3그룹 연구목표 소개

배경민

2023년 7월 5일

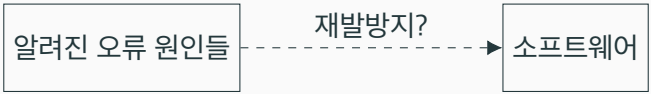
POSTECH 컴퓨터공학과

그룹3 연구목표: 소프트웨어재난 재발방지

소프트웨어

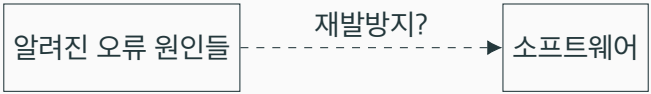
- 동일한 원인에 의해 발생하는 소프트웨어 재난의 재발을 방지

그룹3 연구목표: 소프트웨어재난 재발방지



- 동일한 원인에 의해 발생하는 소프트웨어 재난의 재발을 방지
- 연구내용1: 알려진 오류 원인들에 대한 재난오류 데이터베이스 구축

그룹3 연구목표: 소프트웨어재난 재발방지



- 동일한 원인에 의해 발생하는 소프트웨어 재난의 재발을 방지
- 연구내용1: 알려진 오류 원인들에 대한 재난오류 데이터베이스 구축
- 의문1: 코드기반? 오류 분석이 완료된 동일한 특정 소프트웨어 구현 재검증?

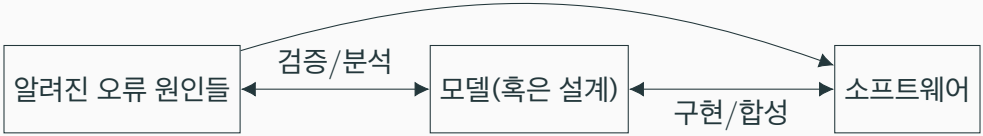
그룹3 연구목표: 소프트웨어재난 재발방지

- 동일한 원인에 의해 발생하는 소프트웨어 재난의 재발을 방지
- 연구내용1: 알려진 오류 원인들에 대한 재난오류 데이터베이스 구축
- 연구내용2: 모델(혹은 설계) 단계에서 소프트웨어 재난의 원인을 분석

그룹3 연구목표: 소프트웨어재난 재발방지

- 동일한 원인에 의해 발생하는 소프트웨어 재난의 재발을 방지
- 연구내용1: 알려진 오류 원인들에 대한 재난오류 데이터베이스 구축
- 연구내용2: 모델(혹은 설계) 단계에서 소프트웨어 재난의 원인을 분석
- 의문2: 모델이나 요구사항이 존재하지 않는 경우? 검증의 복잡성?

그룹3 연구목표: 소프트웨어재난 재발방지



- 동일한 원인에 의해 발생하는 소프트웨어 재난의 재발을 방지
- 연구내용1: 알려진 오류 원인들에 대한 재난오류 데이터베이스 구축
- 연구내용2: 모델(혹은 설계) 단계에서 소프트웨어 재난의 원인을 분석
- 연구내용3: 모델 합성, 요구사항 추론, 오류패턴 기반 모델검증 등 신기술 개발

그룹3 연구내용 개요

시스템 및 재난특성정보

제로베이스
모델합성

- 플랫폼 모델생성
- 재난중심 모델생성
- 모델정제

재난오류
데이터베이스

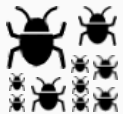
재난오류
데이터베이스화

- 오류 일반화
- 오류 데이터베이스화
- 오류스펙 생성

비정상 상황 감지/복구

재난 정형 모델

순위기반 오류식별



시스템 레벨 재난 요구사항

- 컨트랙트 재사용
- 모듈 컨트랙트 추론
- 모듈 재난원인 추론

자동스펙 생성

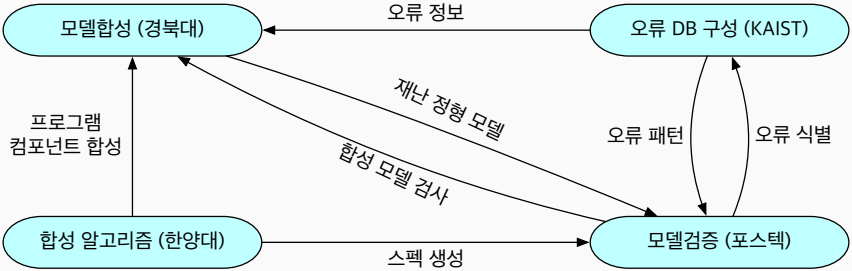
모듈별 재난
요구사항

- 오류 패턴 학습
- 패턴기반 오류탐색
- 패턴기반 요약

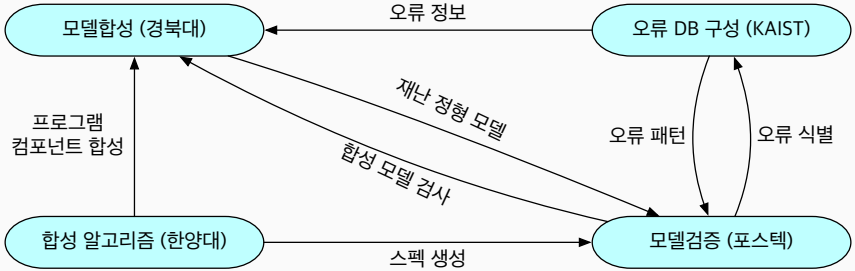
스마트
모델검증

유사 오류 방지 테스트/패치

연구실 별 연구주제 및 인터페이스

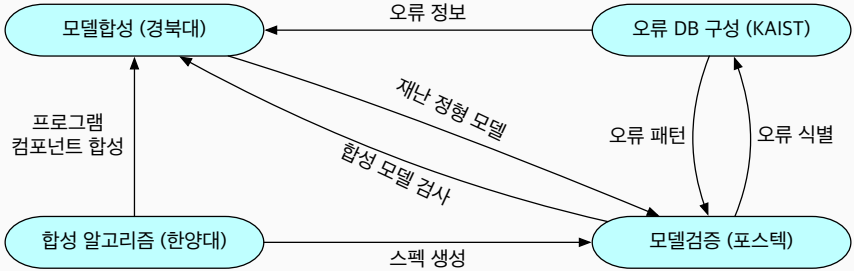


연구실 별 연구주제 및 인터페이스



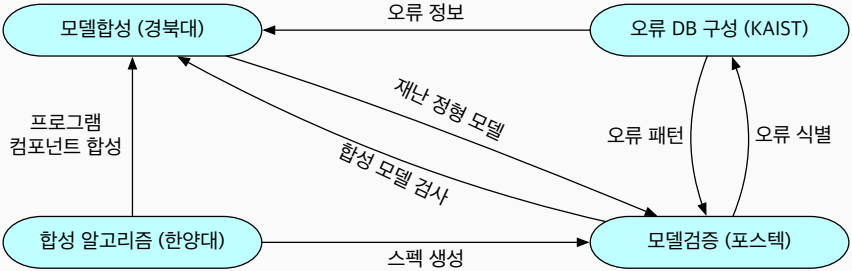
- 모델합성 ⇔ 모델검증
- 합성 알고리즘 ⇔ 모델합성, 모델검증
- 오류 데이터베이스 ⇔ 모델검증, 모델합성

연구실 별 연구주제 및 인터페이스



- 모델합성 ⇔ 모델검증
- 합성 알고리즘 ⇔ 모델합성, 모델검증
- 오류 데이터베이스 ⇔ 모델검증, 모델합성

연구실 별 연구주제 및 인터페이스



- 모델합성 ⇔ 모델검증
- 합성 알고리즘 ⇔ 모델합성, 모델검증
- 오류 데이터베이스 ⇔ 모델검증, 모델합성

POSTECH 모델검증 연구 소개

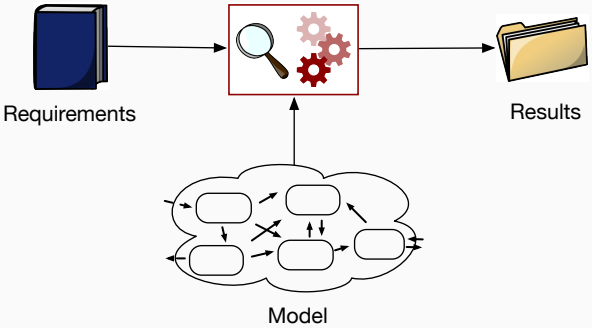
배경민

2023년 7월 5일

POSTECH 컴퓨터공학과

모델검증 (Model Checking)

- 시스템의 오류를 자동으로 찾는 기술
  - 소프트웨어/하드웨어 디자인,
    프로토콜 디자인, 소스 코드, …
  - 다양한 모델검증 도구 존재
- 특징
  - 시스템의 모든 가능한 상태를
    확인하여 “오류 없음” 증명 가능
  - 자동적으로 복잡한 성질을 검증 가능



접근방법: 논리 기반 모델검증

- 모델검증 기법 적용의 장애물
  - Algorithmic challenge: 상태 폭발 문제 (state space explosion)
  - Modeling challenge: 다양한 소프트웨어 시스템의 정형명세

접근방법: 논리 기반 모델검증

- 모델검증 기법 적용의 장애물
  - Algorithmic challenge: 상태 폭발 문제 (state space explosion)
  - Modeling challenge: 다양한 소프트웨어 시스템의 정형명세
- 논리 기반 모델검증

Model		Logic System		Verification
시스템 명세		수학적 모델		
M	$\Longrightarrow$	$\mathcal{R}_M$	$\Longrightarrow$	모델검증
성질 명세		논리식		알고리즘
$spec$	$\Longrightarrow$	φ_{spec}		

논리 기반 모델검증 도구 예제

- Boolean logic
  - CBMC, NuSMV, ⋯
- Satisfiability modulo theories (SMT)
  - nuXmv, MCMT, ⋯
- Rewriting logic
  - Maude, KEVM, RV-Predict, CafeOBJ, ⋯
- Temporal logic of actions
  - TLA+
- ⋯

논리 기반 모델검증 도구 예제

- Boolean logic
  - CBMC, NuSMV, ⋯
- Satisfiability modulo theories (SMT)
  - nuXmv, MCMT, ⋯
- Rewriting logic
  - Maude, KEVM, RV-Predict, CafeOBJ, ⋯
- Temporal logic of actions
  - TLA+
- ⋯

POSTECH 모델검증 연구 (1)

- Algorithmic challenge
  - 논리 시스템의 알고리즘 및 최적화 기법 연구: Rewriting logic 및 SMT

POSTECH 모델검증 연구 (1)

- Algorithmic challenge
  - 논리 시스템의 알고리즘 및 최적화 기법 연구: Rewriting logic 및 SMT
  - STAAR: 패턴 기반 모델검증 알고리즘 및 상태공간축소 기법

POSTECH 모델검증 연구 (1)

- Algorithmic challenge
  - 논리 시스템의 알고리즘 및 최적화 기법 연구: Rewriting logic 및 SMT
  - STAAR: 패턴 기반 모델검증 알고리즘 및 상태공간축소 기법
- Modeling challenge
  - 대상 성질/오류 및 시스템에 최적화된 모델링 및 정형명세 기술 연구

POSTECH 모델검증 연구 (1)

- Algorithmic challenge
  - 논리 시스템의 알고리즘 및 최적화 기법 연구: Rewriting logic 및 SMT
  - STAAR: 패턴 기반 모델검증 알고리즘 및 상태공간축소 기법
- Modeling challenge
  - 대상 성질/오류 및 시스템에 최적화된 모델링 및 정형명세 기술 연구
  - STAAR: 모델검증 인터페이스 개발 및 모델검증 응용

POSTECH 모델검증 연구 (2)

- 모델검증 알고리즘: 패턴을 활용한 모델검증 알고리즘 연구
  - 시스템 상태 및 오류의 패턴에 기반한 모델검증
  - 효과적인 상태공간 탐색 전략 및 추상화 기법 연구

POSTECH 모델검증 연구 (2)

- 모델검증 알고리즘: 패턴을 활용한 모델검증 알고리즘 연구
  - 시스템 상태 및 오류의 패턴에 기반한 모델검증
  - 효과적인 상태공간 탐색 전략 및 추상화 기법 연구
- 모델검증 인터페이스: 프로그래밍 언어의 의미구조 정형명세
  - PROMELA 언어의 Rewriting 기반 의미구조 정의 (번개&포스터: 손병호)
  - Multitask PLC 프로그램의 모델검증 (포스터: 이재서)

POSTECH 모델검증 연구 (2)

- 모델검증 알고리즘: 패턴을 활용한 모델검증 알고리즘 연구
  - 시스템 상태 및 오류의 패턴에 기반한 모델검증
  - 효과적인 상태공간 탐색 전략 및 추상화 기법 연구
- 모델검증 인터페이스: 프로그래밍 언어의 의미구조 정형명세
  - PROMELA 언어의 Rewriting 기반 의미구조 정의 (번개&포스터: 손병호)
  - Multitask PLC 프로그램의 모델검증 (포스터: 이재서)
- 모델검증 응용: 모델검증 사례연구 및 도구 개발
  - TEE API의 정형명세 및 모델검증 (번개&포스터: 류근열)
  - AADL 언어 기반 분산 시스템 모델검증 (포스터: 이재훈)

POSTECH 모델검증 연구 (2)

- 모델검증 알고리즘: 패턴을 활용한 모델검증 알고리즘 연구
  - 시스템 상태 및 오류의 패턴에 기반한 모델검증
  - 효과적인 상태공간 탐색 전략 및 추상화 기법 연구
- 모델검증 인터페이스: 프로그래밍 언어의 의미구조 정형명세
  - PROMELA 언어의 Rewriting 기반 의미구조 정의 (번개&포스터: 손병호)
  - Multitask PLC 프로그램의 모델검증 (포스터: 이재서)
- 모델검증 응용: 모델검증 사례연구 및 도구 개발
  - TEE API의 정형명세 및 모델검증 (번개&포스터: 류근열)
  - AADL 언어 기반 분산 시스템 모델검증 (포스터: 이재훈)
- Bonus: 사이버물리시스템 및 Deep neural network 모델검증 연구
  - Signal Temporal Logic 모델검증 연구 (포스터: 이지아)
  - DNN 검증 시 발생하는 conflict 정보를 이용한 성능 향상 기법 연구 (포스터: 채승현)
  - DNN 검증을 위한 요약 해석 기법 연구 (포스터: 연주은)

진행연구 소개: 패턴을 활용한 모델검증 알고리즘 연구

연구개요: 패턴을 활용한 모델검증

- 목적: 모델검증 시 패턴에 기반한 상태공간 탐색
  - 오류 패턴 등을 활용하여 (알려진) 오류를 효율적으로 탐색
  - 오류와 관계가 적은 상태공간을 효과적으로 요약

연구개요: 패턴을 활용한 모델검증

- 목적: 모델검증 시 패턴에 기반한 상태공간 탐색
  - 오류 패턴 등을 활용하여 (알려진) 오류를 효율적으로 탐색
  - 오류와 관계가 적은 상태공간을 효과적으로 요약
- 필요 기술
  - 패턴의 정의 및 패턴 기반 실행 방법
  - 패턴 공간의 탐색 및 요약

연구개요: 패턴을 활용한 모델검증

- 목적: 모델검증 시 패턴에 기반한 상태공간 탐색
  - 오류 패턴 등을 활용하여 (알려진) 오류를 효율적으로 탐색
  - 오류와 관계가 적은 상태공간을 효과적으로 요약
- 필요 기술
  - 패턴의 정의 및 패턴 기반 실행 방법
  - 패턴 공간의 탐색 및 요약
- 연구 방법
  - Rewriting logic: 높은 표현력을 가진 명세 언어로 다양한 모델링 언어의 의미 정의 가능
  - Rewriting logic으로 명세된 시스템의 패턴 기반 모델검증 연구

Rewriting Logic Specification

- State

  term t (algebraic data type)

- Transition

  rewrite rule $t \longrightarrow t'$ **if** ψ (patterns t and t', and condition ψ)

- Example
  - By rule $f(N) \longrightarrow f(N+1)$ **if** $N < 10$,

Rewriting Logic Specification

- State

  term t (algebraic data type)

- Transition

  rewrite rule $t \longrightarrow t'$ **if** ψ (patterns t and t', and condition ψ)

- Example
  - By rule $f(N) \longrightarrow f(N+1)$ **if** $N < 10$, term $f(5)$ is rewritten to

Rewriting Logic Specification

- State

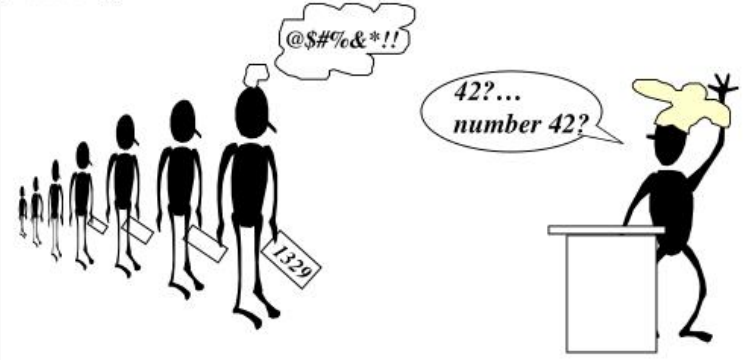
  term t (algebraic data type)

- Transition

  rewrite rule $t \longrightarrow t'$ **if** ψ (patterns t and t', and condition ψ)

- Example
  - By rule $f(N) \longrightarrow f(N+1)$ **if** $N < 10$, term $f(5)$ is rewritten to $f(5+1)$

Example: Lamport's Bakery Algorithm



- Each process receives a ticket number to enter the critical section.
- Process with the smallest ticket number enters the critical section.

Example: Lamport's Bakery Algorithm

- Each state with N processes:

$$n\ ;\ m\ ;\ [i_1, d_1] \ldots [i_N, d_N]$$

  - n: the current number in the bakery's number dispenser
  - m: the number currently served
  - $[i_l, d_l]$: process with id d_l in status $d_l \in \{\texttt{idle}, \texttt{wait}(\mathit{ticket}), \texttt{crit}(\mathit{ticket})\}$

Example: Lamport's Bakery Algorithm

- Each state with N processes:

$$n\ ;\ m\ ;\ [i_1, d_1] \ldots [i_N, d_N]$$

  - n: the current number in the bakery's number dispenser
  - m: the number currently served
  - $[i_l, d_l]$: process with id d_l in status $d_l \in \{\texttt{idle}, \texttt{wait}(\mathit{ticket}), \texttt{crit}(\mathit{ticket})\}$

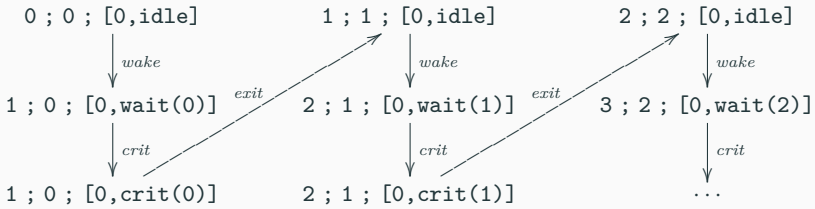
- Rewrite rules (in the Maude syntax)

```
rl [wake]: N ; M ; [K,    idle] PS  =>  N + 1 ; M ; [K,wait(N)] PS .
rl [crit]: N ; M ; [K,wait(M)] PS  =>  N ;      M ; [K,crit(M)] PS .
rl [exit]: N ; M ; [K,crit(M)] PS  =>  N ; M + 1 ; [K,    idle] PS .
```

  - Variables: `N`, `M`, `K` for numbers, and `PS` for process sets

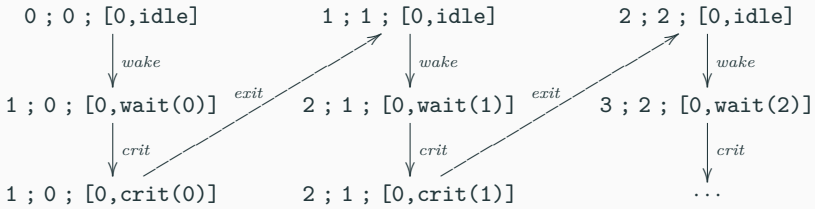
Example: Lamport's Bakery Algorithm

- Rewrite sequences with one process



Example: Lamport's Bakery Algorithm

- Rewrite sequences with one process



- Infinite-state system: unbounded counters n and m

Symbolic Representation Using Logical Terms

- Symbolic state

  t a pattern t with variables

Symbolic Representation Using Logical Terms

- Symbolic state

  $t \parallel \phi$ a pattern t with variables constrained by a formula ϕ

Symbolic Representation Using Logical Terms

- Symbolic state

  $t \parallel \phi$ a pattern t with variables constrained by a formula ϕ

- Example
  - $\texttt{N ; M ; PS} \parallel \texttt{N} \geq \texttt{M}$

Symbolic Representation Using Logical Terms

- Symbolic state

  $t \parallel \phi$ a pattern t with variables constrained by a formula ϕ

- Example
  - $\mathtt{N}\,;\,\mathtt{M}\,;\,\mathtt{PS} \parallel \mathtt{N} \geq \mathtt{M}$
- Express a set of (potentially infinitely many) concrete states
  - $0\,;\,0\,;\,[0, \mathit{idle}], \quad 1\,;\,1\,;\,[0, \mathit{idle}]\,[1, \mathit{idle}], \quad 3\,;\,2\,;\,[0, \mathit{idle}]\,[2, \mathit{wait}(0)], \quad \cdots$

Computing Symbolic State Space

Constrained narrowing $\rightsquigarrow$

- $t \parallel \phi \rightsquigarrow t' \parallel \phi'$ if an instance of $t \parallel \phi$ can be rewritten to t' by some rewrite rule

Computing Symbolic State Space

Constrained narrowing $\rightsquigarrow$

- $t \parallel \phi \rightsquigarrow t' \parallel \phi'$ if an instance of $t \parallel \phi$ can be rewritten to t' by some rewrite rule
- Defines symbolic transitions between logical terms

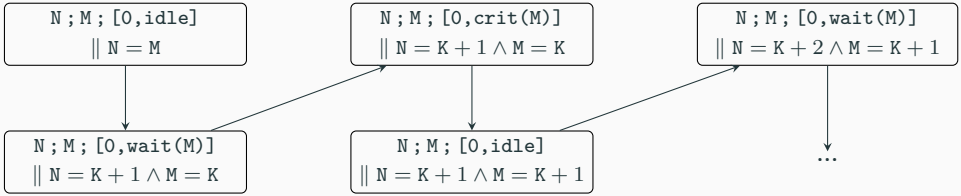
Computing Symbolic State Space

Constrained narrowing $\rightsquigarrow$

- $t \parallel \phi \rightsquigarrow t' \parallel \phi'$ if an instance of $t \parallel \phi$ can be rewritten to t' by some rewrite rule
- Defines symbolic transitions between logical terms
- Sound and complete constrained narrowing methods exist

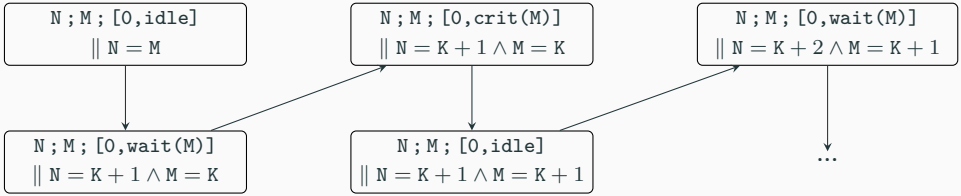
Example

- Constrained narrowing sequences with one process



Example

- Constrained narrowing sequences with one process



- Still an infinite number of symbolic states

Reduction Using Folding

- Folding preorder $\preccurlyeq$ to define more general symbolic states:

$$s_1 \preccurlyeq s_2 \iff s_2 \text{ is more general than } s_1$$

Reduction Using Folding

- Folding preorder $\preccurlyeq$ to define more general symbolic states:

$$s_1 \preccurlyeq s_2 \iff s_2 \text{ is more general than } s_1$$

- Logical terms subsumes all its instances (modulo the equations E)

$$t \parallel \psi \preccurlyeq u \parallel \phi \quad \text{if} \quad t =_E u\theta \ \text{ and } \ \psi \implies \phi\theta \ \text{ for some substitution } \theta$$

Reduction Using Folding

- Folding preorder $\preccurlyeq$ to define more general symbolic states:

$$s_1 \preccurlyeq s_2 \iff s_2 \text{ is more general than } s_1$$

- Logical terms subsumes all its instances (modulo the equations E)

$$t \parallel \psi \preccurlyeq u \parallel \phi \quad \text{if} \quad t =_E u\theta \ \text{ and } \ \psi \implies \phi\theta \ \text{ for some substitution } \theta$$

- Example

$$\texttt{N}\,;\,\texttt{M}\,;\,[\texttt{0},\texttt{idle}]\,\texttt{PS} \parallel \texttt{N} \geq \texttt{M} \quad \preccurlyeq \quad \texttt{N}\,;\,\texttt{M}\,;\,\texttt{PS} \parallel \mathit{true}$$

Reduction Using Folding

- Folding preorder $\preccurlyeq$ to define more general symbolic states:

$$s_1 \preccurlyeq s_2 \iff s_2 \text{ is more general than } s_1$$

- Logical terms subsumes all its instances (modulo the equations E)

$$t \parallel \psi \preccurlyeq u \parallel \phi \quad \text{if} \quad t =_E u\theta \ \text{ and } \ \psi \implies \phi\theta \ \text{ for some substitution } \theta$$

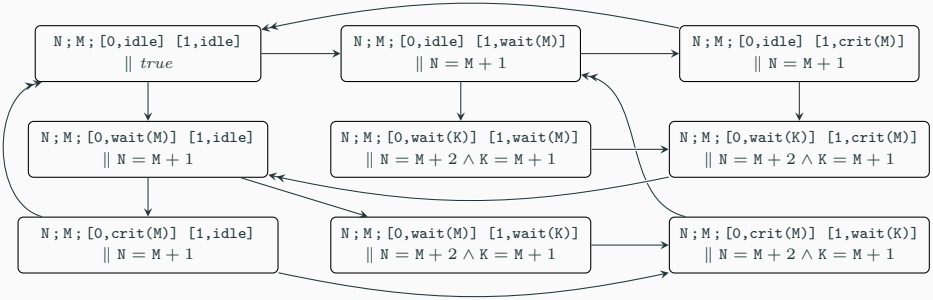
- Example

$$\mathtt{N}\,;\,\mathtt{M}\,;\,[\mathtt{0},\mathtt{idle}]\,\mathtt{PS} \parallel \mathtt{N} \geq \mathtt{M} \quad \preccurlyeq \quad \mathtt{N}\,;\,\mathtt{M}\,;\,\mathtt{PS} \parallel \mathit{true}$$

- Folding reduction
  - collapses a state s into a previously seen t such that $s \preccurlyeq t$

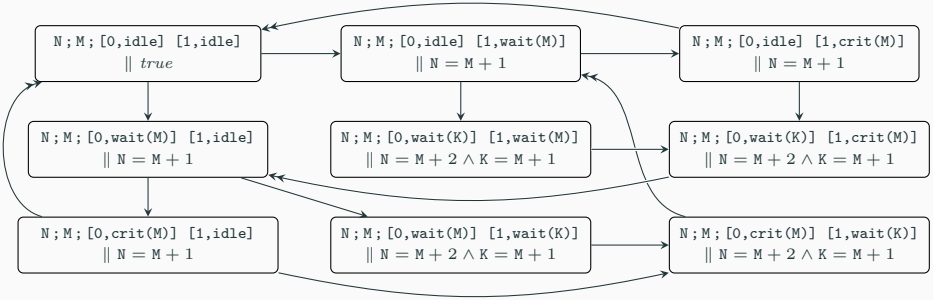
Example

- Bakery algorithm with two processes



Example

- Bakery algorithm with two processes



- Mutual exclusion can be verified for two processes!
- Can be verified for any number of processes, along with abstraction methods

Reachability Analysis using Constrained Narrowing

```
S ← {s_init};
Queue.enqueue(s_init);
while Queue ≠ ∅ do
    u ← Queue.dequeue();
    if an instance of u matches s_goal then
        return True
    foreach u′ such that u ⇝ u′ do
        if u′ is a new symbolic state then
            remove all instances of u′ from
              S and Queue;
            S ← S ∪ {u′};
            Queue.enqueue(u′)
return False
```

- Breadth-first search

Reachability Analysis using Constrained Narrowing

$S \leftarrow \{s_{init}\}$;
$Queue$.enqueue(s_{init});
while $Queue \neq \emptyset$ **do**
  $u \leftarrow Queue$.dequeue();
  if *an instance of* u *matches* s_{goal} **then**
    return True
  foreach u' *such that* $u \rightsquigarrow u'$ **do**
    if u' *is a new symbolic state* **then**
      remove all instances of u' from S and $Queue$;
      $S \leftarrow S \cup \{u'\}$;
      $Queue$.enqueue(u')
return False

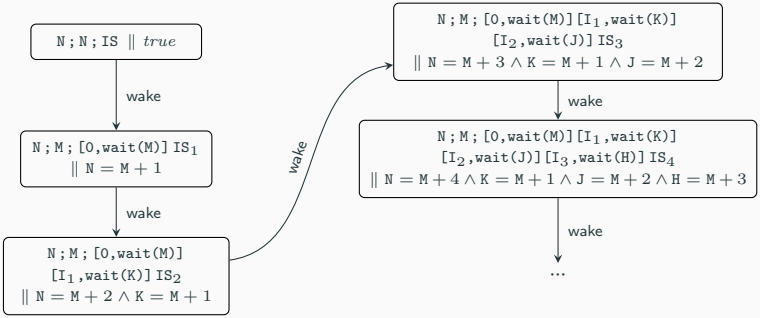
- Breadth-first search
- Refutation-complete

Limitation of the Existing Algorithm

- Bad at finding a counterexample

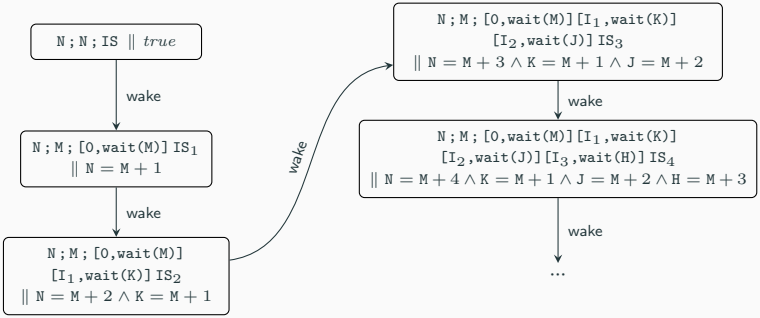
Limitation of the Existing Algorithm

- Bad at finding a counterexample
- BFS can generate a huge amount of useless symbolic states



Limitation of the Existing Algorithm

- Bad at finding a counterexample
- BFS can generate a huge amount of useless symbolic states



- # of symbolic states can increase dramatically as complexity of pattern increases

Our Approaches

- Heuristic search for (constrained) narrowing (강병지)
- Automated abstraction using pattern generalization

Heuristic Search For Narrowing-Based Reachability Analysis

```
S ← {s_init};
PriorityQ.enqueue(s_init);
while PriorityQ ≠ ∅ do
    u ← PriorityQ.dequeue();
    if an instance of u matches s_goal then
        return True
    foreach u′ such that u ⇝ u′ do
        if u′ is a new symbolic state then
            remove all instances of u′ from
              S and PriorityQ;
            S ← S ∪ {u′};
            PriorityQ.enqueue(u′)
return False
```

- Best-first search
- Priorities defined by scoring functions

Heuristic Search For Narrowing-Based Reachability Analysis

```
S ← {s_init};
PriorityQ.enqueue(s_init);
while PriorityQ ≠ ∅ do
    u ← PriorityQ.dequeue();
    if an instance of u matches s_goal then
        return True
    foreach u′ such that u ⇝ u′ do
        if u′ is a new symbolic state then
            remove all instances of u′ from
              S and PriorityQ;
            S ← S ∪ {u′};
            PriorityQ.enqueue(u′)
return False
```

- Best-first search
- Priorities defined by scoring functions

Q: Effective scoring function for narrowing?

Q: Refutation-complete?

Scoring Functions for Narrowing

- $score(t) = g(t) + h(t)$
  - a lower score indicates a higher priority

Scoring Functions for Narrowing

- $score(t) = g(t) + h(t)$
  - a lower score indicates a higher priority
- $g(t)$: the complexity of pattern t
  - e.g., the term size of t

Scoring Functions for Narrowing

- $score(t) = g(t) + h(t)$
  - a lower score indicates a higher priority
- $g(t)$: the complexity of pattern t
  - e.g., the term size of t
- $h(t)$: a heuristic estimation based on prior knowledge
  - e.g., the number of wait processes

Refutation-Completeness for Heuristic Narrowing Search

- There should be no infinite narrowing sequence with non-increasing scores

Refutation-Completeness for Heuristic Narrowing Search

- There should be no infinite narrowing sequence with non-increasing scores

Theorem

Let $Reach^{\leq c}_{\rightsquigarrow}(t) = \{u \mid t \rightsquigarrow^* u, score(u) \leq c\}$. *If* $Reach^{\leq c}_{\rightsquigarrow}(t)$ *is finite for any score* c, *then the heuristic search algorithm returns* $true$ *iff* $\mathcal{R} \vdash (\exists \mathcal{X})\ t \longrightarrow t'$.

Case Study: the OneThirdRule Consensus Algorithm

- OneThirdRule: a round-based distributed consensus algorithm

Initialization :
  $x_p \leftarrow v_p$;
Round r:
  broadcast x_p to all other nodes ; `// init`
  wait for messages from other nodes ; `// waiting`
  if $|ReceivedMsg(r)| > 2/3 \cdot N$ **then**
    $x_p \leftarrow$ the smallest most frequently received value
    if *more than* $2/3 \cdot N$ *of the received values are* x_p **then**
      `Decide` (x_p) ; `// finished`
    else
      proceed to Round $r + 1$;

- N node communicate via (possibly faulty) asynchronous message passing

Case Study: the OneThirdRule Consensus Algorithm

- OneThirdRule: a round-based distributed consensus algorithm

Initialization :
  $x_p \leftarrow v_p$;
Round r:
  broadcast x_p to all other nodes ; `// init`
  wait for messages from other nodes ; `// waiting`
  if $|ReceivedMsg(r)| > 2/3 \cdot N$ **then**
    $x_p \leftarrow$ the smallest most frequently received value
    if *more than* $2/3 \cdot N$ *of the received values are* x_p **then**
      `Decide` (x_p) ; `// finished`
    else
      proceed to Round $r + 1$;

- N node communicate via (possibly faulty) asynchronous message passing
- The algorithm uses $2/3$ as a decision threshold

OneThirdRule: Narrowing-Based Reachability Analysis

- Agreement requirement

  All nodes will agree on the same value when the algorithm terminates

OneThirdRule: Narrowing-Based Reachability Analysis

- Agreement requirement

  All nodes will agree on the same value when the algorithm terminates

- Try to find a counterexample of the agreement condition
  - with different decision threshold $1/2$ for any number of initial nodes

OneThirdRule: Narrowing-Based Reachability Analysis

- Agreement requirement

  All nodes will agree on the same value when the algorithm terminates

- Try to find a counterexample of the agreement condition
  - with different decision threshold $1/2$ for any number of initial nodes

- Failed to find a counterexample using the BFS-based algorithm

Depth bound		1	2	3	4	5	6
No folding	time (s)	0,1	1,1	487,2	3 014,0	-	-
	# state	3	39	341	4 723	-	-
Folding	time (s)	0,1	1,1	484,8	1 450,0	32 858,7	-
	# state	3	24	114	430	1 504	-

OneThirdRule: Experimental Evaluation (1)

- Comparing with the existing method (`vu-narrow`)
  - with different number of components and different thresholds
- Different scoring functions

$score(t) =$	Complexity $g(t)$ + Heuristic Estimate $h(t)$
$score_{BFS}(t) =$	$depth(t)$
$score_g(t) =$	$\#nodes(t)$
$score_1(t) =$	$10 \cdot \#nodes(t) + -9 \cdot \#nodesWithDecision(t) + \sum_{node \in nodes(t)} \max(1, round(node))$
$score_2(t) =$	$10 \cdot \#nodes(t) + -\#nodesWithDecision(t) + \sum_{node \in nodes(t)} \max(1, round(node))$
$score_3(t) =$	$100 \cdot \#nodes(t) + -\#messages(t) + 10 \cdot \sum_{node \in nodes(t)} \max(1, round(node))$

OneThirdRule: Experimental Evaluation (2)

Model			vu-narrow	heuristic-narrow				
n	Θ			$score_{BFS}$	$score_g$	$score_1$	$score_2$	$score_3$
any	$\frac{1}{2} \cdot n$	time (s)	-	-	52 304,9	110,8	391,6	466,4
		depth	-	-	24	24	24	21
		# state	-	-	13 821	565	1 679	2 150
	1	time (s)	-	-	11,7	5,1	5,1	4,1
		depth	-	-	7	7	7	6
		# state	-	-	57	38	38	36
3	$\frac{1}{2} \cdot n$	time (s)	2 325,3	33 788,0	35,4	37,1	36,3	3 076,5
		depth	17	17	24	24	24	17
		# state	N/A	40 550	1 132	1 132	1 132	7 629
	1	time (s)	10,5	28,9	0,4	0,5	0,4	2,0
		depth	6	6	9	9	9	9
		# state	N/A	923	42	42	42	56

OneThirdRule: Experimental Evaluation (2)

Model			vu-narrow	heuristic-narrow			
n	Θ			$score_{BFS}$	$score_g$	$score_1$	$score_2$	$score_3$
any	$\frac{1}{2} \cdot n$	time (s)	-	-	52 304,9	110,8	391,6	466,4
		depth	-	-	24	24	24	21
		# state	-	-	13 821	565	1 679	2 150
	1	time (s)	-	-	11,7	5,1	5,1	4,1
		depth	-	-	7	7	7	6
		# state	-	-	57	38	38	36
3	$\frac{1}{2} \cdot n$	time (s)	2 325,3	33 788,0	35,4	37,1	36,3	3 076,5
		depth	17	17	24	24	24	17
		# state	N/A	40 550	1 132	1 132	1 132	7 629
	1	time (s)	10,5	28,9	0,4	0,5	0,4	2,0
		depth	6	6	9	9	9	9
		# state	N/A	923	42	42	42	56

OneThirdRule: Experimental Evaluation (2)

Model			vu-narrow	heuristic-narrow				
n	Θ			$score_{BFS}$	$score_g$	$score_1$	$score_2$	$score_3$
any	$\frac{1}{2} \cdot n$	time (s)	-	-	52 304,9	110,8	391,6	466,4
		depth	-	-	24	24	24	21
		# state	-	-	13 821	565	1 679	2 150
	1	time (s)	-	-	11,7	5,1	5,1	4,1
		depth	-	-	7	7	7	6
		# state	-	-	57	38	38	36
3	$\frac{1}{2} \cdot n$	time (s)	2 325,3	33 788,0	35,4	37,1	36,3	3 076,5
		depth	17	17	24	24	24	17
		# state	N/A	40 550	1 132	1 132	1 132	7 629
	1	time (s)	10,5	28,9	0,4	0,5	0,4	2,0
		depth	6	6	9	9	9	9
		# state	N/A	923	42	42	42	56

Ongoing and Future Work

- Rewriting logic으로 명세된 시스템의 패턴 기반 모델검증
  - symbolic state-space exploration using constrained narrowing
  - Algorithmic challenge: 상태공간 폭발문제
- 패턴기반 모델검증 성능향상을 위한 접근방법
  - 우선순위에 따른 Best-first search
  - Automated generalization using pattern generalization (not in this talk)

Ongoing and Future Work

- Rewriting logic으로 명세된 시스템의 패턴 기반 모델검증
  - symbolic state-space exploration using constrained narrowing
  - Algorithmic challenge: 상태공간 폭발문제
- 패턴기반 모델검증 성능향상을 위한 접근방법
  - 우선순위에 따른 Best-first search
  - Automated generalization using pattern generalization (not in this talk)
- 진행 연구
  - 알려진 오류 및 이전 검증결과를 바탕으로 scoring function 합성
  - SMT를 활용한 패턴 기반 모델검증의 적용 범위 확대

Ongoing and Future Work

- Rewriting logic으로 명세된 시스템의 패턴 기반 모델검증
  - symbolic state-space exploration using constrained narrowing
  - Algorithmic challenge: 상태공간 폭발문제
- 패턴기반 모델검증 성능향상을 위한 접근방법
  - 우선순위에 따른 Best-first search
  - Automated generalization using pattern generalization (not in this talk)
- 진행 연구
  - 알려진 오류 및 이전 검증결과를 바탕으로 scoring function 합성
  - SMT를 활용한 패턴 기반 모델검증의 적용 범위 확대
- 향후 계획
  - 지진경보시스템 테스트 베드의 Promela 모델에 대한 패턴 기반 모델검증 적용
  - 패턴 기반 모델검증에 “재난오류 데이터베이스”를 활용하는 방안 연구

Thank you!