믿고 쓸만한 AI 에이전트를 위하여

허 기 홍
KAIST 전산학부

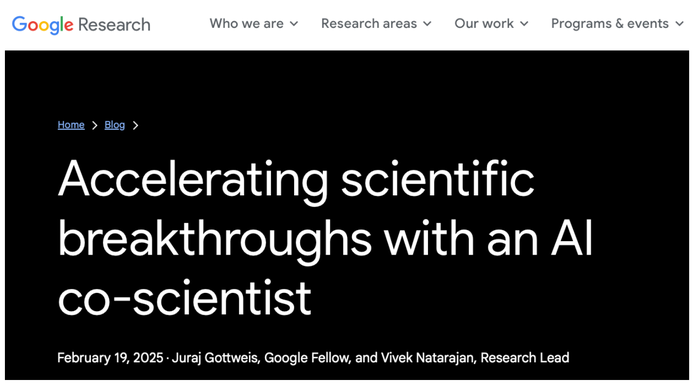
공동연구: 이동재, 이정재, 최치훈, 임영민, 위재영, 오상은, 이선재, 신인식

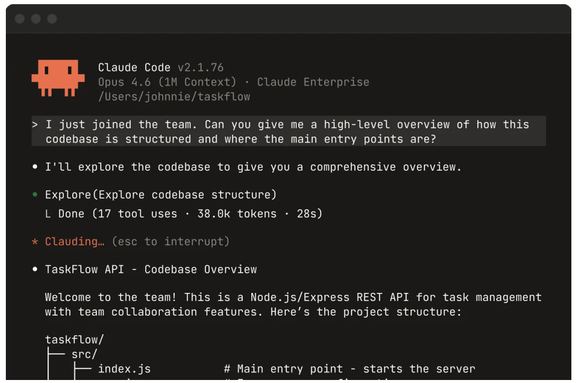
KAIST

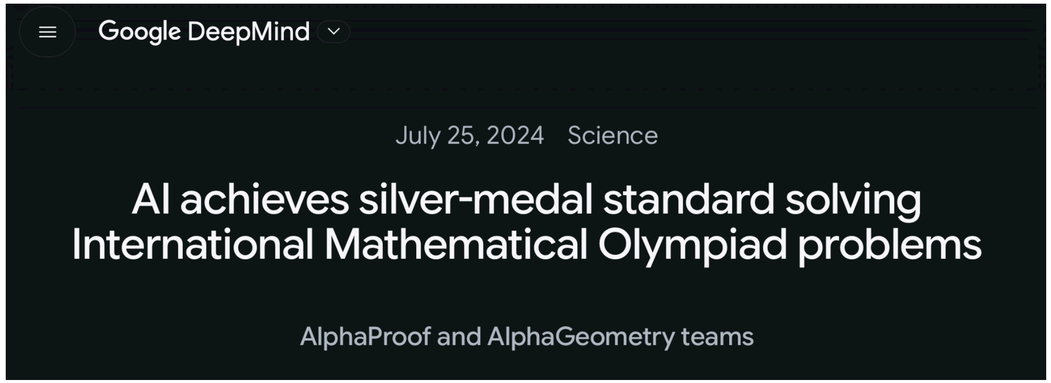
Programming Systems Laboratory

대 에이전트 시대

- 인간 수준 성능으로 자율 실행하는 AI
- 적용분야: SW 구동, 프로그래밍, 수학 증명, 과학적 발견 등





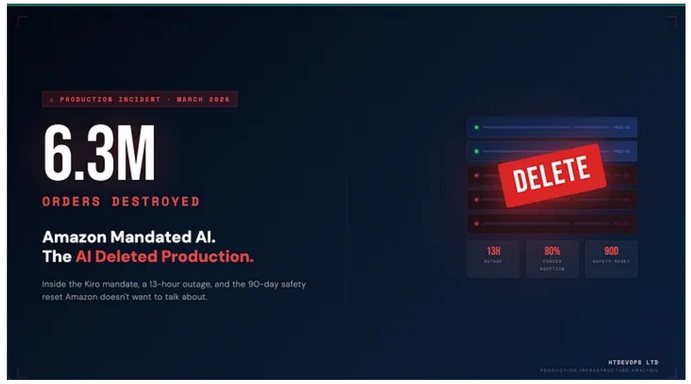


문제: 신뢰성

McDonald's ends AI experiment after drive-thru ordering blunders

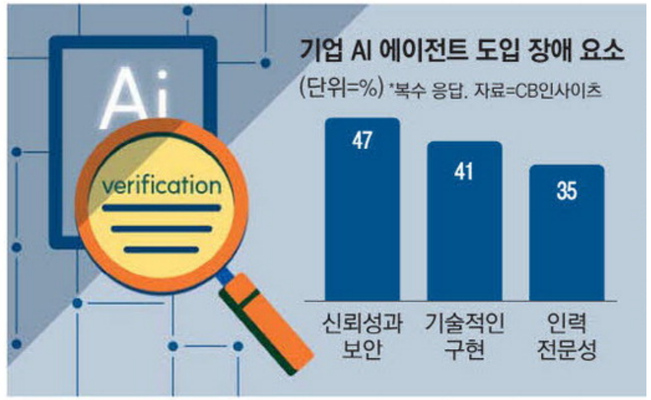
After working with IBM for three years to leverage **AI to take drive-thru orders**, McDonald's called the whole thing off in June 2024. The reason? A slew of social media videos showing confused and **frustrated customers** trying to get the AI to understand their orders.

McDonald (2024)

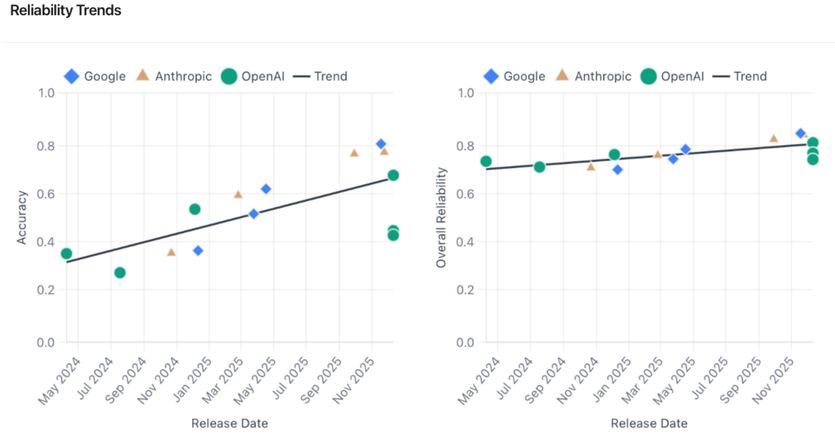


Amazon (2025)

Meta (2026)

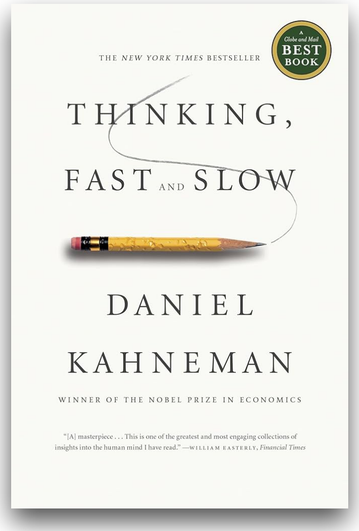


매일경제 (2025)

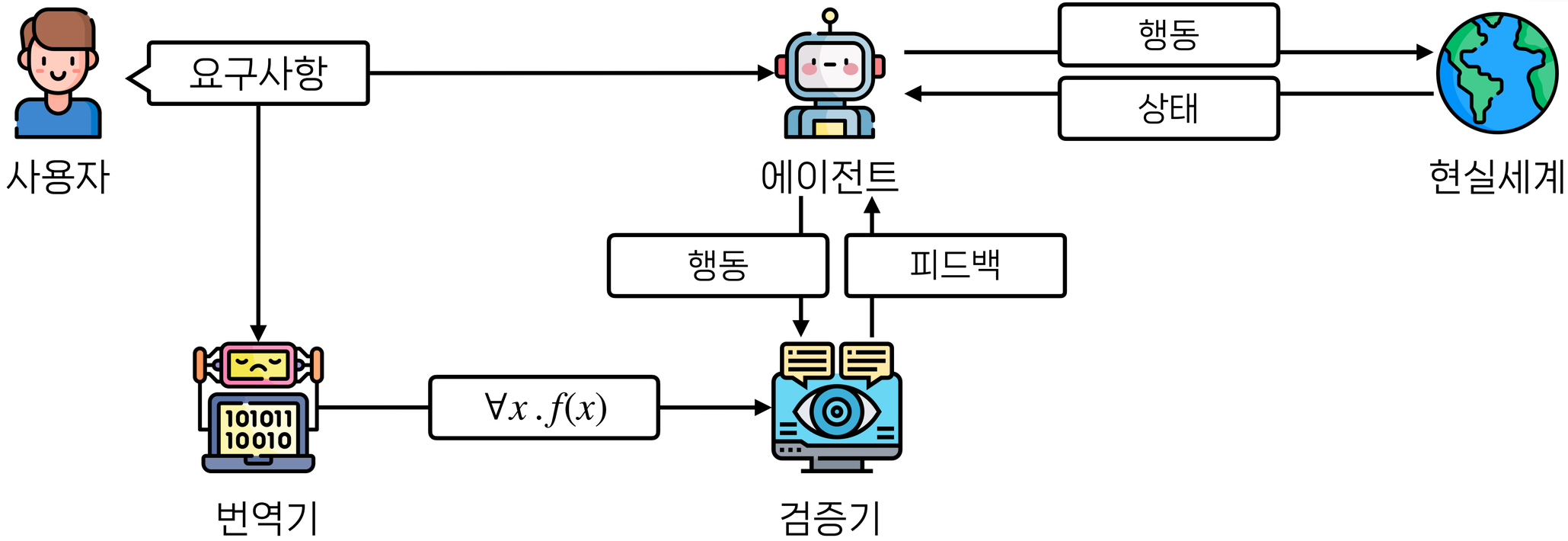


프린스턴대 (2026)

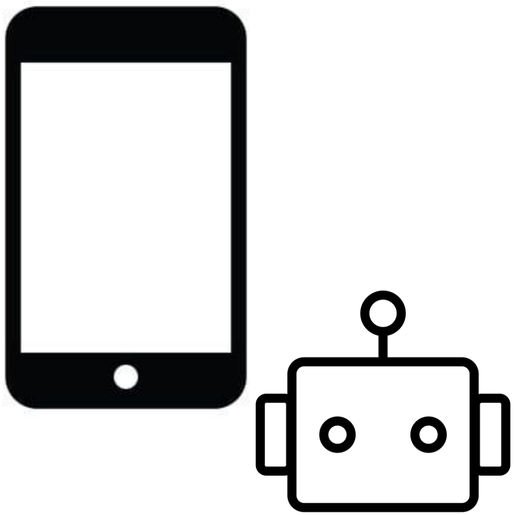
해결책: 논리 + 직관 융합 AI 에이전트



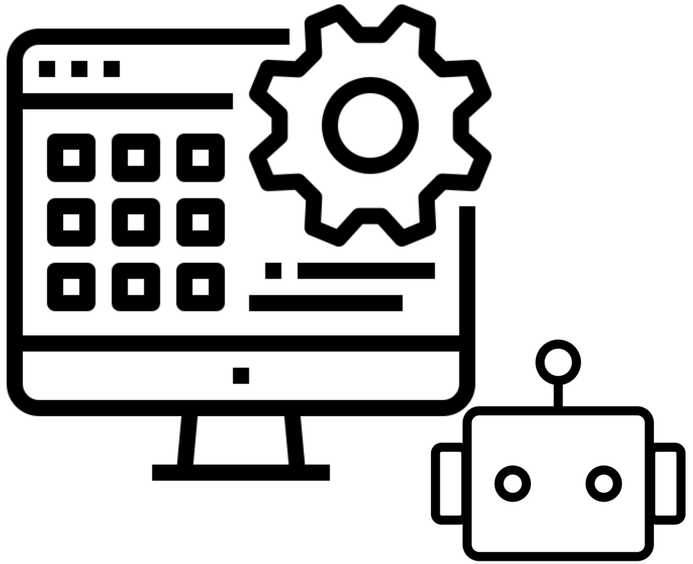
- 뛰어난 직관을 지닌 AI를 통제할 수단?
- 인간이 원하는 대로 AI가 행동하는지 엄밀히 논리 검증: PL, 전산논리 분야의 축적된 노하우
- 이른바, 논리-직관 융합(neurosymbolic) AI: “시스템 I” + “시스템 II”



두 가지 적용 사례



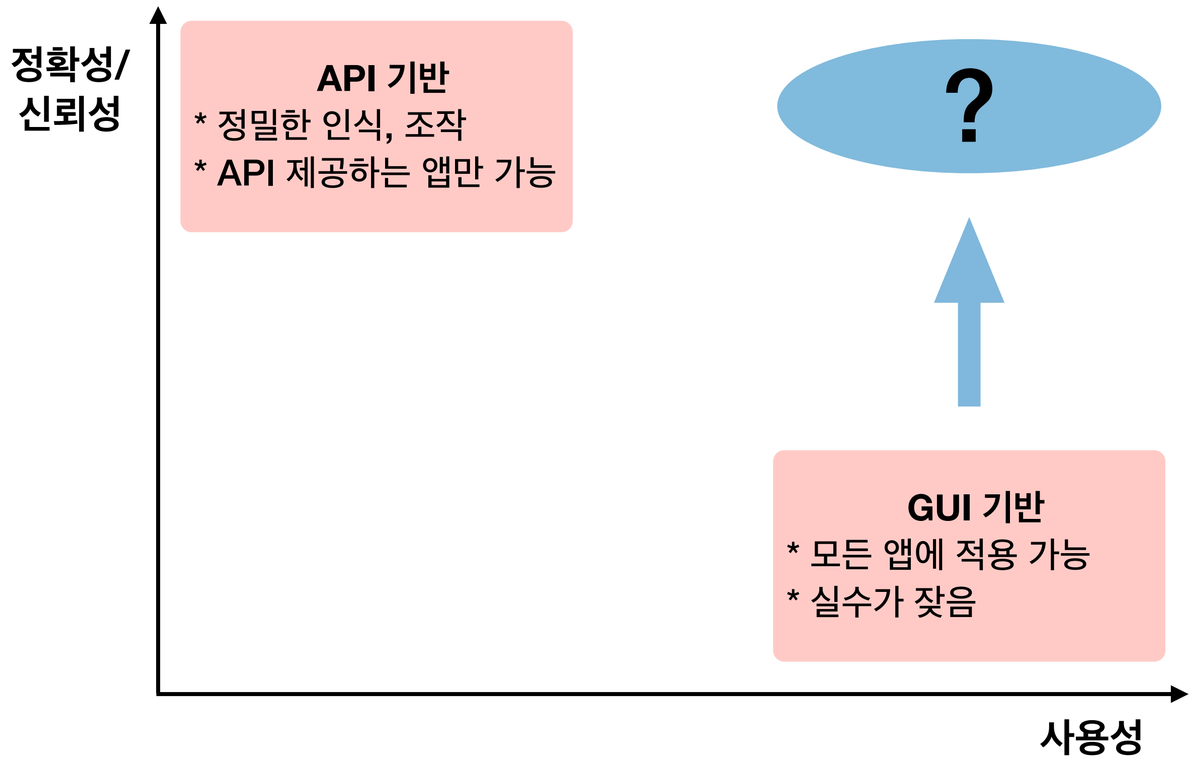
VeriSafe Agent:
믿을만한 모바일 AI 에이전트
[Mobicom'25]



Expecto:
믿을만한 코딩 AI 에이전트
[PLDI'26]

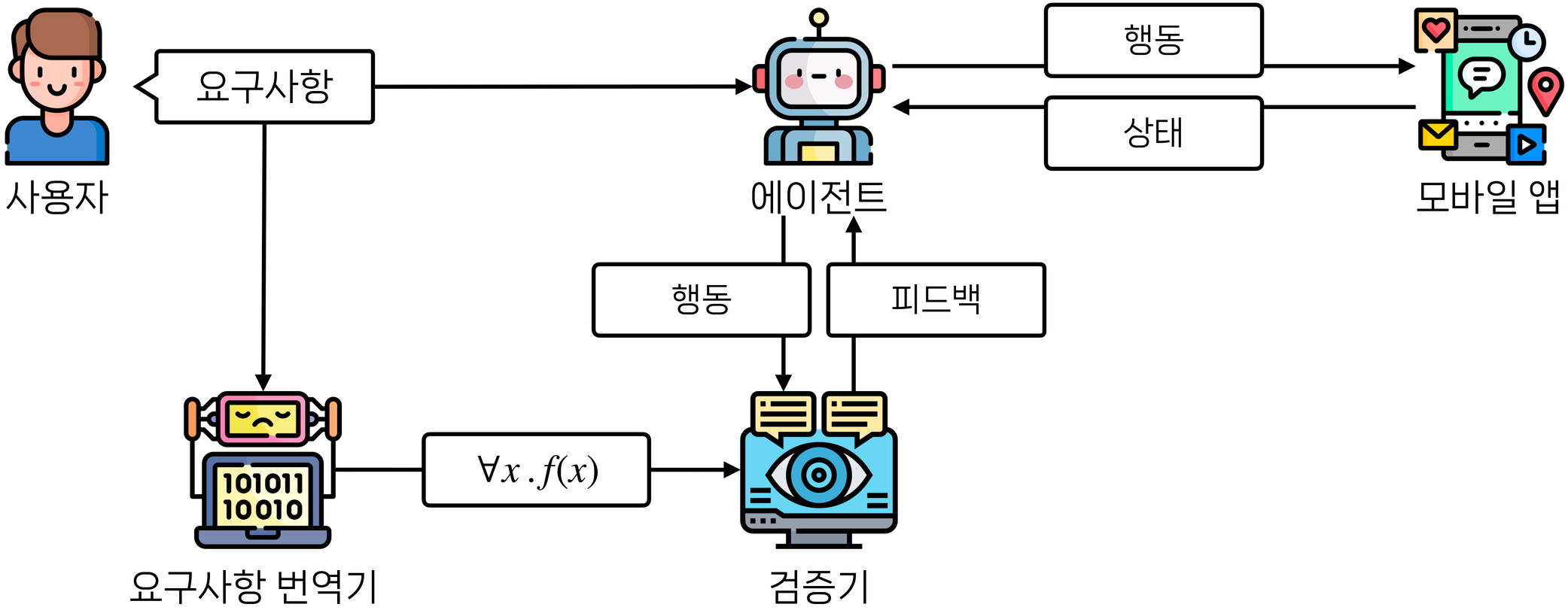
모바일 AI 에이전트

- 사람의 지시에 따라 모바일 앱 자율 실행
- 크게 API 기반, GUI 기반으로 나뉨



GUI 에이전트를 위한
논리기반 행동 검증?

VeriSafe Agent: 논리 기반 에이전트 행동 검증



성능 요약

- 검증 정확도 **38.2%** 
- 복잡한 작업 수행 능력 **130%** 
- 검증 비용 **96%** 

예시: 비행기 표 예약하기 (1)

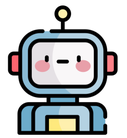
사용자

11월 4일에 출발해서 **11월 9일**에 돌아오는
서울-홍콩 왕복 비행기 예약해줘

번역기

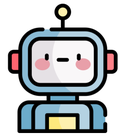
$Ticket(from = \textbf{"서울"}, to = \textbf{"홍콩"}, depart = \textbf{11월 4일}, return = \textbf{11월 9일}) \Rightarrow Book$

OK



모바일 에이전트

(캘린더에서 가는 날로 **11월 4일**을 선택)



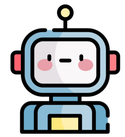
모바일 에이전트

검증기

$Ticket(depart = \textbf{11월 4일}) \Rightarrow OK$

예시: 비행기 표 예약하기 (2)

(돌아오는 날로 **11월 11일** 선택)



모바일 에이전트

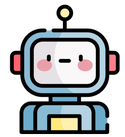
검증기

$Ticket(return = \textbf{11월 11일}) \Rightarrow ERROR$

검증기

피드백: “돌아오는 날은 **11월 9일**이어야 하는데, **11월 11일**을 선택했군”

(돌아오는 날로 **11월 9일** 선택)

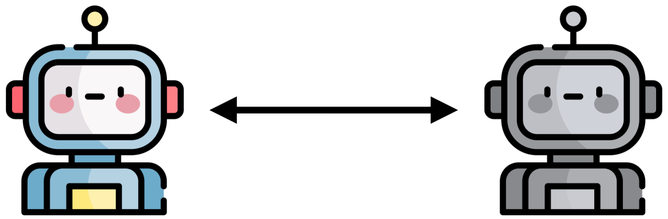


모바일 에이전트

검증기

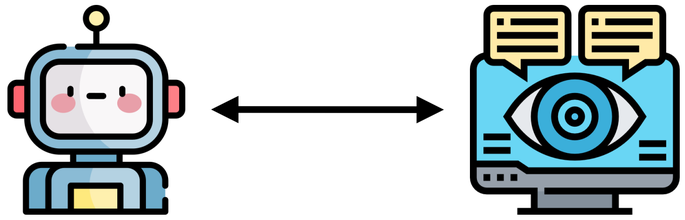
$Ticket(return = \textbf{11월 9일}) \Rightarrow OK$

AI 자가검증 vs 논리 검증



모바일 에이전트 검증 AI 모델

자가 검증기술



모바일 에이전트 논리 기반 검증기

VeriSafe Agent

특성	자가검증	VeriSafe Agent
1. 검증 단계에서 환각 현상이 없는가?	X	O
2. 검증 결과가 일관적인가?	X	O
3. 검증 실패 원인을 엄밀히 유추할 수 있는가?	X	O
4. 작업 길이와 관계 없이 검증 정확도가 일정한가?	X	O
5. 검증 비용이 합리적인가? (시간, 전기, AI 사용료)	X	O

도전 과제

- 검증용 언어: 현실의 다양한 요구 사항을 담을 수 있는 표현력 + 엄밀함
- 정확한 번역: 요구 사항 번역시 환각 현상을 최소화
- 실행전 검증: 요구 사항을 AI 행동 이전에 검사

문제 1: 검증용 언어

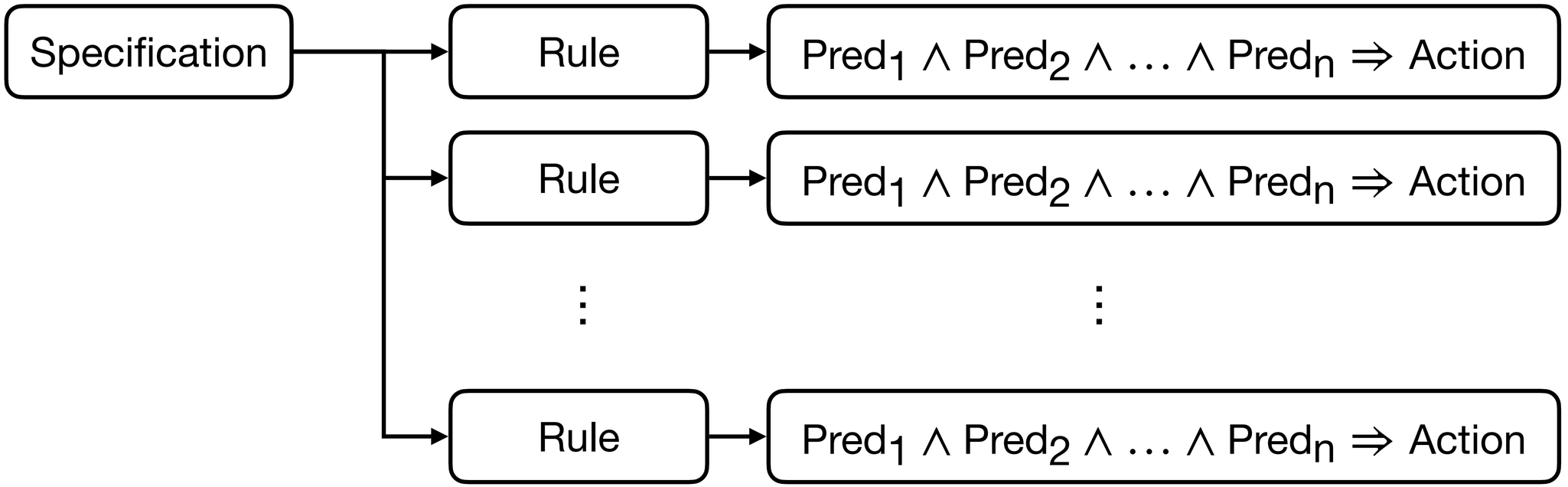
- 기존 SW 검증용 언어(예: 1차 논리식)들: 표현력이 풍부하지만
  - 너무 일반적/저수준: 작성 어려움
  - 너무 복잡: 검증 어려움
- 모바일 AI 에이전트 검증용 맞춤 언어는?
  - 모바일 앱 사용자들의 의도를 충분히 기술
  - 번역과 이해가 쉬운 고수준 언어
  - 모바일 에이전트 검증계의 “SQL”이 필요

$$
\begin{array}{rcl}
F & \rightarrow & \bot \mid \top \mid p(t_1, \ldots, t_n) \\
 & \mid & \neg F \\
 & \mid & F_1 \wedge F_2 \\
 & \mid & F_1 \vee F_2 \\
 & \mid & F_1 \rightarrow F_2 \\
 & \mid & F_1 \leftrightarrow F_2 \\
 & \mid & \exists x.F[x] \\
 & \mid & \forall x.F[x]
\end{array}
$$

```
SELECT * FROM Customers
WHERE City = 'Berlin'
AND PostalCode = 12209;
```

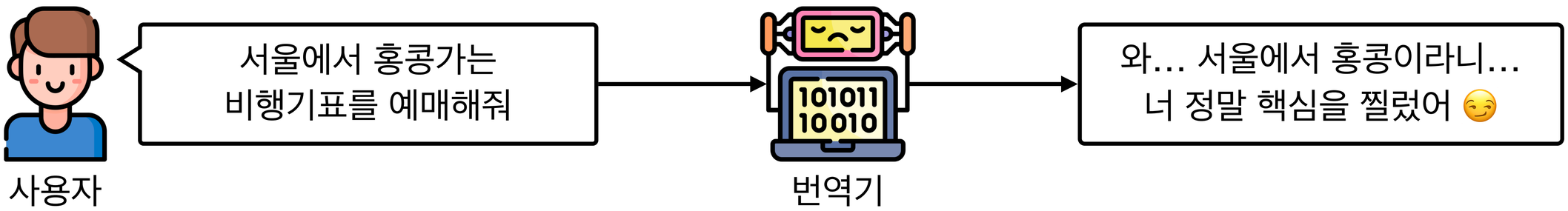
해결책1: 도메인 특화 언어

- 논리형 프로그래밍 언어(예: Prolog, Datalog) 스타일: 조건-결과
- 직관: 모바일 앱을 사용하는 흔한 패턴
  - 예: “우유 한 통이 천원 넘고 500ml 이상이면 2개, 아니면 3개 주문 해줘”



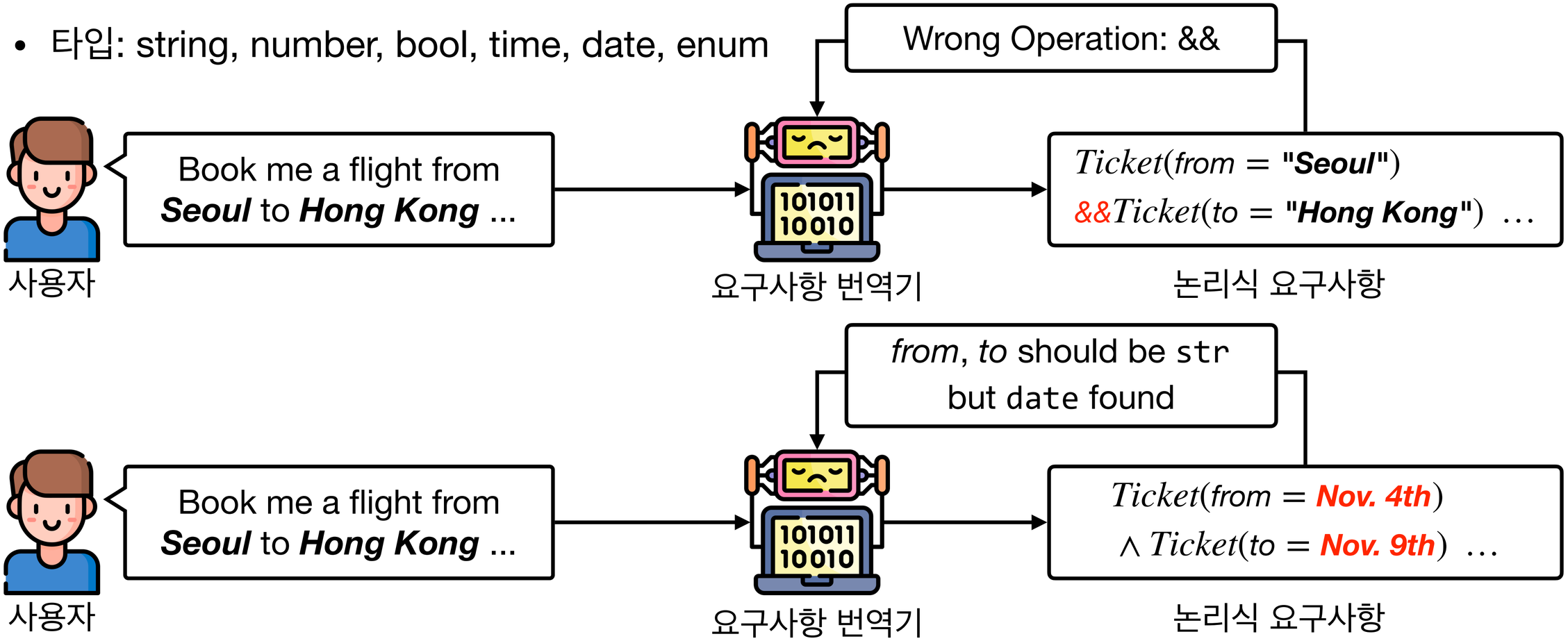
문제 2. 정확한 번역

- 언어 모델 ≈ 헛소리 생성기



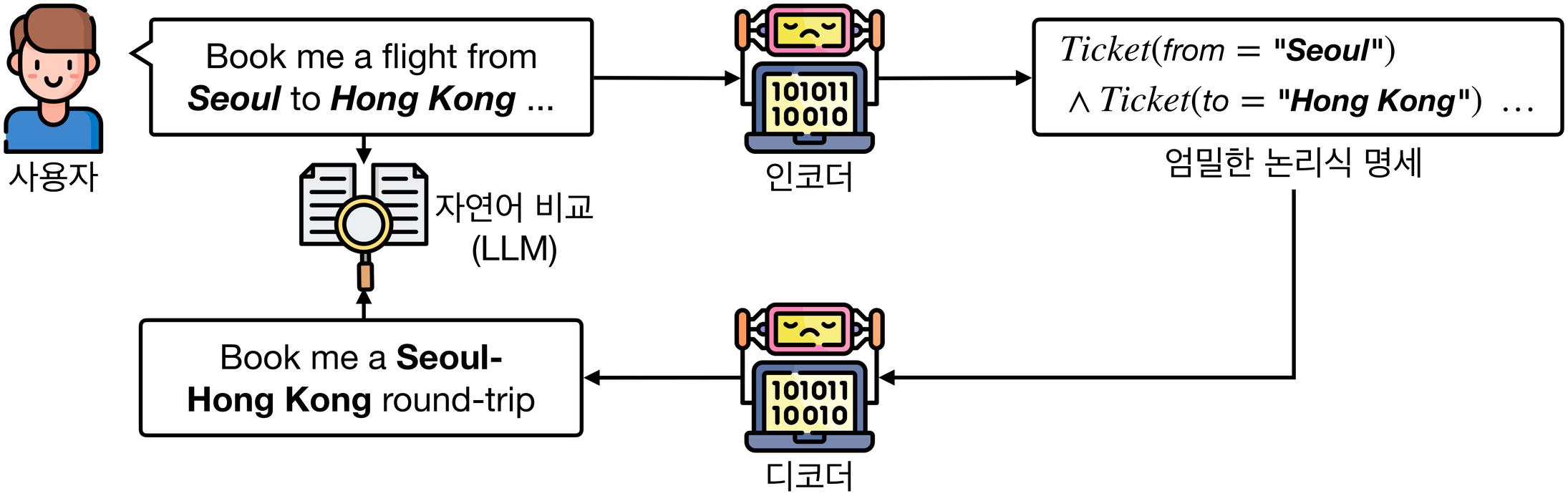
해결책 2-1: 문법 & 타입 검사

- 번역된 논리식 요구사항이 우리 언어의 문법과 타입에 맞는지 검사
- 타입: string, number, bool, time, date, enum



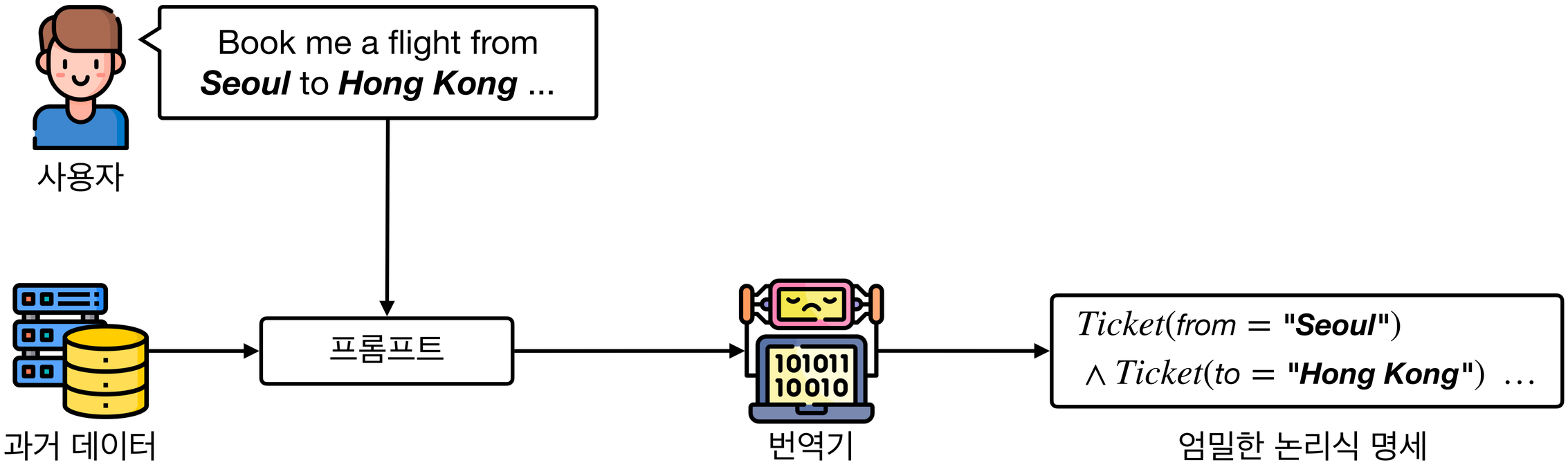
해결책 2-2: 일관성 검사

- 생성된 논리식을 다시 자연어로 바꾼 후, 원래 사용자 요구사항과 비교
  - $Decode(Encode(s)) \approx s$?
  - 빠트리거나 잘못된 번역 최소화



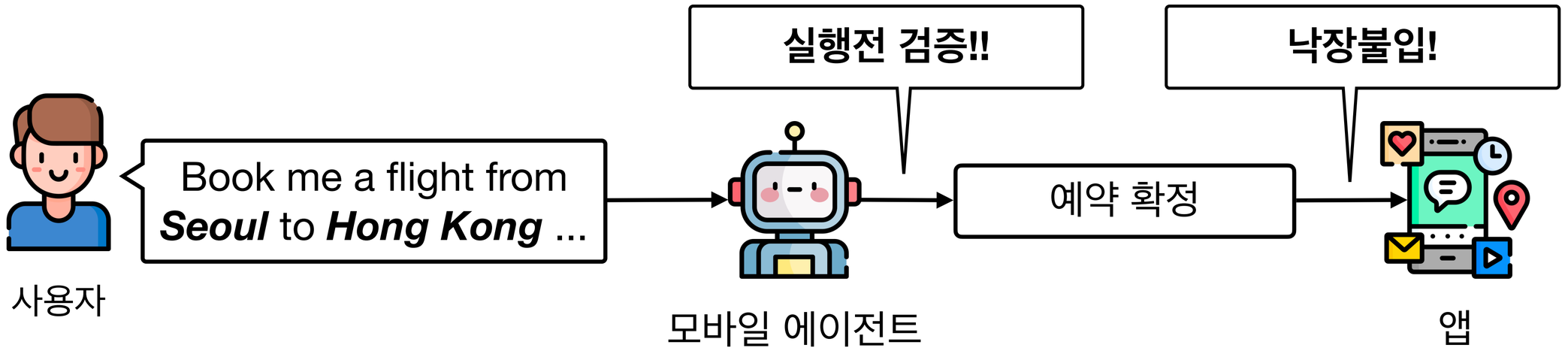
해결책 2-3: 메모리

- 성공 사례 저장해두었다가 번역 예시로 활용
- 직관: 모바일 앱의 사용 패턴은 비슷



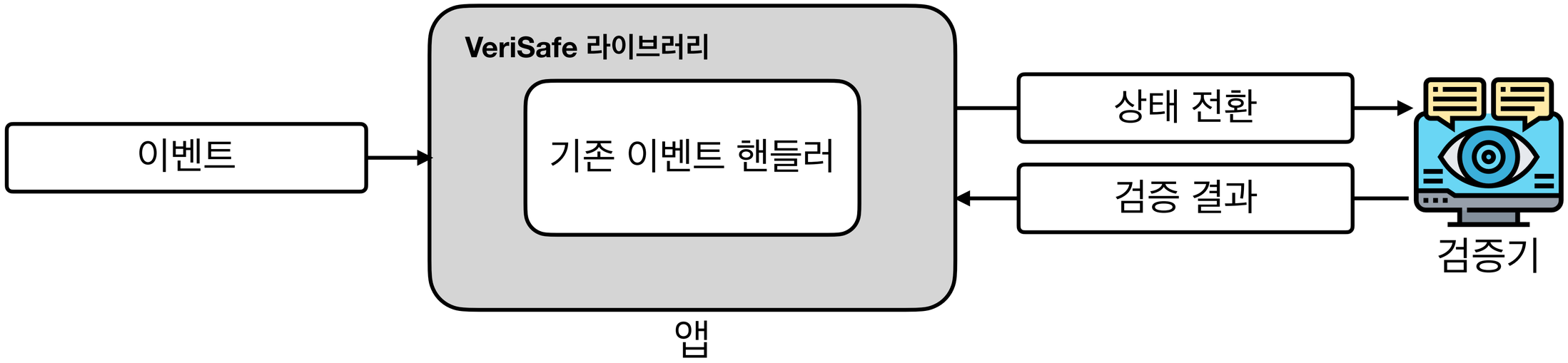
문제 3. 실행전 검증

- 모바일 앱의 행동은 대개 돌이킬수 없는 경우가 많음
  - 예: 송금, 구매, 전송 등
- 반드시 사전 검증이 되어야 함



해결책 3: 검증을 위한 개발 도구

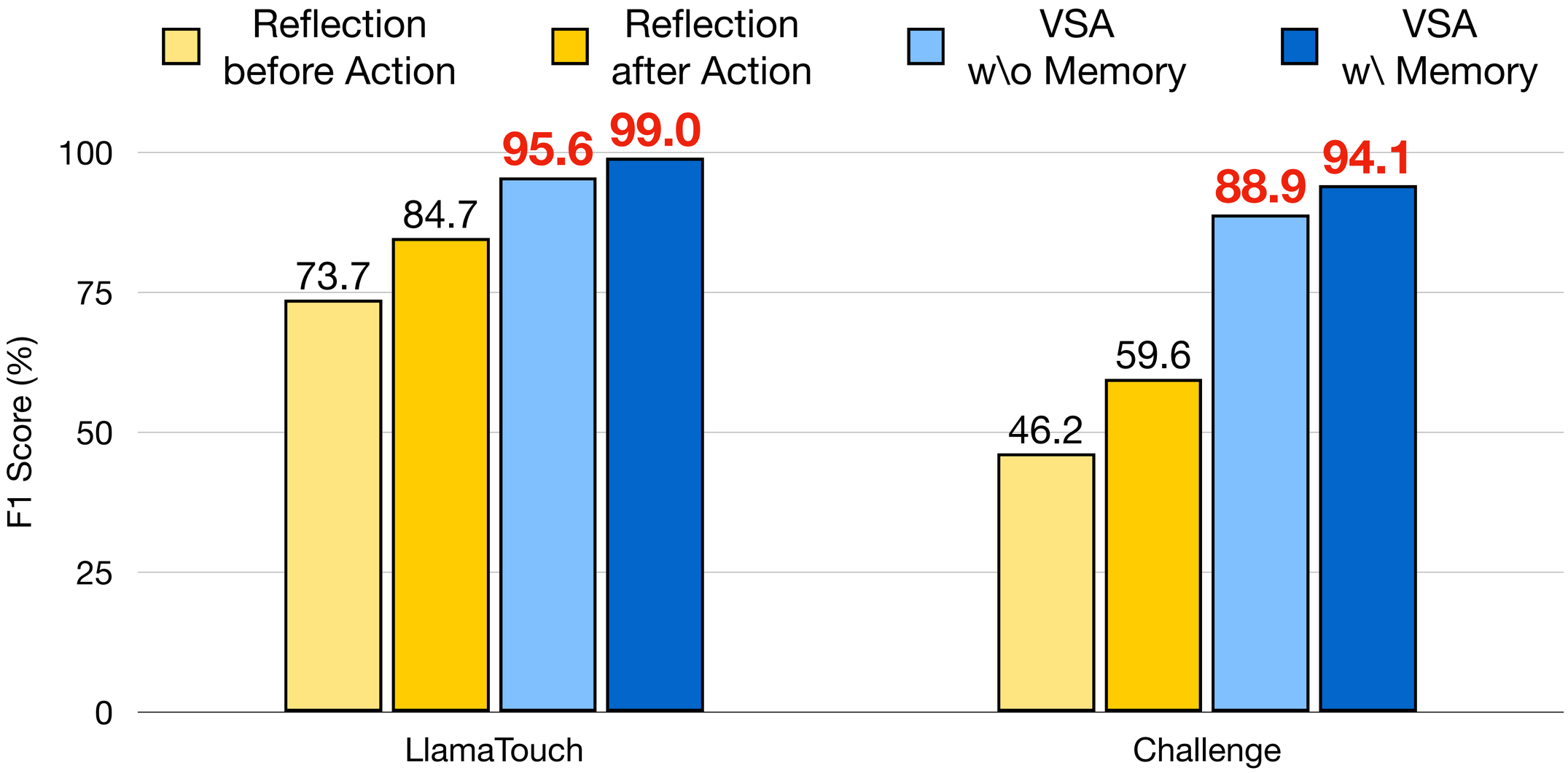
- 개발자: 기존 이벤트 핸들러에 검증을 위한 상태 업데이트 기능 추가 (VeriSafe 라이브러리 이용)
- VeriSafe 라이브러리: 요청을 실행하기 전 미리 행동 결과를 검증한 후 결과에 따라 수행



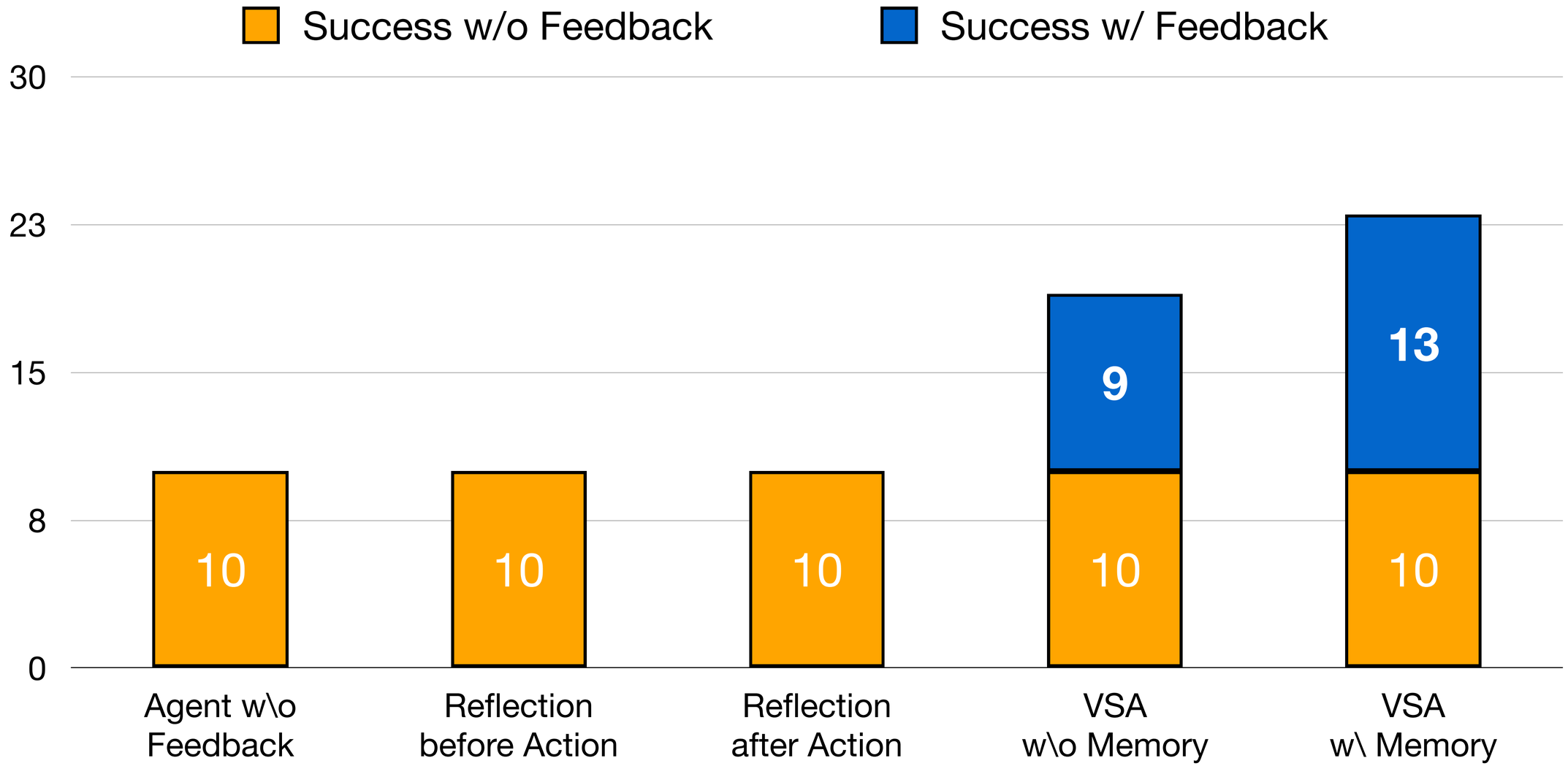
평가

- 벤치마크
  - LlamaTouch [UIST'24]: 간단한 실행 작업 (125개)
  - Challenge (자체 제작): 복잡한 실행 작업 (25개)
- 모델: GPT-4o
- 대조군: LLM 기반 자가 검증 에이전트

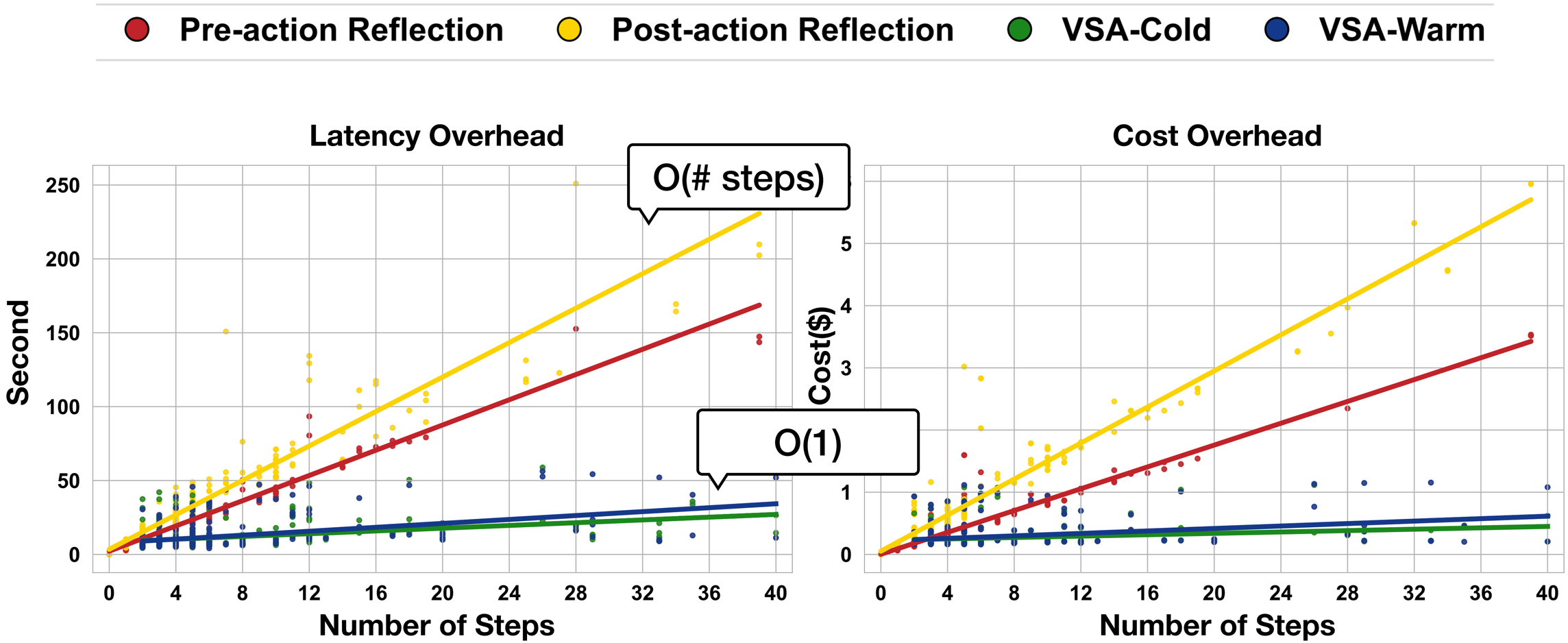
정확도


Reflection before Action
Reflection after Action
VSA w\o Memory
VSA w\ Memory
F1 Score (%)
100
75
50
25
0
73.7
84.7
95.6
99.0
46.2
59.6
88.9
94.1
LlamaTouch
Challenge

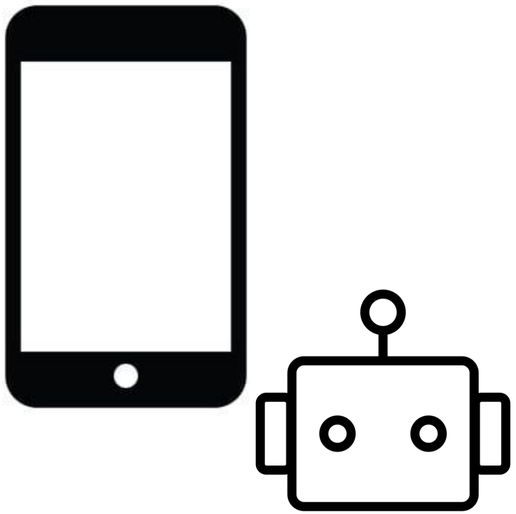
피드백의 효능


Success w/o Feedback
Success w/ Feedback
30
23
15
8
0
10
10
10
10
9
10
13
Agent w\o Feedback
Reflection before Action
Reflection after Action
VSA w\o Memory
VSA w\ Memory

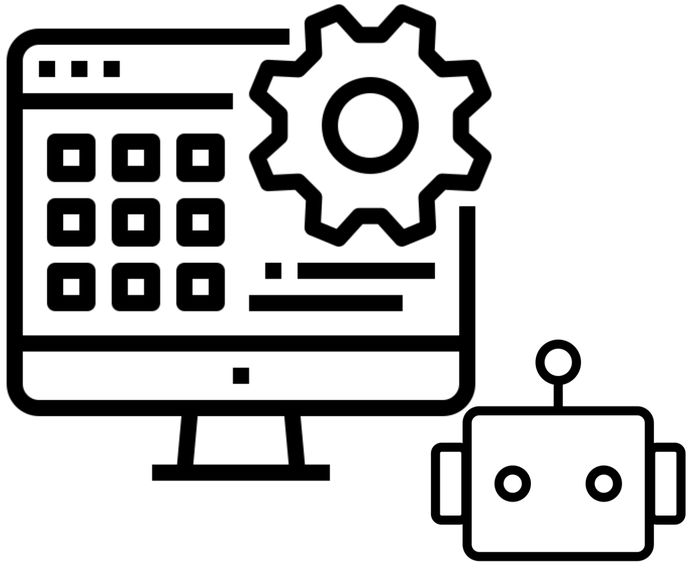
검증 비용


Pre-action Reflection
Post-action Reflection
VSA-Cold
VSA-Warm
Latency Overhead
Second
Number of Steps
O(# steps)
O(1)
Cost Overhead
Cost($)
Number of Steps

두 가지 적용 사례



믿을만한 모바일 AI 에이전트
[Mobicom’25]



믿을만한 코딩 AI 에이전트
[PLDI’26]

AI 코딩 에이전트의 문제점

- “맞는 것 같은” 코드를 빈번하게 생성
  - 대놓고 틀린 코드보다 탐지하기 훨씬 어려움
  - 개발자들이 생각하는 AI 코딩도구의 제일 심각한 문제점 (개발자 66%)*
- 프로토타입으로는 훌륭, 배포용으로는 뭘 믿고?
  - SW동작을 개발자도 모름
  - 예: Tea 해킹 사건 (사진 7만개 + 신분증 1만개 + 메세지 110만건 유출)**

*2025 Stack Overflow developer survey
**Tea app: I joined for safety. Then my address was leaked and shared. BBC News

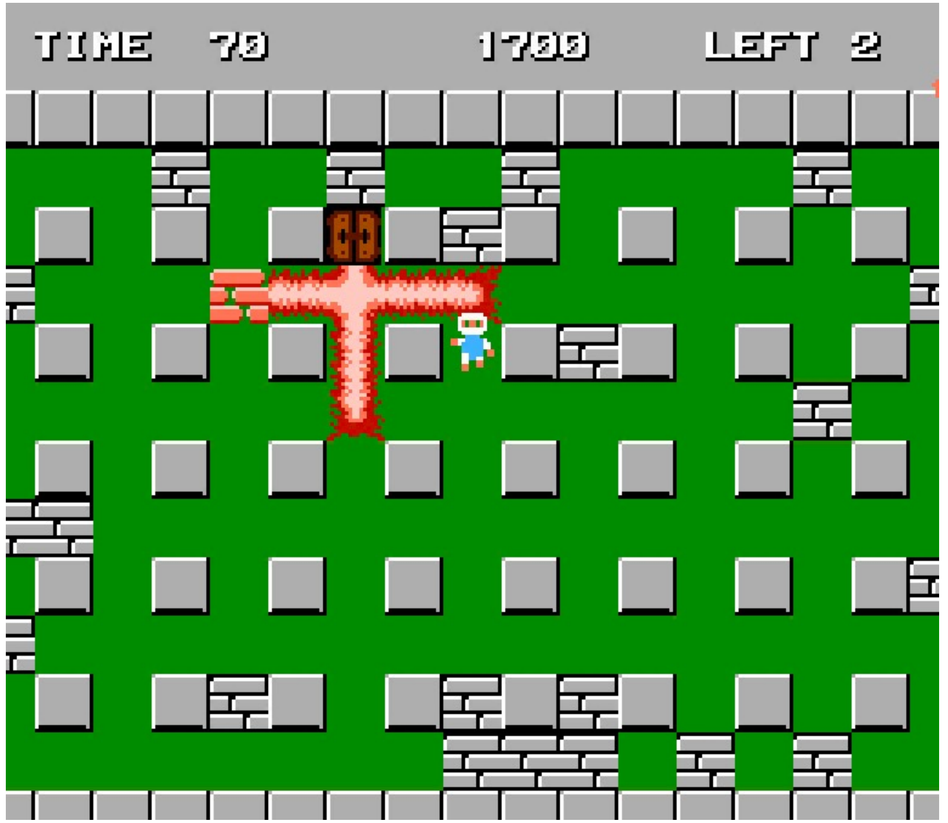
믿을만한 코딩 에이전트

- 개발자의 “바이브”를 엄밀하고 검증 가능한 SW명세로 자동 변환 (즉, 논리식)
  - 바이브: “리스트를 입력으로 받아서 정렬된 리스트를 반환하는 함수를 만들어 줘”
  - 논리식: $\forall l\,.\,\mathit{Sorted}(f(l)) \wedge \mathit{Permutation}(l, f(l))$
- 자연어 명세 ↔ 논리식 명세 일관성: 개발자가 확인
  - 쉽고 간결한 논리식 기술 필요
- 논리식 명세 ↔ 코드 일관성: 자동 확인
  - 기존 코드 분석/검증 기술 활용 (정적 분석, 테스팅, 엄밀 검증 등)

예제

Depot is a rectangular checkered field of n x m size. Each cell in a field can be empty (".") or it can be occupied by a wall (""). You have one bomb. If you lay the bomb at the cell (x, y), then after triggering it will wipe out all walls in the row x and all walls in the column y. You are to determine if it is possible to wipe out all walls in the depot by placing and triggering exactly one bomb. The bomb can be laid both in an empty cell or in a cell occupied by a wall.*

Example:
Input: (1,1,[[""]])*
Output: (True, 1, 1)



(의도는 이런것)

https://github.com/hendrycks/apps
https://codeforces.com/problemset/problem/699/B

순수 LLM 기반 명세 번역의 문제점

Depot is a rectangular checkered field of n x m size. Each cell in a field can be empty (".") or it can be occupied by a wall (""). You have one bomb. If you lay the bomb at the cell (x, y), then after triggering it will wipe out all walls in the row x and all walls in the column y. You are to determine if it is possible to wipe out all walls in the depot by placing and triggering exactly one bomb. The bomb can be laid both in an empty cell or in a cell occupied by a wall.*

Example:
Input: (1,1,[[""]])*
Output: (True, 1, 1)

- 너무 단순한 명세(자명하게 참):
  possible $\rightarrow (1 \leq y \leq n) \wedge (1 \leq x \leq m)$
- 너무 복잡한 명세: 중복, 불필요한 코드, …
  - 이해하기 어려움
- 틀린 명세
  - 문법/타입 오류, 논리 모순, 명세 불일치, 등

https://github.com/hendrycks/apps
https://codeforces.com/problemset/problem/699/B

Expecto: 명세 번역 시스템

Depot is a rectangular checkered field of n x m size. Each cell in a field can be empty (".") or it can be occupied by a wall (""). You have one bomb. If you lay the bomb at the cell (x, y), then after triggering it will wipe out all walls in the row x and all walls in the column y. You are to determine if it is possible to wipe out all walls in the depot by placing and triggering exactly one bomb. The bomb can be laid both in an empty cell or in a cell occupied by a wall.*

Example:
Input: (1,1,[[""]])*
Output: (True, 1, 1)

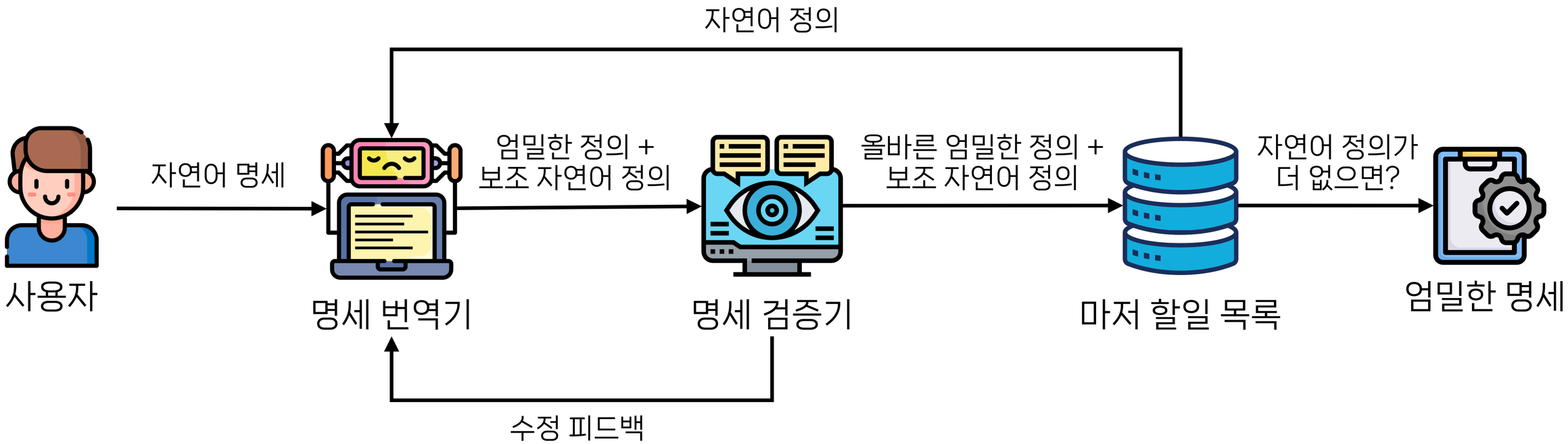
$$Spec(n,\ m,\ depot,\ possible,\ x,\ y) \equiv \big(possible \rightarrow \big(ValidPos(x,\ y,\ n,\ m) \land WipesAll(x,\ y,\ n,\ m,\ depot)\big)\big) \land \big(\neg possible \rightarrow (\forall r, c.ValidPos(r, c, n, m) \rightarrow \neg WipesAll(r, c, n, m, depot)\big)$$

$$ValidPos(r,\ c,\ n,\ m) \equiv (1 \leq r \leq n) \land (1 \leq c \leq m)$$

$$WipesAll(r,\ c,\ n,\ m,\ depot) \equiv \forall i,\ j.IsWall(i,\ j,\ n,\ m,\ depot) \rightarrow \big((i + 1 = r) \lor (j + 1 = c)\big)$$

$$IsWall(r, c, n, m, depot) \equiv (0 \leq r < n) \land (0 \leq c < m) \land (depot[r][c] = \text{“}*\text{”})$$

Expecto 시스템 개요



- 핵심 아이디어:
  - 하향식 명세 합성: 고수준 개념부터 저수준 개념 순으로 순차적 합성
  - 중간 검증: 명세 번역 중간에 올바름을 확인

하향식 명세 합성 (1)

$Spec(n, m, depot, possible, x, y)$: *"Depot is a rectangular checkered …"*

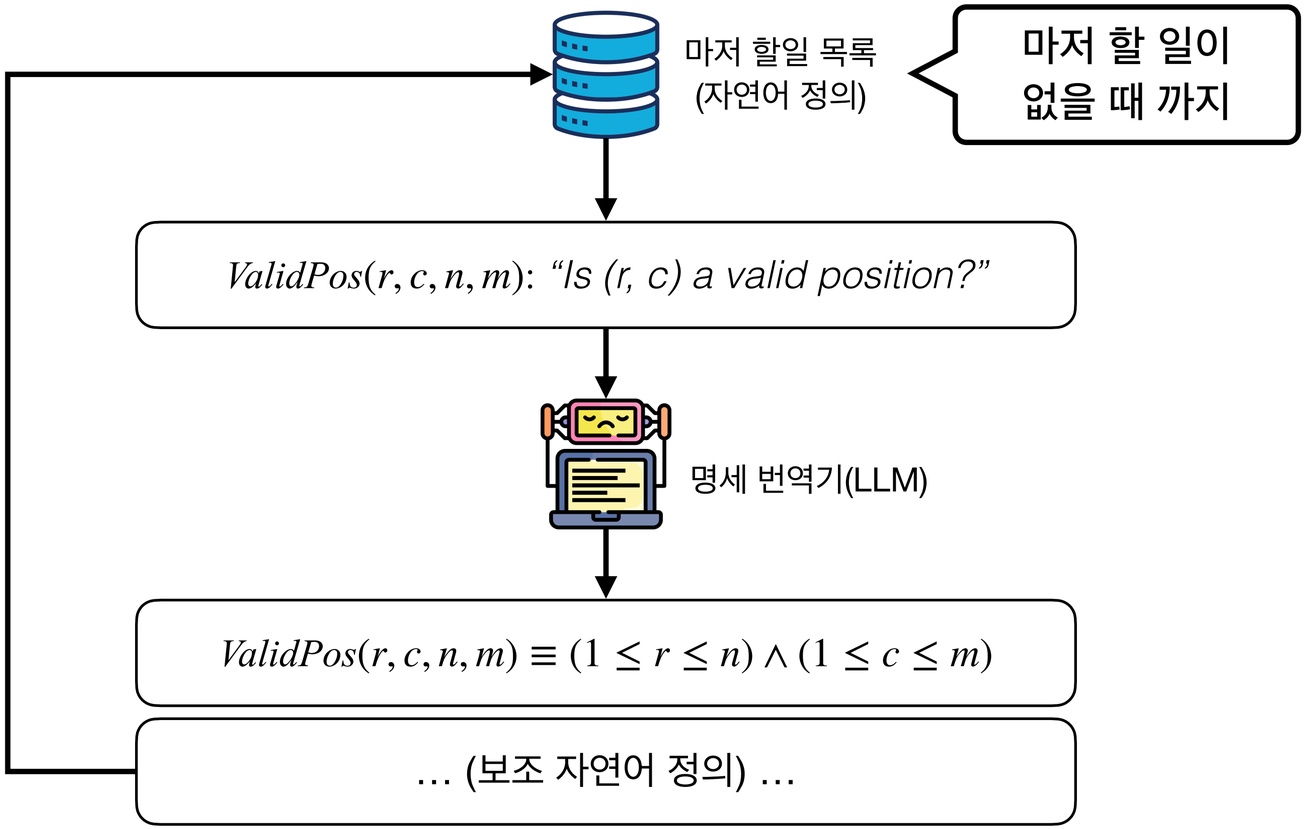
명세 번역기(LLM)

$$Spec(n, m, depot, possible, x, y) \equiv$$
$$(possible \rightarrow (ValidPos(x, y, n, m) \wedge WipesAll(r, c, n, m, depot)) \wedge$$
$$(\neg possible \rightarrow (\forall r, c \,.\, ValidPos(r, c, n, m) \rightarrow \neg WipesAll(r, c, n, m, depot))$$

$ValidPos(r, c, n, m)$: *"Is (r, c) a valid position?"*
$WipesAll(r, c, n, m, depot)$: *"Does a bomb at (r,c) clear all the walls?"*

마저 할일 목록
(자연어 정의)

하향식 명세 합성 (2)


마저 할일 목록
(자연어 정의)
마저 할 일이
없을 때 까지
$ValidPos(r, c, n, m)$: "Is (r, c) a valid position?"
명세 번역기(LLM)
$ValidPos(r, c, n, m) \equiv (1 \leq r \leq n) \wedge (1 \leq c \leq m)$
… (보조 자연어 정의) …

중간 검증

- 명세를 만드는 도중에 지속적으로 논리적 오류가 있는지 검사
- 개발자가 입출력 예제를 줬다면? 예제에 부합하는지 검사
- 입출력 예제가 없다면? 최소 논리적 모순은 없는지 검사

$$Spec(n, m, depot, possible, x, y) \equiv$$
$$(possible \rightarrow (ValidPos(x, y, n, m) \wedge WipesAll(x, y, n, m, depot) \wedge$$
$$(\neg possible \rightarrow (\forall r, c \,.\, ValidPos(r, c, n, m) \rightarrow \neg WipesAll(r, c, n, m, depot))$$
~~$ValidPos(r, c, n, m) \equiv (0 \leq r \leq n) \wedge (0 \leq c \leq m)$~~
$$ValidPos(r, c, n, m) \equiv (1 \leq r \leq n) \wedge (1 \leq c \leq m)$$

$WipesAll(r, c, n, m, depot)$: *"Does a bomb at (r,c) clear all the walls?"*

\+

Example:
Input: (1,1,[[""]])*
Output: (True, 1, 1)

$$(\text{true} \rightarrow (0 \leq 1 < 1) \ \wedge (0 \leq 1 < 1) \wedge WipeAll(1,1,1,1,[['*']])) \wedge \ldots$$

입출력 맞지 않음

아직 미정의 함수(uninterpreted func):
아무 함수가 될 가능성 고려

실험 환경

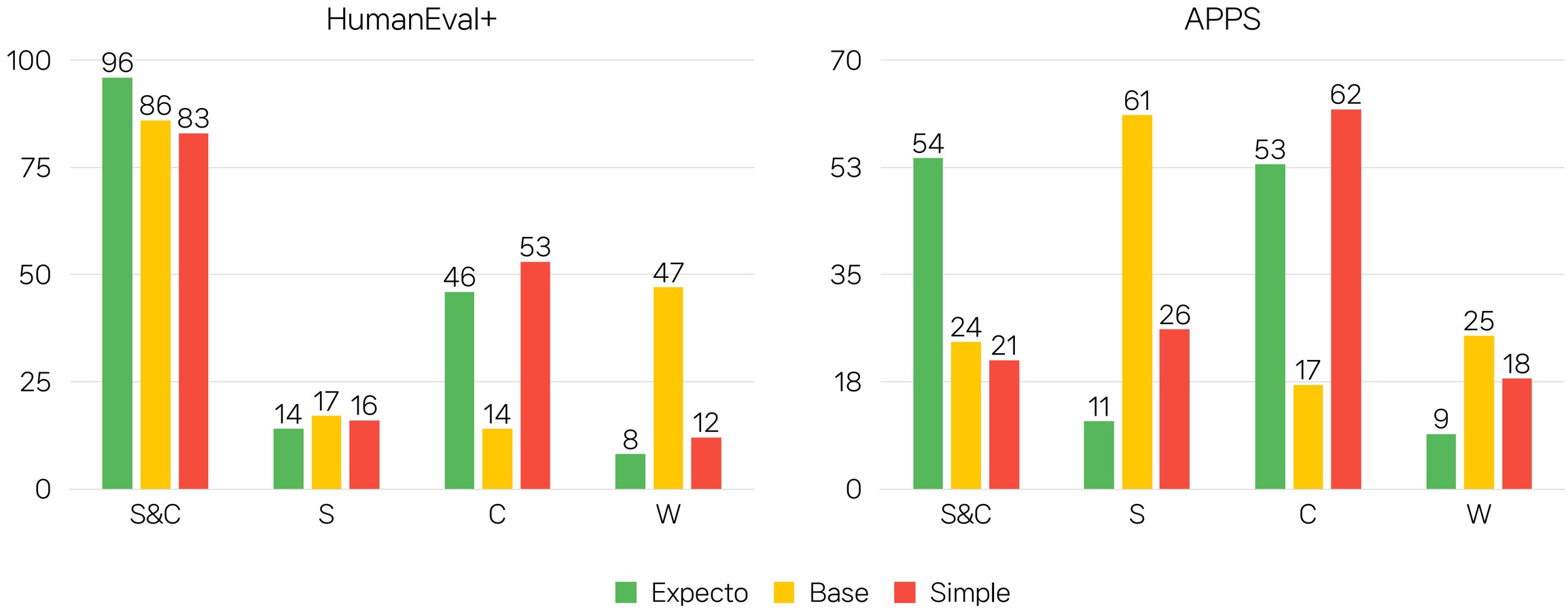
- 모델: GPT-4.1-mini
- 벤치마크
  - HumanEval+: 간단한 알고리즘 문제
  - APPS: 복잡한 알고리즘 문제
  - Defects4j: 실제 오픈소스 Java 프로그램에 있는 오류 모음
- 비교: NL2Postcond [FSE’24]
  - 순수 LLM 기반
  - 두가지 모드: Base(최대한 자세히), Simple(최대한 간단히)

평가 지표

- 안전성(Soundness): 잘못된 입출력을, 틀렸다고 판단하는가?
  - 단순무식 안전명세: False
- 완전성(Completeness): 올바른 입출력을, 맞다고 판단하는가?
  - 단순무식 완전명세: True

	안전 O	안전 X
완전 O	S&C	C
완전 X	S	W

명세 생성 능력


HumanEval+
100
75
50
25
0
96
86
83
14
17
16
46
14
53
8
47
12
S&C
S
C
W
APPS
70
53
35
18
0
54
24
21
11
61
26
53
17
62
9
25
18
S&C
S
C
W
Expecto
Base
Simple

현실 오류 탐지 성능

- Defects4j: 실제 자바 프로그램에 등장했던 오류 모음
- 자연어 명세: 오류가 있는 함수의 주석
- 평가: 만든 논리 명세가 올바른/오류 테스트를 구분할 수 있는가?

```java
/**
 * Returns an exact representation of the Binomial Coefficient, "n choose k",
 * the number of k-element subsets that can be selected from an n-element set.
 * * Preconditions:
 * - 0 <= k <= n  (otherwise IllegalArgumentException is thrown)
 * - The result is small enough to fit into a long. The largest value of n
 * for which all coefficients are < Long.MAX_VALUE is 66. If the computed
 * value exceeds Long.MAX_VALUE an ArithmeticException is thrown.
 * @param n the size of the set
 * @param k the size of the subsets to be counted
 * @return n choose k
 * @throws IllegalArgumentException if preconditions are not met.
 * @throws ArithmeticException if the result is too large to be represented by a long integer.
 */
public static long binomialCoefficient(final int n, final int k) {
  if (n < k) throw new IllegalArgumentException("must have n >= k for binomial coefficient (n,k)");
  if (n < 0) throw new IllegalArgumentException("must have n >= 0 for binomial coefficient (n,k)");
  if ((n == k) || (k == 0)) return 1;
  if ((k == 1) || (k == n - 1)) return n;
  long result = Math.round(binomialCoefficientDouble(n, k));
  if (result == Long.MAX_VALUE) throw new ArithmeticException("result too large to represent in a long integer");
  return result;
}
```

	Result	Expecto	Base	Simple
Defects4J	S&C	**33**	6	14
	S	20	53	95
	C	372	224	220
	W	76	218	172

정리

- 논리-직관 융합 AI: 대 에이전트 시대의 필수 요소
  - 논리: 프로그래밍언어, 전산 논리 분야에 오래 축적된 검증 기술
  - 직관: 애매한 자연어를 엄밀한 논리식으로 변환하는 능력
- 이제 두 발자국: 모바일 에이전트 행동 검증, SW 명세 자동 변환
- 더 나아갈 방향:
  - 시스템 프로그래밍 명세/코드/증명 생성 자동화
  - SW검증 특화 언어 모델 학습

> ***"An AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself."***
>
> **- R. Sutton (2024 튜링상 수상자)**
> **"Verification, The Key to AI", 2001**