

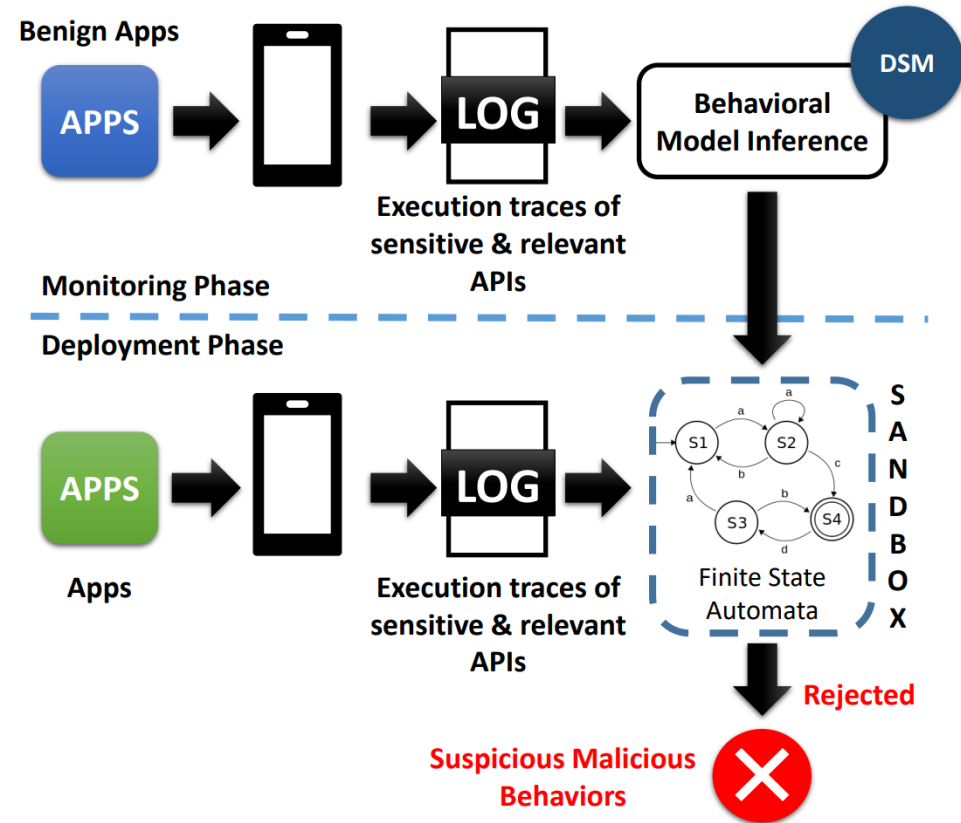
AutoCL: 동적 심볼릭 매퍼 기반 심볼릭 오토마타 능동학습

김요엘, 최윤자
경북대학교 컴퓨터학부
소프트웨어안전공학연구실

2026.02.03 SW재난센터워크숍

동기: 오토마타는 왜 필요한가

- 오토마타: 시스템의 복잡한 행동을 상태와 전이로 요약한 행동 모델
- 오토마타의 활용:
 - 프로그램 스펙 생성 [FSE'18]
 - 모델링 및 검증 [PLDI'25]
 - 모델 기반 테스트 생성 [ICSE'22]
- 목표: 오토마타를 자동으로,
효율적으로 학습



[1] Deep Specification Mining, FSE 2018

[2] Active Learning of Symbolic NetKAT Automata, PLDI 2025

[3] MOREST: Model-based RESTful API Testing with Execution Feedback, ICSE 2022

배경: 심볼릭 오토마타 학습

- 심볼릭 오토마타: 변수들의 값을 관계식(predicate)으로 일반화하여 학습

실행 트레이스 (st : 이전 상태, st' : 다음 상태)

i	st	x_1	x_2	st'
1	<i>off</i>	1	1	<i>on</i>
2	<i>on</i>	1	2	<i>on</i>
3	<i>on</i>	2	3	<i>on</i>
4	<i>on</i>	3	5	<i>off</i>
5	<i>off</i>	5	8	<i>off</i>

트레이스 그룹 ($st = off$)

st	x_1	x_2	st'
<i>off</i>	1	1	<i>on</i>
<i>off</i>	5	8	<i>off</i>

합성 결과: $x_1 = x_2$

트레이스 그룹 ($st = on$)

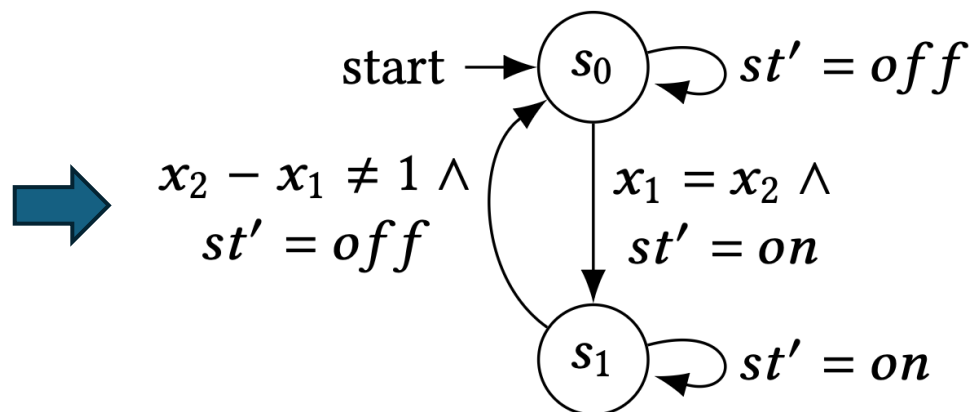
st	x_1	x_2	st'
<i>on</i>	1	2	<i>on</i>
<i>on</i>	2	3	<i>on</i>
<i>on</i>	3	5	<i>off</i>

합성 결과: $x_2 - x_1 \neq 1$

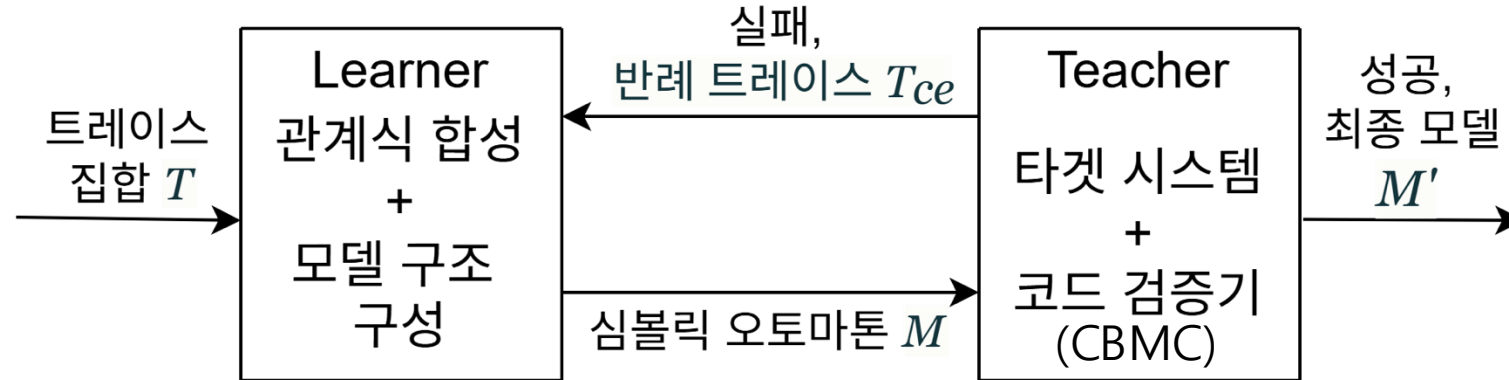
일반화된 트레이스

$1 : x_1 = x_2 \wedge st' = on$
 $2 : st' = on$
 $3 : st' = on$
 $4 : x_2 - x_1 \neq 1 \wedge st' = off$
 $5 : st' = off$

심볼릭 오토마톤



SOTA: 심볼릭 오토마타 능동학습



• Teaching 알고리즘:

1. **반례 후보 선택**: 코드에는 존재하나, 현재 모델 M 에서는 누락된 행동을 반례 후보로 선택
2. **도달 가능성 검사**: 코드 검증기로 그 후보가 시스템에서 실제로 가능한 실행인지 확인
 - 가짜 반례의 경우: 해당 후보를 배제하고 다른 후보를 탐색
 - 진짜 반례의 경우: Learner에게 전달
3. 모든 전이에 대해, 진짜 반례를 찾을 때까지 반복

SOTA의 한계: 두 가지 병목

- Learner의 병목:
 - 매 teaching 이후, 관계식 합성을 반복 (기존 관계식을 버리고 완전히 **처음부터 재합성**)
 - 변수 수/ 값 도메인 및 수집된 트레이스 수가 커질 수록 **합성 탐색공간이 급증**
- Teacher의 병목:
 - 반례 후보들에 대해 **도달 가능성 검사**를 반복 수행해야 함
 - 변수 수 / 값 도메인이 커질 수록 **가능한 반례 후보 조합이 폭발**

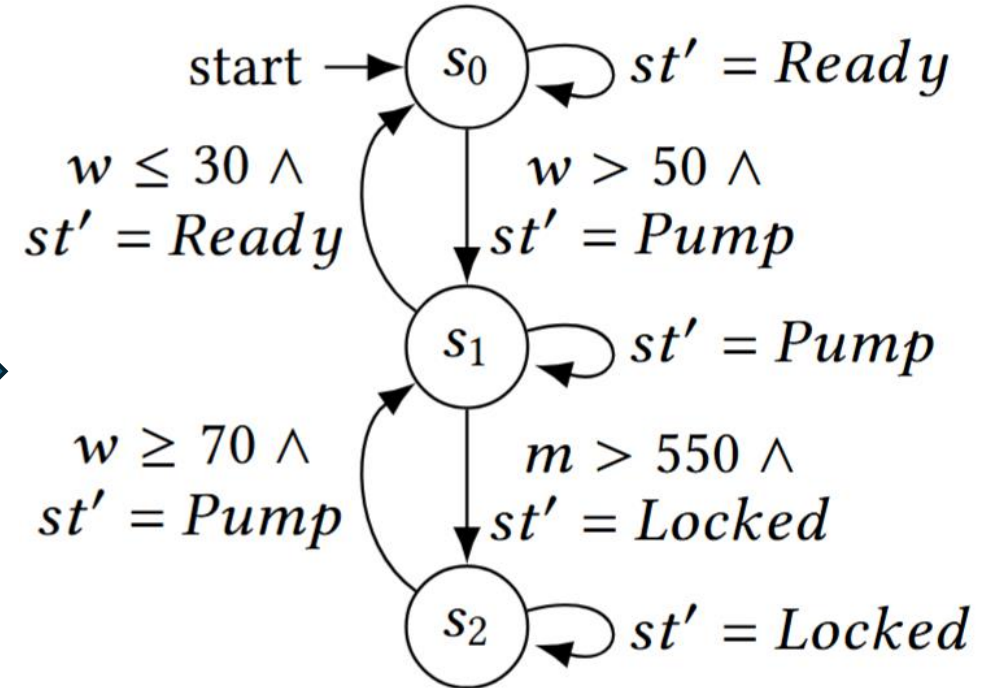
Learner의 병목 예시 (Mine Pump 예제)

초기 실행 트레이스

i	st	w	m	a	h'	st'
1	Ready	31	548	F	0	Ready
2	Ready	73	596	F	0	Pump
3	Pump	28	548	F	1	Ready
4	Ready	75	588	F	0	Pump
5	Pump	72	544	F	1	Pump
6	Pump	68	610	F	2	Locked
7	Locked	67	550	T	1	Locked
8	Locked	71	540	F	0	Pump



학습된 심볼릭 오토마톤



- w (water)와 m (methane)의 넓은 값 도메인 때문에, Predicate 합성 난이도 상승
- Teacher에 의해 반례 트레이스가 쌓일 수록 Predicate 합성 난이도 상승

Teacher의 병목 예시

전이 $s_0 \rightarrow s_1$ 을 $\text{teaching}(st = \text{Ready})$ 에서 $w \geq 50$ 이면, 다음 상태는 항상 *Pump* 인가?)

```
1 : while true do :  
2 :   w := ReadWater()  
3 :   m := ReadMethane()  
  
4 :   if st = Ready  
5 :     h := 0  
6 :     if (m < 600) ∧ (w ≥ 70)  
7 :       st := Pump
```

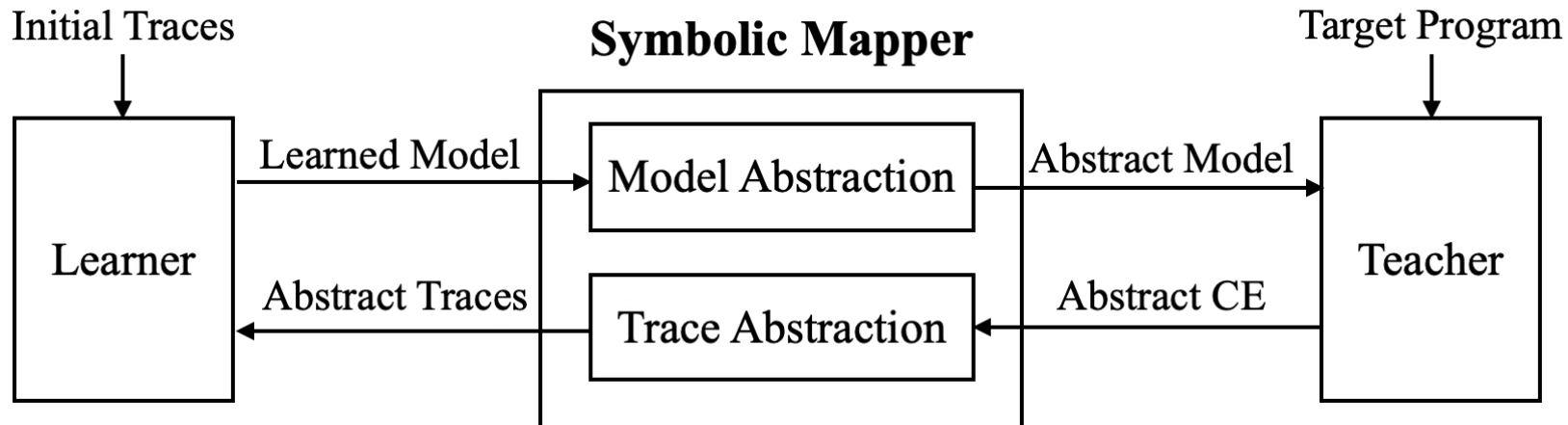
```
8 :   else if st = Pump  
9 :     h := h + 1  
10 :    if (m ≥ 600)  
11 :      st := Locked; a := true  
12 :    else if (w ≤ 30)  
13 :      st := Ready  
14 :    else if (h ≥ 3) ∧ (w ≤ 50)  
15 :      st := Ready
```

```
16 :   else if st = Locked  
17 :     if (h > 0)  
18 :       h := h - 1  
19 :     if (¬a)  
20 :       st := Pump  
21 :     if (m ≤ 550)  
22 :       a := false
```

- 반례 후보 ($st = \text{Ready} \wedge w = 51 \wedge m = 500 \wedge a = \text{true} \wedge h' = 0 \wedge st' = \text{Ready}$)는 위 코드에서 도달 불가능
- 원인: *Ready* 상태에서는 a 값이 항상 *false*이나, 반례 후보에서는 *true*로 가정
- w 와 m 의 넓은 값 도메인 때문에, 위와 같은 원인으로 도달 불가능한 반례 후보들이 많음

(정적) 심볼릭 매퍼

- 핵심: (넓은 값 도메인을 가진) 변수들을 **Boolean Variable (BV)**들로 맵핑
 - BV 예: $b_1 \Leftrightarrow (m < 600)$, $b_2 \Leftrightarrow (w \geq 70)$
- **트레이스 추상화**: 원본 트레이스를 BV 벡터값으로 이루어진 추상 트레이스로 맵핑
- **모델 추상화**: 원본 오토마톤의 Predicate을 BV들의 논리 연산식으로 맵핑



트레이스 추상화

- 심볼릭 매퍼가 다음과 같은 8개의 BV들을 분기 조건식에서 추출했다고 가정

$b_1: (w \leq 30), b_2: (w \leq 50), b_3: (w \geq 70), b_4: (m \leq 550),$

$b_5: (m < 600), b_6: (a = \text{true}), b_7: (h' > 0), b_8: (h' \geq 3)$

실행 트레이스

i	st	w	m	a	h'	st'
1	<i>Ready</i>	31	548	<i>F</i>	0	<i>Ready</i>
2	<i>Ready</i>	73	596	<i>F</i>	0	<i>Pump</i>
3	<i>Pump</i>	28	548	<i>F</i>	1	<i>Ready</i>
4	<i>Ready</i>	75	588	<i>F</i>	0	<i>Pump</i>
5	<i>Pump</i>	72	544	<i>F</i>	1	<i>Pump</i>
6	<i>Pump</i>	68	610	<i>F</i>	2	<i>Locked</i>
7	<i>Locked</i>	67	550	<i>T</i>	1	<i>Locked</i>
8	<i>Locked</i>	71	540	<i>F</i>	0	<i>Pump</i>



**Symbolic
Mapper**



추상 트레이스

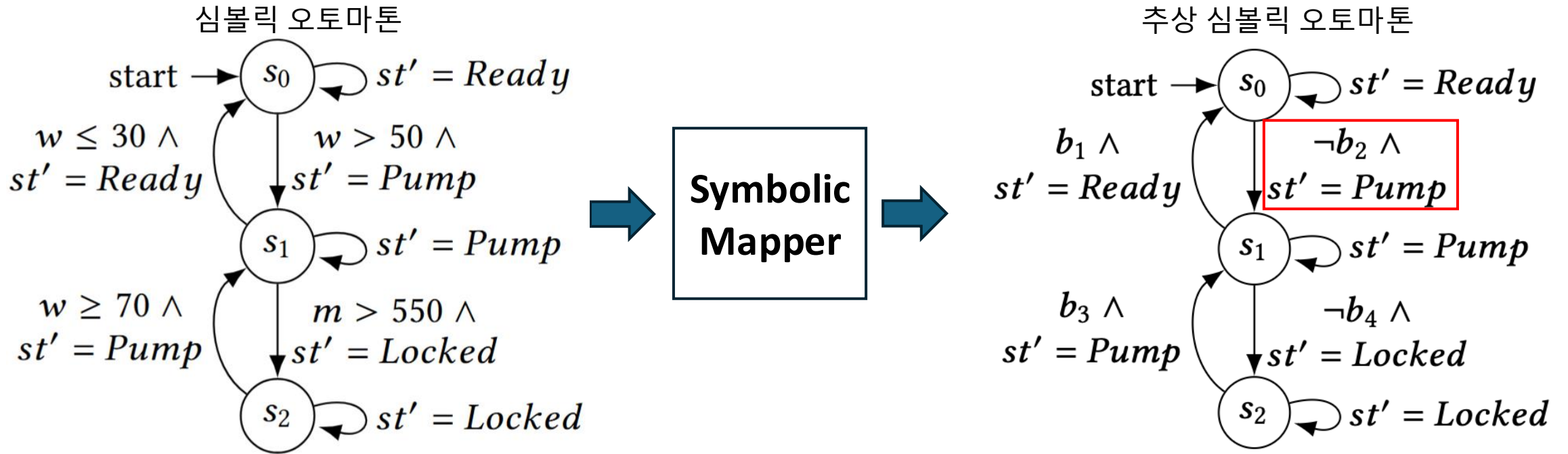
i	st	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	st'
1	<i>Ready</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>Ready</i>
2	<i>Ready</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>Pump</i>
3	<i>Pump</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Ready</i>
4	<i>Ready</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>Pump</i>
5	<i>Pump</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Pump</i>
6	<i>Pump</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Locked</i>
7	<i>Locked</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>Locked</i>
8	<i>Locked</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>Pump</i>

Learner 입장: BV들 간의 논리 연산 (예: and, or, not)만으로도 관계식 합성 가능

모델 추상화

추상 반례 후보 예시 (전이 $s_0 \rightarrow s_1$ 을 teaching 한다고 가정):

- $(st = Ready \wedge b_1 \wedge \neg b_2 \wedge \dots \wedge b_8 \wedge st' = Ready)$



Teacher 입장: BV 벡터 조합만 탐색하면 됨
(예: BV가 n 개면 최대 2^n 개의 추상 반례 후보)

정적 심볼릭 매퍼의 문제점

- (언젠가 필요하지만) 현재 모델/트레이스에서 필요 없는 BV들이 존재함
- 이렇게 사용되지 않는 BV로 인해, 학습 성능이 감소함
- Teaching worst-case 복잡도 = (iteration 횟수) $\times$ (모델의 전이 수) $\times 2^{(\text{BV 개수})}$
- Mine pump 예제:
 - Iteration 1: $B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8\}$; 전이 수 = 7; $\therefore 7 \times 2^8 = 1792$ 번의 검사
 - Iteration 2: $B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8\}$; 전이 수 = 7; $\therefore 7 \times 2^8 = 1792$ 번의 검사
 - Iteration 3: $B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8\}$; 전이 수 = 7; $\therefore 7 \times 2^8 = 1792$ 번의 검사
 - **최악 5376번의 도달 가능성 검사**

핵심 아이디어 1: 동적 심볼릭 매퍼

- **Why 동적?** 현재 모델/트레이스에 알맞게 BV들을 동적으로 정제
 - **모델 기반 BV 정제:** 현재 학습된 오토마타에서 사용된 BV들로만 BV 집합을 구성
 - **반례 기반 BV 정제:** 현재 수집된 반례를 구분할 수 있는 최소한의 BV만 추가

핵심 아이디어 1: 동적 심볼릭 맵퍼

- **Why 동적?** 현재 모델/트레이스에 알맞게 BV들을 동적으로 정제
 - **모델 기반 BV 정제:** 현재 학습된 오토마타에서 사용된 BV들로만 BV 집합을 구성
 - **반례 기반 BV 정제:** 현재 수집된 반례를 구분할 수 있는 최소한의 BV만 추가
- Iteration 1:
 - 모델 기반 BV 정제: $B_{Ready} = \{b_2\}, B_{Pump} = \{b_1, b_4\}, B_{Locked} = \{b_3\}$
 - Teaching 복잡도: $(2 \times 2^1) + (3 \times 2^2) + (2 \times 2^1) = 20$

핵심 아이디어 1: 동적 심볼릭 매퍼

- **Why 동적?** 현재 모델/트레이스에 알맞게 BV들을 동적으로 정제
 - **모델 기반 BV 정제:** 현재 학습된 오토마타에서 사용된 BV들로만 BV 집합을 구성
 - **반례 기반 BV 정제:** 현재 수집된 반례를 구분할 수 있는 최소한의 BV만 추가
- Iteration 1:
 - 모델 기반 BV 정제: $B_{Ready} = \{b_2\}, B_{Pump} = \{b_1, b_4\}, B_{Locked} = \{b_3\}$
 - Teaching 복잡도: $(2 \times 2^1) + (3 \times 2^2) + (2 \times 2^1) = 20$
- Iteration 2:
 - 반례 기반 BV 정제: $B_{Ready} = \{b_2, b_5\}, B_{Pump} = \{b_1, b_4, b_5, b_8\}, B_{Locked} = \{b_3, b_7\}$
 - 모델 기반 BV 정제: $B_{Ready} = \{b_2, b_5\}, B_{Pump} = \{b_1, b_5, b_8\}, B_{Locked} = \{b_7\}$
 - Teaching 복잡도: $(2 \times 2^2) + (3 \times 2^3) + (2 \times 2^1) = 36$

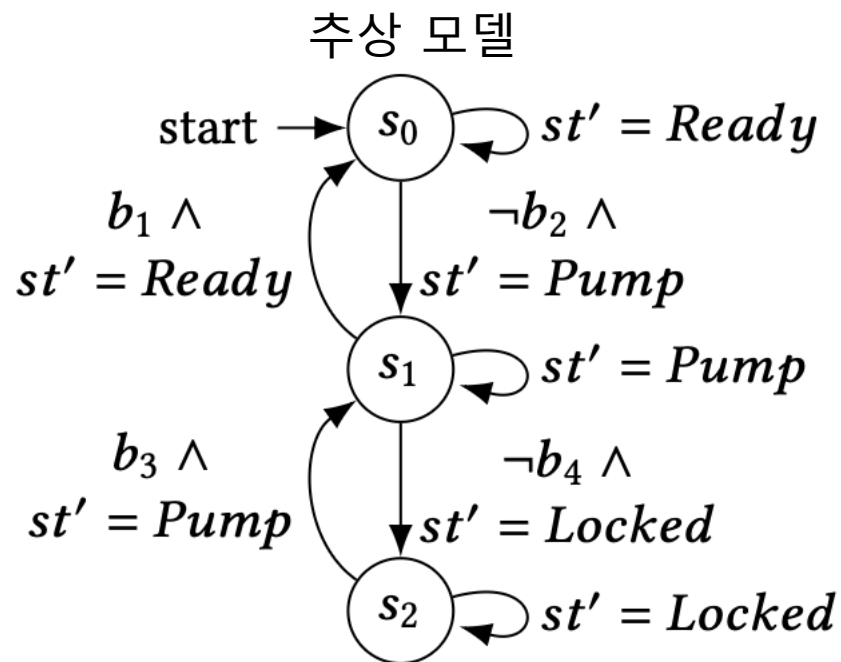
핵심 아이디어 1: 동적 심볼릭 매퍼

- **Why 동적?** 현재 모델/트레이스에 알맞게 BV들을 동적으로 정제
 - **모델 기반 BV 정제:** 현재 학습된 오토마타에서 사용된 BV들로만 BV 집합을 구성
 - **반례 기반 BV 정제:** 현재 수집된 반례를 구분할 수 있는 최소한의 BV만 추가
- Iteration 1:
 - 모델 기반 BV 정제: $B_{Ready} = \{b_2\}, B_{Pump} = \{b_1, b_4\}, B_{Locked} = \{b_3\}$
 - Teaching 복잡도: $(2 \times 2^1) + (3 \times 2^2) + (2 \times 2^1) = 20$
- Iteration 2:
 - 반례 기반 BV 정제: $B_{Ready} = \{b_2, b_5\}, B_{Pump} = \{b_1, b_4, b_5, b_8\}, B_{Locked} = \{b_3, b_7\}$
 - 모델 기반 BV 정제: $B_{Ready} = \{b_2, b_5\}, B_{Pump} = \{b_1, b_5, b_8\}, B_{Locked} = \{b_7\}$
 - Teaching 복잡도: $(2 \times 2^2) + (3 \times 2^3) + (2 \times 2^1) = 36$
- Iteration 3: $B_{Ready} = \{b_3, b_5\}, B_{Pump} = \{b_1, b_2, b_5, b_8\}, B_{Locked} = \{b_7\}$
 - **최악 $20 + 36 + 60 = 116$ 번의 도달 가능성 검사만 진행하면 됨**

모델 기반 BV 정제

- 추상 모델에서 st 값 종류별로, 실제로 사용된 BV들로만 집합을 구성
- 예: $B_{Ready} = \{b_2\}$, $B_{Pump} = \{b_1, b_4\}$, $B_{Locked} = \{b_3\}$
- 추상 반례 후보 예시: $(st = Pump \wedge \neg b_1 \wedge \neg b_4 \wedge st' = Pump)$

최악 4번의 도달 가능성 검사만 필요



반례 기반 BV 정제

- **트레이스 충돌:** 같은 st 및 BV 벡터 값을 가지지만, 서로 다른 st' 값이 있는 현상

st	b_1	b_4	st'
<i>Pump</i>	<i>T</i>	<i>T</i>	<i>Ready</i>
<i>Pump</i>	<i>F</i>	<i>T</i>	<i>Pump</i>
<i>Pump</i>	<i>F</i>	<i>F</i>	<i>Locked</i>
<i>Pump</i>	<i>F</i>	<i>F</i>	<i>Pump</i>

← 새롭게 찾은 추상 반례

- 충돌된 트레이스만을 최소한으로 구분할 수 있는 새로운 BV 합성

st	w	m	a	h'	st'
<i>Pump</i>	68	610	<i>F</i>	2	<i>Locked</i>
<i>Pump</i>	68	599	<i>F</i>	2	<i>Pump</i>

전체 중 극히 일부의 트레이스만 사용
합성 결과: $b_5 := (m < 600)$

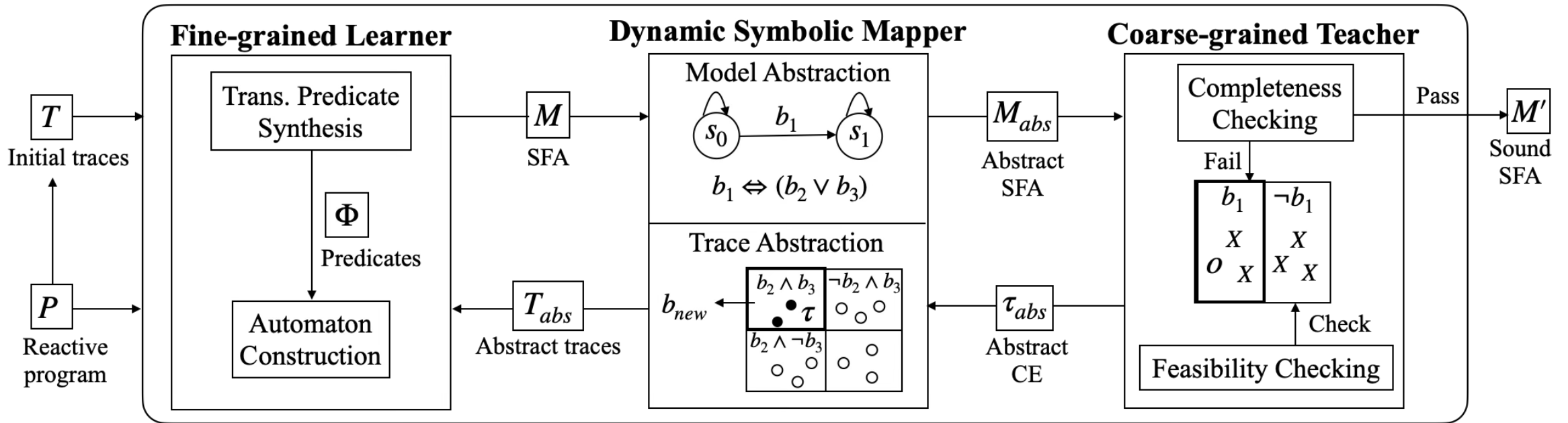
st	b_1	b_4	b_5	st'
<i>Pump</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Ready</i>
<i>Pump</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>Pump</i>
<i>Pump</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>Locked</i>
<i>Pump</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Pump</i>

정제된 BV 집합으로 추상화된 트레이스

핵심 아이디어 2: 세분도 교대 전략

- Teaching: BV 개수가 적음 → 도달 가능성 검사 횟수 감소 → Teaching 시간 감소
- Learning: BV 개수가 많음 → 반례 기반 BV 정제에 필요한 예제 수 감소 → Learning 시간 감소
- 이를 쉽게 조절할 수 있는 방법 중 하나는 **BV의 세분도(granularity)**을 조절하는 것

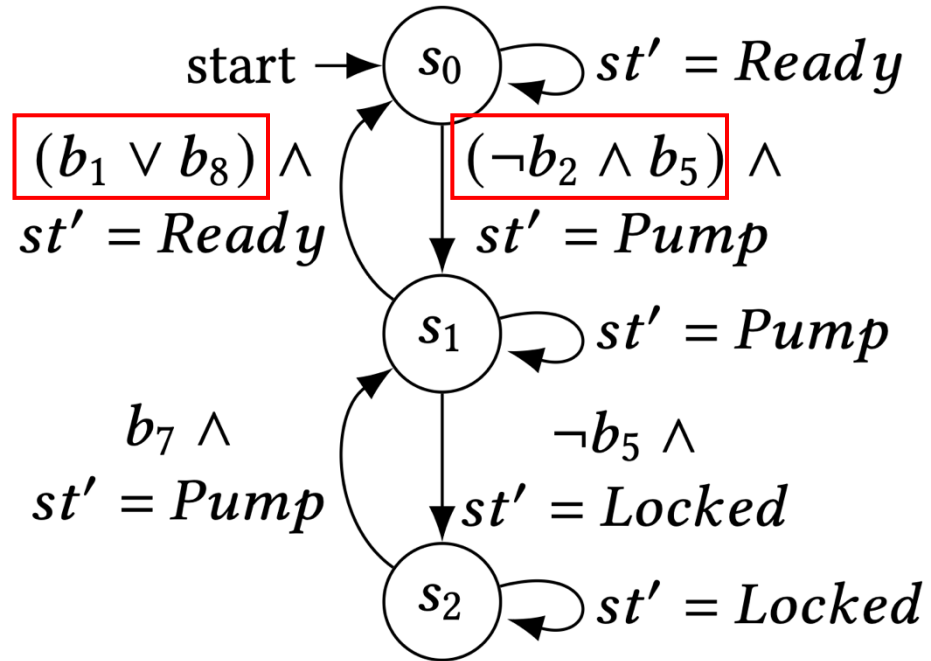
AutoCL



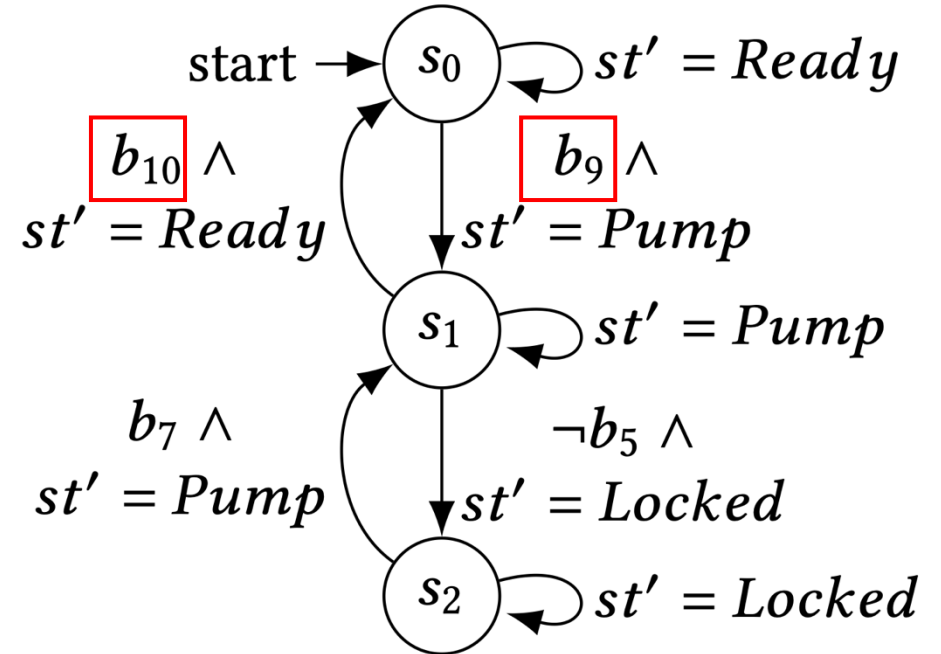
Coarse-grained Teaching

- 모델 기반 BV 정제 시 coarse-grained BV들을 생성
 - 예: 새로운 BV $b_9 := (\neg b_2 \wedge b_5)$ 및 $b_{10} := (b_1 \vee b_8)$ 을 생성
 - 정제된 BV 집합: $B_{Ready} = \{b_9\}$, $B_{Pump} = \{b_5, b_{10}\}$, $B_{Locked} = \{b_7\}$

추상 모델 (Fine-grained BVs)



추상 모델 (Coarse-grained BVs)



Fine-grained Learning

- 반례 기반 BV 정제 직전, 다시 BV들을 fine-grained하게 나눔
 - 기존 BV 집합: $B_{Ready} = \{b_9\}$, $B_{Pump} = \{b_5, b_{10}\}$, $B_{Locked} = \{b_7\}$
 - Fine-grained BV 집합: $B_{Ready} = \{b_2, b_5\}$, $B_{Pump} = \{b_1, b_5, b_8\}$, $B_{Locked} = \{b_7\}$

추상 트레이스 (Coarse-grained BVs)

st	b_5	$b_{10} (b_1 \vee b_8)$	st'
<i>Pump</i>	<i>T</i>	<i>T</i>	<i>Ready</i>
<i>Pump</i>	<i>T</i>	<i>F</i>	<i>Pump</i>
<i>Pump</i>	<i>F</i>	<i>F</i>	<i>Locked</i>
<i>Pump</i>	<i>T</i>	<i>F</i>	<i>Pump</i>
<i>Pump</i>	<i>T</i>	<i>T</i>	<i>Ready</i>
<i>Pump</i>	<i>T</i>	<i>T</i>	<i>Locked</i>

BV 합성에 3개의 반례가 필요

추상 트레이스 (Fine-grained BVs)

st	b_1	b_5	b_8	st'
<i>Pump</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>Ready</i>
<i>Pump</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Pump</i>
<i>Pump</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>Locked</i>
<i>Pump</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Pump</i>
<i>Pump</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>Ready</i>
<i>Pump</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>Locked</i>

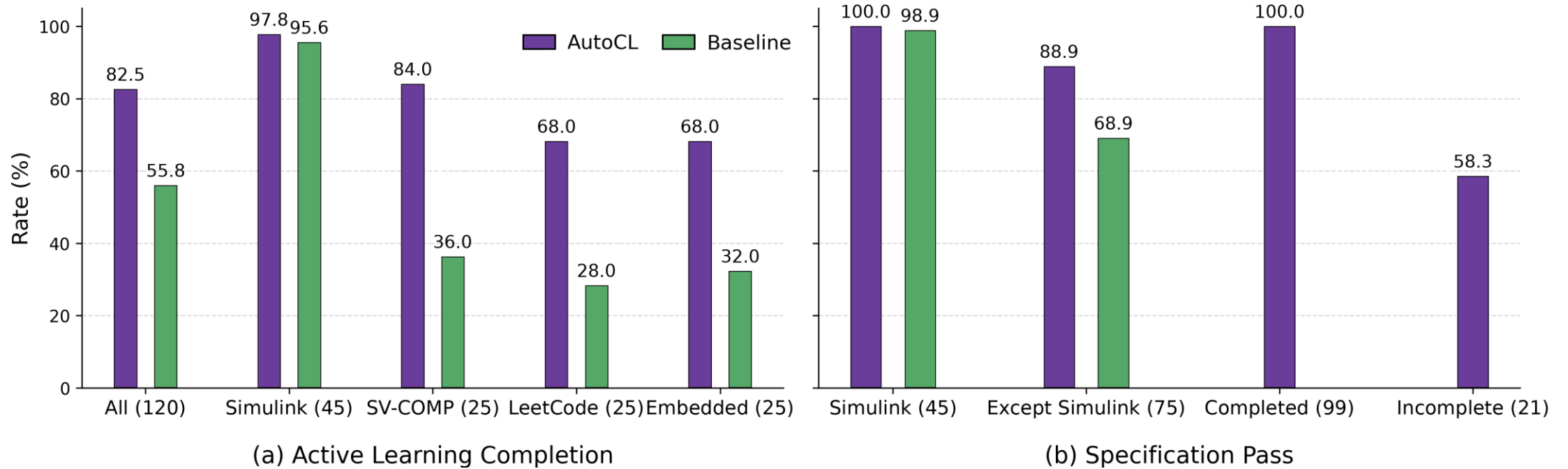
오직 2개의 반례만 필요

실험

- RQ1: AutoCL vs. Baseline
- RQ2: Dynamic vs. Static symbolic mappers
- RQ3: Coarse- vs. Fine-grained AutoCL variants
- 벤치마크: 총 120개 C 프로그램
 - Simulink: Baseline 논문에서 사용된 예제 (총 45개)
 - SV-Comp, LeetCode (각각 25개)
 - Embedded: 엘리베이터, 차량, 로봇, 인공위성, 드론 제어 예제 (총 25개)

Benchmark	$ P $	# Nest	LoC	$ Dom(t) $	$ X $	# Bool	# Enum	# Num
Simulink	45	5	236	3.7	7.0	1.2	4.3	1.4
SV-COMP	25	5	1,526	4.6	34.8	1.6	11.0	22.2
LeetCode	25	17	186	5.2	25.9	4.4	1.5	20.0
Embedded	25	16	1,242	5.0	19.1	1.3	3.5	14.3
Avg.			704	4.5	19.2	2.0	5.0	12.3

RQ1: AutoCL vs. Baseline



- AutoCL은 총 99개, Baseline은 67개 예제 제한시간 (3시간) 내 학습 성공
- Simulink외 나머지 벤치마크들의 성공률은 **73.3% (AutoCL) vs. 32% (Baseline)**
- 학습된 오토마타로 프로그램 스펙 (assertion 형태)을 생성하여 검사한 결과, 학습 성공한 99개 예제의 경우 모든 스펙이 각 프로그램에서 통과

RQ2: Dynamic vs. Static Mappers

- 정적 매퍼를 구현하여 실험한 결과, 120개 예제 중 21개에서 **트레이스 충돌**로 인해 실패
- 21개의 충돌 케이스가 모두 성공한다고 해도, **동적 매퍼의 99개에는 못 미침 ($63 + 21 = 84$)**
- **평균 34.4개의 BV** 수로 인해, 동적 매퍼 (평균 6.5개)에 비해 teaching 성능이 낮음을 확인
- 반면, 동적 매퍼의 **BV 정제 오버헤드**는 평균 34초 정도로 낮음을 확인

Mapper	Completed	Trace conflict	Learn	Teach	# BV	$\mathcal{A}(\mathcal{M})$	$\mathcal{A}(\mathcal{T})$	Refine($\mathcal{M}$)	Refine($\mathcal{T}$)	Overhead
AutoCL	86	–	121.8s	1,646.8s	6.5/1.5	$< 0.1s$	6.1s	8.5s	19.7s	34.4s
Static-Mapper	63	21/120	4.1s	3,991.4s	34.4	$< 0.1s$	$< 0.1s$	–	–	$< 0.1s$

† These statistics exclude trace-conflict cases (excluded: 21, remaining: 99).

RQ3: Coarse- vs. Fine-grained

- **AutoCL**: Fined-grained learning & Coarse-grained teaching
- **AutoCL-Coarse**: Coarse-grained learning & teaching
- **AutoCL-Fine**: Fine-grained learning & teaching

Variant	Completed	# Iter	# BV (learn/teach)	Learn	Obs. usage	BV synth (#/time)	Teach	$ \mathcal{T}_{inf} $
AutoCL	99	36.6	7.7/1.7	298.1s	14.8%	15.4/65.5s	1,977.2s	31.2
AutoCL-Coarse	96	31.6	11.2/1.7	847.8s	49.5%	30.4/664.8s	1,592.7s	29.4
AutoCL-Fine	94	21.4	9.1/8.7	293.3s	17.2%	10.7/86.4s	2,495.3s	627.1

- Obs. usage: 전체 트레이스 수 대비, BV 합성에 사용된 트레이스 수의 비율 (평균)
- $|\mathcal{T}_{inf}|$: 도달 불가능한 반례 후보 개수 (평균)

진행 중인 작업 (FSE'26 Major Revision)

1. Embedded 벤치마크 추가 실험
 - 완료. 기존 5개 예제에서 25개까지 확장
2. 정적 매퍼 vs. 동적 매퍼 비교 실험
 - 완료. Codex을 이용하여 최대한 빠르게 정적 매퍼 구현
3. 처음부터 끝까지 동작하는 running example 작성
 - 완료. Mine Pump 예제를 사용
4. 알고리즘 soundness 증명
5. Related work 보강