LOTOS 정형명세의 시각적 분석 지원 도구

조 수 선† · 이 광 용†† · 오 영 배†††

요 약

본 논문에서는 LOTOS 정형 명세의 확인을 위한 시각적 분석 지원 도구의 개발을 소개한다. 이 도구는 대표적인 두 가지 기능을 제공하는데 명세의 시각적 시뮬레이션과 시각적 확장 기능이 그것이다. 정형 기법 지원도구에서 명세의 분석을 위한 시각적인 방법을 제공하는 것은 시스템 개발에 실질적인 적용 사례가 드문 현실에서, 산업계 등으로 정형기법을 확산시키기 위한 중요한 요인이 된다. 그럼에도 불구하고 기존의 정형 기법 지원 도구에 관한 연구에서는 이 부분을 소홀히 다루었다. 본 연구에서 개발된 도구는 LOTOS 명세 작성자로 하여금 비주얼 전이 트리를 이용하여 보다 직관적이고 편리하게 명세를 분석할 수 있도록 지원한다.

A Visual Analysis Tool for LOTOS Specifications

Soo-Sun Cho† · Kwang-Yong Lee†† · Young-Bae Oh†††

ABSTACT

In this paper, we introduce a development of a visual analysis tool for LOTOS specifications. The tool has two major functions, which are visual simulation and visual expansion of LOTOS specifications. Providing visual analyzing function is very important to extend the formal methods in industry where the practically applied cases are rarely found. Nevertheless, there are few tools that concern the visual analyzing functions. The developed tool supports that a LOTOS specifier is able to use visual transition trees in order to analysis the LOTOS specification more intuitively and comfortably.

1. 서 론

정형 기법은 엄격한 수학적 바탕을 가진 정형 명세 언어(formal specification language)를 사용하여 시스템을 명세하고, 이러한 정형 언어로 작성된 요구 명세는 애매모호함이 없는 간결하고 정확한 표현으로 이루어져 있으므로 수학적 검증(verification) 및 확인(validation)이 가능하다. 최근 소프트웨어 시스템은 산업의 전 분야로 사용 영역이 확산되고 있고 고 안전성 및 고 신뢰성을 요하는 시스템의 개발에 이러한 정형 기법(formal methods)을 적용하려는 시도가 늘고있다. 정형 기법을 사용함으로써 요구 명세의 분석 단계에서 미리 심각한 오류들을 제거할 수 있고 이 오류들이 설계 및 구현으로 파급되는 것을 막을 수 있기 때문에 보다 효과적으로 고품질의 소프트웨어 시스템을 개발할 수 있다[16].

정형 기법은 오류를 검출하는 효과면에서는 인정을 받아왔으나 사용상의 어려움과 교육 및 홍보 부족, 그

※ 본 연구는 1997년 정보통신부 연구비 지원에 의한 결과임.
† 정 회 원 : 한국전자통신연구원 컴퓨터·소프트웨어기술연구소 선임연구원
†† 정 회 원 : 숭실대학교 생산기술연구소 연구원
††† 정 회 원 : 한국전자통신연구원 컴퓨터·소프트웨어기술연구소 선임연구원 개발자동화지원도구, 영역기반재사용 기술 과제책임자
논문접수 : 1998년 3월 2일, 심사완료 : 1998년 10월 13일

리고 지원 도구의 미비 등으로 현실적인 적용 사례를 찾아보기가 쉽지 않았다. 하지만 근래에는 소프트웨어 명세 분야 또는 하드웨어 검증 분야에서 실제로 산업계에 적용하여 정형 기법의 이점을 얻는 경우가 점차 늘고 있다[3].

일반적으로 정형 기법의 적용이라 함은 수학적으로 잘 정의된 정형 명세 언어를 사용하여 명세를 작성하고 이것의 검증 및 확인 작업을 통하여 올바른 명세를 획득하며, 또한 명세를 더 구체적인 형태로 변환(transform) 또는 정제(refine)함으로써 구현에 이르게 하는 일련의 과정을 말한다. 따라서 이 과정을 지원하는 자동화 도구의 사용은 필수적인 요소로 인식되고 있다.

본 논문에서는 LOTOS(Language Of Temporal Ordering Specification) 정형 명세의 확인을 위한 분석 지원 도구[17]를 소개하고자 한다. ViLAT(a Visual LOTOS Analysis Tool)이라 명명된 이 도구는 LOTOS 명세의 시뮬레이션과 확장을 시각적으로 표현함으로써 사용자로 하여금 명세된 시스템의 행위 분석을 손쉽게 할 수 있도록 지원한다. 먼저 2장에서 관련 연구를 소개하고 3장, 4장에서는 개발된 도구의 대표적인 기능인 시뮬레이션과 명세 확장에 대해 설명하며, 5장에서는 실제로 진행된 구현 작업을 소개한다. 마지막으로 6장에서는 결론과 함께 앞으로의 과제를 도출한다.

2. 관련 연구

2.1 LOTOS

정형 명세언어 LOTOS는 1989년 국제 표준화 기구인 ISO에 의해 제정되었다. LOTOS는 분산, 동시성 정보 처리 시스템의 정형적 표현을 위해 개발되었는데, 특히 OSI(Open Systems Interconnection) 서비스와 프로토콜의 정형 명세를 목적으로 하였다[12].

LOTOS에서는 시스템을 하나의 프로세스로 나타내고 프로세스는 또한 서브 프로세스를 가질 수 있어서 계층적으로 표현된다. 프로세스는 관측되지 않는 내부 동작(internal action)을 취할 수도 있고, 외부(environment)로부터 관측 가능한 동작을 취할 수도 있다. 프로세스들 간의 상호작용(interaction)은 관측 가능한 동작들이 프로세스들 사이에서 동시에 발생함으로써 동기화(synchronization)될 때 일어난다. 동기화가 일어나는 장소를 게이트라 한다. 결국 LOTOS에서 시스템의 행위는 외부에서 관측되는 동작들 간에 순차적인 관계를 정의함으로써 표현된다[14].

LOTOS는 크게 두 가지 구성 요소로 이루어져 있는데 첫번째 요소는 프로세스의 행위와 상호작용을 나타내기 위한 부분으로 프로세스 대수인 CCS(Calculus of Communication Systems)[15]에 기초를 두고 있고, 두번째 요소는 자료구조 및 자료 값 표현을 위한 것으로 추상 자료형 언어인 ACT ONE[6]을 도입한 것이다[2][14].

2.2 LOTOS 명세의 확인

LOTOS 명세의 확인 작업은 명세의 검증, 시뮬레이션, 프로토타이핑, 시험, 변환 등을 포함한다[4].

명세의 검증(verification)이란 어떤 명세가 또 다른 명세 – 대개 설계 과정에서 더 이전에 개발된 명세와 올바른 관계를 가지고 있음을 증명하거나 명세가 논리적으로 표현된 특성들 – 명세의 정확성(correctness), 안전성(safety), 일관성(consistency), 활인성(liveness) 등을 가지고 있음을 증명하는 것이다.

명세의 시뮬레이션은 주어진 단계(state)에서 동기화가 가능한 동작들 중 하나를 선택함으로써 시뮬레이션 도구와 상호 작용하여 명세를 분석하는 작업이고, 프로토타이핑은 실행 가능한 언어로 명세를 변환하여 실행시켜 보는 방법이다.

시험(testing)은 초기 단계의 명세에서 테스트 케이스를 생성하여 이를 보다 구체적인 명세에서 실행시켜 보는 방법이다.

2.3 기존 도구

LOTOS 명세의 확인 기법 중 시뮬레이션과 명세의 변환 및 확장을 지원하는 도구를 조사해 보면 다음과 같다.

대표적인 LOTOS 시뮬레이션 도구로는 네덜란드의 Twente 대학에서 개발한 SMILE[5]과 캐나다 Ottawa 대학에서 개발한 ISLA[9][10]가 있다. 이들은 모두 단계별(step-by-step) 시뮬레이션을 주 기능으로 한다. 즉 사용자가 시뮬레이션 환경의 역할을 하여 주어진 동작들 중 하나를 선택하면 다음 상태로의 전이가 일어나고 계속 이러한 작업을 되풀이하면서 명세의 행위를 분석한다.

SMILE은 ISLA에 비해 심볼형 전이(symbolic transition)를 지원한다는 점에서 한 차원 높은 기능을 제

정된다. ISLA는 사용자로 하여금 동작을 선택할 때 변수 값 또한 입력하게 되어있기만 SMILE에서는 변수를 그대로 유지한 채 심볼형의 전이 트리를 생성함으로써 전이를 자동화 할 수 있다. SMILE에서는 전이 트리에서 관심 있는 노드(즉, 상태)를 펼침(unfolding)으로써 다음 상태로의 전이를 이룬다.

전통적인 의미의 시뮬레이션이 사용자와의 상호작용을 통하여 관심이 있는 동작의 경로를 쫓아가는 것이라면 일정한 depth만큼 자동적으로 전이 트리를 생성하는 도구 또한 LOTOS 분석 작업에 유용하게 쓰일 수 있다. 여기서 depth란 순차적으로 나열된 동작의 수를 말한다. 이러한 전이 트리의 생성은 명세의 확장(expansion)에 의해 가능하며 전이 트리는 LOTOS의 의미론에서 LTS (Labeled Transition System)에 해당하므로 LTS를 자동 생성하는 것이 중요한 이슈가 된다. 이를 지원하는 도구로는 스페인 Madrid대학의 LOLA[13]와 캐나다 Ottawa대학의 SELA[11][9]가 있다. 이 두 도구는 모두 LOTOS의 확장 정리(Expansion Theorem)를 이용하여 심볼형 전이 트리를 생성한다. LOLA는 전이 트리를 생성하기 위해 또 다른 LOTOS 명세로의 변환 방법을 취하는 반면 SELA는 LOTOS 명세를 Prolog로 바꾼 후에 전이 트리를 생성한다.

이 두 도구와 유사한 기능을 제공하는 것으로 프랑스 INRIA에서 개발한 CAESAR[8]가 있다. CAESAR는 상호유사성 검사 도구인 ALDEBARAN과 함께 툴박스CADP[7]에 포함되어 보급되며 가장 최근까지 업데이트되고 있다. LTS의 생성을 위하여 중간 단계로 Petri Net을 이용하며 가장 많은 적용 사례를 가지고 있다.

소개된 도구들은 조금씩 다른 접근 방식으로 LOTOS 명세의 분석을 위한 연구를 하였고 대부분이 대학을 중심으로 개발되었다. 사용 환경은 대부분 Unix 환경을 기반으로 하고 있고 LOLA와 CADP는 각각 PC의 DOS와 Linux를 지원하고 있다. 대학에서 개발된 것이 대부분이므로 사용자 인터페이스를 고려한 전이 트리의 시각적 표현을 구현한 것은 없다.

3. 시뮬레이션

3.1 LOTOS 명세의 시뮬레이션

LOTOS는 그 의미론이 LTS(Labeled Transition System)로 표현되기 때문에 주어진 명세를 동작(action)의 선택에 의한 상태 전이(sate transition)의 연속으로 실행시킬 수 있다. 이것을 명세의 시뮬레이션이라 하고 이 작업은 사용자와 상호 상용하는 시뮬레이션 도구를 이용하여 수행된다.

LOTOS 명세의 시뮬레이션은 다음과 같은 절차를 통해 이루어진다.

- 시뮬레이션 도구는 현재 상태에서 동기화(synchronization) 가능한 모든 종류의 동작을 계산하여 메뉴로 제시한다.
- 사용자는 동작 메뉴에서 관심 있는 하나의 동작을 선택한다. 이 때 사용자는 명세된 시스템의 환경으로 작용하여 선택한 동작으로 동기화를 이루는 역할을 한다.
- 시뮬레이션 도구는 현재 상태와 선택된 동작으로 다음 상태를 계산하여 그 상태로 전이한 후 다시 동작 메뉴를 제시한다.
- 사용자의 동작 선택에 의해 상태 전이가 반복된다.

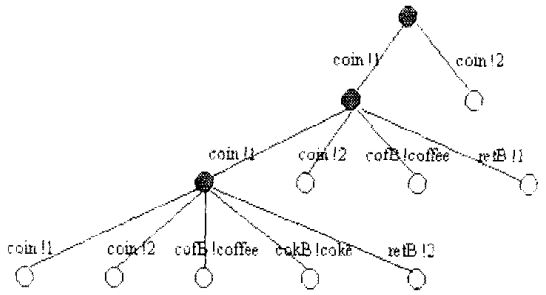
LOTOS 시뮬레이션에서는 이와 같은 방법을 계속 적용하여 명세에 표현된 시스템의 행위가 명세자의 의도대로 올바르게 실행되는지 손쉽게 분석할 수 있다.

LOTOS 시뮬레이션을 보다 이해하기 쉽게 소개하기 위해 간단한 예제를 사용하여 설명해 보자. 다음 명세는 간단한 자동판매기를 나타내는 LOTOS 행위 명세이다.

```
behaviour
  VM[coin, cofB, cokB, retB] (0)
where
process VM[coin, cofB, cokB, retB] (n:Nat) : noexit :=
  coin !1; VM[coin, cofB, cokB, retB] (n+1)
  []
  coin !2; VM[coin, cofB, cokB, retB] (n+2)
  []
  cofB !coffee [n ge 1]; VM[coin, cofB, cokB, retB] (n-1)
  []
  cokB !coke[n ge 2]; VM[coin, cofB, cokB, retB] (n-2)
  []
  retB !n [n gt 0]; VM[coin, cofB, cokB, retB] (0)
endproc
```

추상 자료형 1, 2는 자연수로 정의할 수 있고 여기서 자세한 정의는 생략한다. 명세된 시스템은 외부 환경과 coin 게이트를 통해 동전 1, 2를 받거나 cofB, cokB 게이트를 통해 커피 또는 콜라를 내보내는 동작을 한다. 또한 retB 게이트를 통해 남은 동전을 내보내기도 한다.

이 명세를 이용하여 시뮬레이션을 하였을 때, 예를 들면 동전 1을 두 번 받아들였을 때 시스템이 어떤 상태를 나타내는가 살펴보고자 한다면 이러한 상황을 표현해주는 시뮬레이션 트리가 필요하게 된다. (그림 1)의 시뮬레이션 트리는 외부 환경, 즉 사용자가 선택 가능한 동작 중 coin !1을 연속으로 두 번 선택했을 때의 상황을 보여준다. 사용자는 이 상태에서 시스템이 동전 1, 2를 받거나 커피 또는 콜라를 내보내거나 동전을 리턴할 수 있음을 알 수 있다. 따라서 coin !1을 연속으로 두 번 발생시켰을 때 이러한 상태에 도달하는 것이 원래의 의도에 부합하는지 확인할 수 있게 된다.



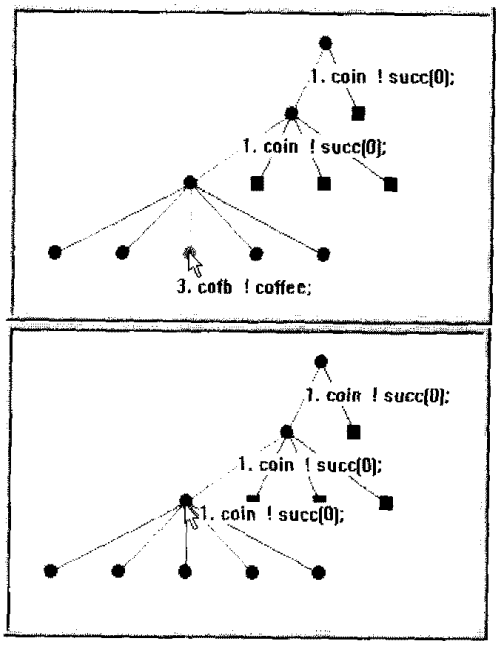
(그림 1) LOTOS 시뮬레이션 트리
(Fig. 1) Simulation Tree of LOTOS Spec.

3.2. 시각적 시뮬레이션

앞에서 살펴본 바와 같이 LOTOS 명세의 시뮬레이션은 도구와 사용자간의 상호 작용이 매우 빈번하게 일어나는 작업이므로 무엇보다 편리한 사용자 인터페이스를 요구한다. 또한, 상호 작용에 의한 결과와 과정이 시뮬레이션 트리로 표현될 때 가장 자연스럽게 이해된다. 본 연구에서는 LOTOS 시뮬레이션의 이러한 특징을 도구에 어떤 식으로 반영할 것인지를 중요하게 다루었다. 기존의 LOTOS 시뮬레이션 도구들에 비해 빠르고 쉬운 상호작용을 지원하고 현재 상태와 경로를 시각적으로 이해할 수 있도록 하기 위해 ViLAT 도구에서는 다음과 같은 기능을 선택하였다.

- (그림 1)과 같은 개념적인 시뮬레이션 트리를 직접 화면에 시각적으로 표시해 준다.
- 사용자는 그래픽 시뮬레이션 트리상에서 도구와 상호 작용할 수 있다. 즉, 현재 상태의 노드들 중 하나를 마우스 클릭으로 선택하여 다음 상태로 천이를 이룰 수 있다.
- 사용자의 선택에 의해 새로 구성된 그래픽 시뮬레이션 트리가 화면에 재신시된다.
- 사용자는 마지막 부모 노드를 클릭함으로써 최근의 상태 천이를 취소하고 이전 상태로 되돌아 갈 수 있다.
- 현재까지의 경로를 나타내는 노드들의 연결선들을 구분되는 색으로 표시하여 한눈에 알아 볼 수 있게 한다.

ViLAT은 LOTOS 시뮬레이션을 위해 이와 같은 기능을 지원함으로써 다른 도구들에 비해 매우 빠르고 편리한 사용자 인터페이스를 제공한다. 또한, 시각적인 시뮬레이션을 통해 전체 분석 과정에 대한 직관적인 이해를 도움으로써 명세의 분석 작업에 대한 효율성을 향상시킨다.



(그림 2) ViLAT의 시뮬레이션 트리
(Fig. 2) Simulation Tree in ViLAT

(그림 2)는 ViLAT 시뮬레이션 트리의 화면상의 모습이다. 앞에서의 자동판매기 예제를 그대로 적용한 결과이며 위쪽 그림은 사용자가 현재 상태에서 동작 "cofb !coffee"를 선택하는 순간을 보여주고 있으며 아래쪽 그림은 상태 천이의 취소를 위해 부모 노드를 클릭하는 순간을 보여주고 있다.

4. 명세 확장

4.1. LOTOS 명세의 확장

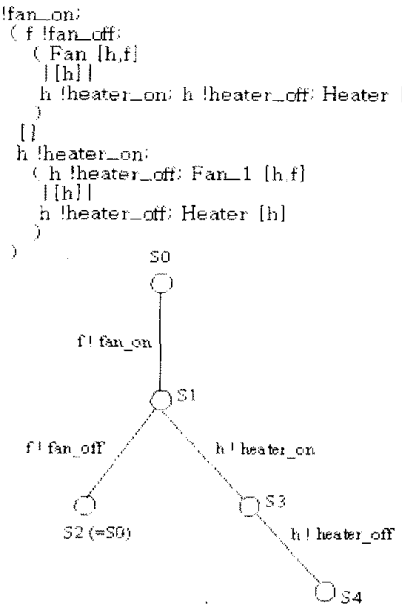
명세 확장(Expansion)이란 LOTOS명세를 동작 선행자(action prefix)와 선택(choice) 연산자로만 표현한 것이다. LOTOS는 Temporal Ordering 이란 이름에서 나타나듯이 이벤트의 동시성을 결국 모든 가능한 종류의 이벤트의 순차적인 발생으로 해석하므로 여러 가지 병렬 연산자는 모두 동작 선행자와 선택 연산자만을 가진 명세로 확장될 수 있다.

사용자는 현재 상태를 나타내는 LOTOS 명세를 확장해 봄으로써 가능한 전이의 종류와 개수를 알 수 있고 교착점(deadlock)이 발생하는지 살펴볼 수 있다. 시뮬레이션이 한가지 경로를 선택하여 단계별로 상태전이를 점검하는 분석 방법이라면 명세 확장은 모든 가능한 전이를 한번에 계산하여 그 결과를 통계치로 알아보는 분석 방법이다.

명세 확장의 구체적인 결과를 보여주기 위해 간단한 예제를 통해 설명해 보자. 아래 명세는 전체 명세에서 서브시스템을 나타내는 프로세스 Fan과 프로세스 Heater가 게이트 h를 통하여 동기화되는 Fan_Heater시스템을 표현하고 있다.

```
specification Fan_Heater[h, f] : noexit
type Opn is
sorts Op
opns
  fan_on, fan_off, heater_on, heater_off : -> Op
endtype
behaviour
  Fan[h, f] |[h]| Heater[h]
where
  process Fan[h, f] : noexit :=
        f !fan_on; Fan_1[h, f]
  where
        process Fan_1[h, f] : noexit :=
          f! fan_off; Fan[h, f]
        []
          h !heater_on; h !heater_off; Fan_1[h, f]
        endproc
  endproc
  process Heater[h] : noexit :
        h !heater_on; h !heater_off; Heater[h]
  endproc
endspec
```

실현대이션에서와 마찬가지로 LOTOS 명세의 확장도 트리로 표현되면 직관적인 이해를 할 수 있다. (그림 3)은 이 명세의 행위 표현식 'Fan[h, f] |[h]| Heater[h]'를 초기 상태로 하여 확장했을 때의 결과와 그에 대응하는 확장 트리의 모양을 보여준다. 확장 깊이(depth)는 3으로 준 경우이다. 여기서 확장 깊이란 연속되는 동작의 최대 개수를 말한다.



(그림 3) LOTOS 확장 트리
(Fig. 3) Expansion Tree of LOTOS Spec.

일반적인 확장 규칙에서 확장을 멈추는 경우는 반복 상태나 교착점에 도달했을 때와 주어진 확장 깊이에 도달했을 때이다. (그림 3)은 이러한 일반확장의 결과를 보여준다.

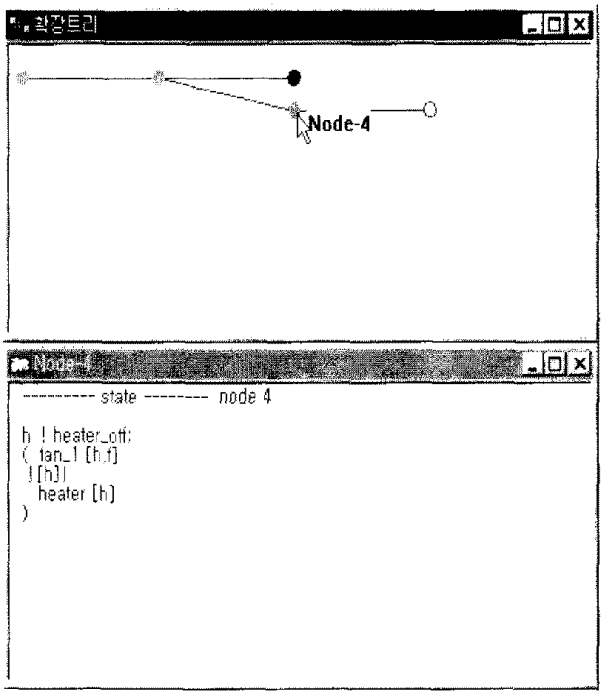
생성된 확장 트리는 모두 4개의 에지(전이)와 5개의 노드(상태)를 가진다. 확장 깊이를 3으로 지정하였기 때문에 루트 노드로부터 마지막 노드까지의 최대 에지 수는 3이 된다. 루트 노드를 초기 상태 S0라고 하였을 때 두 번의 동작, 'f ! fan_on', 'f ! fan_off'가 일어난 후 도달한 상태S2는 초기 상태 S0와 같은 상태이다. 즉 반복 상태가 발견되어 더 이상의 전이가 일어나지 않고 상위 노드로 되돌아 간다. 상태 S1에서는 게이트 h를 통해 두 번의 동작, 'h ! heater_on', 'h ! heater_off'가 일어난 후 지정된 확장 깊이에 도달하였으므로 더 이상의 전이가 일어나지 않고 멈춘다. 이 때 도달

된 상태 S4는 분석되지 않았으므로 이전 상태의 반복 상태인지 또 다른 상태인지 알 수 없다. 따라서 위의 예에서 분석된 총 상태의 종류는 3가지이고 발생한 전이의 수는 4이며, 반복 상태 수는 1이다. 교착점은 나타나지 않았다.

4.2. 시각적 명세 확장

명세 확장은 시뮬레이션만큼 도구와 사용자간의 빈번한 상호작용이 필요하지 않다. 그러나 확장 트리의 그래픽 표현은 사용자의 이해를 돕고 상태 분석에 대한 전체적인 파악에 큰 도움을 준다. 더구나 확장 결과에 의해 생성된 상태들의 종류를 그래픽으로 한눈에 구분할 수 있다면 더욱 유용할 것이다. 따라서 ViLAT 도구에서는 LOTOS 명세 확장을 위해 다음과 같은 기능을 제공하였다.

- 명세 확장에 의해 생성된 모든 상태를 노드로 표시하고 전이를 에지로 표시하는 그래픽 트리를 생성하여 제시한다.
- 노드들을 분석된 상태, 반복 상태 등 상태의 종류에 따라 다른 색으로 구분해 준다.
- 마우스 클릭으로 노드를 선택하면 자세한 상태 정보를 제공해 준다.



(그림 4) ViLAT의 확장 트리
(Fig. 4) Expansion Tree in ViLAT

그림 4는 앞에서 사용한 Fan_Heater 예제 명세를 ViLAT을 사용하여 확장시킨 결과이다. 확장 깊이를 마찬가지로 3으로 준 경우이며 위쪽, 왼쪽부터 일련의 노드 번호가 부여된다. ViLAT의 그래픽 확장 트리에서는 녹색으로 표시되는 분석 노드 수가 총 3개이며 청색의 반복 노드 수가 1임을 알 수 있고 이들 중 하나를 선택했을 때 그 상태를 나타내는 확장된 명세를 확인 할 수 있다. 그림에서는 노드 4의 상태를 나타내고 있다. 또한, 확장 깊이에 도달하여 분석되지 못한 상태는 백색의 노드로 표시하고 있다.

ViLAT은 LOTOS 명세 확장을 위해 이와 같은 기능을 제공함으로써 상태들간의 연결 관계, 즉 전이의 전후 관계를 파악할 수 있게 하고 관심 있는 상태가 어떤 경로를 통해 도달되는지 쉽게 알아볼 수 있게 한다. 이 것은 다른 도구들이 단순히 결과 통계치만 제시하는데 비해 보다 명확한 명세의 분석 작업이 이루어질 수 있도록 지원한다.

5. ViLAT 구현

5.1. 연구 목적 및 개발 환경

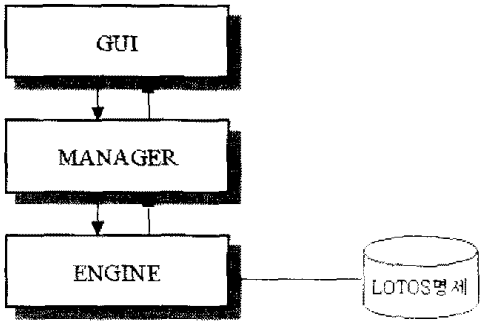
3, 4장에서 소개한 바와 같이 ViLAT은 LOTOS 명세 분석 과정을 쉽게 이해할 수 있게 하고, 사용자와의 상호 작용을 빠르고 편리하게 할 수 있도록 지원한다. 본 연구에서는 이와 같은 실용적인 정형 명세 분석 도구의 개발 및 보급에 의해 궁극적으로 LOTOS 정형 기법의 산업계 확산에 기여하는 것을 목적으로 한다.

시스템 개발 및 사용 환경은 팬티엄PC의 Win95를 기반으로 한다. 대학을 중심으로 시험적으로 개발된 대부분의 LOTOS명세 분석 지원 도구들은 워크스테이션의 Unix기반이고, 성능이 안정화 되어있지 않으며, 쓸만한 GUI의 제공에 소홀하므로 산업계에 사용이 확산되는데 한계가 있다. 따라서 많은 사용자들에게 친숙한 사용 환경에서 시각적 분석 기능을 지원함으로써 보급 효과를 한층 높일 수 있으리라 기대된다.

5.2. 구성

ViLAT은 (그림 5)와 같이 크게 3 종류의 계층으로 구성된다. GUI는 사용자와 직접 상호작용하기 위한 부분으로 사용자 입력을 받아들이고 시뮬레이션 및 확장 결과를 여러 가지 다양한 윈도우에 선시하는 역할을

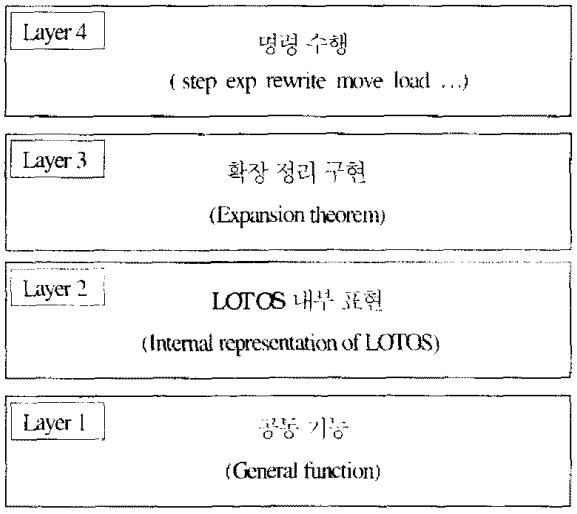
담당하고, 관리기(MANAGER)는 GUI를 통해 입력된 자료를 가공 처리한 후 구동기(ENGINE)를 호출하는 작업과 구동기로부터의 결과를 GUI에 전시하기 위한 작업을 담당한다. 마지막으로 구동기는 관리기로부터 받아들인 명령과 옵션으로 시뮬레이션 및 확장의 기본적인 처리를 하고 결과를 관리기에 넘겨준다.



(그림 5) ViLAT 구성
(Fig. 5) ViLAT Architecture

5.3. 구동기

ViLAT 도구는 시뮬레이션과 명세 확장의 시각적 지원 방법을 통해 도구의 실용성을 높이는 것을 목적으로 하므로, 빠른 시간 내에 쓸모 있는 도구를 개발하기 위해 시스템의 구성 요소 중 구동기에 해당하는 부분을 LOLA의 재사용에 의한 라이브러리를 구축하여 구성한다.



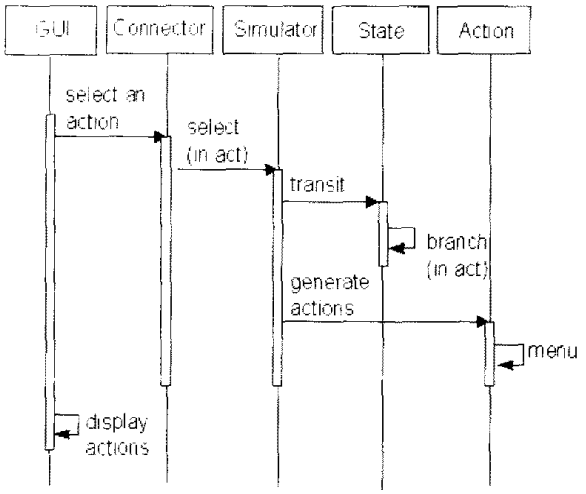
(그림 6) LOLA 라이브러리의 계층 구조
(Fig. 6) Layers of LOLA Library

본 장에서 소개한 바 있는 LOLA는 명세의 시뮬레이션 기능과 명세 확장 기능을 함께 제공하고, 연구실에서 개발중인 다른 도구와 공통의 내부 표현을 사용하므로 가장 적당한 재사용 대상으로 선택되었다.

(그림 6)는 LOLA라이브러리의 계층 구조이다. 이와 같은 구조에서 관리기는 명령 수행 계층을 호출하여 사용한다.

5.4. 관리기

관리기의 설계에서는 객체 통신 모형이 사용되었다. 객체 통신 모형은 클래스 통신도와 실행 추이도의 두 가지 다이어그램으로 이루어진다. (그림 7)은 시뮬레이션 기능에서 동작 선택에 의한 상태 전이 부분을 설계한 실행 추이도이다.



(그림 7) LOTOS 시뮬레이션 실행 추이도
(Fig. 7) Execution Sequence Diagram of LOTOS Simulation

(그림 7)에서 클래스 GUI를 제외한 4개의 클래스(Connector, Simulator, State, Action)는 모두 관리기 모듈에 포함된다. GUI로부터 전달된 동작 선택 결과는 Connector, Simulator 객체를 거쳐 실질적인 관리기 객체에 해당하는 State와 Action에 전달되고, 이 두 객체는 LOLA라이브러리의 해당 함수를 호출하여 그 결과를 전달 받아 임시 저장 장소에 보관한다. 이후 GUI 모듈에서는 이 임시 저장 형태의 데이터를 참조하여 결과를 전시한다.

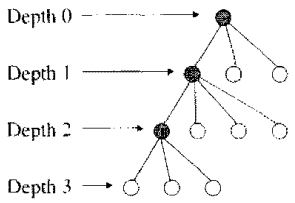
이와 같이 관리자 모듈의 설계에 객체 통신 모형을 이용함으로써, 설계로부터 일관성 있는 C++언어의 도출이 가능하였다.

5.5. GUI

ViLAT 도구에서는 사용자 친화적인 GUI의 개발에 역점을 두었으며 사용자와의 원활한 상호 작용을 구현하는 방법을 중요하게 다루었다. 그 방법의 대표적인 두 가지 요소가 3, 4장에서 소개한 그래픽 시뮬레이션 트리와 그래픽 확장 트리이다. 여기서는 대표로 4장의 그래픽 확장 트리 구현을 간단히 소개한다.

그래픽 확장 트리의 구현은 LOLA라이브러리에서 제공하는 depth list를 이용한다. (그림 8)은 depth list와 이에 해당하는 트리의 모양을 나타낸다.

```
                                  | Transits | States
0 - 3/2 - 3/2 - 3/1 - 2/1 - 2 |   10    |   3
1 - 2/0 - 1/0 - 1/.
```



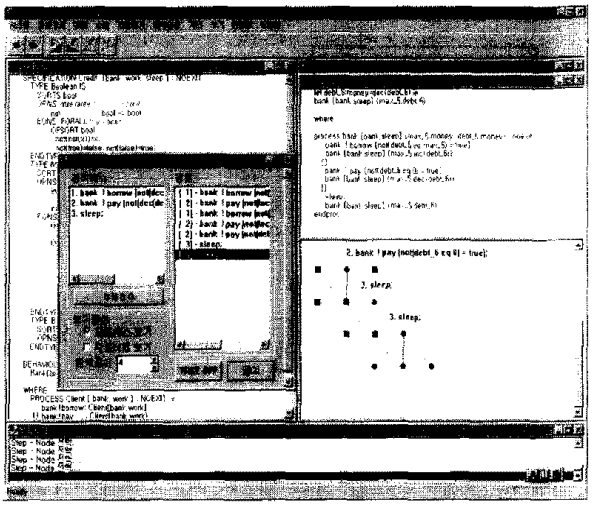
(그림 8) Depth List와 확장 트리
(Fig. 8) Depth List and Expansion Tree

Depth list의 첫번째 덤인 0 - 3/ 은 depth 0에서 3에 이르는 전이가 존재함을 의미한다. 이어지는 2 - 3/ 은 마지막 생성된 depth 2의 노드로부터 다시 depth 3에 이르는 또 다른 전이가 존재함을 의미한다. 이와 같은 방법으로 트리를 구성해 보면 (그림 8)에서와 같은 모양이 된다. 짙은 색으로 표시된 노드는 반복이 아닌 상태를 나타낸다. (그림 8)은 일반 확장의 예를 보여주므로 마지막 depth에 이르기 이전에 확장을 멈춘 노드들은 모두 반복 상태를 나타낸다. ViLAT에서는 이 depth list를 이용하여 (그림 4)와 같은 그래픽 확장 트리를 생성한다.

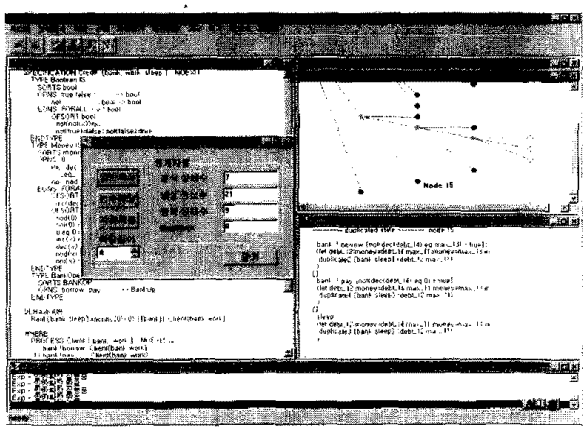
5.6. 대표 화면

다음의 (그림 9)와 (그림 10)은 ViLAT의 대표적인 기능인 시뮬레이션 및 명세 확장이 진행되는 화면이다.

(그림 9)는 시뮬레이션 진행 화면으로서 선택된 경로에 의한 시뮬레이션 트리가 화면의 오른쪽 아래에 나타나 있고 이 트리 상에서 바로 입력 처리가 가능하게 되어있다. 즉, 트리의 마지막 depth 노드들 중 하나를 더블 클릭함으로써 다음 상태로의 전이를 유발시키고, 전이가 일어난 후 바로 상위의 노드를 더블 클릭함으로써 전이를 취소 시킬 수도 있다. 한편, 왼쪽 중앙에는 선택 가능한 동작의 종류와 지금까지의 경로를 텍스트 형식으로 보여주는 다이얼로그 창이 존재한다.



(그림 9) 시뮬레이션 실행 화면
(Fig. 9) Display of LOTOS Simulation



(그림 10) 명세 확장 실행 화면
(Fig. 10) Display of LOTOS Expansion

(그림 10)은 명세 확장의 진행 화면인데 화면 왼쪽의 바탕에 나타나있는 전체 명세 중에서 관심 있는 부분을 먼저 선택하여 부분적인 명세를 확장할 수 있고, 그 결과가 오른쪽 상단의 확장 트리로 표현된다. 한편, 왼쪽 중앙에는 확장의 종류와 깊이를 받아들이고 확장의 결과 계산된 '분석 상태 수', '생성 전이 수', '교착점' 등이 표현된 다이얼로그 창이 존재한다.

6. 결 론

본 논문에서는 LOTOS로 작성된 시스템의 행위 명세를 시뮬레이션 및 명세 확장 방법을 통해 시각적으로 분석하는 기법과 도구를 소개하였다. LOTOS는 시스템의 행위를 프로세스 대수에 기반하여 명세하므로 시뮬레이션을 통해 행위를 분석하기에 적당한 명세 언어이다. 그러나 지금까지 개발된 도구들은 시뮬레이션의 효과를 극대화 시킬 수 있는 시각적 인터페이스를 제공하지 못하였고 본 연구에서는 이 점에 착안하여 시각적 시뮬레이션 및 명세 확장 기능을 제공하는 도구 ViLAT를 개발하였다.

논문 [11]은 정형 명세 언어 교육에서 도구의 시각화와 상호 작용이 얼마나 중요한 영향을 미치는지 연구한 결과이다. 저자는 이 연구에서 정형 언어 교육에 시각적이고 상호작용을 돕는 지원 도구를 사용함으로써 학생들이 그래픽을 통한 즉각적인 피드백을 얻을 수 있었다고 소개한다. 이처럼 정형 명세의 교육 및 실사용자 층에 대한 보급을 위해서는 본 연구에서 다룬 시각적 분석 인터페이스가 중요한 요소임을 알 수 있다.

<표 1>은 기존의 LOTOS 지원 도구와 ViLAT의 기능 비교표이다.

〈표 1〉 LOTOS 도구 기능 비교표
〈Table 1〉 Function Table of LOTOS Tool s

Tool	ViLAT	SMILE	ISLA/ SELA	CAESAR	LOLA
Simulation	O	O	O	X	O
Expansion	O	△	O	O	O
Symbolic Execution	O	O	△	O	O
GUI	O	O	X	O	X
Visual Tree	O	X	X	X	X
Environment	PC/ Win95	Sun Sparc/ X11	Sun Sparc/ X11	Sun Sparc/ X11, Linux	Sun Sparc/ X11, PC/Dos
Organization	SERI	Univ. of Twente	Univ. of Ottawa	INRIA	Univ. of Madrid

ViLAT은 Win95환경에서 Visual C++ 5.0을 사용하여 개발되었으며 구동기는 LOLA의 필요한 기능을 라이브러리로 작성하여 개발하였고 관리기는 객체 통신 모형을 사용하여 개발하였다. ViLAT의 핵심적인 기능인 비주얼 전이 트리의 구현은 depth list를 이용하여 프로그램 하였다.

향후 관심 있는 분야는 개발된 도구를 이용하여 시스템의 명세 및 분석 사례에 대한 연구를 하는 것이다. 이러한 연구를 위해서는 LOTOS 정형명세를 효과적으로 적용할 수 있는 시스템 개발 사례를 산업계에서 찾고 대상 업체와 협력하여 시스템 명세를 LOTOS로 작성하여야 할 것이며 도구를 사용하여 이를 분석 및 정제해 나가는 과정을 거쳐야 할 것이다. 이와 같은 향후 연구를 통해 LOTOS 정형 명세 및 개발된 지원도구 ViLAT의 효용성이 입증될 수 있을 것으로 기대한다.

참 고 문 헌

[1] P. Ashkar, "Symbolic Execution of LOTOS Specifications," Master's Thesis, University of Ottawa, 1992.

[2] T. Bolognesi and E. Brinksma, "Introduction to the ISO Specification Language LOTOS," Computer Networks and ISDN Systems, Vol.14, No. 1, pp.25-59, 1987.

[3] E. M. Clarke and J. M. Wing et al, "Formal Methods: State of the Art and Future Directions," ACM Computing Surveys, Vol.28, No.4, Dec. 1996.

[4] E. H. Eertink, editor, 'Simulation Techniques for the Validation of LOTOS Specification', Proefschrift, 1994.

[5] E. H. Eertink 'SMILE user manual (release 4.0), Tele-Informaics and Open Systems Group, 1993.

[6] H. Ehrig and B. Mahr, 'Fundamentals of Algebraic Specification 1: Equations and Initial Semantics', EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1985.

[7] J. Fernantez, H. Garavel, A. Kerbrat, R. Mateescu, L Mounier, and M. Sighireanu, "CADP: A Protocol Validation and Verification Toolbox," in

Proc. 8th Conference on Computer_Aided Verification, Aug. 1996.

[8] H. Garavel and J. Sifakis, "Compilation and Verification of LOTOS Specifications," in Proc. of 10th International Symposium on Protocol Specification, Testing and Verification, pp.379-394, June. 1990.

[9] B. Ghribi and L. Logrippo, "A Validation Environment for LOTOS," Protocol Specification, Testing and Verification, XIII, Elsevier Science publishers B. V., pp.93-108, 1993.

[10] M. Haj Hussein, "An Interactive System for LOTOS Applications," Master's Thesis, University of Ottawa, 1988.

[11] M. Procopiuc, O. Procopiuc, and S. H. Rodger, "Visualization and Interaction in the Computer Science Formal Languages Course with JFLAP," in Proc. Frontiers in Education '96, Salt Lake City, USA, Nov. 6-9, 1996

[12] ISO, 'International Standard ISO8807', 1st Ed., 1989.

[13] D. Larrabeiti , S. Pavon and G. Rayvay, 'LOLA user manual (version 3R6), Department De Ingenieria Telematica, 1995.

[14] L. Logrippo, M. Faci and M. Haj-Hussein, "An introduction to LOTOS: learning by examples," Computer Networks and ISDN Systems, Vol.23, No.5, pp.325-342, 1992.

[15] R. Miller, 'Communication and Concurrency', International Series in Computer Science, Prentice-Hall, 1989.

[16] S. L. Pfleeger, "Investigating the Influence of Formal Methods," Computer, pp.33-43, Feb. 1997.

[17] 시스템공학연구소, "정형기법기반 소프트웨어 검증 및 자동생성 기술 개발에 관한 연구", 연구보고서, 1997. 12.

조 수 선

e-mail : scho@etri.re.kr

1987년 서울대학교 자연과학대학 계산통계학과 졸업(학사)

1989년 서울대학교 대학원 계산통계학과(이학석사)

1989년~1994년 (주)웅잔미디어 CBE 개발부 연구원

1994년~현재 한국전자통신연구원 컴퓨터·소프트웨어 기술연구소 선임연구원

관심분야 : 소프트웨어공학 (특히 정형기법, 정형명세, 객체지향 개발방법, CASE)

이 광 용

e-mail : kylee@cub.etri.re.kr

1991년 숭실대학교 전자계산학과 졸업(학사)

1993년 숭실대학교대학원 전산학과(공학석사)

1997년 숭실대학교대학원 전산학과(공학박사)

1997년~1998년 한국전자통신연구원 컴퓨터·소프트웨어기술연구소 박사후 연수연구원(Post-Doc.)

1996년~현재 숭실대학교 생산기술연구소 연구원

관심분야 : 소프트웨어공학, 실시간 시스템, 객체지향 모델링/시뮬레이션, 정형화기법, 분산처리, 인공지능.

오 영 배

e-mail : ybou@etri.re.kr

1982년 고려대학교 기계공학과 졸업(학사)

1995년 인하대학교 전자계산공학과(공학석사)

1995년~1996년 캘리포니아대학(UCI) 객원연구원

1983년~현재 한국전자통신연구원 컴퓨터·소프트웨어 기술연구소 선임연구원 개발자동화지원도구, 영역기반재사용기술 과제책임자

관심분야 : 컴포넌트 기반 재사용 기술, 정형기법, 실시간시스템