기호적 시뮬레이션을 이용한 심층추론 방법론

Deep Reasoning Methodology Using the Symbolic Simulation

지승도
Sung-Do Chi

Abstract

Deep reasoning procedures are model-based, inferring single or multiple causes and/or timing relations from the knowledge of behavior of component models and their causal structure. The overall goal of this paper is to develop an automated deep reasoning methodology that exploits deep knowledge of struture and behavior of a system. We have proceeded by building a software environment that uses such knowledge to reason from advanced symbolic simulation techniques introduced by Chi and Zeigler. Such reasoning system has been implemented and tested on several examples in the domain of diagnosis, performance evaluation, and event-based control.

1. 서 론

추론(Reasoning)은 주어진 지식으로부터 새로운 지식을 얻어내는 과정으로서 AI 분야의 기본이라 할 수 있다. 최근 시스템의 발전 경향을 보면, 단순성에서 복잡성으로, 그리고 구체적인 것에서 추상적인 것으로 이동하고 있다. 즉, 생체학, 전자공학등의 복잡한 시스템들이 연구되어 질수록 그 복잡성에 대처할 수 있고 이해하기 위해서는 보다 추상화된 개념들이 필요케되는데 추상화와 관련된 기호적 논리는 이제까지 연구되어온 추론방법중 가장 적절한 방법론 중의 하나로 일컬어지며, AI의 항구적 목적의 하나는 컴퓨터 시스템을 통한 기호적 추론 과정의 자동화라고 할 수 있다 [1].

한편, 이산사건 모델링은 복잡한 공장 자동화, 통신, 그리고 컴퓨터 시스템등의 분석 및 설계 분야에 있어서 점증적 응용 추세를 보이고 있다. 하바드대학의 Ho는 IEEE 등을 통해 이 분야의 중요성을 이미 인식시켜왔으며[2] 강력한 프로그램 언어의 개발 및 Workstation의 출현은 이산사건 시스템의 컴퓨터 시뮬레이션을 위한 모델링 기법에 많은 발전을 제공했으나, 아직도 이산사건 시스템에 대한 본질적 이해도는 연속 시스템(continuous system) 연구에 비해 유아 단계에 불과하다고 볼 수 있다. Zeigler에 의해 개발된 DEVS 형식론 (Discrete EVent System Specification formalism)은 이산사건 모델들의 계층구조적 모듈화 방법을 제공해 준다[3, 4, 5, 6, 7]. 이론적으로 완벽한 체계에 의한 완성된 DEVS 형식론은 시뮬레이션 모델링 분야와 AI기법의 강력한 매개체 역할을 해왔다. Chi와 Zeigler는 기존 DEVS의 시간대를 실수에서 선형다항 기호식으로 확장시킴에 의해 사건 시간의 기호적 표현을 가능케 함으로써 확장된 기호적 DEVS를 발표하였다[8, 9].

본 연구는 기호적 DEVS 시뮬레이션을 이용하여, 시스템상의 상징적 모델이 가지고 있는 구조와 기능의 분석을

통한 추론기법[10, 11]인 심층추론의 문제에 접근하여 추론과정의 자동화 방법론을 제시한다. 즉, 기호적 DEVS를 통해 획득한 궤적 tree로부터 추론에 필요한 지식을 자동 분석 및 생성시키는 자동화된 지식획득 방법론[12, 13]을 중점적으로 설명한다. 제안된 방법론은 고장진단분야를 비롯한 성능평가, 사건중심 제어문제등에 이르는 각종 응용분야에 대한 사례연구를 통해 각 분야에 대한 새로운 각도의 접근가능성을 타진하고 있다.

본 논문은 순서는 다음과 같다. 먼저, 기호적 DEVS 시뮬레이션의 개요와 야구경기를 통한 간단한 예가 소개된 후, 심층추론 방법론을 제안한다. 다음으로, 화학반응 냉각시스템을 예로한 고장진단문제, 컴퓨터 구조를 통한 성능분석문제, 그리고 고양이와 쥐의 미로게임을 통해 본 사건중심 제어문제등의 응용사례들이 차례로 설명된다.

2. 기호적 DEVS 시뮬레이션 개요

Chi와 Zeigler는 기존 DEVS의 시간대를 실수에서 선형다항 기호식으로 확장시킴에 의해 사건 시간의 기호적 표현을 가능케 하였다[8, 9]. 기호연산 기능을 첨가한 이산사건 모델 형식론은 지능제어, 고자치 시스템 설계등의 지능 시스템 설계등의 지능 시스템 연구에 유용하게 적용되어져 왔다[6, 14, 15]. 예를들면, "5 + 2 * 기차시간 + 지연시간" 이라는 표현은 합법적(실질적) 시간값이 될 수 있으며, 여행에 필요로하는 시간을 의미한다고 볼 수 있다. 여기서 "기차시간"과 "지연시간"은 실수를 대신하는 값이며 차후 실수로 대체될 수 있다. 이와같이 어떤 사건시간의 정확한 지식은 기호값을 실수값으로 나타냄에 의해 쉽게 대체될 수 있다.

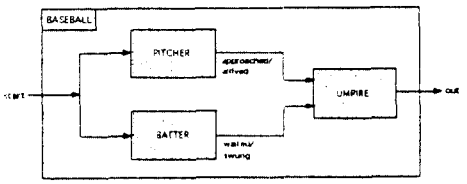
그러므로 기호적 DEVS는 기존 DEVS이론의 확장이라 할 수 있으며, 기호적 DEVS의 모든 시간값들이 실수값으로 대체될 때 기존 DEVS로 복귀하게 된다. 이것은 미분방정식 모델의 질적 표현방법(qualitative models)들이 보다 많은 정보획득(질적 정보)에 따른 댓가로 정확한 시간값들을 손실해야한다는 특징을 갖는 것과 좋은 대조를 이룬다.

기호적 DEVS 구성원들의 결합 모델(coupled model) 또는 network은 실수의 사건시간을 갖는 기존 DEVS의 경우와는 달리 발생가능한 사건 시간들 중 먼저 일어나야 할 사건시간이 결정지워 질 수 없기 때문에 시뮬레이션 궤적은 Tree 형태로 분리되어져야 한다. 예를들면, 발생가능 사건시간의 집합이 {t1, t2}인 경우 t1의 값이 작을수도 (t1 < t2), 또는 t2의 값이 작을 수도(t1 < t2)있다. 이와 같은 경우 기호적 시뮬레이션은 두개의 가지를 쳐 나감(branching)에 의해 각 경우를 모두 시뮬레이션해야만 한다. 또한, "t1 < t2"의 궤적 다음에 "t2 < t3"라는 조건의 경우를 시뮬레이션할때, 시뮬레이션의 효율적 관리를 위해 "t1 < t3"라는 앞의 두 조건의 함축적 조건을 추론해야 할 필요가 생기는데, 이와같이 선형다항식으로 표현되는 기호적 시간조건들의 효율적 관리와 추론을 위한 알고리즘은 이미 Chi와 Zeigler에 의해 개발되어 발표된 바 있다[8, 9, 16].

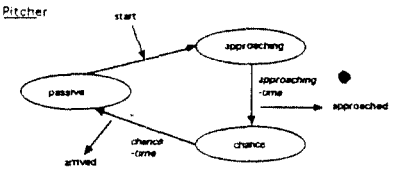
Chi와 Zeigler는 또한 개발된 기호적 DEVS를 이용하여 로보트 시스템의 고장진단에 필요로하는 오류의 인과응보적 전파(causal propagation)에 성공적으로 적용하였다[11, 17, 18]. 그들은 사건시간을 기호적으로 표현함에 의해, 주어진 오류에 기인한 시스템의 모든 비정상적 행동들의 궤적을 생성시킬 수 있었고, 그 궤적들의 적절한 분석에 의해 구조적으로 복잡하고 시간관계에 밀접한 영향을 갖는 시스템의 고장원인과 증상에 대한 지식을 획득할 수 있었다. 기호적 DEVS는 시간을 모르거나, 알지만 변하는 경우등의 모든 경우에 쉽게 적용시킬 수 있으므로, 시스템에 대한 시간적 영향을 표현한다는 점에서 기존의 모델베이스 고장진단을 능가한다고 볼 수 있다.

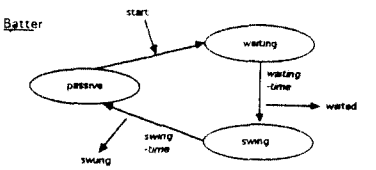
* 기호적 시뮬레이션의 예 : 야구경기

기호적 시뮬레이션의 한 예로서, 투수와 타자 사이의 시간적 관계를 고려해 보자. 투수로부터 떠난 공은 홈 프래트를 통과 하면서 타자로 하여금 공을 맞출 기회를 허용하게 된다. 만약 공을 맞추지 못하면 공은 포수에게 갈 것이다. 한편, 타자는 공이 좋은 위치에 올때까지 기다리다가 공을 치기위해 방망이를 휘두를 것이다. 〈그림 1a)〉는 야구경기의 결합 모델을 보이고, 〈그림 1b)〉는 결합모델의 구성원인 투수와 타자의 상태변환 그래프를 각각 나타낸다. 투수모델이 start 명령을 받으면, 상태는 passive 에서 approaching 상태로 바뀐다. approaching—time(투수가 던진 시간부터 홈 프래트 직전까지의 시간)이 경과하면 approached 라는 출력을 심판모델(umpire)에게 전달함과

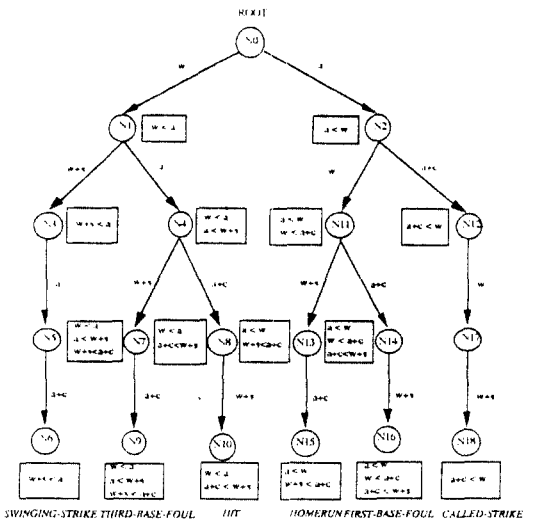


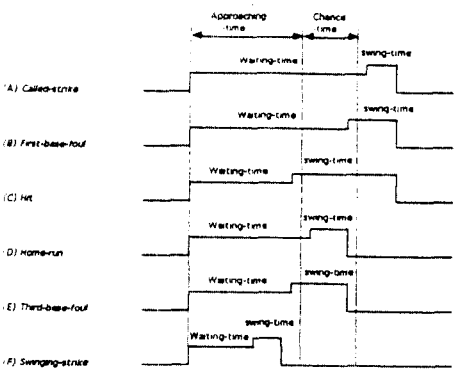
(a) Model Structure of Baseball





(b) State Transition Graphs





(d) Timing Relations

Where a: approaching-time, w: waiting-time, c: chance-time, s: swing-time

Note: Branch is labelled by the time chosen to be set minimum and constraints of each node are shown in an associated box.

(c) Generated Tree of Trajectories

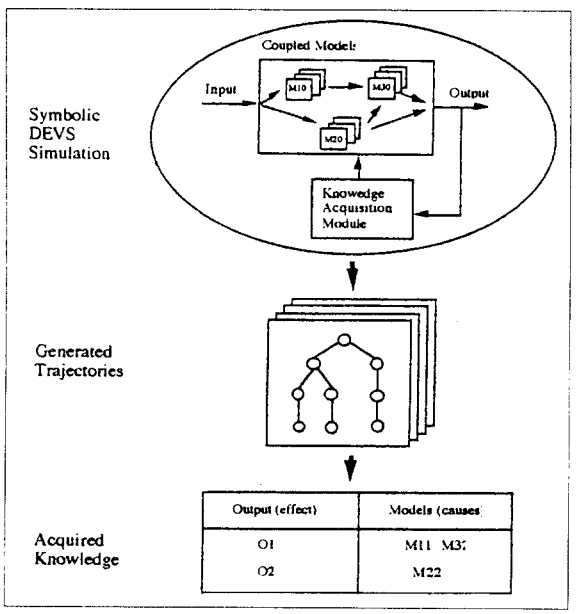
〈그림 1〉 야구경기의 기호적 시뮬레이션

동시에 chance 라는 상태로 바뀌어 chance—time(홈 프래트 통과 시간) 이라는 시간동안 유지하게 된다. chance—time 이 경과하면 심판모델에 arrived 라는 메세지를 전달함과 동시에 최초의 상태인 passive로 복귀한다. 타자모델 역시 투수모델과 근본적으로 유사한 상태변환을 갖으나, 다른 이름을 사용하고 있다. 기호적 시뮬레이션은 〈그림 1c)〉와 같은 궤적 tree를 생성시키는데 그 시간적 의미는 〈그림 1d)〉에 나타낸 바와 같다. 각 노드에 부착된 네모 상자는 그 노드의 constraints를 나타낸다. 예를들면 노드 N1은 constraint w 〈 a에 의해 생성되는데, 이것은 두 기호 시간인 w(waiting-time)와 a(approaching-time)중에 w가 최소 시간으로 간주된 것을 의미한다. 노드 N8의 경우, constraints는 w 〈 a, a 〈 w+s (노드 N4로부터의 조건들), 그리고 a+c 〈 w+s (새로 생성된 조건)등 세개로 구성될 것이나, 추론 알고리즘에의해 중복된 constraint인 a 〈 w+s는 자동적으로 제거됨으로써 결과되는 constraints는 w 〈 a

과 a+c < w+s의 두개가 된다. <그림 1c)>에 나타난 모든 끝 노드(leaf node)들은 투수와 타자간에 나타날 수 있는 모든 가능한 시간관계를 나타낸다.

3. 심층추론 방법론

기호적 DEVS 시뮬레이션을 이용한 심층추론 방법론은 DEVS 시뮬레이션 S/W환경위에 확장구현되었으며, 뒷절에 설명될 사례연구를 통하여 그 적용성을 입증하였다. 구현된 방법론은 다음과 같이 진행된다.



<그림 2> 기호적 시뮬레이션을 이용한 심층추론 방법론 개념도

(1) 대상 시스템 각 구성원의 구조적 지식과 상호간의 coupling관계를 나타낸다. DEVS 시뮬레이션 S/W환경에서 제공하는 System Entity Structure 개념[4]을 통하여 시스템의 구조적 지식 및 coupling관계를 표현한다.

(2) 각 구성원의 행동적 지식을 기호적 이산 사건 모델링을 통해 표현한다. 기호적 DEVS 형식론을 통하여 시스템 구조상에 나타난 atomic 레벨의 모델들을 각각 표현한다.

(3) 구조적 지식과 행동적 지식의 결합을 통해 전체적인 기호적 시뮬레이션 모델을 완성한다.

(4) 추론에 필요로하는 goal을 선정한 뒤 시뮬레이션을 행한다. 기호적 시뮬레이션의 초기조건으로 선정된 goal은 시뮬레이터에 의해 검색되며 그에따라 최종적으로 goal 궤적을 얻을 수 있다.

본 방법론에 있어서 구조적 지식과 행동적 지식은 System Entity Structure[3, 4]와 기호적 DEVS 형식론을 통하여 표현할 수 있다. 대상 시스템의 구조적 그리고 행동적 모델이 완성되고 추론에 필요한 최종 goal이 선정되면, 기호적 시뮬레이션을 위한 준비가 갖춰진다.

<그림 2>에 나타난 바와 같이, 기호적 시뮬레이션은 시뮬레이션 모델에 주어진 입력명령을 가함에 의해 시작된다. 시뮬레이션이 진행되면서, 입력명령은 구성원들간의 결합관계와 기호적으로 표현되는 시간들의 타이밍관계에 따라 주어진 goal조건을 만족시킬 때까지 트리궤적을 자동생성하면서 연결된 구성원으로 전파되어 나간다. 이와 같은 방법으로 기호적 시뮬레이션은 발생가능한 모든 사건들의 시간 궤적을 자동생성 시킴에 의해 주어진 goal 상태에 이르는 모든 사건과 시간관계를 추론해 낸다. 시뮬레이션의 결과로 나타나는 시뮬레이션 궤적은 goal상태에 도달하는 인과응보적 사건들의 연결관계뿐만 아니라 그에 따른 시간적 정보도 제공해준다.

제안된 방법론은 이산 사건 시스템 형식론을 토대로 계층구조적이며 모듈화 모델링 환경을 이용한 시스템이론적이면서 시간중심적인 기법이라는 점에서 Reiter등의 타방법론들을 능가한다고 볼 수 있다 [12, 13, 19, 20, 21, 22, 23, 24, 25]. 연구된 방법론의 주요한 특징은 다음과 같다.

(1) 이산 사건 시스템 형식론에 따른 모델링 환경을 제공하므로 표현력이 강하다.

(2) 계층구조적 모듈화 방식으로 설계됨으로 구현이 용이하다.

(3) 구성원 모델간 혹은 가능한 사건들간의 결합관계의 표현이 용이하다.

(4) 시스템 구성원의 모델링뿐아니라 인간, H/W, 환경적 요인등 시스템의 추론에 영향을 미치는 각종 사건들의 모델링이 용이하다.

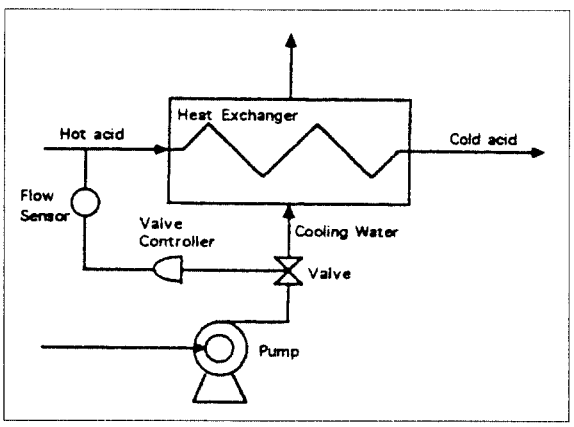
(5) 근본적으로 타이밍에 따른 영향을 나타내므로 인과응보적 추론과정을 표현한다.

(6) 선형 프로그램밍기법을 이용한 선형다항식 알고리즘을 통해 효율적 궤적 생성을 보장한다.

4. 응용사례 연구

4.1. 고장진단: 화학반응기의 냉각시스템

신뢰도 연구의 전형적인 예중의 하나가 화학반응기에 있어서 펌프, 열교환기, 밸브 제어시스템, 기타 장비 또는 주변환경등의 오동작으로 인해 초래될 수 있는 반응기의 과열상태에 대한 인과응보적 과정의 분석이다. 〈그림 3〉은 화학반응기의 냉각시스템을 도식하는데, 여기서 펌프는 냉각수를 밸브를 통해 열교환기로 공급해주며, 화학반응을 통해 뜨거워진 액체는 열교환기를 통해 식혀진다.

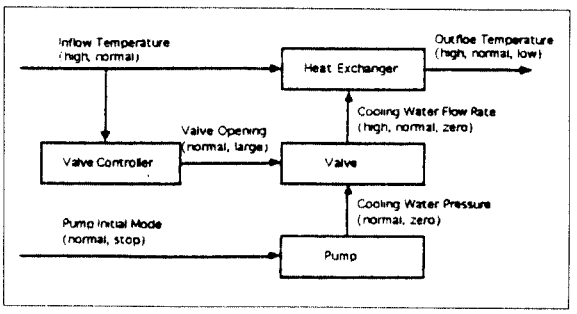


〈그림 3〉 화학반응기의 냉각시스템 개념도

[23]

입/출력관계에 의해 표현될 수 있는 시스템 구성원간의 결합관계는 〈그림 4〉와 같다. 펌프의 내부적 모드는 시스템에 대한 입력으로 간주되는데, 입력은 "정상작동"과 "무작동"중 하나의 사건으로 표현된다. 냉각수압력은 펌프의 출력으로 나와서 밸브의 입력으로 들어가는데, "정상압력"과 "공압력"의 두종류가 있다. 밸브열림은 밸브에 대한 또 하나의 입력으로 작용한다. 밸브의 출력은 냉각수 유출량으로 나타나는데, 유출량은 "과유출", "정상유출", "공유출"등 세종류로 나눌 수 있다.

밸브 제어기는 열교환기로 입력되는 액체의 온도가 높을때 밸브열림을 증가시키는 방향으로 작동케 된다. 따라서, 밸브열림은 "정상열림"과 "큰열림"의 두 상태가 가능하며, 밸브 제어기에는 "정상작동"과 "비정상작동"의 두 사건중의 하나가 발생할 수 있다.



〈그림 4〉 냉각시스템의 입/출력관계 구성도

열교환기는 냉각수 유출량과 입력액체의 온도등 두개의 입력을 갖는다. 온도는 "정상온도"와 "과열온도"로 나뉘는데, 여기서 온도는 시스템의 입력으로 간주한다.

열교환기의 출력은 출력액체의 온도로 나타내는데, "과열온도", "정상온도", "냉각온도"등으로 구분된다. 여기서, "과열온도"로 출력된 액체는 다시 반응기로 순환되어 사용되므로 큰 재난을 초래시키게 된다. 그러므로, 본 응용사례를 통해 분석하고자 하는 추론의 goal은 "열교환기에서 방출되는 액체의 온도가 과열로 되는 상태"로 정의할 수 있다.

각 구성원들의 의사결정표(입/출력관계)는 표 1에 나타나 있다. 예를들어, 펌프의 입/출력관계는 표에 표시되어 있는 바와 같이 펌프의 "무동작"은 "공압력"을 유발시키고, "정상동작"은 "정상압력"을 보장한다. 〈그림 5〉는 기호적 시뮬레이션 방법론을 통해 구현된 모델구조를 나타낸다.

기호적 시뮬레이션은 시뮬레이션 모델에 "정상온도" 또는 "과열온도"등의 입력을 가함에 의해 시작된다. 가해진 입력은 열교환기(EXCHR), 밸브제어기 (CONTRL), 제어기의 오동작 결함 (BG-B), 펌프(PUMP), 펌프 무동작 (BG-A)등의 모델들을 동시에 활성화시킨다. 각 모델은 지정된 기호적 시간만큼 지정된 상태로 변환되는데, 이경우 시뮬레이션은 각 기호적 시간들간에 비결정론적 조건으로 인해 트리를 쳐나가면서 깊이우선방식에의해 시뮬레이션을 행한다. 이와같은 방법으로 모든 정상적 혹은 비정상적 사건들이 테스트되어진다.

〈그림 6〉은 기호적 시뮬레이션을 통해 생성된 궤적을 나타내는데, leaf에 표시된 "*"은 goal 조건 (열교환기에서 방출되는 액체의 온도가 과열로 되는 상태)을 만족시키는 궤적을 의미한다. 이 궤적들은 "정상온도"의 입력으로부

〈표 1〉 냉각시스템 구성원들의 의사결정표

Input	Output
Initial Mode	Cooling Water Pressure
Normal Stop	Normal Zero

(a) Pump

Input		Output
Cooling Water Pressure	Valve Opening	Cooling Water Flow Rate
Normal	Normal	Normal
Normal	Large	High
Zero	-	Zero

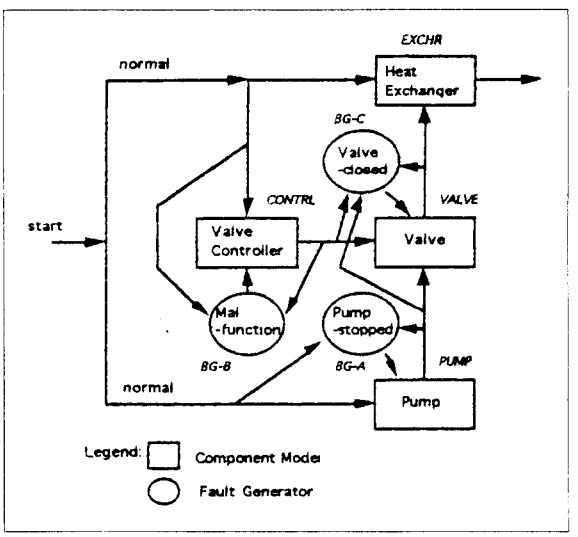
(b) Valve

Input	Output
Inflow Temperature	Valve Opening
Normal	Normal
High	Large

(c) Valve Controller

Input		Output
Cooling Water Flow Rate	Inflow Temperature	Outflow Temperature
Normal	High	High
Normal	Normal	Normal
High	High	Normal
High	Normal	Low
Zero	-	High

(d) Heat Exchanger



〈그림 5〉 냉각시스템의 기호적 DEVS 모델구조

터 goal 조건에 이르기 까지 발생가능한 사건들과 냉각기 시스템 구성원들 사이의 모든 인과응보적 발생순서를 보여준다. 〈그림 6〉으로부터 자동획득된 고장트리가 표2에 나타나 있는데, 열교환기의 출력을 과열온도로 이끄는 원인은 "밸브잠김", "펌프 무동작", "제어기 결함에 이은 밸브잠김", "제어기 결함에 이은 펌프 무동작", 그리고 "제어기 결함과 밸브잠김과 펌프 무동작이 함께 일어날 때"임을 알 수 있다.

4.2. 성능평가 : 컴퓨터 구조의 성능평가

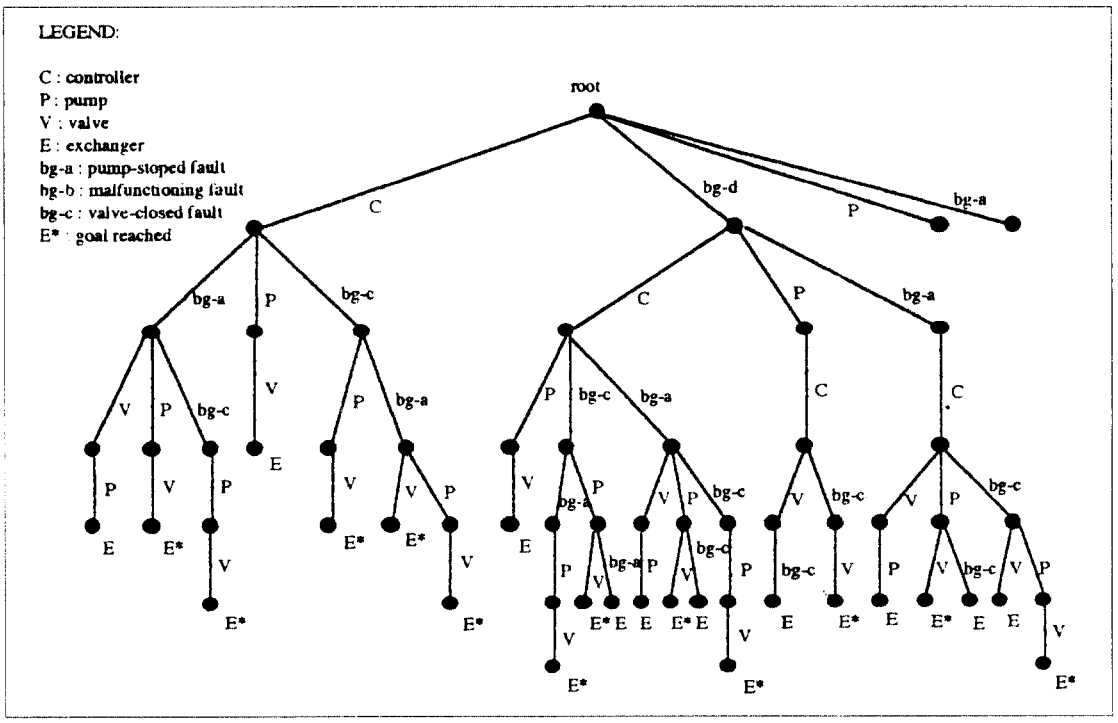
컴퓨터 구조들에 대한 성능평가 문제는 많은 관심의 대상이 되어왔다. 대부분의 경우 제안된 각종 컴퓨터 구조의 구현을 통한 검증은 불가능하지만 모델링에의한 시뮬레이션은 가능하여 왔다. 본 사례연구는 컴퓨터의 전형적인 구조들에 대한 성능평가의 자동화 방안에 대하여 진행되었다.

단일 (Single) 프로세서 시스템과 Multiserver, Divide& Conquer, 그리고 Pipeline 시스템등의 다중프로세서 시스템과의 성능 비교에 있어서 주요 성능평가 요소로는 1) Turnaround-time:시스템이 하나의 job을 처리하는데 필요로하는 평균시간, 2) Thruput:시스템으로부터 시간당 처리될 수 있는 job의 양등을 들 수 있다. 즉, 모든 프로세서가 가급적 busy상태를 유지하면서도 job을 잃어버리지 않을 때, 최적의 성능을 나타낸다고 할 수 있다.[4]

〈그림 7〉은 네 종류의 전형적인 컴퓨터 구조를 나타내고 있는데, 하나의 프로세서만을 사용하는 단일 프로세서 구조와는 달리 다중 프로세서 구조들은 부여된 job을 종속 프로세서에 보내고 처리결과를 받는 역할을 담당하는 co-ordinator를 갖고 있다. Multi-server구조에서는 입력된 job을 passive상태에 있는 프로세서들 중 하나에 할당해 준다. Pipeline 구조에 입력된 job은 미리 정해진 순서에 따라 각 프로세서를 거치게 되는데, 각 프로세서는 job의 일부분을 담당하게 된다. Divide&Conquer 구조에서는 입력된 job이 divider를 통해 작은 부분으로 나누어져 각 프로세서에서 동시에 병렬처리된 후 conqueror에 의해 함께 모여져서 출력하게 된다.

EF(Experimental Frame)는 각 컴퓨터 구조의 성능평가를 위한 독립된 시험모듈을 말하는데, G(Generator)는 각 구조에 대한 입력(job)을 일정시간 간격으로 생성시키며, T(Transducer)는 컴퓨터 구조로 부터 출력된 job을 받아들임으로써, 성능분석을 담당한다. 이와같이 기호적 DEVS에서 제공하는 모듈화 개념은 실험적 방법을 통한 손쉬운 시스템분석 수단을 아울러 제공해 준다.

각 컴퓨터 구조의 성능 평가는 각 프로세서들의 처리시



〈그림 6〉 생성된 시뮬레이션 트리궤적

〈표 2〉 최종결과로 생성된 고장진단표

Top Event	Inflow	Fault Tree
High-temperature in Heat Exchanger	Normal	valve-closed pump-stop controller-malfunction, valve-closed controller-malfunction, pump-stop valve-closed, pump-stop controller-malfunction, valve-closed, pump-stop

간 및 job의 평균발생시간을 기호적으로 표현함에 의해 시뮬레이션되며, 기호적 시뮬레이션의 궤적결과 분석에 의해 최적의 성능을 나타낼 수 있는 시간관계를 추론해 낼 수 있다. Goal 조건을 만족시키는 트리궤적의 모델 및 해당 상태의 인과응보적 관계에 촛점을 둔 고장진단 문제와는 달리, 성능평가에 있어서는 Goal 조건을 만족시키는 궤적의 time constraints의 분석에 의한 추론을 필요로 한다.

실험을 단순화 시키기 위해, 각 프로세서의 일처리 시간은 'p-time으로, G의 평균발생시간은 'interarrival-time으로 고정하였고, Goal 조건으로는 최적의 성능보장을 위하여, 첫째 프로세서가 최대한 busy상태를 유지하면서, 둘째 job을 잃어 버리지 않는 조건으로 초기화 하였다. 다중 프로세서 구조들의 경우는 세개의 프로세서를 각각 갖는 것으로 모델링하였으며, Divide&Conquer 구조에서는 divid 모델과 conquer 모델, 모두 'd&c-time의 일처리 시간을 갖는 것으로 초기화하였다.

앞절에 제시한 심층추론방법론을 통해, 다음과 같은 순서로 각 구조들에 대한 기호적 시뮬레이션을 진행하였다.

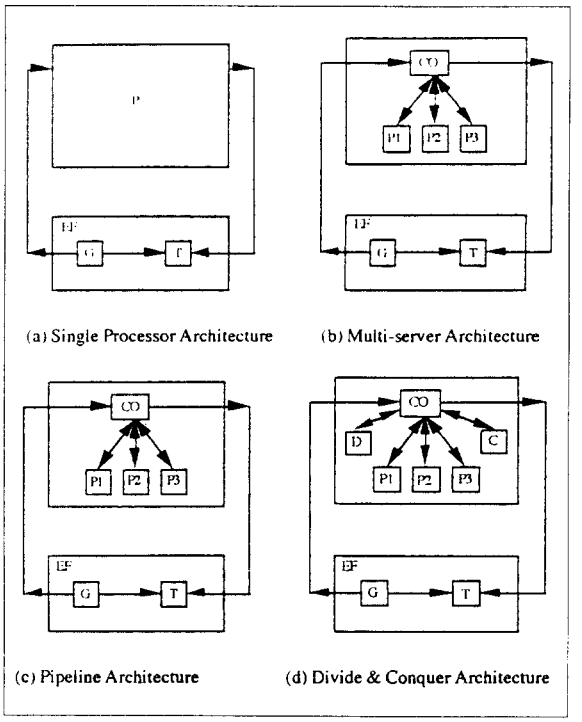
(1) 최적의 성능보장을 위한 goal조건을 설정

(2) 기호적 시뮬레이션을 통한 모든 트리궤적의 생성

(3) Goal 조건을 만족시키는 트리궤적을 검색

(4) 생성된 timing constraints를 통한 최적 성능 관계분석

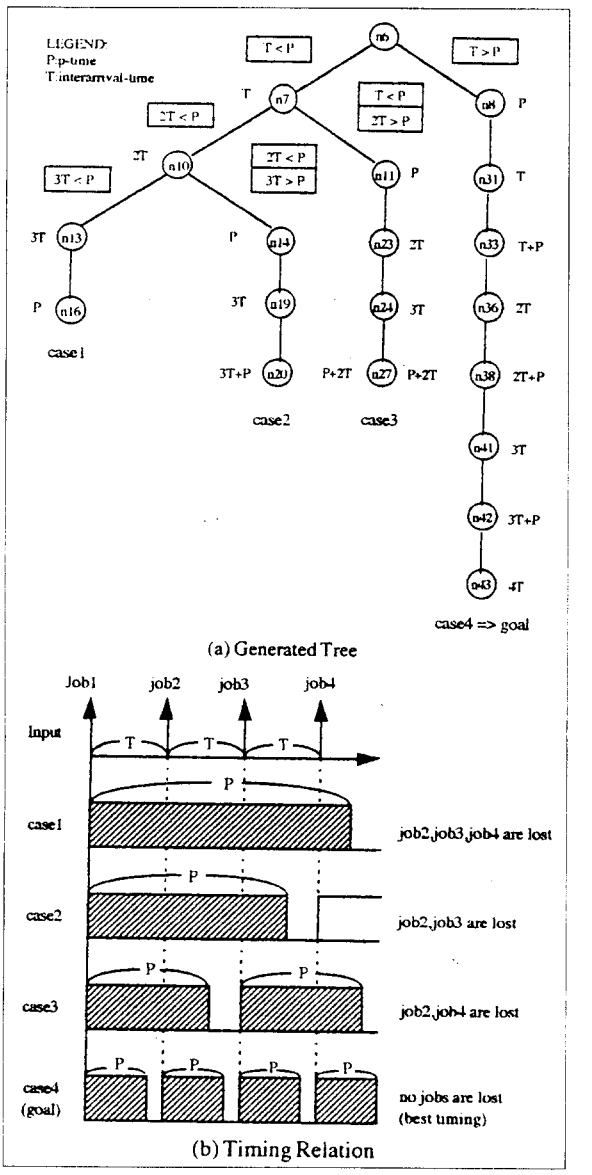
〈그림 8〉은 단일 프로세서 구조인 경우 3개의 job을 입력한 경우 생성된 트리궤적과 각 궤적별 시간관계를 도식하고 있다. 〈그림 8(a)〉에서 goal 조건을 만족시키는 궤적(case4)을 보면 "p-time 〈= interarrival-time"의 constraint를 갖게 됨을 알 수 있다. 즉, job 평균발생시간 (interarri-



〈그림 7〉 단일 및 다중 컴퓨터 구조의 결합모델 구성도

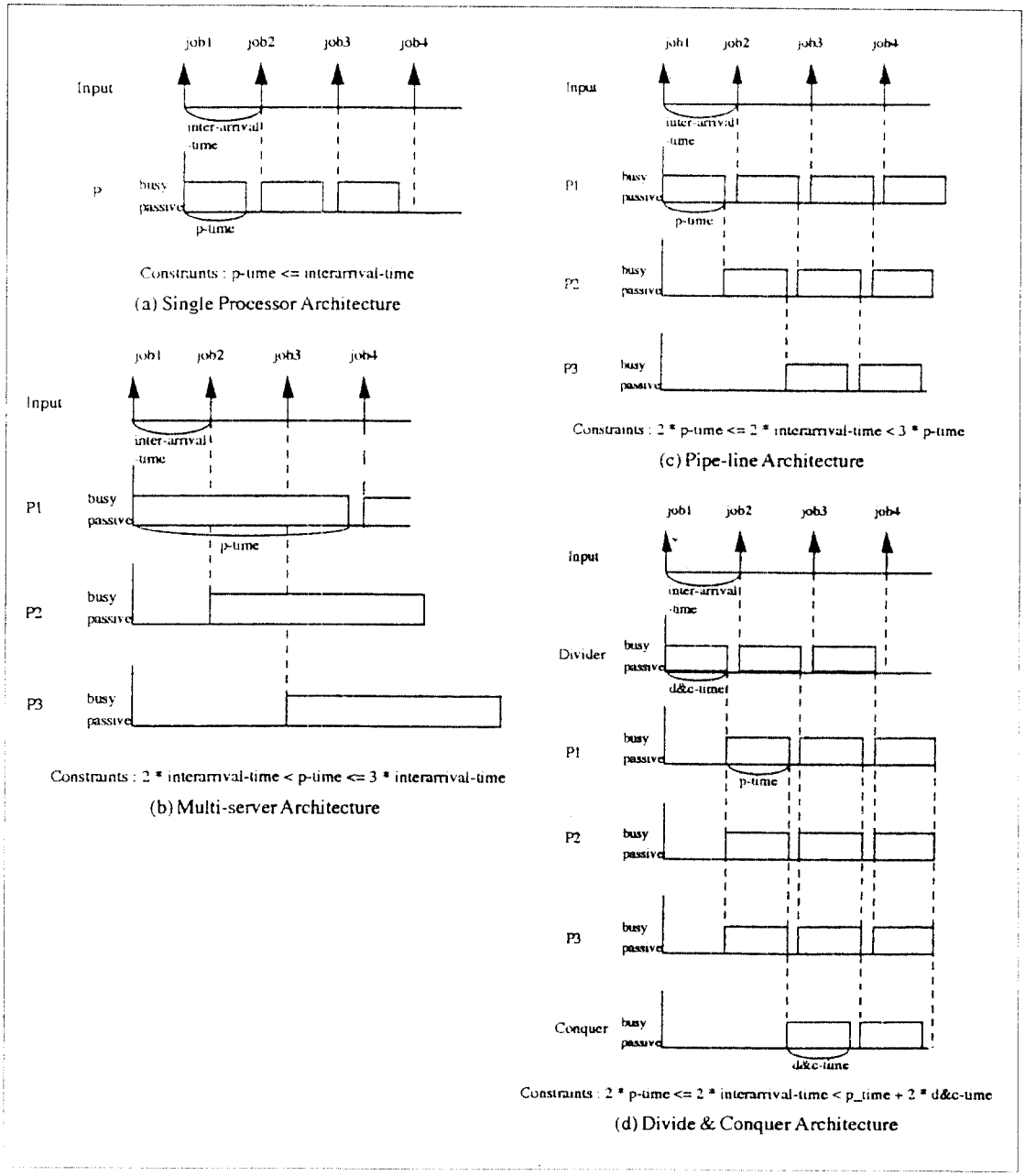
val-time)이 프로세서의 일처리시간 (p-time)보다 클 경우, 프로세서는 다음 job이 도착하기 전에 일을 마치게 되므로 job을 잃어버리지 않게 된다. 물론, p-time이 interarrival-time과 동일한 경우는 job을 잃지 않음은 물론 프로세서의 활용을 극대화 (busy상태를 지속)할 수 있어서 최적의 성능을 나타냄을 알 수 있다. 반면, interarrival-time이 p-time보다 작은 경우(case1, 2, 3), 프로세서가 busy상태인 동안에 다음 job이 도착하므로 job을 잃어버리는 경우가 발생된다. (본 예에서는 queue가 없는 프로세서를 가정하였다.)

이와같이, 기호적 시뮬레이션의 결과로 얻은 최적의 시간관계는 각각 〈그림 9〉에 나타낸 바와 같이 각 구조들에 대한 질적인 성능분석을 가능케 함을 알 수 있다. 예를들면, 〈그림 9(b)〉의 Multi-server 구조의 경우, 최적의 시간관계는 "2 * interarrival-time 〈 p-time 〈= 3 * interarrival-time"으로 나타나는데, 만약 "interarrival-time 〈 p-time 〈= 2 * interarrival-time"의 시간관계를 가정한다면, job을 잃어버리지는 않지만 프로세서 P3는 항상 passive상태를 유지하게 될 것이다. 즉, 그와같은 시간관계에 있어서는 오



〈그림 8〉 단일 프로세서 구조의 경우 생성된 트리궤적 및 시간관계

직 P1과 P2 두 개의 프로세서만으로도 충분하다는 결론이 도출될 수 있다. 〈그림 9(c)〉의 divide&conquer 구조에서는 만약 d&c-time이 p-time과 같다면 〈그림 9(d)〉의 pipeline구조와 동일한 최적의 시간관계를 갖음을 알 수 있다. 물론 이러한 경우 우리는 총 5개(divider와 conquer 포함)의 프로세서를 사용하는 divide&conquer구조보다는 총 3개의 프로세서를 사용하는 pipeline구조가 더 경제적일 것



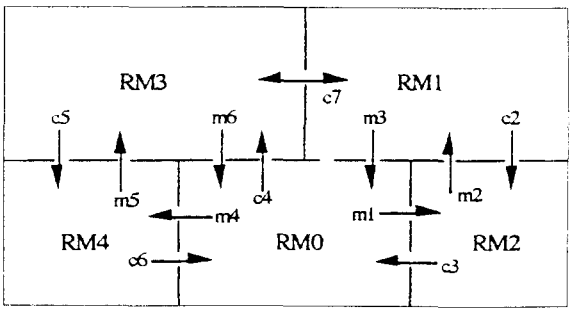
〈그림 9〉 최적의 성능을 보장하는 시간관계

으로 판단할 수 있을 것이다. 이와같이, 기호적 시뮬레이션은 컴퓨터 구조의 성능평가 문제에 있어서도 기존의 수치적 시뮬레이션에 의한 방법론[3, 4]에서는 불가능한 자동화된 기호적 추론기법을 제공한다.

4.3. 제어문제 : 쥐와 고양이의 미로에서의 제어법칙

이 문제는 이산사건 시스템에 대한 전형적인 제어문제

로서 Ramadge 와 Wonham에 의해 제시된 바 있다[26]. 〈그림 10〉에서와 같이 쥐와 고양이가 있는 미로가 있는



〈그림 10〉 쥐와 고양이의 미로게임

데, 초기조건으로 쥐는 방4에, 고양이는 방2에 각각 있다. 각각의 방문인 Mi (i = 1, 6)와 Cj (j = 1, 7)는 고양이와 쥐를 위한 전용문을 나타내며 통과 방향은 그림에 지적된 바와 같다. C7을 제외한 각각의 문은 제어법칙에 따른 명령에 의해 열리거나 닫힐 수 있다. 여기서 주어진 제어문제는 아래의 조건을 충족 시킬 수 있는 제어법칙을 찾는 것이다.

(조건1) 쥐와 고양이는 동시에 동일한 방에 존재할 수 없다.

(조건2) 초기조건(방4와 방2)으로의 복귀는 언제든지 가능하다.

물론, 제어가능한 한도내에서, 각 동물들에게는 최대한의 자유통행이 보장되어야 할 것이다.

Ramadge와 Wonham은 그들이 제안한 제어용 언어를 이용하여 제어 알고리즘을 발표하였는데, 결과된 최적 제어법칙은 다음과 같이 요약된다[26].

(1) 두 동물이 초기조건에 있을 때는 C3와 M5를 열어두되,

(2) 만약 고양이가 방2를 떠나려면, 쥐를 방4에 머물도록 한다. 즉, M5와 C5를 닫는다.

(3) 마찬가지로, 쥐가 방4를 떠나려면, 고양이를 방2에 머물게 한다. 즉, M1과 C3를 닫는다.

이 접근방법은 최소한의 제어법칙을 보장해 주기는 하나, 두 동물에게 최대한의 자유를 보장해 주지는 못하고 있다. 즉, 한 동물이 이방저방을 자유로이 배회하는 동안 다른 한 동물은 한 방에 갇혀 지내야 하기 때문이다. 본 논문에서는 그와같은 고립조건을 피하기 위해 Ramadge와 Wonham이 제시한 두 조건에 다음의 조건을 추가하였다.

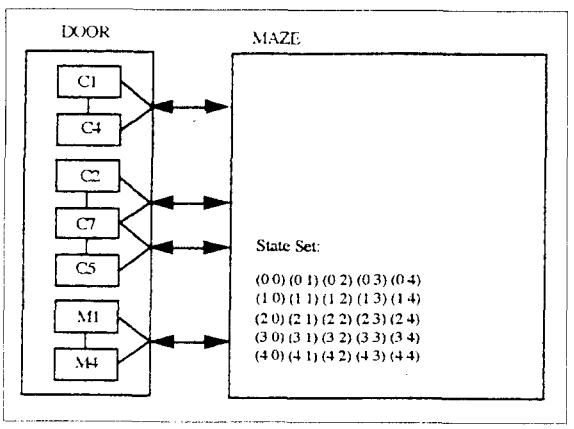
(조건3) 두 동물은 계속적으로 방을 옮겨 다녀야 한다.

이상의 세가지 조건들은 다음과 같은 제어 조건으로 일반화될 수 있다.

- Safeness : 두 동물이 동시에 동일한 방에 존재해서는 않된다.

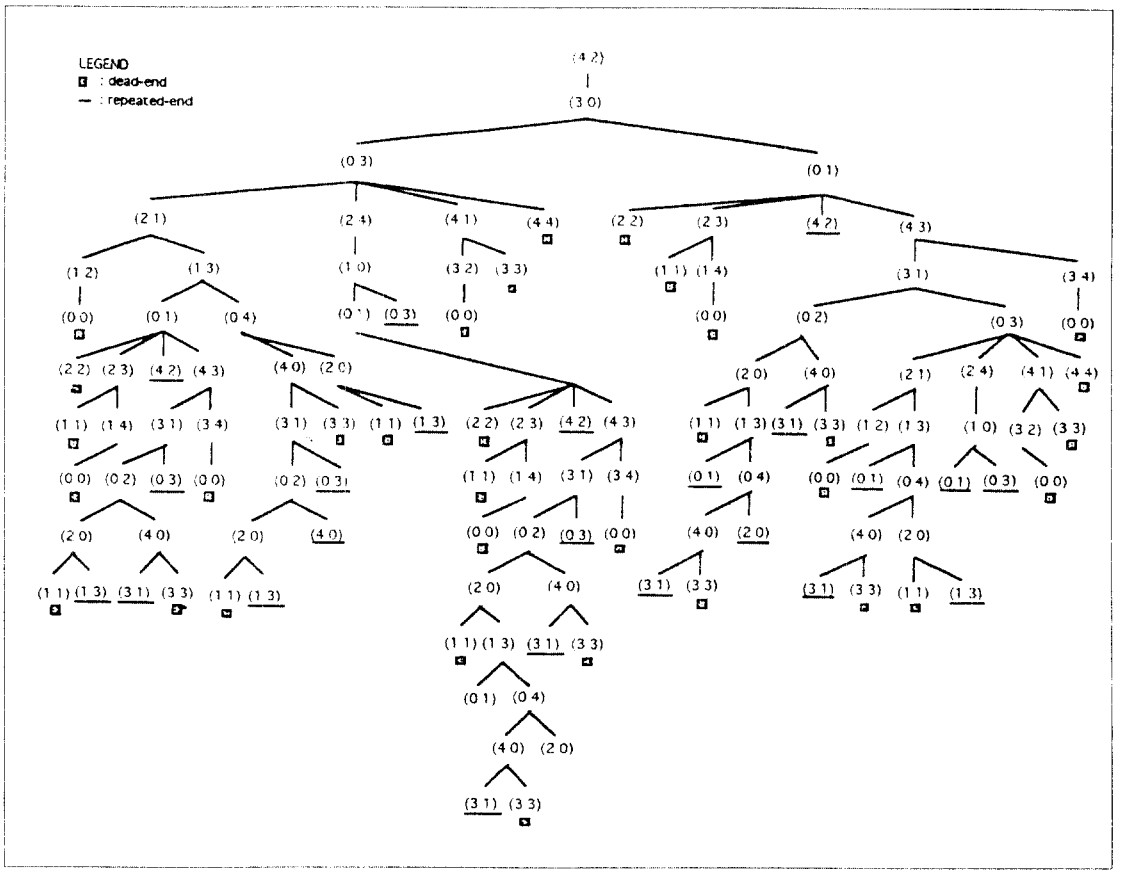
- Liveness : 두 동물은 계속적으로 방을 옮겨 다녀야 한다.

이제 우리가 찾고자 하는 제어법칙은 위의 두 조건을 만족시키는 최소한의 법칙을 의미하게 되는데, 이를 위해서 먼저 (쥐가 있는 방, 고양이가 있는 방)을 초기조건인 (4 2)의 상태로부터 시작하여 발생가능한 모든 상태궤적들을 확인한 뒤, Safeness를 위배하는 상태, 즉 (Mi Ci)상태, 그리고 제어 불가능 상태, 즉 고양이가 C7을 통과 하거나, 또는 출구가 하나인 방에 있을 경우(이 경우는 Liveness조건 때문에 제어가 곤란하게 된다.)등을, 벗어날 수 있는 최소 조건을 찾는 것이다..



〈그림 11〉 고양이와 쥐의 미로 모델

〈그림 11〉은 고양이와 쥐가 만들어 낼 수 있는 모든 가능한 궤적을 생성시키기 위한 미로모델의 구조를 나타낸다. 각 동물이 하나의 방에 체류하는 시간은 'staying-time으로 초기화하였고, 두 동물이 이동할 방들의 모든 경우수를 발생시키기 위해 각 방문들을 모델화하여 독립적인 기호적 시간을 갖게 하였다. 예를들면, 방1에 있는 고양이는 C7과 C2를 통해 각각 방3 또는 방2로 갈 수 있는데, 이 경우 모델 C7과 C2의 처리시간인 'c7-passing-time과 'c2-passing-time간의 비결정적 관계 때문에 트리가 발생되



〈그림 12〉 생성된 모든 상태 트리궤적

어 각각의 경우가 테스트되어 진다. 제어법칙의 발견을 위한 기호적 시뮬레이션의 Goal 조건으로 (Mi Ci)상태를 선정하여 시뮬레이션한 결과, 〈그림 12〉에 표현된 트리궤적을 얻었고, 이들 궤적들의 분석을 통해 최종적으로 다음과 같은 제어법칙을 얻을 수 있었다

(1) (0 1)의 상태에 있으면, M1을 닫아라.
(2) (0 3)의 상태에 있으면, M4를 닫아라.
(3) (2 0)의 상태에 있으면, C1을 닫아라.
(4) (2 1)의 상태에 있으면, C2를 닫아라.
(5) (4 0)의 상태에 있으면, C4를 닫아라.
(6) (4 3)의 상태에 있으면, C5를 닫아라.

예를들면, 〈그림 12〉의 좌측상단의 (0 0)로 끝나는 궤적을 보면, (4 2), (3 0), (0 3), (2 1), (1 2), (0 0)의 상태변환 궤적임을 알 수 있는데, (0 0)의 상태를 피하려면 (2 1)의 상태에서 (1 2)의 상태로 진입해서는 안된다는 것을 알 수 있다. 즉, 제어법칙(4)에 명시된 바처럼 (2 1)의 상태에서 C2를 닫아야만 (1 2)의 상태를 벗어날 수 있는 것이다. 이와같이, 기호적 시뮬레이션은 모든 상태공간을 자동적으로 테스트할 수 있어서 (즉, 초기상태로부터 도달될 수 있는 uncontrollable상태와 controllable상태 모두를 테스트한다.) uncontrollable상태로의 접근을 차단하고자 하는 사건중심 제어문제에 있어서 손쉬운 해결방법을 제공해 준다.

5. 결 론

본 논문에서는 복잡성과 추상화의 경향을 갖는 시스템의 구조적 행동 분석에 따르는 난제를 자동적으로 해결해 줄 수 있는 심층추론 방법론이 제시되었다. 제안된 방법론은 시간개념을 사용한 시뮬레이션 기법을 이용함으로써

시간 개념이 고려되지 않은 Reiter등에 의한 타방법론을 능가하는 새로운 가능성을 제공한다.

제안된 심층추론 방법론은 여러 응용분야에 있어서 성공적으로 적용되었는데, 고장진단문제에 있어서는 고장의 원인과 결과의 인과응보적 변환관계를 나타내는 시뮬레이션 궤적을 통한 자동분석을 가능케 하였으며, 성능분석문제에 있어서도 각종 컴퓨터 구조의 최적효율을 보장하는 최적의 타이밍관계를 자동생성해 낼 수 있는 새로운 방법이 제안되었다. 그리고 아직 학문적으로 초기 단계에 불과한 사건중심 제어문제에 있어서도 주어진 제어 조건에 따른 controllability를 보장해주는 최적의 제어법칙을 자동생성해 낼 수 있어서 이 분야에 많은 발전가능성을 제시하고 있다. 아울러, 제안된 방법론은 본 논문을 통해 제시된 응용분야외에도 인공지능 및 시스템의 여러 분야에 있어서 새로운 각도의 접근 방법을 가능케할 것으로 사료되는 바, 계속적 연구가 진행되어야 할 것이다.

감사의 글

본 연구는 한국학술진흥재단의 93' 신진교수지원 과제에 의해 수행되었으며, 관계자 여러분께 감사드린다.

참고문헌

[1] L. Wos, "Automated Reasoning" In: Artificial Intelligence, A Knowledge-Based Approach edited by M.W. Firebaugh, PWS-KENT Pub., 1988.

[2] Y. Ho, "Editors Introduction", Special Issue on Dynamics od Discrete Event Systems, Proc. of IEEE, Vol.77, No.1, 1989.

[3] B.P.Zeigler, Multifacetted Modelling and Discrete Event Simulation, Academic Press, 1984.

[4] B.P.Zeigler, Object-Oriented Simulation with Hierarchical Modular Models: Intelligent Agents and Endomorphic Systems, Academic Press, 1990.

[5] B.P.Zeigler, "DEVS Representation of Dynamical Systems: Event-based Intelligent Control", IEEE Proc., Vol.77, No.1, Jan., pp.72-80, 1989.

[6] B.P. Zeigler, F.E. Cellier, and J.W. Rozenblit, "Design of a Simulation Environment for Laboratory Management by Robot Organizations", J.Intelligent and Robotic Systems, Vol.1, pp.299-309, 1988.

[7] S.D. Chi and B.P. Zeigler, "DEVS-Based Intelligent Control of Space Adapted Fluid Mixing", Proc. of 5th Coonf. on Artificial Intelligence for Space Application, May, 1990.

[8] B.P. Zeigler and S.D. Chi, "Symbolic Discrete Event System Specification", IEEE Trans. on System, Man, and Cybernetics, 1992.

[9] S.D. Chi and B.P. Zeigler, "Linear Polynormial Constraints Inferencing Algorithm", Submitted to J.of Applied Data & Knowledge Engineering, 1992.

[10] 정보과학회, 정보과학회지, 특집, 인공지능-상식 추론, 제10권 제4호, 1992년.

[11] S.D. Chi and B.P. Zeigler, "Hierarchical Model-based Diagnosis for High Autonomy Systems" accepted to J. of Intelligent and Robotic Systems, Kluwer Pub., 1992.

[12] R.C. DeVries, "An Automated Methodology for Generating a Fault Tree", IEEE Trans. on Reliability, Vol.39, No.1, 1990, April.

[13] F.A. Patterson-Hine and B.V. Koen, "Direct Evaluation of Fault Trees Using Object-Oriented Programming Techniques", IEEE Trans. on Reliability, Vol.38, No. 2, 1989, June.

[14] S.D. Chi, Modelling and Simulation for High Autonomy Systems, University of Arizona, Ph.D Dissertation, 1991.

[15] S.D. Chi, B.P Zeigler, and F.E. Cellier, "Model-Based Task Planning Systems for a Space Laboratory Environment", Proc.on SPIE Conf.Cooperative Intelligent Robotics in Space, Boston, Nov., 1990.

[16] A. Orden, "On the Solution of Linear Equation/Inequality Systems", Mathematical Programming, Vol.1, pp.137-152, 1971

[17] NASA, The Space Station Program, NASA Publication, 1985.

[18] B.P. Zeigler, "Systems Formulation of a Theory oh Diagnosis from First Principles", submitted to IEEE Trans on Reliability, 1992.

[19] R. Reiter, "Theory of Diagnosis from First Principles",

Artificial Intelligence, Vol.32, pp.57-95, 1987.

[20] R. Greiner, B.A. Smith and R.W. Wilkerson, "A Correction to the Algorithm in Reiter's Theory of Diagnosis", Artificial Intellicence, Vol.77, No.1, 1989.

[21] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior", Artificial Intelligence, Vol.24, pp. 347-410, 1984.

[22] J.B. Fussell, G.J. Powers, and R.G. Bennetts, "Fault-trees—A state of the Art Discussion", IEEE Trans. on Reliability, Vol. R-23, No. 1, April, 1974.

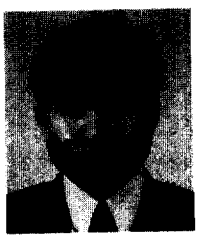
[23] E.J. Henley, H. Kumamoto, Reliability Engineering and Risk Assessment, Prentice-Hall Pub., 1981.

[24] S.A. Lapp, G.J. Powers, "Computer-aided Synthesis of Fault-Trees," IEEE Trans. on Reliability, Vol. R-26, pp. 2 - 13, April, 1977.

[25] F.A. Patterson-Hine and B.V. Koen, "Direct Evaluation of Fault Trees Using Object-oriented Programming Techniques," IEEE Trans. on Reliability, Vol. 38, No. 2, June, 1989.

[26] W.M. Wonham, P.J. Rarmadge, "On the Supremal Controllable Sublanguage of a Given Language", SIAM J. Control and Optimization, Vol.25, No.3, pp.653-656, 1987.

● 저자소개 ●



지승도(池 承 道)

1959년 2월 19일생. 1982년 연세대 전기공학과 졸업.
1984년 동 대학원 전기공학과 졸업 (석사).
1991년 아리조나대 전기 및 전산공학과 졸업 (박사).
1985 - 86년 두산 컴퓨터 (현 한국 디지탈) 근무.
1991 - 92년 미국 SIMEX S/W 회사 S/W 담당자로 근무.
현재 한국항공대학교 컴퓨터공학과 조교수.
관심분야 : 모델링 및 시뮬레이션, 지능 시스템 설계