

# SNU 4541.664A Program Analysis

## Spring 2006

### Note 22

Prof. Kwangkeun Yi

## 소프트웨어 모델 검증 Software Model Checking

# 모델 검증 Model Checking

- “모델” = 검증할 대상 = 실행과정/의미구조
- “검증” = 요구하는 성질을 만족하는지 검증

역사적으로 모델 검증(*model checking*)은 하드웨어 검증을 목표로

- 정확한 모델은 항상 유한: 유한 오토마타(*finite automata*)
- 모델 만들기 뻘해. 상태가 많아서 그렇지:  $2^{|\text{bits}|}$
- 소프트웨어에 적용하려면? 모델 만들기가 문제거리

# 검증에 주안점

이론	모델계산	검증
요약해석	✓	
타입분석	✓	✓
제약식분석	✓	
모델검증		✓

프로그램 분석 = 프로그램 모델 계산 + 모델에 대한 검증  
 = 데이터 모으기 + 정보 추출하기

# 모델 검증 과정

- 유한한 모델 만들기(*finite modeling*)
  - 상태 전이 그래프(*state transition graph*)
  - hw: 애초부터 유한
  - sw: 무한한 의미구조를 유한하게 계산해야 (요약!)
- 검증할 성질 정의하기(*specification*)
  - 프로그램 논리(*program logic, dynamic logic, modal logic*)라는 언어를 사용
- 모델이 성질을 만족하는 지 확인하기(*verification*)
  - 그래프 따라가서 확인해 보기(*graph reachability*)
  - 요약된 모델이었다면 반쪽 결론만 가능

# 딜레마

딜레마 (프로그램 분석이 가지는 공통된)

- 소프트웨어의 유한한 모델이 필요하고
- 유한한 모델은 요약이 필수
- 검증 결과가 반쪽이 되지 않게 어떻게 요약해야?

알 수 없다

- 허위 오류로 부터 요약을 다듬는(*abstraction refinement*) 힌트를 받는다.
- 그렇게 다시 요약한 모델을 가지고 모델 검증
- 위의 과정을 반복.
- 검증되거나 진짜 오류가 찾아질 때 까지; 안 끝날 수도

# 취할점

취할점 (모델 검증 동네에서 피어오르는 연기들)

- 허위 오류로부터 자동으로 요약을 다듬는(*abstraction refinement*) 기술
- 검증할 성질을 엄밀하고 정확하게 표현하는 언어:  
프로그램 논리(*program logic*)들
- 검증에 사용되는 경제적인 데이터 구조
- 오류의 이유를 자동으로 설명하기

# 모델 검증의 성공담

- $10^{20}$ 개의 상태를 가진 모델(hw 회로) 검증: 1980년대 후반
- $10^{120}$ 개의 상태를 가진 모델(hw 회로) 검증: 1991년
- IEEE Futurebus+ cache coherence protocol 검증/오류발견: 1992년
- 2500줄 C 프로그램(로봇 제어) 검증/오류발견: 2001년
- 10K C 프로그램(CDMA 통화 처리) 검증/오류발견: 2002년



# 모델 = 의미구조 [★]

유한한 상태 전이 그래프(*labeled state-transition graph*) (크립케 구조(*Kripke structure*))라고 함) 로 정의

- 무한한 상태들 사이의 전이과정을 유한하게 계산해 놔야  
안전하게 요약하는 것이 필수
- 요약한 모델이기 때문에 반쪽 검증만 가능
- 검증 후 오류 있으면: 진짜 오류인지는 알 수 없다
- 검증 후 오류 없으면: 오류는 진짜로 없다

모델: 상태 전이 그래프(*state transition graph*) 예

## 필요한 성질 정의하기(*specification*)

프로그램 논리(*program logic, dynamic logic, modal logic*)이라는 언어를 사용

- 시간 논리(*temporal logic*): 시간 흐름을 염두한 논리식들
- 계산 트리 논리 CTL(*computational tree logic*): 상태 전이 그래프를 (무한한) 트리로 이해하면서 시간 흐름을 염두한 논리식들
  - 트리: 모든 실행 경로가 드러남

계산 트리(*computational tree*)의 예

상태전이 그래프

$$\{(a, b), (a, c), (b, a), (b, c), (c, c)\}$$

위의 그래프는 다음의 계산 트리과정으로 풀어 읽을 수 있다

$$\{(a_i, b_i), (a_i, c_i), (b_i, a_{i+1}), (b_i, c_{i+1}), (c_i, c_{i+1}) \mid i \in \mathbb{N}\}$$

# CTL(*computational tree logic*) 식의 예

$$\begin{aligned} \text{formula} &\rightarrow \text{modality (predicate)} \\ \text{modality} &= \{A, E\} \times \{G, F, X, U\} \end{aligned}$$

이고

- Does variable  $v$  remain positive?

$$M \models \text{AG}(v = \oplus)$$

- Can variable  $v$  be positive?

$$M \models \text{EF}(v = \oplus)$$

- Does variable  $v$  remain positive until  $w$  is negative?

$$M \models \text{AU}(v = \oplus, w = \ominus)$$

- "After state  $s$ , does variable  $v$  remain positive?"

계산 트리 논리식(*computational tree logic*)  $f$  검증 알고리즘의 복잡도:

$$O(|f| \times |M|)$$

$|M|$ 은 상태 전이 그래프(계산 트리로 풀지않은) 모델에서 상태와 연결선 갯수의 합

# 모델 검증 취할 기술들

- 허위 오류로부터 자동으로 요약을 다듬는(*abstraction refinement*) 기술
- 검증할 성질을 엄밀하고 정확하게 표현하는 언어: 프로그램 논리(*program logic*)들
- 검증에 사용되는 경제적인 데이터 구조: BDD(*binary decision diagram*)
- 오류의 이유를 자동으로 설명하기

읽기:

Model Checking, *E. Clarke, O. Grumberg, and D. Peled,*  
MIT Press, 2002

# 요약 다듬기(*abstraction refinement*) 기술의 예

Microsoft's SLAM system