속성 명세 지원 시스템

전 승 수 권 기 현

경기대학교 정보과학부 소프트웨어 공학 연구실

Property Specifications Guided System

Seungsu Jun Gihwon Kwon

Software Engineering Laboratory, Department of Computer Science, Kyonggi University

요 약

본 논문에서는 패턴 기반의 시각적 속성 명세 연구를 통해 모든 명세 논리를 포괄하는 요구 속성 명세의 단일 프레임워크와 자동화 지원 도구를 제시한다. 또한 유도 질문을 통한 속성 명세와 속성의 구조 및 상호관계 표현 방법을 보인다. 본 연구에서는 패턴 기반의 시각적 속성 명세 언어(PVSL)를 정의했다. 요구 속성은 속성도를 통해 표현되며 패턴 다이어그램과 속성 and-or 트리로 의미 및 구조를 해석한다. PVSL 과 속성도는 검증자의 기존 지식을 최대한 활용할 수 있도록 계층형 유한 상태 기계의 표기법을 활용한다. 그리고 Nu-SMV 에서 실제 사용된 CTL 예제를 속성도로 명세하고 이를 해석하는데 적용했다. 그 결과 배경 지식을 최소화할 수 있었으며 빠른 명세와 해석이 가능했다. 또한 명세의 오류를 방지할 수 있었으며 속성의 구조와 상호관계를 쉽게 파악할 수 있었다.

1. 연구 배경

모형 검사의 주요 장애 요소 중 하나는 요구 속성 명세의 문제이다[1]. 모형 검사를 하기 위해서 검증자는 검사할 속성을 정확히 표현해야 한다. 속성 명세는 명세 논리와 오토마타, 속성 지향 명세 언어 등이 사용되는데 이를 이용하여 속성을 명세하고 해석하기가 매우 어렵다. 현재의 속성 명세 방법은 검증자에게 많은 배경 지식을 강요한다. 요구 속성의 명세에는 LTL, CTL, CTL*, GIL, QREs, Modal Mu-Calculus 등이 사용된다. 또한 각각의 속성 명세 논리는 표현력의 차이와 고유한 특성을 갖고 있어 검사 대상 시스템의 상태와 요구 속성에 따라 검사 방법과 모형검사기의 선택이 달라진다. 우리의 연구는 속성 명세의 주요 장애 요소에 있어 배경 지식의 최소화, 검증자 편의 보장, 명세 오류 방지, 명세 논리의 재사용, 단일 프레임워크 제시, 그리고 자동화 가능한 지원 도구 개발을 목적으로 진행되었다. 본 연구와 관련하여 기존 연구를 조사한 결과 명세 전용 언어[2], 패턴 이용[3], 시각적 명세 언어[4] 등이 우리의 연구 방향에 부합된 것으로 밝혀졌다. 하지만 기존의 연구는 특정 논리에 한정되어 있고 그 적용과 활용 범위가 제한적이다. 따라서 모든 명세 논리를 포괄할 수 있는 패턴 기반의 시각적 명세 언어(PVSL: Pattern based Visual Specification Language)를 정의했다. 또한 배경 지식의 최소화 및 속성의 의미와 구조의 효과적인 표현 및 해석을 위해 속성도와 패턴 다이어그램, 속성 and-or 트리를 사용했다. 그 결과 검증자의 명세 및 해석이 용이하게 했으며 단일한 속성 명세 프레임워크를 제공할 수 있었다. 또한 요구 속성의 의미 뿐만 아니라 구조의 이해를 도울 수 있었으며 패턴 기반의 자동화 가능한 설계를 통해 요구 속성 명세에 대한 일반 개발자의 접근을 가능하게 했다. 본 연구에서는 속성 명세 지원 방식을 상호 보완적으로 결합하여 가장 효율적인 속성 명세 방법이 되도록 설계한다. 우선 명세의 편의와 정형성을 지키기 위해 패턴을 활용하며 패턴의 확장과 결합이 용이하도록 패턴 한정자를 추가한다. 또한 배경 지식이 적은 일반 개발자의 요구 속성 명세를 위해 유도 질문을 통한 명세 접근 방법을 설계한다. 한편, 요구 속성의 의미와 구조에 대한 효과적인 명세 및 해석을 위해 속성도 및 속성 and-or 트리, 패턴 다이어그램을 사용한다.

2. 패턴 기반의 시각적 속성 언어

패턴 기반의 시각적 속성 명세 시스템의 처리 과정은 자연어로 기술된 요구 속성을 검증자가 분석하고 결과에 맞는 패턴을 선택하여 이를 그림 2 와 같은 속성도로 기술한다. 중심 패턴에 속하는 속성 원소나 부 패턴을 결합시켜 속성 명세가 끝나면 시스템은 검증자가 원하는 명세 논리의 패턴 사상 테이블을 이용하여 실제 논리를 생성한다. 패턴을 구성하는 전체 과정은 그림 1 과 같으며 검증자는 요구 속성에 대한 질의에 응답하면서 명세를 진행하게 된다. 이는 하향식 명세이며 각 패턴 간의 상속은 속성과 패턴으로 나누어진다.

2.1 패턴 기반 시각적 속성 명세 언어(PVSL)

대분류로는 Occurrence 와 Order 가 있으며, 중 분류에는 8 개의 패턴(Absence, Universality, Bounded Existence, Existence, Response, Precedence, Chain

Response, Chain Precedence)이 있다. 속성 패턴은 모든 경로에서 만족된다면 강하다고 하며, 속성을 만족하는 경로가 존재한다면 약하다고 한다. 따라서 상태와 행동은 강함(A:all path) 과 약함(E:some path) 다시 세분된다. PVSL 은 요구 속성을 명세하기위해 속성의 구조와 관계를 패턴 관계 그래프로 표현한다.

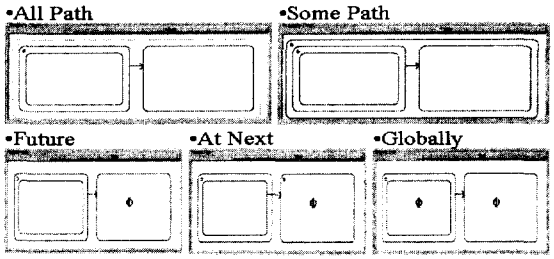


그림 1 PVSL 의 한정자

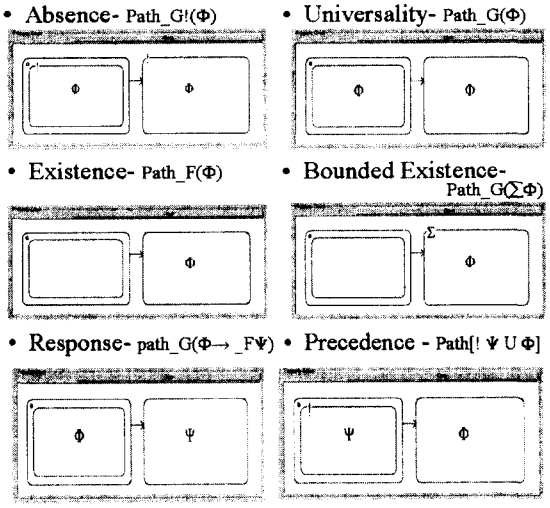
2.2 구문 규칙

PVSL 의 구문 규칙과 패턴 한정자 및 연사자는 다음과 같다.

$\Phi ::= P \mid \prod \mid \chi \mid \Xi \mid \Sigma \mid \approx \mid \Phi_1 \& \Phi_2 \mid \Phi_1 \mid \Phi_2 \mid !$

속성(Φ)은 패턴(P)과 패턴 한정자의 결합으로 정의되어진다.각 패턴은 Dwyer 의 의미 기반 패턴 분류와 같다. 하지만 속성의 표현력을 높이기 위해 모든 한정자를 패턴 한정자로 확장했다. P = { Absence, Universality, Existence, Bounded Existence, Response, Precedence, Chain Response, Chain Precedence }

표 1 패턴 기반 시각적 속성 명세 언어

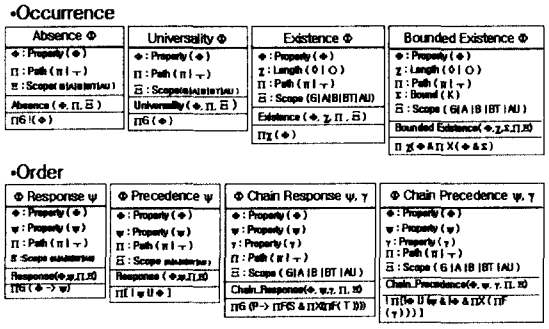


2.3. 의미

PVSL 은 패턴 한정자를 통해 표현력을 높이고 요구 속성의 구조를 쉽게 해석할 수 있도록 설계했다. 또한 Dwyer 의 속성 분류를 세분화하여 속성간의 결합을 가능하게 했다. 모든 패턴은 시스템(M)에 대하여 경로(∏)에서 만족되며 범위(Scope: Ξ) 상에서 수행된다 속성 Φ와 ψ는 항상 γ에 앞에서 만족한다.

식 Absence 와 Universality 는 시스템의 안전성을 검사하는 패턴이며 Existence 는 궁극성을 검사하는 패턴이다. Bounded Existence 는 순환성과 출현성을 나타내며 Response 와 Precedence, Chain 패턴은 공평성과 연쇄성을 검사하는 패턴이다. 각 패턴의 클래스 형태는 표 1 과 같이 정의될 수 있으며 그림 3 과 같은 패턴 연산자를 통해 결합된다. 각 패턴은 고유한 의미를 갖을 수 있는 형태이며 패턴(P)은 검증자의 세부적 요구를 변수로 받는다. 모든 패턴은 속성(Φ)과 경로(∏), 범위(Ξ)를 변수로 갖는다. 속성은 패턴(P) 혹은 조건(Φ)을 상속 받을 수 있으며 경로의 경우 강함(all path:Π)과 약함(some path:T)을 변수 값으로 갖는다.

표 2 패턴 형태 및 분류



2.4 명세 작성에 적용

일반적인 검증 과정에서 검증자는 자연어를 통한 요구 속성의 기술을 선행하고 이를 분석하여 특정 명세 논리로 기술한다. 본 연구에서는 자연어의 모호성을 피하기 위해 실제 Nu-SMV[7]에서 사용된 CTL 식을 그림 2 와 같이 속성도로 표현하고 CTL 식과 양상뮤 논리 식을 생성했다.

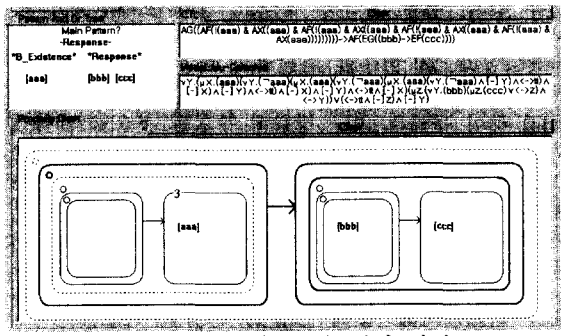


그림 2 PVSL 시스템을 통한 명세

3. 속성 명세 지원 시스템

우리의 연구는 요구 속성 명세의 단일 프레임워크 제

시와 자동화 가능한 지원 도구 개발을 목적으로 진행됐다. PVSL 과 상태도, 속성 and-or 트리, 패턴 다이어그램, 유도질문 등은 속성 명세 지원 도구를 통해 생성된다. PVSL 지원 도구의 사용자 인터페이스는 그림 3 과 같다.

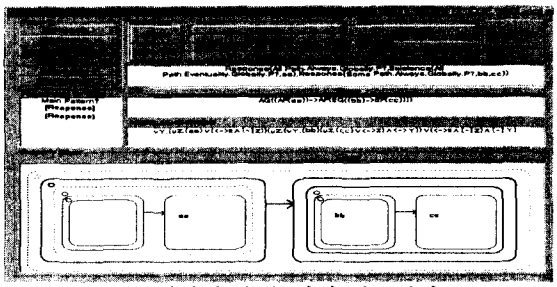


그림 3 시각적 속성 명세 시스템의 GUI

주요 메뉴는 파일을 관리는 File, 속성 명세의 정보 관리 및 갱신에 위한 Wizard, 출력 방식을 선택하는 Mode, 그리고 패턴 다이어그램으로 구성된다. 속성에 대한 주요 정보는 속성 Wizard 를 통해 입력된다. 검증자가 입력한 요구 속성 정보는 속성도, 속성 and-or 트리, 패턴 다이어그램, 논리 등의 생성 함수를 통해 처리되어 출력된다. 우리의 PVSL 지원 도구는 속성 명세 지원의 기본적인 개념을 설계한 것이며 이러한 지원 도구를 통해 검증자의 다양한 요구 사항 정보를 얻고자 한다. 이러한 요구 정보들은 향후 개발될 정식 버전의 주요 기능으로 추가될 것이다.

표 3 Nu-SMV(CTL)의 패턴 분류 결과

패턴 유형	수
Absence	40
Universally	6
Existence	58
Bounded Existence	0
Response	154
Precedence	60
Response Chain	0
Precedence Chain	0
UNKNOWN	0
Total	318

4. 기존 연구와의 비교 및 기여도

우리의 시각적 명세를 위한 패턴은 Dwyer 의 의미 기반 속성 분류 체계를 따르고 있지만 모든 명세 논리와 실제 명세의 정확한 분류를 위해 각 패턴의 제한자를 확장하고 세분화했다. 우리는 시각적 속성 명세를 위한 패턴을 통해 Nu-SMV 에서 사용된 실제 속성 명세 논리(CTL)식을 분류했다. 그 결과 160 개의 식에서 318 개의 패턴을 추출했으며 그 중 Response 패턴이 절반을 차지했다. 우리는 실제 명세 논리의 패턴 분류를 통해 패턴 분류 체계의 안전성을 확인했으며 그 결과는 다음의 표 3 과 같다. 한편, 최근에 진행된 양상 뮤 논리의 속성 명세 패턴 연구를 통해 실제 사용된 복잡한 속성 명세가 패턴 단위로 결합하는 것과 해석에 있어 패턴 포함 관계를 활용하는 것이 효율적이라는 것을 입증했다. 따라서 우리는 가장 일반적 형태를 유지하면서 확장이 가능한 형태로 패턴을 재정의했다. 시각적 속성 명세 언어를 통해 복잡한 속성간의 관계와 구조를 잘 표현하고 해석했다. 또한 패턴 다이어그램과 속성 and-or 트리는 속성간의 계층과 상호관계를 쉽게 해석할 수 있도록 한다.

5. 결론 및 향후 연구

우리의 패턴 기반 시각적 속성 명세 연구는 모든 명세 논리를 포괄하는 요구 속성 명세의 단일 프레임워크와 자동화 및 효율적인 속성 명세 방법을 제시했다. 유도 질문 방법은 효율적인 속성 명세 접근을 가능하게 하며 검증자의 요구 속성을 자연어로 기술, 이를 확인할 수 있게 했다. 또한 해석에 있어 속성도와 속성 and-or 트리, 패턴 다이어그램을 통해 속성의 의미와 구조 및 속성 원소 상속을 쉽게 이해할 수 있도록 했다. 이것은 일반 개발자의 모형 검사 접근을 용이하게 하며 패턴 기반의 시각적 속성 명세 시스템은 모형 검사 소프트웨어 개발에 기여한다. 본 논문에서는 속성의 명세 및 해석의 편이 보장과 속성의 구조 표현에 중점을 두었다. 향후 연구는 패턴 기반 시각적 속성 명세 시스템의 편의, 안전, 기능을 확장하는 것이다. 이를 위해 속성 추출 및 범위 적용, 계층적 명세에 관한 연구를 진행할 것이다.

참고문헌

[1] M.B.Dwyer, et.al, "Model Checking Generic Contai-ner Implementations", Proceedings of Dagstuhl Se-minar, Lecture Notes in Computer Science, 2000.

[2] J.C.Corbett, et.al, "A Language Framework For Expressing Checkable Properties of Dynamic Software", Proceedings of the SPIN Software Model Checking Workshop, LNCS 1885, 2000.

[3] M.B. Dwyer, et.al, "Property Specification Patterns for Finite-State Verification", Proceedings of the Workshop on Formal Methods in Software Practice, 1998.

[4] A.Del Bimbo, et.al, "Visual Specification of Branching Time Temporal Logic", In Proceedings of the 11th IEEE Symposium on Visual Lang-uages, September 1995.

[5] D.Harel, et.al, "The STATEMATE Semantics of Statecharts", ACM Transactions on Software Engineering and Methodology,5(4):293-333,October1996.

[6] R.Alur, et.al "Model Checking of Hierar-chical State Machines", Proceedings of the 6th ACM Symposium on Foundations, 1998.

[7] Nu-SMV, http://nusmv.irst.itc.it/