS는 인공지능 이론을 시뮬레이션의 구성 요소로 사용하여 확장한 예가 된다.

RG-DEVS는 시뮬레이션 모델에서 상태를 변화시키는 액션(action)을 결정하기 위하여, 인공 지능 계획(planning) 시스템의 목표 역행(goal-regression) 계획 메커니즘을 사용하는 DEVS의 이론적인 확장이다[12]. 이렇게 함으로써 모델 작성자는 계획(plan)[2] 생성을 위한 규칙(rule; 지식표현의 한 형태)의 추가만으로 지능적인 시뮬레이션 모델을 작성하기가 용이하다. RG-DEVS에서 반영한 계획(planning) 기술은 한 단계(level)에서의 계획이다. 그러나, 현실 상황에서 문제해결을 위한 계획은 그 단위가 크고 복잡하여 처음(초기상태)부터 끝(목표상태) 까지 구체적인 단계의 모든 계획이 아닌, 중요한 사항부터 일단 결정하고 세부 계획을 세우는 경우들이 많다. 즉, 계층적 계획(hierarchical planning) 이 필요하다는 것이다. 계층적 계획을 함으로써 상위 단계의 계획에 따라 하위 단계에서의 구체적인 계획을 생성하는데, 이러한 특징은 탐색공간(searching space)을 훨씬 작게 하여 계획 생성 비용을 줄일 수 있다[13]

본 논문에서는 인공지능의 계층적 계획(hierarchical planning) 개념을 시뮬레이션에 통합하여 대상 시스템을 모델로 정의하는데 있어 좀 더 체계적인 모델링 방법을 제안하며, 이후 그 이름을 HRG-DEVS(Hierarchical RG-DEVS)라 하겠다. 이 후, 2장에서는 배경이론이 되는 DEVS, RG-DEVS, 인공지능에서의 계층적 계획을 설명하고, 3장에서는 본 연구에서 제안하는 HRG-DEVS 모델링 방법론을 정의한다. 4장에서는 계획 문제의 고전적인 예제인 블록 쌓기 시스템(ABSTRIPS)을 예로 들어 HRG-DEVS를 이용한 모델 정의 및 실행 절차를 설명한다. 마지막으로, 5장에서는 연구 내용에 대한 결론을 맺는다.

2) 액션 순서인계획(plan)과 이 계획(plan)을 생성하는 과정인 계획(planning)을 일반적인 국문 표기법에 따라 같은 단어로 사용하니 이후의 '계획' 용어는 문맥으로 이해해주기 바란다. 다만, 문맥상 의미가 모호 할 경우에는 원어를 함께 기재하겠다.

2. 배경이론

2.1 DEVS 형식론

DEVS[7]는 계층적이고 모듈화된 이산 사건 시스템을 표현하기 위한 방법론으로서, 집합이론을 기반으로 체계적으로 정립된 형식론이다. DEVS에서 대상 시스템은 시간을 기반으로 하는 입력, 상태, 출력, 상태 변환 함수들로 표현되며, 함수들은 현재 상태와 입력을 근거로 하여 다음 상태와 출력을 결정하게 된다. DEVS 형식론에서 시스템을 기술하기 위한 두 가지 모델 유형, 기본(basic) 모델과 결합(coupled) 모델이 있다. 기본 모델(M)은 시스템의 동작 (behavior)의 단위가 되는 시스템의 구성 요소들을 표현하기 위한 것이고, 결합 모델(DN)은 시스템의 구성 요소 간의 상호작용을 의미하는 구조(structure)를 표현하기 위한 것이다. 다음은 이 두 모델의 구성 요소이다.

기본모델 $M = < X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ta >$ 이며, 이 때 X, S, Y는 각각 입력 이벤트, 상태, 출력 이벤트의 집합이고, δ_{int}, δ_{ext}는 각각 내부 상태 변이, 외부 상태 변이 함수이며, λ와 ta는 출력 및 시간 갱신 함수이다. 결합모델 $DN = < D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, select >$ 이며, 이 때 D는 구성 요소가 되는 모델들의 이름 집합을, M_i 은 구성 요소가 되는 i번째 모델을, I_i 는 모델 i가 영향 주는 다른 모델들의 집합을 의미하고, $Z_{i,j}$ 는 모델 i에서 모델 j로의 연결 함수를 의미한다.

2.2 인공지능 계획(planning)

대상 시스템의 초기 상태에서 문제 해결을 표현하는 목표 상태에 이르기까지의 액션(action, 행동)들을 하나의 순서(sequence)로 만든 것을 계획(plan)이라고 하며, 이러한 순서를 찾아 가는 과정을 계획(planning) 이라고 한다[14,15]

계획(planning)을 위한 기술로는 생산 시스템(production system) 기술이 사용되는데, 생산 시스템[16]은 규칙 데이터베이스로 이루어지며, 각각의 규칙은 복합적인 조건문과 그 조건문이 충족 될 때 수행되어야 하는 몇 가지 행동으로 구성된다. 이러한 규칙 기반의 생산 시스템은 주로 if-then 구문들의 목록과 하나 이상의 규칙이 동시에 충족되는 경우에 사용될 수 있는 여러 가지 충돌 분석 메커니즘으로 이루어진다.

규칙 데이터베이스를 기초로 계획을 생성하는 방법에는 순방향(forward) 탐색에 의한 순행(progression) 계획과 후방향 (backward) 탐색에 의한 역행(regression) 계획이 있다. 순방향 탐색은 초기상태에서 목표상태를 찾아가고, 후방향 탐색은 목표상태로부터 초기상태로 유추해 간다.

순향탐색은 초기상태에 순향규칙을 정의하여 적용하고, 후향탐색은 목표상태에 후향규칙을 적용하는 것이다. 후향탐색에서는 후향규칙을 적용한 부목표가 초기상태가 되면 계획이 완성되는 것인데, 이때 후향규칙을 순향규칙에 기초를 두고 사용할 수 있다. 즉, 목표 G를 부목표(sub-goal) G'으로 변환시키는 후향규칙은 논리적으로 G' 상태목록에 적용되어 G 상태목록을 생성하는 순향규칙에 기초를 두게 되는데, 이렇게 순향규칙을 통해 G에서 G'를 얻어내는 과정을 역행(regression)이라 한다. 일반적으로 목표가 명확하게 주어지는 경우, 역행을 이용한 탐색은 순방향 탐색보다 더 효율적이다고 알려져 있다.

2.3 RG-DEVS 형식론

RG-DEVS(ReGression-DEVS)[12]는 앞 절에서 설명한 인공지능의 계획(planning) 능력을 DEVS 모델에 부여하여 확장시킨 모델링 방법론이다. 이는 시뮬레이션 모델이 변화해 나갈 상태 및 상태 변화 관련한 액션 순서(계획;plan)를 시뮬레이션 진행 중 계획(planning) 기술을 이용하여 정의하는 것이다. 즉, 계획(planning) 결과가 동적으로 변하는 모델이 되고, 계획(plan) 실행이 시뮬레이션이 되는 것이다. 계획을 생성하기 위한 메커니즘으로 후향탐색의 역행(regression)을 사용하는데, 이는 목표상태를 외부 입력 이벤트 형태로 명확히 받기 때문이다. 계획과정에서 선정된 규칙들은 목표(goal)를 얻기 위한 계획(plan)을 만드는 액션 순서가 되고, 모델의 순차적인 상태들은 역행과정동안 모델의 상태를 묘사하기 위해 미리 정의된 서술식 (predicate formula)과 액션들에 대응되는 부목표를 결합함으로써 정의된다. 따라서, 모델의 순차적 상태들은 모델의 입력으로 들어오는 목표-모델이 도달 할 수 있는 가능한 목표들 중에 하나인 - 에 따라 달라지므로 자연스럽게 시뮬레이션 모델이 동적으로 정의된다.

다음은 이러한 기능을 가능하게 하는 RG-DEVS 모델링 형식론의 구성 요소를 나타낸 것이다.

- $M_{RG} = \langle X, S_R, Y, \delta_{ext}, \delta_{int}, \lambda, ta, F \rangle$

이 때,

X, Y : 입력 이벤트 집합, 출력 이벤트 집합

S_R : 기존 DEVS에서 S에 해당하는 순차적 상태 집합; $S_R = S^C \times S^D{}_R$,
S_C : 일반적인 상태 집합
$S^D{}_R$: 목표의 역행 계획에 의한 상태서술 집합

δ_{int} : 내부 상태 변이 함수;
$S_R \times (Fu \cup \emptyset) \rightarrow S_R$,
$Fu = \{ fu \mid f \in F, u \text{ is mgu}^{3)} \}$
Fu: 목표의 역행 계획에 의한 규칙 인스턴스 집합

δ_{ext} : 외부 상태 변이 함수; $Q \times X \rightarrow S_R$
$Q = \{(s,e) \mid s \in S_R,\ 0 \le e \le ta(s)\}$,
e: 마지막 이벤트 이후로 경과된 시간

λ : 출력 함수; $S_R \rightarrow Y$

ta : 시간 갱신 함수; $S_R \rightarrow R^+{}_{0,\infty}$,
$R^+{}_{0,\infty}$: 0에서 무한대까지의 실수 집합

F : 계획(planning)을 위한 규칙 집합

외부 입력 이벤트를 통해 주어지는 목표에 따라 달라지는 모델의 순차적 상태는 역행된 상태 서술(부목표 또는 목표 상태를 표현한) 집합 $S^D{}_R$이 되며, $S^D{}_R = \{ s^d \mid s^d = R[E:fu] \vee E \} \cup \{ s^d{}_0 \}$ 이다. 이때 E는 목표 서술식이고, R[E:fu]은 서술식 E가 규칙 인스턴스(instance) fu에 의해 역행된 결과이며, $s^d{}_0$는 초기 모델 상태를 의미한다. 내부 상태 변이 함수의 $S_R \times (Fu \cup \emptyset) \rightarrow S_R$ 에서, Ø는 모델이 계획의 마지막 액션 규칙 실행이 끝나고 초기 상태로 다시 돌아오는 경우에 필요하다. λ는 DEVS에서와 같이 출력함수이며 정의역이 S_R이라는 것만 다르다. Fu는 계획에 의해 선정된 규칙들의 집합이며, 각 규칙은 if-then 구문의 전제조건과 액션(상태 서술의 삭제와 추가를 포함)으로 구성되며, 해당 규칙을 실행 할 때 필요한 처리 시간 요소가 추가 된다. 계획된 상태 집합 $S^D{}_R$에서 목표 상태 E에 대한 서술식은 $L \wedge G_{1,M} \wedge \cdots \wedge G_{J,M}$ 로 표현되는데, L은 fu를 적용하여 얻어지는 상태에 대한 서술식이며 $G_{i,M}$은 L을 제외한 fu에 의해 획득되지 않은 i번째 서술식을 의미한다.

다음은 역행되는 과정 한 단계를 표현한 식이다.

$s^d{}_i = L_i G_i$,
; $L_i G_i$는 $L \wedge G1,i \wedge \cdots \wedge G_{j,i}$ 의 생략된 표현으로, i가 M일 때 $L_i G_i$는 계획의 목표상태이고 i가 1일 때 $L_i G_i$는 계획의 초기상태이다.

3) mgu : most general unifier

$s^d_{i-1} = R[L_iG_i:f_iu_i]$
$= P_iu_i \wedge R'[G_i:f_iu_i]$; P_iu_i는 i번째 역행 식
$= L_{i-1}G_{i-1}$

위의 과정을 통해, 목표 상태에서 초기(현재) 상태까지 이르면 계획(fu의 순서)이 완성된다. 목표 상태는 모델에 외부 입력을 통해 설정이 되고, 외부 입력의 처리를 하는 δ_{ext}에 의해 계획이 진행된다. 이렇게 외부 입력 처리의 결과로 생성된 계획은 역으로 초기상태부터 목표상태까지 δ_{int}을 통해 내부 상태 변이 과정을 거치며 실행된다.

다음은 i-1상태에서 i상태로의 상태 변이 과정의 전개식이다.

$\delta_{int}(s_{i-1},f_iu_i) = s_i$, 즉,
$\delta_{int}(s_{i-1},f_iu_i) = \delta_{int}((s^c, s^d_{i-1}), f_iu_i)$
$= \delta_{int}((s^c, L_{i-1}G_{i-1}), f_iu_i)$
$= (s^c, L_iG_i)$
$= (s^c, s^d_i)$
$= s_i$

2.4 계층적 계획

인공지능 개념이 소개된 이래, 계층적 문제 해결은 계획의 계산 비용을 줄이기 위한 방법으로 사용되어 왔다[13]. 계층적 문제 해결 개념은 목표와 중요도가 다른 액션들을 구별하여 중요한 문제를 먼저 풀겠다는 것을 의미한다. 이러한 개념은 구체적인 행위들은 잠시 무시하고 특정 행위(activity)만을 우선적으로 강조함으로써, 구체적인 행위 단계(level) 에서의 계획(plan)을 찾아가는 탐색 공간 (search space)을 훨씬 작게 하여 계획의 계산 비용을 줄일 수 있는 것이다.

그림 1은 세 단계를 거쳐 계획이 생성되는 것을 표현한 것이다. 상세 계획을 단일 계층의 계획으로 할 경우에는, 하나의 액션을 결정하기 위한 후보 액션이 모든 액션이 되지만, 여기에서는 상위 단계의 계획에 의한 액션을 가능하게 하는 것과 연관된 상세 계획의 액션들만 후보 액션이 된다. 계층성은 계획의 비용 외에도 모델 설계자가 계획 규칙을 정의하고 관리하는 것을 용이하게 하는 효과가 있다. 오늘날, 많은 문제 해결 시스템들이 이러한 계층적 계획 개념을 반영하고 있다[17-20].

3. HRG-DEVS 형식론

이 장에서는 본 논문에서 제안하는 계층적 계획 개념

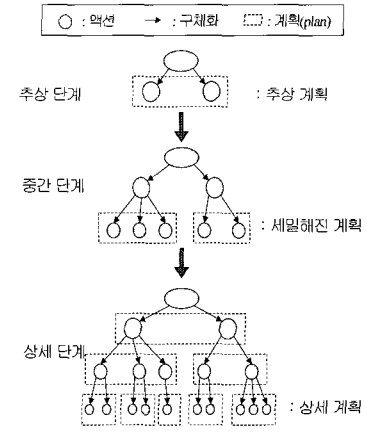

그림 1. 계층적 계획

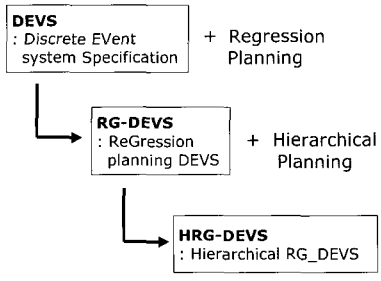

그림 2. DEVS, RG-DEVS, HRG-DEVS

이 반영된 이산 사건 시뮬레이션 모델링 형식론인 HRG-DEVS(Hierarchical ReGression DEVS)를 정의한다. 앞 장에서 인공지능의 계획(planning) 기술을 시뮬레이션에 통합하기 위해 DEVS를 확장한 RG-DEVS를 살펴보았는데, HRG-DEVS는 이러한 RG- DEVS에 계층적 계획 기술을 추가하여 한 번 더 확장시킨 것이다. 그림 2는 이러한 DEVS와 확장된 형식론과의 관계를 보여준다.

본 장에서는 우선 모델링을 위한 구성 요소를 정의하고, 각 구성 요소들을 이용하여 규칙을 선정하고 상태가 생성되는 식을 전개함으로써 시뮬레이션 모델을 동적으로 정의하기 위해 계획의 계층성이 어떻게 반영되는지를 정의한다. 또, HRG-DEVS 형식론에 의한 모델의 실행관점에서, 하나의 외부입력 처리를 하기 위해 계획이 생성되고 그 계획을 실행시킴으로써 시뮬레이션이 진행되는 절차를 설명한다.

3.1 HRG-DEVS 구성 요소

HRG-DEVS에 의한 기본 모델(M_{HRG})의 구성 요소는

다음과 같다.

- $M_{HRG} = < X, S_R, Y, \delta_{ext}, \delta_{int}, \lambda, ta, F >$ 이때,

N 개(1,2,…,n)의 계층을 가정하며, 각 요소는 다음의 의미를 갖는다.

X, Y: 입력 이벤트 집합, 출력 이벤트 집합

S_R :DEVS에서 S에 해당하는 순차적 상태집합;

$S_R = S^C \times S^D{}_R$,

S^C : 일반적인 상태 집합,

$S^D{}_R$: 목표의 역행 계획에 의한 상태 서술(description) 집합,

$S^D{}_R = S^{D1}{}_R \times S^{D2}{}_R, \times \cdots \times S^{Dn}{}_R$

($S^{Dk}{}_R$: k 계층의 역행 계획에 의한 상태 서술 집합, $1 \le k \le n$)

δ_{int} : 내부 상태 변이 함수;

$S_R \times (F^n u \cup \emptyset) \rightarrow S_R$,

$F^n u = \{ f^n u \mid f^n \in Fn, u \text{ is mgu} \}$,

:목표의 역행 계획에 의한 최하위 단계 n에서의 규칙 인스턴스 집합

δ_{ext} : 외부 상태 변이 함수; $Q \times X \rightarrow S_R$

$Q = \{(s,e) \mid s \in S_R, \ 0 \le e \le ta(s)\}$,

e: 마지막 이벤트 이후로 경과된 시간

λ : 출력 함수; $S_R \rightarrow Y$

ta : 시간 갱신 함수; $S_R \rightarrow R^+{}_{0,\infty}$

F : $F = \{ F^k \mid k$ 계층에서의 계획을 위한 규칙 집합, $1 \le k \le n \}$

기본 구조는 앞에서 설명한 RG-DEVS 모델과 동일하지만, 모델의 상태 S_R과 오퍼레이션 규칙 F의 표현식에 계층성을 반영한다는 것이 다르다.

계층이 최상위 1단계부터 최하위 n단계까지 정의되어 있다고 가정 할 때, 오퍼레이션 규칙 집합인 F는 계층적 계획에서의 각 단계에 해당하는 오퍼레이션 규칙 집합들 ($F^1, F^2, \cdots, F^n$)의 집합이 되고, 계획된 상태 집합 S_R은 각 단계의 상태 집합들($S^{D1}{}_R, S^{D2}{}_R, \cdots, S^{Dn}{}_R$)의 카티션곱이 된다.

각 단계의 상태집합 $S^{Dk}{}_R$은 k단계에서의 오퍼레이션 규칙 집합 F^k에 의해 계획되는 상태 집합이며, $S^{Dk}{}_R = \{ s^{dk} \mid s^{dk} = R^k[E^k:f^k u] \vee E^k \} \cup \{ s^{dk}{}_0 \}$ 로 표현된다. 이 때, E^k는 목표 서술식이고, $R^k[E^k:f^k u]$ 은 서술식 E^k가 규칙 인스턴스 $f^k u$에 의해 역행된 결과이다.

규칙의 인스턴스 집합인 Fu는 최하위 계층 n단계의 규칙 인스턴스 집합 $F^n u = \{ f^n u \mid f^n \in F^n, u \text{ is mgu} \}$ 가 된다. 최하위 단계의 규칙을 정의 할 때는 일반적인 if-then 구문의 규칙에서 액션 처리에 필요한 시간 값 요소가 추가 되어야 하는데, 이는 시뮬레이션 실행시 이벤트 처리 시간으로 반영시키기 위한 것이다. 반면, 최하위 단계까지 오기 위한 상위 단계들의 규칙은 최하위 단계의 액션을 결정하기 위한 (실제 시뮬레이션 실행을 위한 것이 아닌) 과정으로써 의미가 있는 것이므로 액션 처리 시간 값은 정의되지 않는다.

3.2 규칙 선정과 상태 생성

$S^D{}_R$는 계획이 진행되는 동안의 부목표 및 목표의 상태를 기술한 값들의 집합으로, 각 계층별 상태들 $S^{D1}{}_R, S^{D2}{}_R, \cdots, S^{Dn}{}_R$의 카티션곱으로 정의된다. 이 $S^D{}_R$가 생성되는 과정은 다음과 같다.

$$
\begin{aligned}
&S^{Dk}{}_R \\
&= \{ s^{dk} \mid s^{dk} = R^k[E^k:f^k u] \vee E^k \} \cup \{ s^{dk}{}_0 \} \\
&= \{ s^{dk} \mid s^{dk} = R^k[L^k \wedge G^k{}_{1,M} \wedge \cdots \wedge G^k{}_{j,M} :f^k u] \\
&\qquad \vee (L^k \wedge G^k{}_{1,M} \wedge \cdots \wedge G^k{}_{j,M}) \} \\
&\quad \cup \{ s^{dk}{}_0 \}
\end{aligned}
$$

이 때, 서술식 $L^k \wedge G^k{}_{1,M} \wedge \cdots \wedge G^k{}_{j,M}$ 이 목표 상태 E^k를 의미한다. L^k은 $f^k{}_u$를 적용하여 얻어지는 상태에 대한 서술식이며 $G^k{}_{j,M}$은 L^k을 제외한, 즉 $f^k{}_u$에 의해 획득되지 않은, j번째 서술식을 의미한다.

계층적 계획의 깊이우선 탐색 기법을 사용했을 때를 기준으로, 계획을 통해 모델의 상태가 생성되어 가는 과정은 다음 식과 같다.

$$
\begin{aligned}
&s^d{}_i \\
&= (s^{d1}{}_i, s^{d2}{}_i, \cdots, s^{dn}{}_i) \\
&= (L^1{}_i G^1{}_i, L^2{}_i G^2{}_i, \cdots, L^n{}_i G^n{}_i)
\end{aligned}
$$

$$
\begin{aligned}
&s^d{}_{i-1} \\
&= (L^1{}_i G^1{}_i, L^2{}_i G^2{}_i, \cdots, R^n[L^n{}_i G^n{}_i : f^n{}_i u_i]) \\
&\quad \vee (L^1 i G^1 i, L^2 i G^2 i, \cdots, R^{n-1}[L^{n-1}{}_i G^{n-1}{}_i : f^{n-1}{}_i u_i], \\
&\qquad\qquad R^n[L^n{}_i G^n{}_i : f^n{}_i u_i]) \\
&\quad \cdots \\
&\quad \vee (L^1{}_i G^1{}_i, R^2[L^2{}_i G^2{}_i : f^2{}_i u_i], \cdots, R^n[L^n{}_i G^n{}_i : f^n{}_i u_i]) \\
&\quad \vee (R^1[L^1{}_i G^1{}_i : f^1{}_i u_i], R^2[L^2{}_i G^2{}_i : f^2{}_i u_i], \cdots, \\
&\qquad\qquad R^n[L^n{}_i G^n{}_i : f^n{}_i u_i])
\end{aligned}
$$

$$= (L^1_iG^1_i, L^2_iG^2_i, \cdots, P^n_iu_i \wedge R^{n'}[G^n_i:f^n_iu_i])$$
$$\vee (L^1_iG^1_i, L^2_iG^2_i, \cdots, P^{n-1}_iu_i \wedge R^{n-1} \prime [G^{n-1}_i:f^{n-1}_iu_i]), P^n_iu_i \wedge R^n \prime [G^n_i:f^n_iu_i])$$
$$\cdots$$
$$\vee (L^1_iG^1_i, P^2_iu_i \wedge R^2 \prime [G^2_i:f^2_iu_i], \cdots, P^n_iu_i \wedge R^n \prime [G^n_i :f^n_iu_i])$$
$$\vee (P^1_iu_i \wedge R^1 \prime [G^1_i:f^1_iu_i], P^2_iu_i \wedge R^2 \prime [G^2_i:f^2_iu_i], \cdots, P^n_iu_i \wedge R^n \prime [G^n_i:f^n_iu_i])$$
$$= L^1_{i-1}G^1_{i-1}, L^2_{i-1}G^2_{i-1}, \cdots, L^n_{i-1}G^n_{i-1}$$

위의 식에서 $L^k_iG^k_i$는 i번째 상태에서 k단계의 목표(또는 부목표)상태를 의미하는 $L^k \wedge G^k_{1,i} \wedge \cdots \wedge G^k_{j,i}$ 의 간략한 표현이고, $f^k_iu_i$는 i번째 상태에서 k단계의 상태 변화를 일으키는 오퍼레이션 규칙 인스턴스이다. $P^k_iu_i$는 k단계에서 $f^k_iu_i$에 의해 역행된 이후에 변화된 부분의 상태 서술식이다.

i번째 상태가 결정되는 것은 궁극적으로 계층의 최하위 단계인 n계층에서의 오퍼레이션 규칙 인스턴스 $f^n_iu_i$에 의해 이루어지지만, $f^n_iu_i$가 결정되기 위해 상위 단계에서의 상태 변화가 함께 일어나야 하는 경우들도 있다. 즉, 모델에서의 상태 변화 결과는 최하위 단계의 $f^n_iu_i$를 포함한 상위단계의 오퍼레이션 규칙들 ($f^1_iu_i$, $f^2_iu_i$, ···, $f^{n-1}_iu_i$)에 대해 각 단계의 역행(regression) 계획이 실행되는 것을 고려한 모든 가능한 경우들의 논리합이 된다.

위의 과정을 거치는 동안 선정되는 최하위 단계의 규칙 인스턴스인 $f^n_iu_i$가 계획(plan)을 완성하기 위한 하나의 액션으로 추가된다.

3.3 계획 기반의 상태 변이

DEVS 기반의 모델에서는 외부 상태 변이 와 내부 상태 변이가 정의되는데, 이에 해당하는 함수가 각각 δ_{ext}, δ_{int}이다.

HRG-DEVS에서 외부 상태 변이 함수 δ_{ext}는 모델이 도달해야 하는 목표 상태 정보를 갖는 외부 입력 이벤트에 대한 처리 함수이다. 외부 이벤트가 발생하면 시뮬레이터는 목표를 시작으로 하는 역행 계획(planning)을 실행 시키고, 계획(planning)의 결과로 부목표가 되는 상태들(s^d_1, s^d_2, ···, s^d_m)과 선정된 규칙들의 순서인 계획($f^n_1u_1$, $f^n_2u_2$, ···, $f^n_{m-1}u_{m-1}$, ø)이 생성된다. 이렇게 계획이 생성된 후에는 δ_{ext}에 의해 아래의 전개식과 같은 상태 변이가 일어난다.

δ_{ext} : $Q \times X \rightarrow S_R$
; $Q = \{(s, e) | s \in S_R, 0 \le e \le ta(s)\}$
$\delta_{ext}(s_0, e, x)$
$= \delta_{ext}((s^c, s^d_0), e, x)$
$= (s^c, (L^1_1G^1_1, L^2_1G^2_1, \cdots, L^n_1G^n_1))$
$= (s^c, (s^{d1}_1, s^{d2}_1, \cdots, s^{dn}_1))$
$= (s^c, s^d_1)$
$= s_f$

δ_{ext}와 달리, 내부 상태 변이 함수 δ_{int}는 자동적으로 발생하는 내부 이벤트에 대한 처리 함수로서, 외부 입력 처리 과정에서 생성된 상태들을 계획(plan)에 있는 규칙 인스턴스 $f^n_iu_i$를 이용하여 상태 변이를 실행한다. 아래 식은 δ_{int}에 의해 i-1번째 상태에서 i번째 상태로의 상태 변이 과정을 전개한 것이다.

δ_{int} : $S_R \times F_u \rightarrow S_R$; $Fu = F^nu$
$\delta_{int}(s_{i-1}, f^n_iu_i) = s_i$, 즉,
$\delta_{int}(s_{i-1}, f^n_iu_i) = \delta_{int}((s^c, s^d_{i-1}), f^n_iu_i)$
$= \delta_{int}((s^c, (s^{d1}_{i-1}, s^{d2}_{i-1}, \cdots, s^{dn}_{i-1})), f^n_iu_i)$
$= \delta_{int}((s^c, (L^1_{i-1}G^1_{i-1}, L^2_{i-1}G^2_{i-1}, \cdots, L^n_{i-1}G^n_{i-1})), f^n_iu_i)$
$= (s^c, (L^1_iG^1_i, L^2_iG^2_i, \cdots, L^n_iG^n_i))$
$= (s^c, (s^{d1}_i, s^{d2}_i, \cdots, s^{dn}_i))$
$= (s^c, s^d_i)$
$= s_i$

3.4 HRG-DEVS 모델의 실행

시뮬레이션은 대상 시스템을 표현한 모델을 시뮬레이터를 통해 실행시키는 것인데, 시스템의 명세는 모델에 있는 상태 변수들에 대한 초기값과 입력에 대한 시간 구분 값들로 기술되며, 시뮬레이터는 실행되는 모델의 상태와 출력의 경로를 생성하는 역할을 담당한다[7].

하나의 HRG-DEVS 모델을 정의 및 실행시키기 위해서는 시스템의 명세가 기술된 모델(HRG-DEVS-Model)은 동적으로 변하는 모듈(a)과 계획(planning)을 위한 지식 베이스 모듈(b)이 필요하고, 시스템 명세를 생성하고 변이시키는 주체인 시뮬레이터(HRG-Simulator)는 이벤트 스케줄링을 담당하는 모듈(c)과 계획 생성 모듈(d)이 필요하다. 이러한 네 모듈들이 서로 연동하여, 하나의 외부입력 처리를 하기 위한 계획이 생성되고, 그 계획을 실행시킴으로써 시뮬레이션이 진행된다. 여기에서 외부 입력이라 함은 모델이 변화해 나갈(또는 계획해 나가야 할) 목표가 되는 상태를 의미한다는 것을 상기하자. HRG-

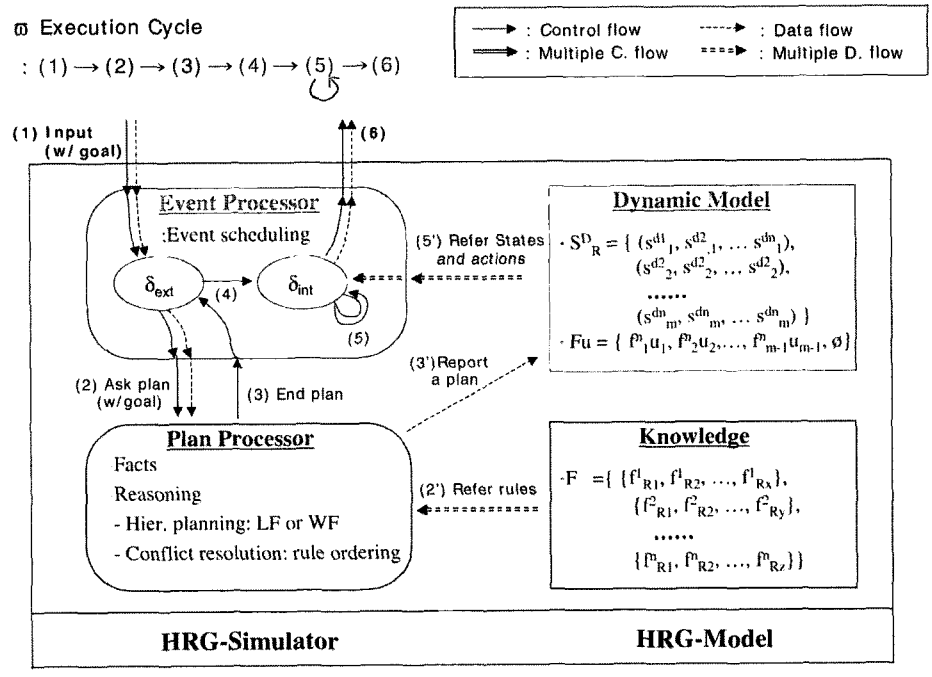

그림 3. HRG-DEVS 모델의 실행 주기

DEVS 모델이 하나의 외부 입력 이벤트에 대한 처리를 완료하기까지의 실행 주기는 그림 3과 같다. 실행 주기에 참여하는 구성 요소들은 앞서 언급한 네 개의 모듈 a, b, c, d인데, 이는 각각 Dynamic Model, Knowledge, Event Processor, Plan Processor 이다. 즉, Dynamic Model과 Knowledge은 모델의 실행에 필요한 정보(대상 시스템의 명세)가 정의된 부분이며, Event Processor와 Plan Processor는 모델 실행을 담당하는 프로세서들이다.

그림에서 괄호 숫자들의 순서는 외부 입력에 대한 처리 과정의 순서를 표기한 것이다. 모델에서의 전체적인 실행 흐름은 Event Processor가 주도하는데, 이 실행 주기는 모델의 상태가 변이 되어 최종적으로 도달해야 할 목표상태의 정보를 갖고 있는 입력(1)을 받는 것으로 시작된다. Event Processor는 입력이 들어오면 우선 외부사건에 대한 처리함수(δ_{ext})를 실행하는데, 상태 변이 실행에 앞서 목표까지의 계획을 생성한다(2). 계획 생성은 Plan Processor를 통해 이루어지는데, 이때 Knowledge 모듈에 있는 각 단계별 오퍼레이션 규칙에 근거하여(2') 이루어진다. 현 모델에서 계층적 계획 알고리즘으로는 도메인에 따라 깊이 우선 또는 너비 우선 방식을 따르며, 충돌 해결(conflict resolution) 방법으로는 규칙이 정의된 순서에 우선순위를 둔 방식(rule ordering)을 따른다. 계획 생성이 끝나면 Event Processor에게 제어가 이동하고(3), 완성된 계획을 Dynamic Model 모듈에 저장함으로써(3') 모델이 변화될 상태 및 인스턴스 규칙 리스트가 완성된다. 계획이 생성된 이후에는 모델의 초기상태 s^d_0에서 첫번째 상태 s^d_1로 상태 변이를 함으로써 δ_{ext} 처리를 마치고, 외부 상태 변이가 끝난 후에는 δ_{int}를 통해 내부 상태 변이가 실행된다(4). 내부 이벤트에 대한 처리 함수 δ_{int}에서는, 계획의 결과로 정의된 Dynamic Model의 정보를 참고하여(5') 상태 변이를 실행한다(5). δ_{int}는 Dynamic Model에 정의되어 있는 오퍼레이션 규칙 $f^n_{m-1}u_{m-1}$에 의한 마지막 상태 $s^d_m = (s^{d1}_m, s^{d2}_m, \cdots, s^{dn}_m)$ 에 이르기까지 반복되며, 마지막으로 널(null) 오퍼레이션 규칙인 Ø에 의한 상태 변이를 통해 모델을 s^d_0 상태로 바꾸는 것으로 내부 상태 변이 δ_{int}의 반복을 마친다. 내부 상태 변이가 다 끝난 후에는 이벤트 제어권을 모델 밖으로 넘긴다(6).

지금까지의 과정을 요약하면, 외부 입력이 들어온 후 (2), (3)과정을 통해 모델이 실행할 순차적 규칙 인스턴스 및 상태들이 자동으로 생성되고, 이의 역순으로 (5)과정의 내부 상태 변이를 반복하며 시뮬레이션이 실행된다.

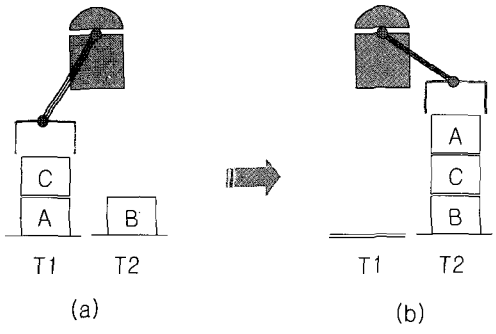

그림 4. 블록쌓기: 현재상태 (a), 목표상태 (b)

표 1. 상태 표현을 위한 술어(predicate)

Predicate	Description
On (a, b)	블록 a는 블록 b위에 있다
ClrBlk (a)	블록 a위에 아무 것도 없다
AtTbl (a,t)	블럭 a는 탁자 t에 있다
HHold (a)	로보트 손이 블록 a를 쥐고 있다
HAt (t)	로보트 손이 탁자 t에 있다.
HEmpty	로보트 손이 비어 있다

- a,b : 임의의 블록을 표현한 변수
- t : 블록을 놓을 수 있는 임의의 탁자 변수

4. 블록 쌓기 문제의 HRG-DEVS 모델링

이 장에서는 AI에서 계획 문제의 고전적인 예로 쓰이고 있는 블록 쌓기 문제[15]에서의 계층적 접근 시스템인 ABSTRIPS를 약간 수정하여 HRG-DEVS 모델로 작성하고 이의 실행 과정을 보인다.

4.1 블록 쌓기 문제

블록 쌓기 문제에서는 세 가지 구성 물질이 있는데, 이들은 블록이 쌓일 수 있는 탁자, 동일한 크기의 정사각형 블록, 블록을 옮기는 주체가 되는 로보트이다. 로보트는 몸체는 고정되어 있는 상태에서 팔과 손을 움직여 블록을 빈 탁자 혹은 다른 블록 위로 쌓을 수 있으며 한 번에 하나의 블록만 들 수 있다. 단, 여기에서 탁자는 블록 한 줄만 위로 쌓을 수 있는 공간이라도 가정한다.

그림 4는 블록 쌓기 문제를 표현한 초기 상태와 목표 상태의 한 예로서, 앞으로 전개될 HRG-DEVS모델에서의 계획 생성 및 시뮬레이션(계획 실행)을 위해 가정되는 상태이다.

표 2. 로보트의 행동(operation) 규칙

Lev.	Operation	Rule		
		Pre-condition	Action	
			Delete	Add
1	stack (a,b)	ClrBlk (a) ClrBlk (b)	ClrBlk (b)	On (a,b)
2	pickup (a,t)	AtTbl (a,t) ClrBlk (a) HAt (t) HEmpty	AtTbl (a,t) HEmpty	HHold (a)
	putdown (a,t)	HHold (a) HAt (t)	HHold (a)	HEmpty AtTbl (a,t)
	move (t1,t2)	HAt (t1)	HAt (t1)	HA t(t2)

표 3. 로보트 팔의 액션 처리 시간

Operation	Processing Time
Pickup (a,t)	3.0
PutDown (a,t)	3.0
Move (t1,t2)	1.5

4.2 블록 쌓기 문제에서의 계획(planning)

블록 쌓기 문제에서 대상 시스템의 상태를 표현하기 위한 술어(predicate)들은 표 1과 같고, 목표가 되는 형태로 블록을 쌓기 위한 로보트의 행동 규칙은 표 2와 같다.

시뮬레이션 모델을 작성하여 생성하기 위해서는 최하위 단계에서 각 액션들을 실행시키기 위한 처리 시간을 정의해 주어야 한다. 표 3은 표 2에서 최하위 단계인 level 2에 해당하는 액션들의 처리 시간이다.

그림 4에서의 초기 상태와 목표 상태는 표 1의 상태 표현 술어들을 이용하여 표현하면, 각각 다음과 같다.

- 초기 상태: AtTbl(A,T1)∧AtTbl(C,T1)∧On(C,A)
  ∧ClrBlk(C)∧AtTbl(B,T2)∧ClrBlk(B)
  ∧HAt(T1)∧HEmpty
- 목표 상태 : AtTbl(B,T2)∧AtTbl(C,T2)∧On(C,B)
  ∧AtTbl(A,T2)∧On(A,C)∧ClrBlk(A)
  ∧HAt(T2)∧HEmpty

HRG-DEVS 모델에서 목표 상태로부터 초기 상태를 발견할 때까지의 각 단계별 계획 생성은 그림 5의 과정을 거쳐 다음과 같은 계획 결과를 만들어 낸다.

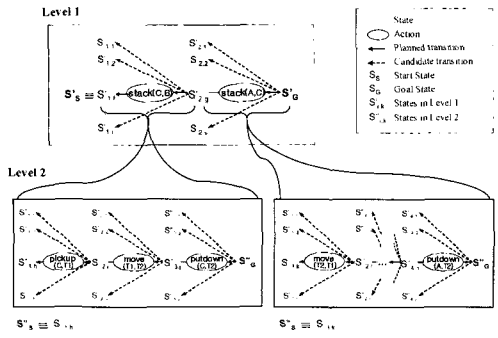

그림 5. 계층적 계획 과정

- Level 1에서 목표 상태로의 계획
:{*stack*(C,B), *stack*(A,C)}

- Level 2에서 목표 상태로의 계획
:{pickup(C,T1), move(T1,T2), putdown(C,T2), move(T2,T1), pickup(A,T1), move(T1,T2), putdown(A,T2)}

이 중 최하위 단계인 Level 2에서의 계획은 이후 시뮬레이션을 진행하기 위한 액션 순서가 되는데, 마지막 액션 putdown(A,T2)이 실행되면 주어진 목표 상태(On(A,C)∧On(C,B)∧ClrBlk(A), AtTbl(B,T2)∧HAt(T2)∧HEmpty)에 대한 시뮬레이션 실행 주기는 끝나게 된다.

계획(planning) 과정이 끝난 후에는 $S^D{}_R$과 Fu가 다음과 같은 내용으로 갱신된다.

$S^D{}_R$
$=\{s^d{}_1(\mathrm{On(C,A)}\wedge\mathrm{ClrBlk(C)}\wedge\mathrm{ClrBlk(B)},\ \mathrm{AtTbl(C,T1)}\wedge\mathrm{HAt(T1)}\wedge\mathrm{HEmpty}),$
$s^d{}_2(\mathrm{ClrBlk(C)}\wedge\mathrm{ClrBlk(B)},\mathrm{HAt(T1)}\wedge\mathrm{HHold(C)}),$
$\cdots\cdots,$
$s^d{}_8(\mathrm{On(A,C)}\wedge\mathrm{On(C,B)}\wedge\mathrm{ClrBlk(A)},\ \mathrm{AtTbl(B,T2)}\wedge\mathrm{HAt(T2)}\wedge\mathrm{HEmpty})\ \}$

Fu = { pickup(C,T1), move(T1,T2), putdown(C,T2), move(T2,T1), pickup(A,T1), move(T1,T2), putdown(A,T2) }

4.3 블록 쌓기 문제에서의 시뮬레이션(계획 실행)

계획(planning)의 결과로 생성된 $S^D{}_R$ 와 Fu를 기초로 시뮬레이션(계획 실행)을 통해 모델의 상태 변이가 일어나는데, 다음 식은 그 첫번째 상태 변이가 진행되는 과정을 전개한 것이다.

$$
\begin{aligned}
&\delta_{ext}\ (s_0,\ e,\ x)\\
&=\delta_{ext}((s^c,s^d{}_0),\ e,\ x)\\
&=\delta_{ext}(((\sigma,\ v_d),\ s^d{}_0),\ e,\ x)\\
&=\delta_{ext}(((\sigma,\ (v^1{}_d,\ v^2{}_d)\ ),\ s^d{}_0),\ e,\ x)\\
&=\delta_{ext}(((\infty,(\mathrm{On(C,A)}\wedge\mathrm{ClrBlk(C)}\wedge\mathrm{ClrBlk(B)},\mathrm{AtTbl}\\
&\qquad(\mathrm{A,T1})\wedge\mathrm{AtTbl(B,T2)}\wedge\mathrm{AtTbl(C,T1)}\\
&\qquad\wedge\mathrm{HAt(T1)}\wedge\mathrm{HEmpty}))\\
&\qquad,\ \mathrm{passive}\ ),\\
&\qquad 0,\ (\mathrm{On(A,C)}\wedge\mathrm{On(C,B)}\wedge\mathrm{ClrBlk(A)},\\
&\qquad\quad\mathrm{AtTbl(B,T2)}\wedge\mathrm{HAt(T2)}\wedge\mathrm{HEmpty}\ )\\
&\quad)\\
&=((0,\ (\mathrm{On(C,A)}\wedge\mathrm{ClrBlk(C)}\wedge\mathrm{ClrBlk(B)},\\
&\qquad\mathrm{AtTbl(A,T1)}\wedge\mathrm{AtTbl(B,T2)}\wedge\mathrm{AtTbl(C,T1)}\\
&\qquad\wedge\mathrm{HAt(T1)}\wedge\mathrm{HEmpty}),\\
&\qquad(\mathrm{On(C,A)}\ \wedge\mathrm{ClrBlk(C)}\ \wedge\ \mathrm{ClrBlk(B)},\\
&\qquad\mathrm{AtTbl(C,T1)}\wedge\mathrm{HAt(T1)}\wedge\mathrm{HEmpty}\ )\\
&\quad)\\
&=((\sigma,\ v_d),\ (s^{d1}{}_1,s^{d2}{}_1)\ )\\
&=(s^c,\ s^d{}_1)\\
&=s_1
\end{aligned}
$$

이전의 passive 상태에서 $S^D{}_R$의 첫번째 상태 서술 값인 $s^d{}_1$(현재 상태 s의 계획에 의한 상태 서술)로 갱신한다. $s^d{}_1$ 상태는 목표 상태에 도달하기 위해 정의된 초기상태로서, 상태를 유지시키는 시간 σ은 0이다. σ가 0인 것은 역행 계획 직후에 상태변이가 실행됨을 의미한다. 즉, 외부 상태 변이에 대한 처리 시간은 0이 되므로, 이어지는 내부 상태 변이 이벤트는 즉시 발생된다.

δ_{ext}가 끝난 직후에는 Fu에 있는 첫번째 액션($f^2{}_1u_1$: pickup(C,T1)) 실행을 내용으로 하는 외부 목표 입력에 대한 첫번째 내부 상태 변이 일어난다. 다음 식은 이 과정을 전개한 것이다.

$$
\begin{aligned}
&\delta_{int}\ (s_1,\ f^2{}_1u_1)\\
&=\delta_{int}\ ((s^c,\ s^d{}_1),\ f^2{}_1u_1)\\
&=\delta_{int}\ (((\sigma,\ v_d),\ (s^{d1}{}_1,\ s^{d2}{}_1),\ f^2{}_1u_1)\\
&=\delta_{int}\ (((\sigma,\ (v^1{}_d,\ v^2{}_d)),\ (s^{d1}{}_1,\ s^{d2}{}_1)\ ),\ f^2{}_1u_1)\\
&=\delta_{int}\ (((0,\ (\mathrm{On(C,A)}\wedge\mathrm{ClrBlk(C)}\wedge\mathrm{ClrBlk(B)},\\
&\qquad\mathrm{AtTbl(A,T1)}\wedge\mathrm{AtTbl(B,T2)}\wedge\\
&\qquad\mathrm{AtTbl(C,T1)}\wedge\mathrm{HAt(T1)}\wedge\mathrm{HEmpty})),
\end{aligned}
$$

(On(C,A) ∧ClrBlk(C) ∧ ClrBlk(B),··· (1)
AtTbl(C,T1)∧HAt(T1)∧HEmpty)) ··· (2)
, **pickup(C,T1)**)
= ((3.0, (ClrBlk(C)∧ClrBlk(B), AtTbl(A,T1)∧
AtTbl(B,T2)∧HAt(T1)∧HHold(C))),
(ClrBlk(C)∧ClrBlk(B), HAt(T1)∧HHold(C))··· (3)
)
= ((σ, v_d), ($s^{d1}{}_2$, $s^{d2}{}_2$))
= (s^c, $s^d{}_2$)
= s_2

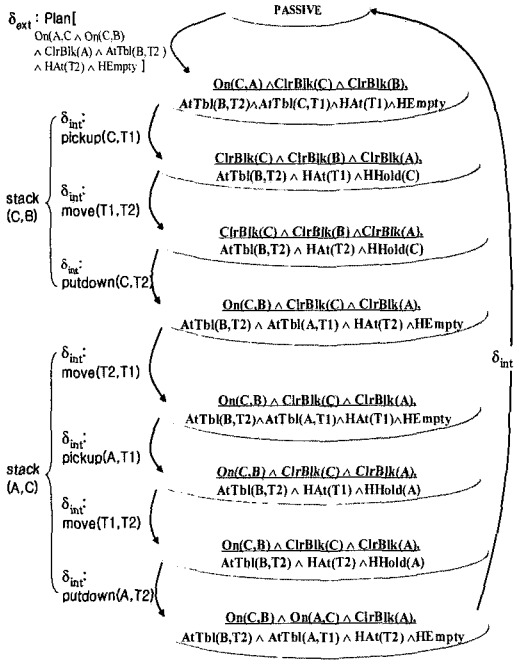

그림 6. HRG-DEVS 기반 모델의 계획 실행에 따른 상태 변화도: 블록 쌓기 예

즉, 첫번째 내부 상태 s_1 가 첫번째 액션으로 인해 두번째 s_2로 바뀌는 과정을 기술한 것이다.

상태 s_1은 s^c와 $s^d{}_1$으로 구성되는데, s^c는 순서쌍(σ,v_d)이다. 계획에 의한 규칙 인스턴스 의 처리 시간의 의미를 갖는 σ이 0으로 되어 있는데, 이는 s_0에서 s_1으로의 상태 변이가 δ_{ext}에 의해 실행되면서 계획을 생성하는 δ_{ext}의 처리 시간을 0으로 가정하였기 때문이다. 모델의 현재 상황에 대한 모든 predicate의 논리곱인 상태 변수 v_d는 각 계층에서의 prediate들을 모두 갖고 있다. 즉, $v_d = (v^1{}_d, v^2{}_d)$이 된다. 그러나, 현재 상태를 나타내는 것으로 규칙 인스턴스 $f^2{}_1u_1$와 관련한 상태 값을 갖는 $s^d{}_1$은 오퍼레이션 pickup(C,T1)을 실행시킬 조건의 상태가 되는 각 계층별 $s^{d1}{}_1$, $s^{d2}{}_1$로 구성된다. 전개식에서 (1), (2)는 각각 $s^{d1}{}_1$, $s^{d2}{}_1$에 해당하는 상태 정보들이다. 'pickup' 액션으로 내부 상태 변이 함수가 실행된 이후에는 다음 이벤트까지의 경과 시간 σ가 'pickup' 오퍼레이션의 처리 시간인 3.0으로 바뀌고, 모델의 상태 또한 (3)와 같이 바뀐다. (3)은 변화된 각 계층의 상태 $s^{d1}{}_2$, $s^{d2}{}_2$이 되고, 이는 $s^d{}_2$가 되며 s^c와 순서쌍을 이루어 s_2가 된다.

위의 상태 변이 과정은 이미 계획된 상태 리스트를 따라 목표 상태에 도달 할 때까지 계속되며, 그림 6은 그림 4에서와 같은 초기 상태에서 목표 상태가 외부 입력으로 들어왔을 때의 시뮬레이션 진행 과정을 상태 변이도로 표현한 것이다. 타원으로 표현된 상태는 각 계층에서의 predicate들을 카티션곱으로 표현한 것이며, predicate들 중 밑줄이 있는 것은 계획 level 1 관점에서 상태, 밑줄이 없는 것은 level 2 관점에서의 상태를 구분한 것이다. 각 상태를 지면상 전체 predicates 중 단계별 주요 predicate만을 이용하여 기술하였다. 첫번째 상태 변이는 δ_{ext}에 의해 계획(plan)이 생성된 직후 실행되고, 두번째부터는 계획된 상태 순서를 따라 목표에 도달 할 때까지 δ_{int}에 의해 실행된다.

4.4 계층적 계획에 따른 탐색 공간 축소

계층적 계획과 비계층적 계획 사이의 계획 비용을 탐색 공간을 기준으로 비교해보자. 탐색 공간은 세 가지 변수를 가정하여 간단히 계산 할 수 있다[21]. 세 변수는 계층적 계획에서 계층의 깊이(depth)인 d, 각 계층에서 상위 계층의 액션 하나를 구체화시킨 계획(plan)이 갖는 액션(action)의 수인 a, 각 계층에서의 탐색 가지수(branching factor)[4]인 b이다. 탐색 공간에서 최악의 경우의 탐색비용은,

- 계층적 계획에서 탐색 비용
  $= b^a \cdot 1 + b^a \cdot a^1 + \cdots + b^a \cdot a^{d-1}$
- 비계층적 계획의 탐색 비용
  $= b^n$; 이 때 $n = a^d$

이 된다.

4) 가지수(branching factor) : 탐색 트리에서 노드의 서브트리(sub-tree)의 개수를 의미하는데, 이는 이론적인 분석을 하기 위해 종종 모든 노드가 같은 가지수를 갖는 것으로 가정된다[14].

위의 식을 앞에서의 블록 쌓기 문제의 경우로 적용시키기 위해 세 변수 값을 살펴보자. 계층의 깊이인 d = 2이고, 각 계층에서 계획의 액션 수 a_1, a_2는 각각 2(stack, stack: 주어진 초기 상태에서 목표 상태로 가기 위한 액션 수), 3.5(level 1의 stack을 수행하기 위한 level 2에서의 평균 액션 수)이며, 각 계층의 가지 수(오퍼레이션 규칙 수)인 b_1, b_2는 각각 1(stack), 3(pickup, move, putdown)이다. 이때, 계층적 계획의 탐색 비용은 다음과 같다.

- 계층적 계획의 탐색 비용
  $= b_1^{a1} + b_2^{a2} \cdot a_1$
  $= 1^2 + 3^{3.5} \cdot 2$
- 비계층적 계획의 탐색 비용
  $= b_2^n$; 단, $n = a_1 \cdot a_2 = 2 \cdot 3.5$
  $= 3^7$

위의 두 계산식을 통해 단일 계층의 계획 보다는 여러 단계를 거치는 계층적 계획이 전체 탐색 공간을 분할함으로써 계획(planning) 비용을 줄여줄 수 있음을 알 수 있다. 또, 이러한 탐색 공간의 축소의 효과는 변수 a의 값이 커질수록 더욱 더 커지는데, 이는 HRG-DEVS 모델 실행 시 계획(planning)과정이 포함되어 있는 δ_{ext}의 실행시간을 줄어들게 한다.

5. 결 론

본 논문은 시뮬레이션 모델링 방법론에 관한 연구로서, 계층적 계획이 필요한 시뮬레이션 모델을 체계적으로 구성 할 수 있는 환경을 제안하였다. DEVS에 인공지능의 계획 기술을 반영하여 확장한 RG-DEVS 형식론을 기초로 하여, 계층적 계획 기술을 추가 반영하기 위하여 RG-DEVS를 이론적으로 더 확장 시킨 계층적 계획 이산 사건 시뮬레이션 모델링 방법론인 HRG-DEVS를 정의하고, 간단한 블록 쌓기 문제를 통해 모델링의 예시를 보였다.

시뮬레이션 모델에 계층적 계획 개념을 반영함으로써, 계획(planning)단계에서 계획 (plan)을 찾아가는 탐색공간을 작게 할 수 있고, 모델링 대상 시스템의 계층적 작업 흐름을 반영하기가 용이하다. 또한, 시뮬레이션 모델에서 계층적 계획의 특징을 시뮬레이션 결과를 분석하기 위해서도 유용할 것으로 기대되는데, 이러한 특징은 향후 연구 논문에서 이를 뒷받침 할만한 실제 시스템으로의 적용을 보일 것이다.

참 고 문 헌

1. Zeigler, B.P., Cho, T.H., and Rozenblit, J.W. "A Knowledge-Based Simulation Environment for Hierarchical Flexible Manufacturing." IEEE Trans. on Systems, Man and Cybernetics, vol. 26, no. 1, 1996.
2. Javor, A., Benko, M., Leitereg, A., More, G., "AI controlled simulation of complex systems." Computing & Control Engineering Journal, vol. 5, no. 2, 1994.
3. Russell, C.S., Elmaghraby, A.S., Graham and J.H., "An Investigation of a Standard Simulation-Knowledge interface." Proc. of the 24th conference on Winter simulation *table of contents*, Arlington, Virginia, 1992.
4. Yang, M., Katayama, A., Mnabe, K. and Aikawa, N., "An AI process control system with simulation database and adaptive filter for V-bending." Proc. Of Intelligent Process of Material and Manufacturing, 1999.
5. Lehmann, A., "Taxonomy and Applications of Expert systems in Simulation." Proc. of the International Symposium on AI, Expert System and Languages in Modeling and Simulation, North-Holland Publish company, 1987.
6. Monostori, L., Viharos, Z.J., "Hybrid, AI- and simulation-supported optimization of process chains and production plants." Annals of the CIRP, vol. 50, no. 1, 2001.
7. Zeigler, B.P., Praehofer, H. and Kim, T.G., Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, 2000.
8. Chow, A., "Parallel DEVS: a Parallel, Hierarchical, Modular Modeling Formalism and Its Distributed Simulator." Trans. of SCS, vol. 13, no. 2, 1996.
9. Zeigler, B.P. and Chi, S.D., "Symbolic Discrete Event System Specification." IEEE Tans. On Systems, Man, and Cybernetics, 1992.
10. Barros, F.J., "Dynamic Structure Discrete Event System Specification: Formalism, Abstract Simulators and Applications." Trans. of the Society of Computer Simulation, vol. 13, no. 1, 1996.
11. Cho, S.M. and Kim, T.G., "Real-Time DEVS Simulation: Concurrent, Time-Selective Execution of Combined RT-DEVS Model and Interactive Environment." Proc. of SCSC, Reno, NV, 1998.
12. Cho, T.H., "Embedding Intelligent Planning Capability

to DEVS Models by Goal Regression Method." Trans. of The Society for Modeling and Simulation International, vol. 78, no. 12, 2002.
13. Yang, Q., Intelligent Planning. Springer Verlag, Berlin, 1997
14. Rich, E., Artificial intelligence. McGraw-Hill, 1991
15. Nilsson, N.J., Artificial Intelligence: A New Synthesis. Morgan Kaufmann, San Francisco, California, 1998
16. Russell, S.J. and Norvig, P., Artificial Intelligence: A Modern Approach, Prentice Hall, 1995
17. Sacerdoti, E., A Structure for Plans and Behavior. American Elsevier, New York, 1977
18. Stefik, M., "Planning with Constraints." Artificial Intelligence, vol. 16, no. 2, 1981
19. Unruh, A. and Rosenbloom, P.S., "Abstraction in Problem Solving and Learning." Proc. of IJCAI-89, San Mateo, CA, 1989
20. Wilkins, D., "Domain-independent Planning: Representation and Plan Generation." vol. 22, no. 3, 1984
21. Wilensky R., "An Introduction to Artificial Intelligence.", http://www.cs.berkeley.edu/~wilensky/cs 188/ lectures/index.html

이 미 라 (yimira@mmu.ac.kr)

1998 성균관대학교 정보공학과 학사
2000 성균관대학교 전기전자 및 컴퓨터공학과 석사
2005 성균관대학교 전기전자 및 컴퓨터공학과 박사
2005～현재 목포해양대학교 해양전자통신공학부 전임강사

관심분야 : 모델링 방법론, 인공지능과 시뮬레이션, 시뮬레이션 개발 환경, ERP, 자율운항시스템