

VeriSafe Agent

안전한 모바일 GUI 에이전트를 위한 논리 기반 에이전트 행동 검증
이동재

공동 연구: 이정재, 최치훈, 임영민, 위재영, 허기홍, 오상은, 이선재, 신인식

2025.07.29 ERC



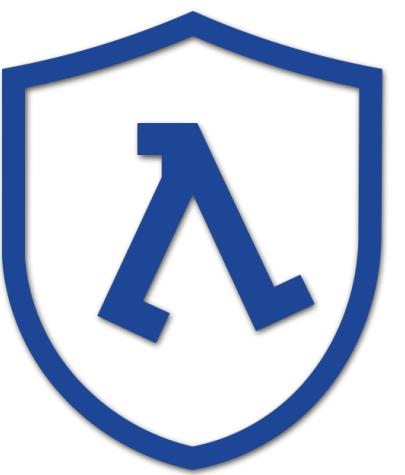
소프트웨어 재난 재발 방지



STAAR

Prosys Lab의 목표: 소프트웨어 재난의 재발 방지

소프트웨어 재난 재발 방지



STAAR

Prosys Lab의 목표: 소프트웨어 재난의 재발 방지



"어떤" 소프트웨어?

신종 소프트웨어의 등장

신종 소프트웨어의 등장

```
def hello():  
    print("Hello!")
```

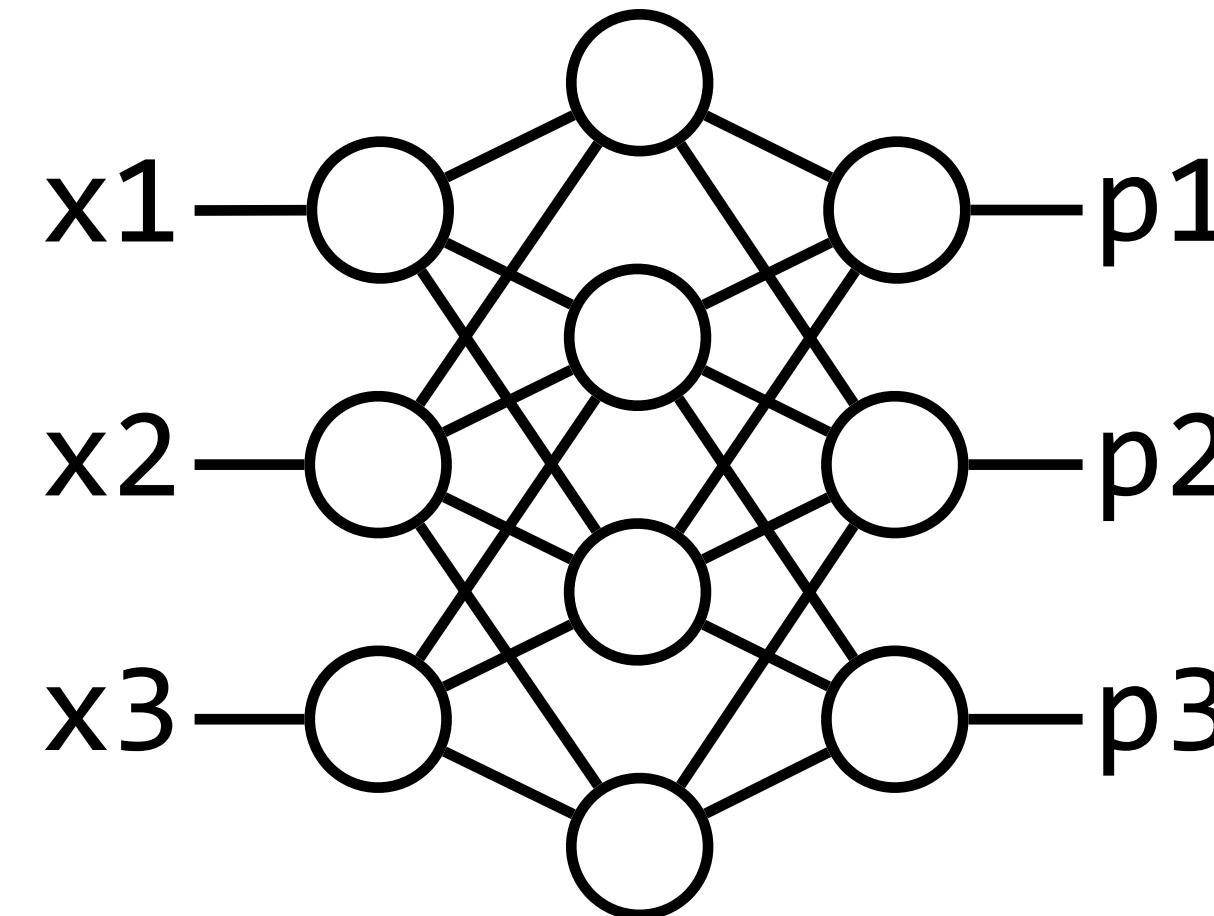
1세대 소프트웨어

소스코드

신종 소프트웨어의 등장

```
def hello():
    print("Hello!")
```

1세대 소프트웨어
소스코드

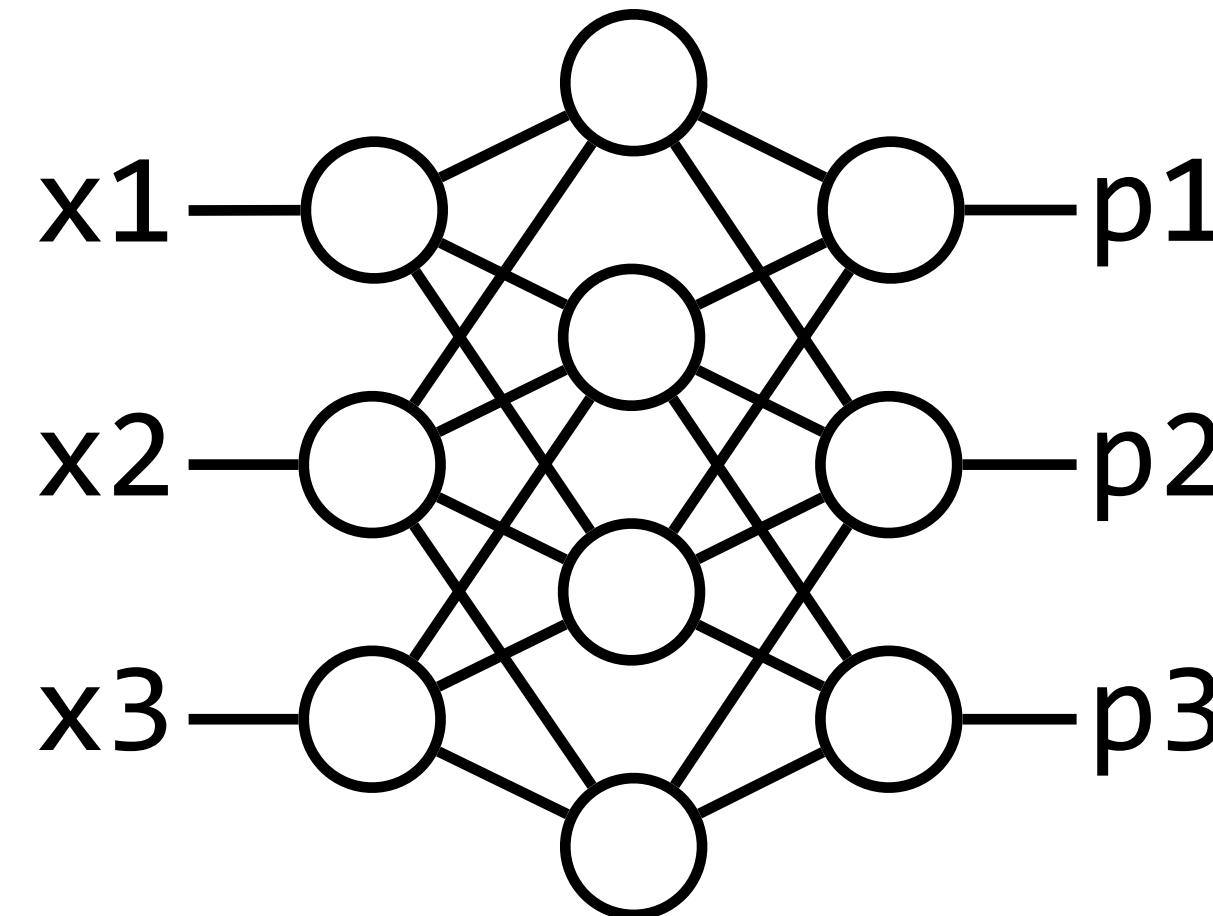


2세대 소프트웨어
데이터셋 + 신경망

신종 소프트웨어의 등장

```
def hello():
    print("Hello!")
```

1세대 소프트웨어
소스코드



2세대 소프트웨어
데이터셋 + 신경망

"당신은 최고의
SW 엔지니어입니다.
버그를 고쳐주세요"

3세대 소프트웨어
프롬프트 + 언어모델

4세대 소프트웨어: AI 에이전트

4세대 소프트웨어: AI 에이전트

- 목표 달성을 위해 자율적으로 동작하는 소프트웨어

4세대 소프트웨어: AI 에이전트

- 목표 달성을 위해 자율적으로 동작하는 소프트웨어
 - Issue를 자동으로 해결, 버그를 찾아서 제보

4세대 소프트웨어: AI 에이전트

- 목표 달성을 위해 자율적으로 동작하는 소프트웨어
 - Issue를 자동으로 해결, 버그를 찾아서 제보
 - 외부 소프트웨어와 스스로 상호작용 가능

4세대 소프트웨어: AI 에이전트

- 목표 달성을 위해 자율적으로 동작하는 소프트웨어
- Issue를 자동으로 해결, 버그를 찾아서 제보
- 외부 소프트웨어와 스스로 상호작용 가능

```
GEMINI

Tips for getting started:
1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. ./help for more information.

> write a short paragraph about why Gemini CLI is awesome

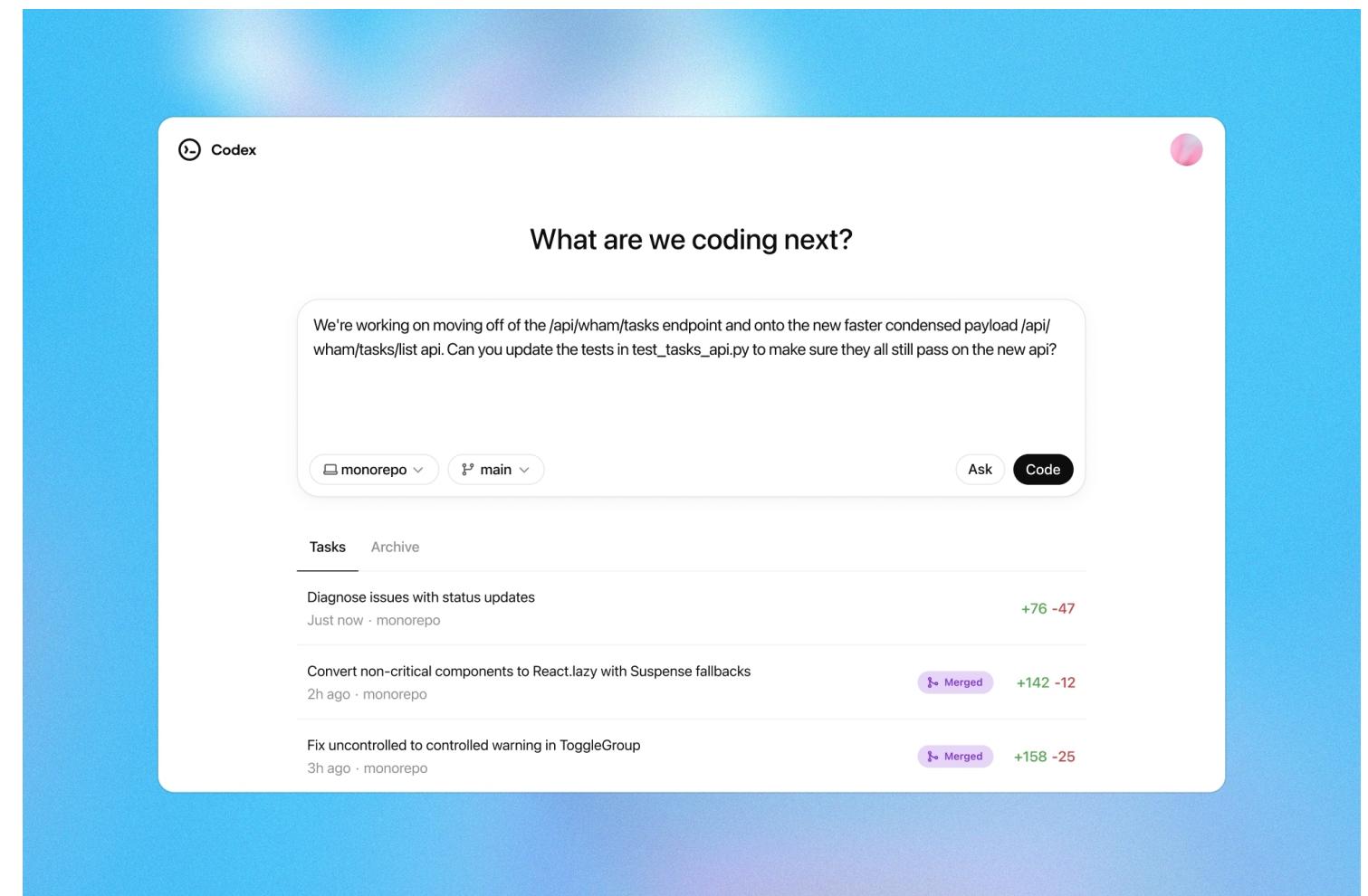
+ I will start by searching the web for "Gemini CLI" to understand its main features and purpose. Following that, I'll examine the README.md file for a project-specific overview. I will then consult docs/index.md and docs/cli/commands.md to gain a deeper understanding of its capabilities. Finally, I will write a short paragraph summarizing why Gemini CLI is awesome.

- GoogleSearch Searching the web for: "Gemini CLI features and purpose"

: Uncovering Gemini's Awesome (esc to cancel, 21s)

Using 3 GEMINI.md files

~/code/gemini-cli (release*)      no sandbox (see /docs)      gemini-2.5-pro (99% context left)
```



에이전트의 문제점

에이전트의 문제점

1. 환각 (Hallucination) 문제

에이전트의 문제점

1. 환각 (Hallucination) 문제

- 화면 상태를 잘못 이해, 버튼의 기능을 오해, 지시사항을 잘못 이해

에이전트의 문제점

1. 환각 (Hallucination) 문제

- 화면 상태를 잘못 이해, 버튼의 기능을 오해, 지시사항을 잘못 이해

2. 맥락 파악 문제

에이전트의 문제점

1. 환각 (Hallucination) 문제

- 화면 상태를 잘못 이해, 버튼의 기능을 오해, 지시사항을 잘못 이해

2. 맥락 파악 문제

- 모델의 컨텍스트 창 길이 제한을 초과하거나, 맥락 파악에 실패하는 경우

에이전트의 문제점

1. 환각 (Hallucination) 문제

- 화면 상태를 잘못 이해, 버튼의 기능을 오해, 지시사항을 잘못 이해

2. 맥락 파악 문제

- 모델의 컨텍스트 창 길이 제한을 초과하거나, 맥락 파악에 실패하는 경우
- 현재 무엇을 하고 있었는지, 최종 목표가 무엇인지 중간에 잊어버림

에이전트의 문제점

1. 환각 (Hallucination) 문제

- 화면 상태를 잘못 이해, 버튼의 기능을 오해, 지시사항을 잘못 이해

2. 맥락 파악 문제

- 모델의 컨텍스트 창 길이 제한을 초과하거나, 맥락 파악에 실패하는 경우
- 현재 무엇을 하고 있었는지, 최종 목표가 무엇인지 중간에 잊어버림

3. 일관성 문제

에이전트의 문제점

1. 환각 (Hallucination) 문제

- 화면 상태를 잘못 이해, 버튼의 기능을 오해, 지시사항을 잘못 이해

2. 맥락 파악 문제

- 모델의 컨텍스트 창 길이 제한을 초과하거나, 맥락 파악에 실패하는 경우
- 현재 무엇을 하고 있었는지, 최종 목표가 무엇인지 중간에 잊어버림

3. 일관성 문제

- 의미적으로 동일한 질문을 했을 때 항상 같은 결과가 나온다는 보장 없음

에이전트의 문제점

1. 환각 (Hallucination) 문제

- 화면 상태를 잘못 이해, 버튼의 기능을 오해, 지시사항을 잘못 이해

2. 맥락 파악 문제

- 모델의 컨텍스트 창 길이 제한을 초과하거나, 맥락 파악에 실패하는 경우
- 현재 무엇을 하고 있었는지, 최종 목표가 무엇인지 중간에 잊어버림

3. 일관성 문제

- 의미적으로 동일한 질문을 했을 때 항상 같은 결과가 나온다는 보장 없음
- 비결정적 생성 알고리즘 (샘플링 기반 알고리즘) 사용 시 매번 결과가 달라짐

실제 문제 사례

1. 맥도날드 드라이브스루에 도입된 에이전트가 주문을 잘못 이해 ([The Guardian, 2024.06.17](#))

실제 문제 사례

1. 맥도날드 드라이브스루에 도입된 에이전트가 주문을 잘못 이해 ([The Guardian, 2024.06.17](#))
 - 잘못된 품목을 추가하거나, 개수를 잘못 설정

실제 문제 사례

1. 맥도날드 드라이브스루에 도입된 에이전트가 주문을 잘못 이해 ([The Guardian, 2024.06.17](#))
 - 잘못된 품목을 추가하거나, 개수를 잘못 설정
2. Claude 기반 LLM Agent가 부트로더 (GRUB) 설정을 손상시킴 ([Shlegeris, 2024.09.30](#))

실제 문제 사례

1. 맥도날드 드라이브스루에 도입된 에이전트가 주문을 잘못 이해 ([The Guardian, 2024.06.17](#))
 - 잘못된 품목을 추가하거나, 개수를 잘못 설정
2. Claude 기반 LLM Agent가 부트로더 (GRUB) 설정을 손상시킴 ([Shlegeris, 2024.09.30](#))
 - 부팅 시스템이 망가져 컴퓨터가 켜지지 않음

실제 문제 사례

1. 맥도날드 드라이브스루에 도입된 에이전트가 주문을 잘못 이해 ([The Guardian, 2024.06.17](#))
 - 잘못된 품목을 추가하거나, 개수를 잘못 설정
2. Claude 기반 LLM Agent가 부트로더 (GRUB) 설정을 손상시킴 ([Shlegeris, 2024.09.30](#))
 - 부팅 시스템이 망가져 컴퓨터가 켜지지 않음
3. Cursor 상담 봇이 존재하지 않는 회사의 정책을 고객에게 알림 ([Benj Edwards, 2025.04.18](#))

실제 문제 사례

1. 맥도날드 드라이브스루에 도입된 에이전트가 주문을 잘못 이해 ([The Guardian, 2024.06.17](#))
 - 잘못된 품목을 추가하거나, 개수를 잘못 설정
2. Claude 기반 LLM Agent가 부트로더 (GRUB) 설정을 손상시킴 ([Shlegeris, 2024.09.30](#))
 - 부팅 시스템이 망가져 컴퓨터가 켜지지 않음
3. Cursor 상담 봇이 존재하지 않는 회사의 정책을 고객에게 알림 ([Benj Edwards, 2025.04.18](#))
 - Cursor가 기기 1대에서만 로그인 가능하다는 메일을 보냈으나, 그런 정책은 없음

신종 소프트웨어의 안전성 확보 기술은?

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...
- 2세대 소프트웨어 (데이터셋 + 신경망)

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...
- 2세대 소프트웨어 (데이터셋 + 신경망)
 - 신경망 검증, 적대적 공격 (Adversarial Attack) 방어를 위한 학습, ...

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...
- 2세대 소프트웨어 (데이터셋 + 신경망)
 - 신경망 검증, 적대적 공격 (Adversarial Attack) 방어를 위한 학습, ...
- 3세대 소프트웨어 (프롬프트 + 언어모델)

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...
- 2세대 소프트웨어 (데이터셋 + 신경망)
 - 신경망 검증, 적대적 공격 (Adversarial Attack) 방어를 위한 학습, ...
- 3세대 소프트웨어 (프롬프트 + 언어모델)
 - 정렬 (Alignment), 안전성 판별 모델, 모델 해석 (Mechanistic Interpretability)

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...
- 2세대 소프트웨어 (데이터셋 + 신경망)
 - 신경망 검증, 적대적 공격 (Adversarial Attack) 방어를 위한 학습, ...
- 3세대 소프트웨어 (프롬프트 + 언어모델)
 - 정렬 (Alignment), 안전성 판별 모델, 모델 해석 (Mechanistic Interpretability)
- 4세대 소프트웨어 (AI 에이전트)

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...
- 2세대 소프트웨어 (데이터셋 + 신경망)
 - 신경망 검증, 적대적 공격 (Adversarial Attack) 방어를 위한 학습, ...
- 3세대 소프트웨어 (프롬프트 + 언어모델)
 - 정렬 (Alignment), 안전성 판별 모델, 모델 해석 (Mechanistic Interpretability)
- 4세대 소프트웨어 (AI 에이전트)
 - 🤖🤖🤖

신종 소프트웨어의 안전성 확보 기술은?

- 1세대 소프트웨어 (소스코드)
 - 테스팅, 검증, 정적 분석, 퍼징, ...
- 2세대 소프트웨어 (데이터셋 + 신경망)
 - 신경망 검증, 적대적 공격 (Adversarial Attack) 방어를 위한 학습, ...
- 3세대 소프트웨어 (프롬프트 + 언어모델)
 - 정렬 (Alignment), 안전성 판별 모델, 모델 해석 (Mechanistic Interpretability)
- 4세대 소프트웨어 (AI 에이전트)
 - **VeriSafe Agent (MobiCom '25)**

모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어

모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어
- **API 기반:** API를 사용하여 앱을 인식하고 조작

모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어
- **API 기반:** API를 사용하여 앱을 인식하고 조작
 - 장점: API를 활용하기 때문에 정밀하게 앱 상태를 인식하고 조작할 수 있음

모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어
- **API 기반:** API를 사용하여 앱을 인식하고 조작
 - 장점: API를 활용하기 때문에 정밀하게 앱 상태를 인식하고 조작할 수 있음
 - 단점: API를 제공하지 않는 앱에는 적용 불가능. 확장성 떨어짐

모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어
- **API 기반:** API를 사용하여 앱을 인식하고 조작
 - 장점: API를 활용하기 때문에 정밀하게 앱 상태를 인식하고 조작할 수 있음
 - 단점: API를 제공하지 않는 앱에는 적용 불가능. 확장성 떨어짐
- **GUI 기반:** 인간처럼 앱 화면 (캡처된 화면, XML)을 보고 앱을 인식 및 조작

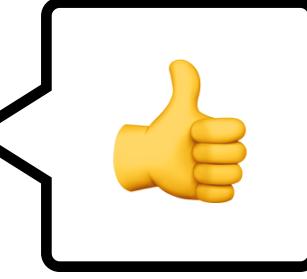
모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어
- **API 기반:** API를 사용하여 앱을 인식하고 조작
 - 장점: API를 활용하기 때문에 정밀하게 앱 상태를 인식하고 조작할 수 있음
 - 단점: API를 제공하지 않는 앱에는 적용 불가능. 확장성 떨어짐
- **GUI 기반:** 인간처럼 앱 화면 (캡처된 화면, XML)을 보고 앱을 인식 및 조작
 - 장점: 인간이 사용하는 모든 앱에 적용 가능

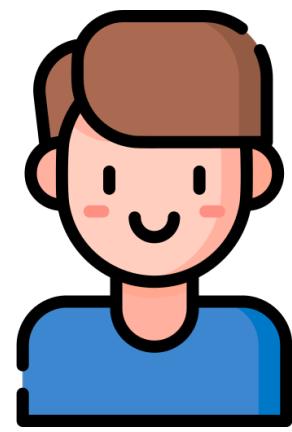
모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어
- **API 기반:** API를 사용하여 앱을 인식하고 조작
 - 장점: API를 활용하기 때문에 정밀하게 앱 상태를 인식하고 조작할 수 있음
 - 단점: API를 제공하지 않는 앱에는 적용 불가능. 확장성 떨어짐
- **GUI 기반:** 인간처럼 앱 화면 (캡처된 화면, XML)을 보고 앱을 인식 및 조작
 - 장점: 인간이 사용하는 모든 앱에 적용 가능
 - 단점: 정밀한 앱 화면 인식 및 조작이 어려움

모바일 에이전트

- 모바일 환경에서 사용자의 지시사항을 자동으로 수행하는 AI 소프트웨어
- **API 기반:** API를 사용하여 앱을 인식하고 조작
 - 장점: API를 활용하기 때문에 정밀하게 앱 상태를 인식하고 조작할 수 있음
 - 단점: API를 제공하지 않는 앱에는 적용 불가능. 확장성 떨어짐
- **GUI 기반:** 인간처럼 앱 화면 (캡처된 화면, XML)을 보고 앱을 인식 및 조작
 - 장점: 인간이 사용하는 모든 앱에 적용 가능 
 - 단점: 정밀한 앱 화면 인식 및 조작이 어려움

해결책: 논리식으로 에이전트 행동 검증하기



사용자

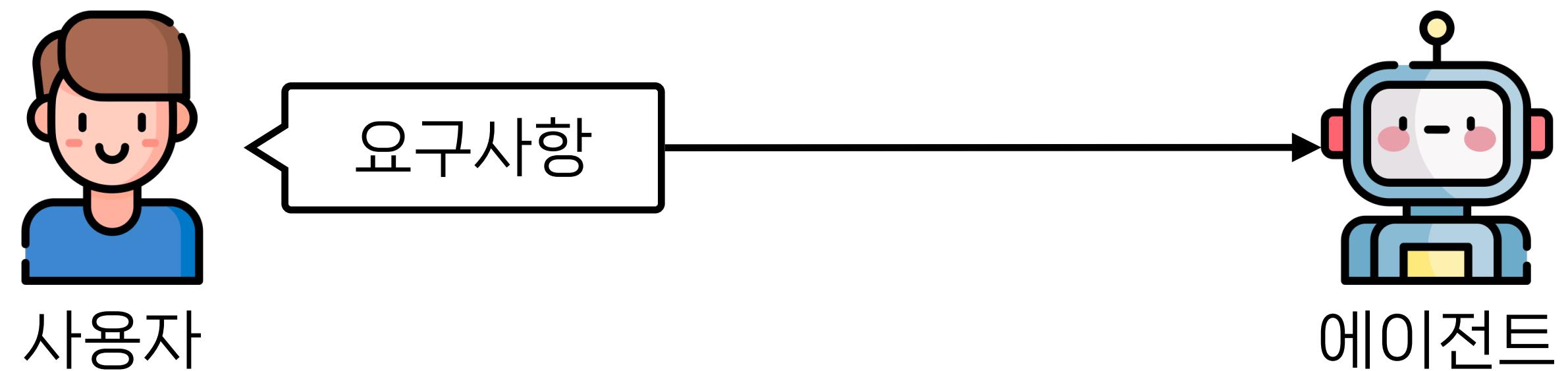
해결책: 논리식으로 에이전트 행동 검증하기



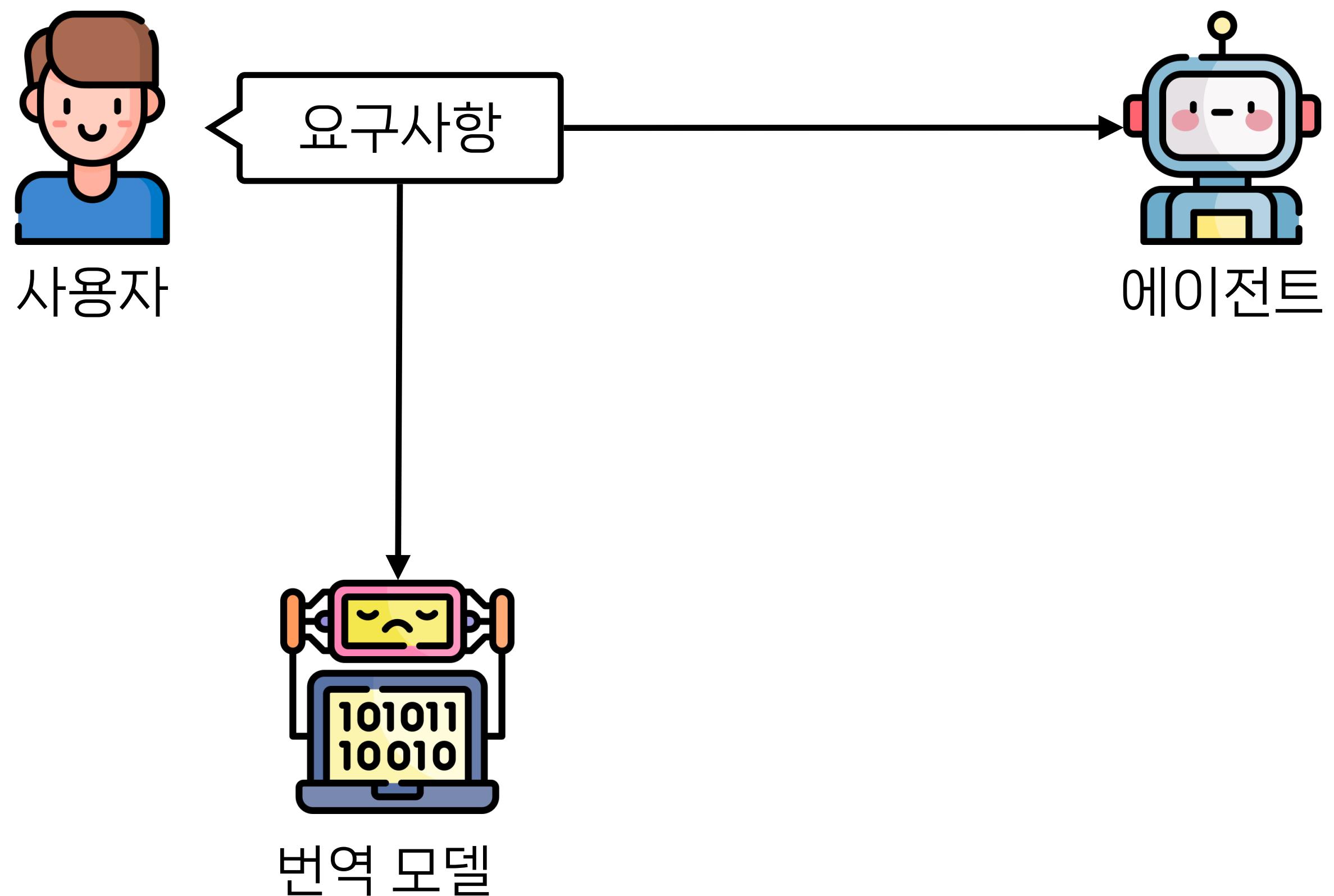
요구사항

사용자

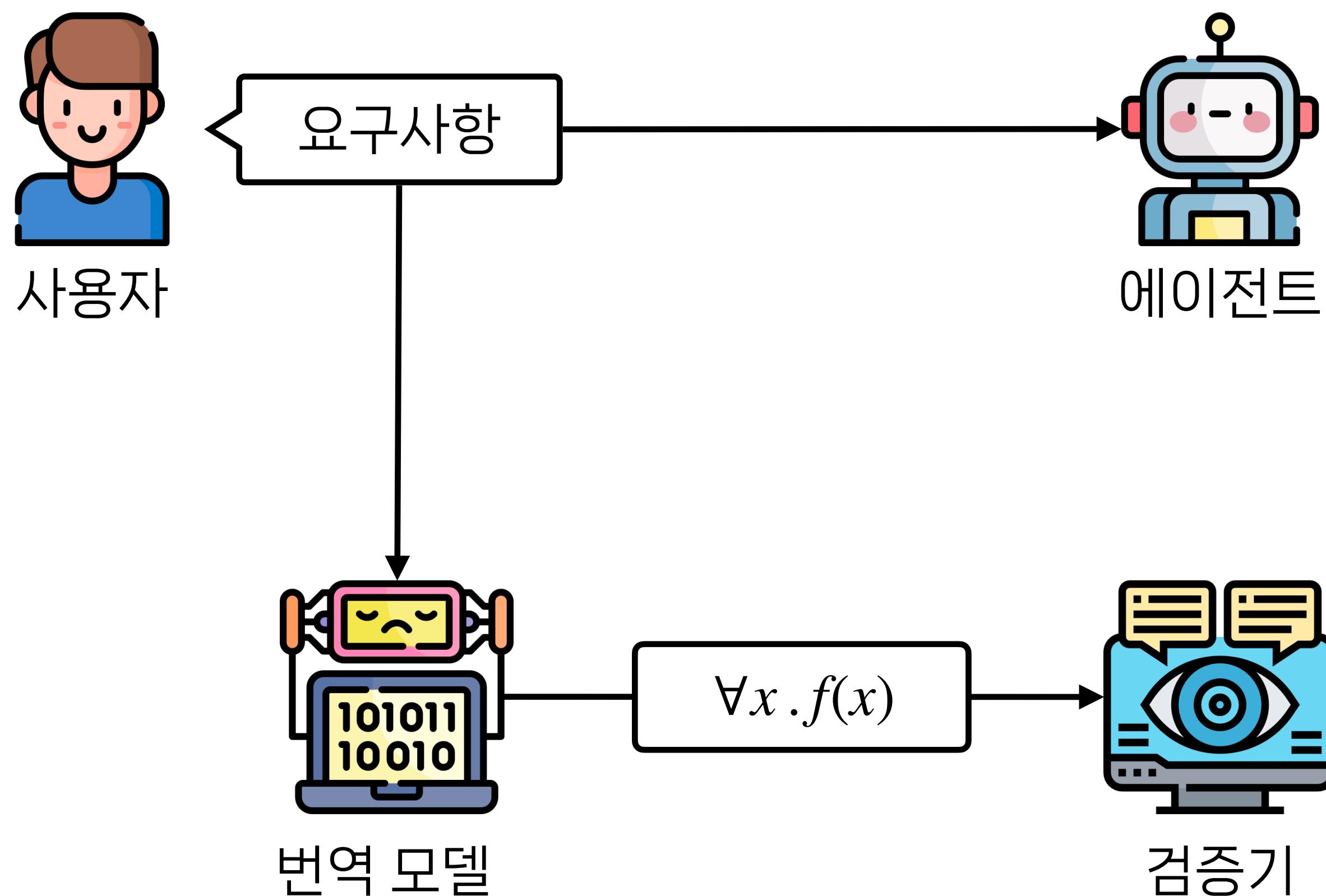
해결책: 논리식으로 에이전트 행동 검증하기



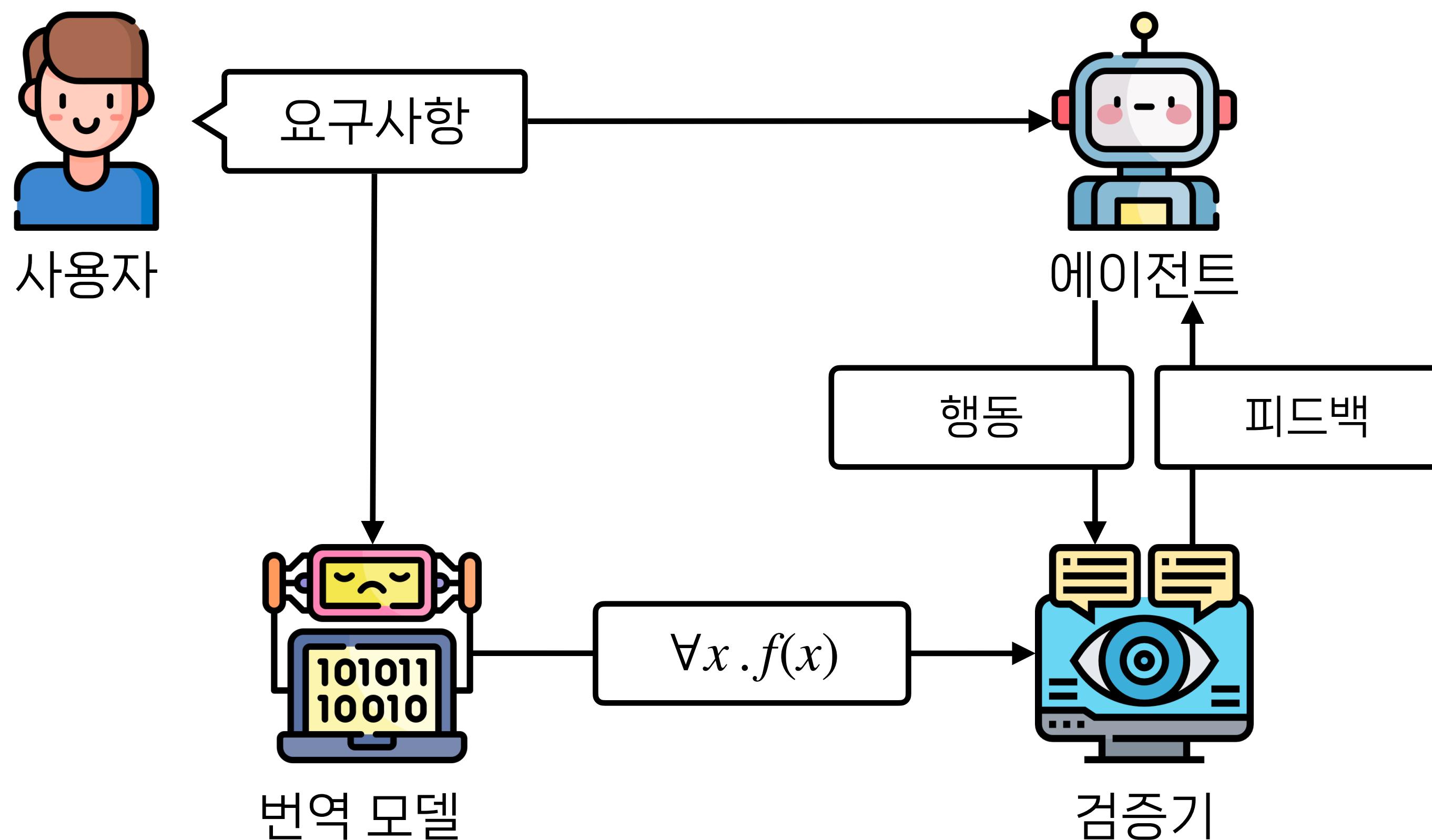
해결책: 논리식으로 에이전트 행동 검증하기



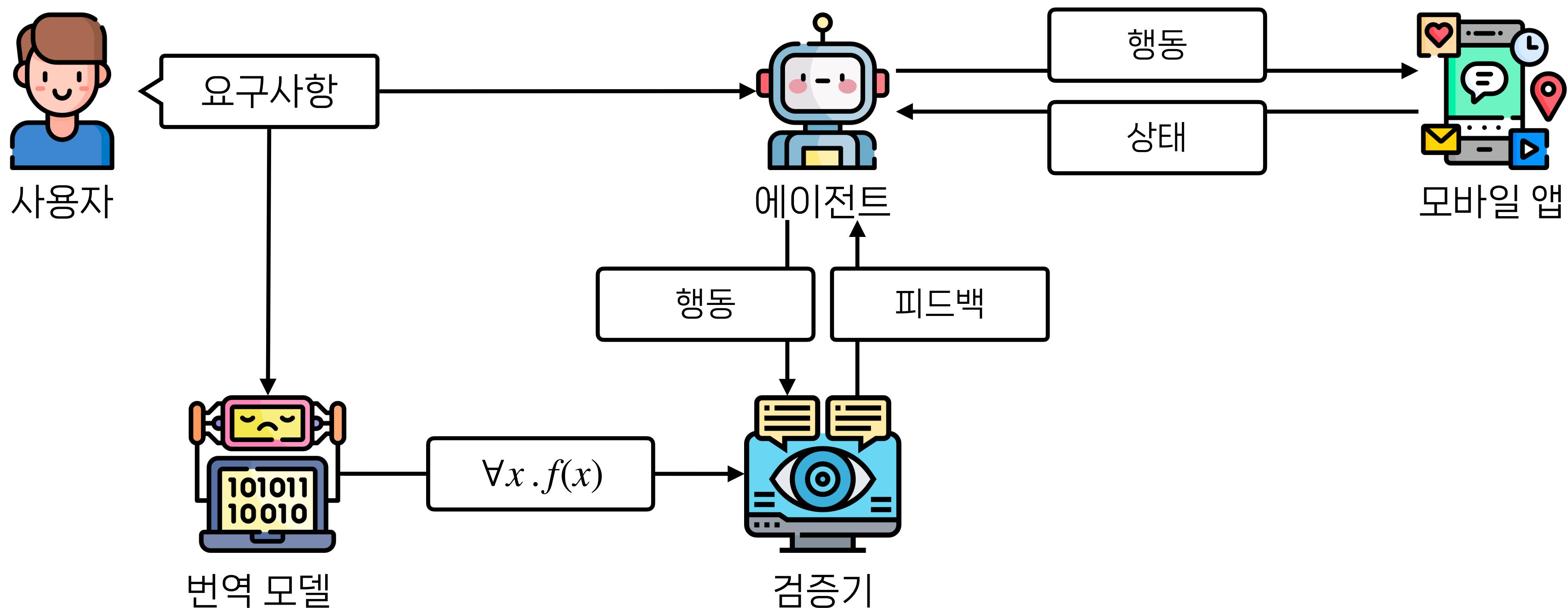
해결책: 논리식으로 에이전트 행동 검증하기



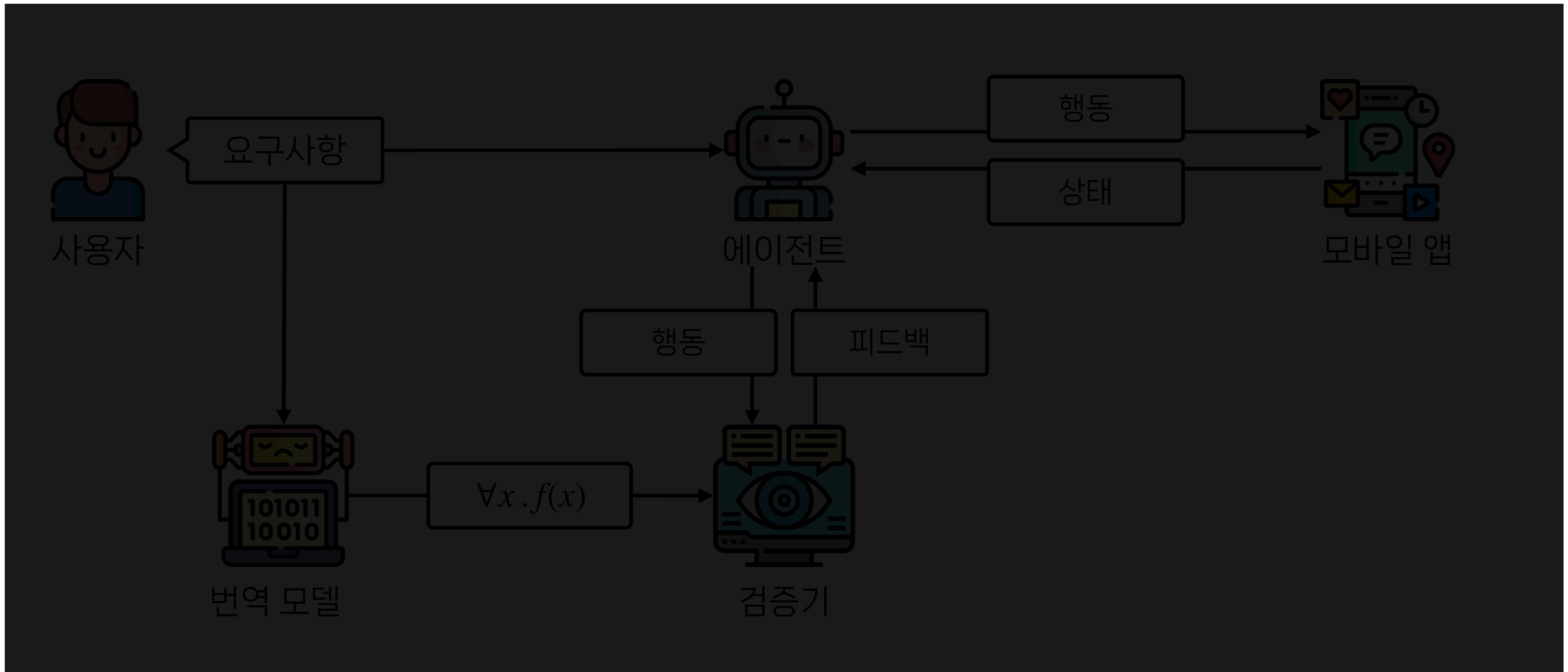
해결책: 논리식으로 에이전트 행동 검증하기



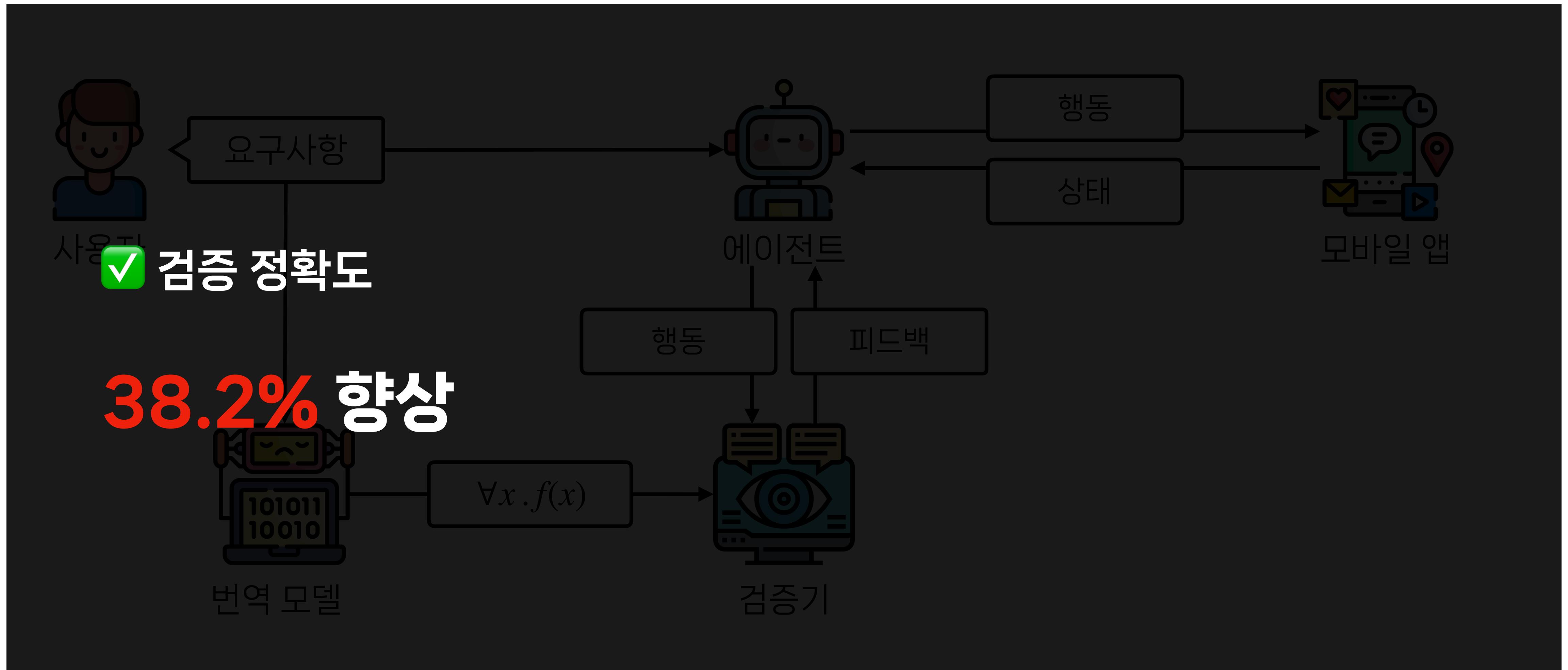
해결책: 논리식으로 에이전트 행동 검증하기



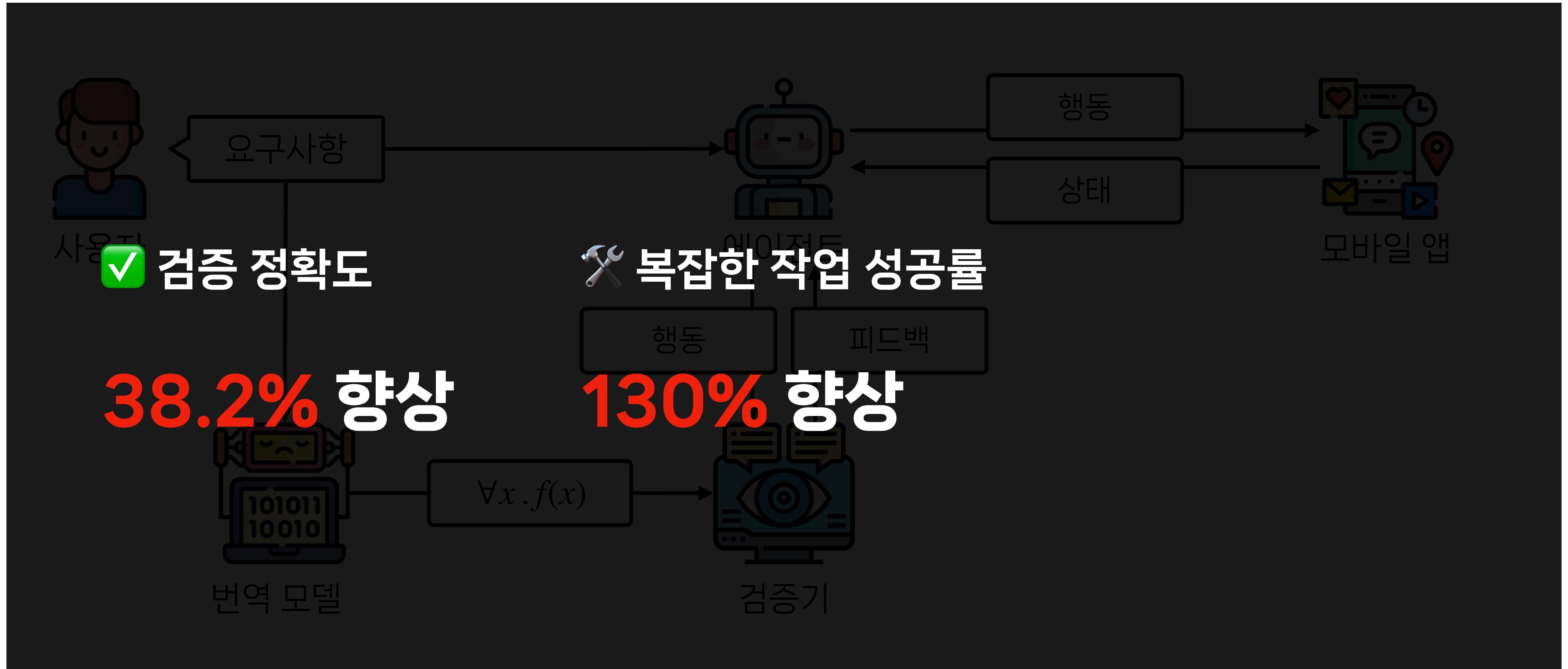
해결책: 논리식으로 에이전트 행동 검증하기



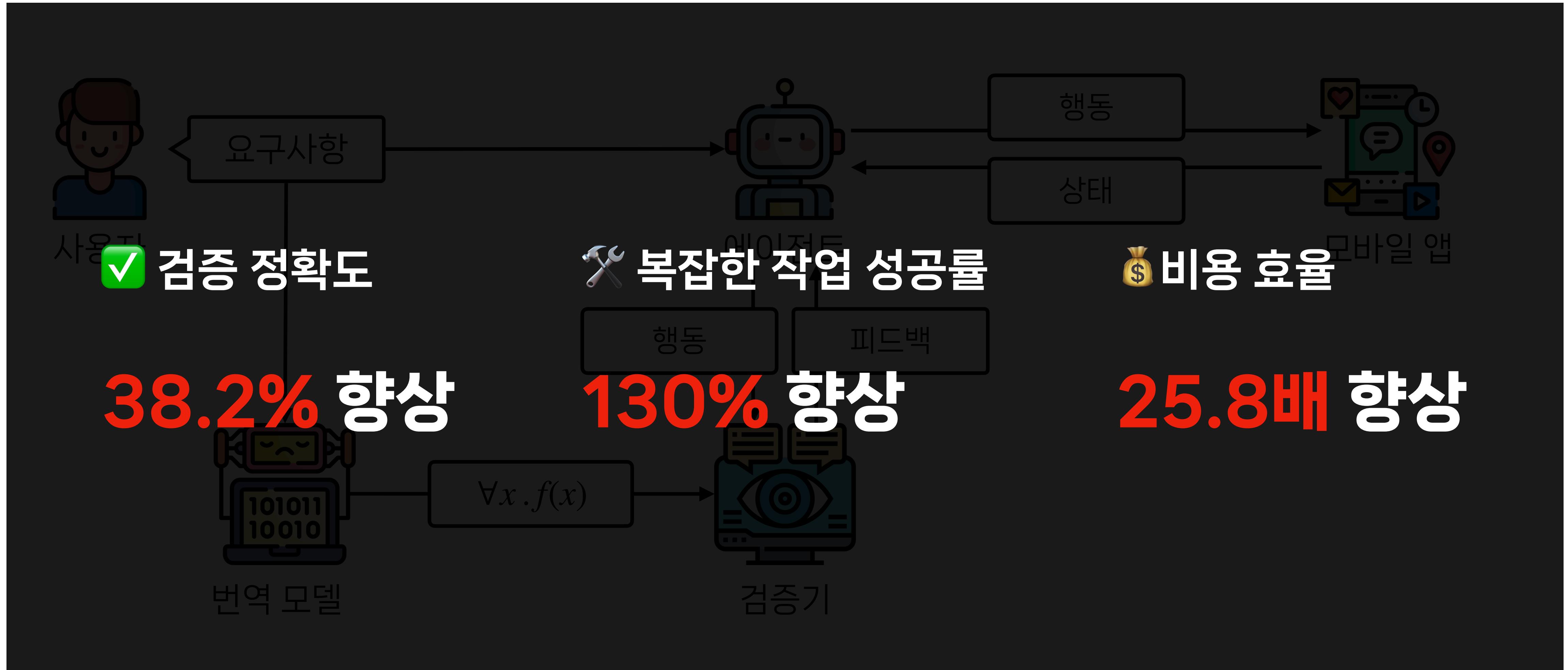
해결책: 논리식으로 에이전트 행동 검증하기



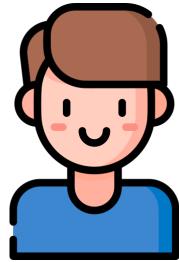
해결책: 논리식으로 에이전트 행동 검증하기



해결책: 논리식으로 에이전트 행동 검증하기



예시: 비행기 표 예약하기



사용자

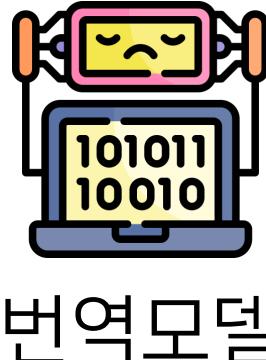
3월 7일에 출발해서 3월 9일에 돌아오는 일본 비행기 예약해줘. 가격은 30만원 이하로!

예시: 비행기 표 예약하기



사용자

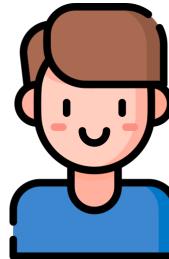
3월 7일에 출발해서 3월 9일에 돌아오는 일본 비행기 예약해줘. 가격은 30만원 이하로!



번역모델

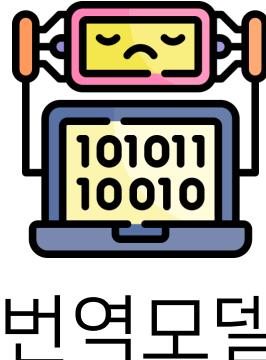
$Ticket(\text{depart} = 03.07, \text{return} = 03.09, \text{dest} = \text{Japan}, \text{price} \leq 300,000) \Rightarrow Book$

예시: 비행기 표 예약하기



사용자

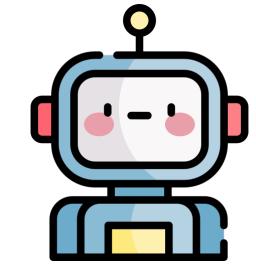
3월 7일에 출발해서 3월 9일에 돌아오는 일본 비행기 예약해줘. 가격은 30만원 이하로!



번역모델

$Ticket(\text{depart} = 03.07, \text{return} = 03.09, \text{dest} = \text{Japan}, \text{price} \leq 300,000) \Rightarrow Book$

네 알겠습니다



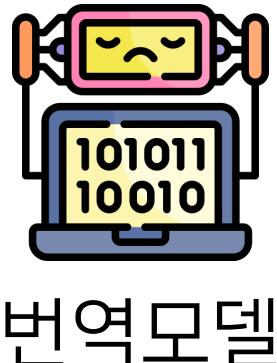
에이전트

예시: 비행기 표 예약하기



사용자

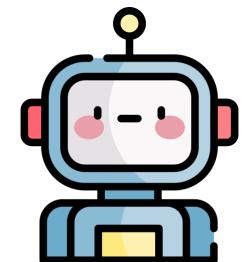
3월 7일에 출발해서 3월 9일에 돌아오는 일본 비행기 예약해줘. 가격은 30만원 이하로!



번역모델

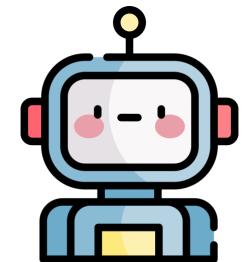
$Ticket(\text{depart} = 03.07, \text{return} = 03.09, \text{dest} = \text{Japan}, \text{price} \leq 300,000) \Rightarrow Book$

네 알겠습니다



에이전트

캘린더에서 2025.03.07을 선택



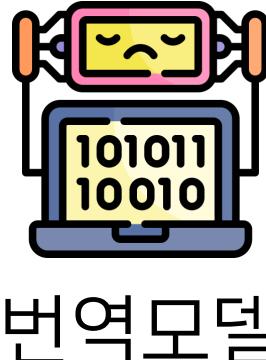
에이전트

예시: 비행기 표 예약하기



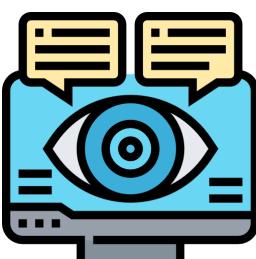
사용자

3월 7일에 출발해서 3월 9일에 돌아오는 일본 비행기 예약해줘. 가격은 30만원 이하로!



번역모델

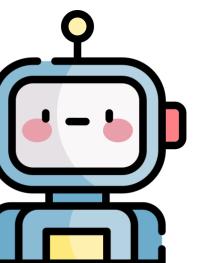
$Ticket(\text{depart} = 03.07, \text{return} = 03.09, \text{dest} = \text{Japan}, \text{price} \leq 300,000) \Rightarrow Book$



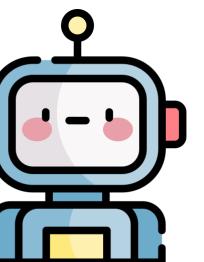
검증기

$Ticket(\text{depart} = 03.07) \Rightarrow OK$

네 알겠습니다



에이전트

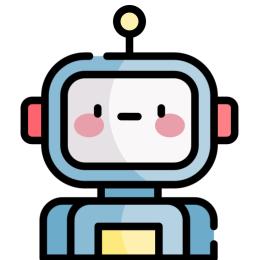


에이전트

캘린더에서 2025.03.07을 선택

예시: 비행기 표 예약하기

캘린더에서 2025.03.11을 선택



에이전트

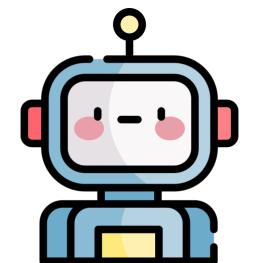
예시: 비행기 표 예약하기



검증기

Ticket(return = 03.11) ⇒ ERROR

캘린더에서 2025.03.11을 선택



에이전트

예시: 비행기 표 예약하기



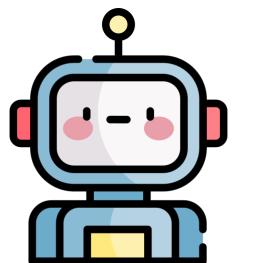
검증기



검증기

Ticket(return = 03.11) \Rightarrow ERROR

캘린더에서 2025.03.11을 선택



에이전트

예시: 비행기 표 예약하기



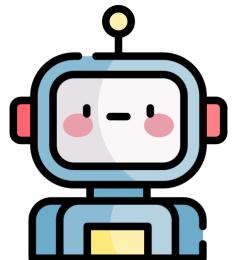
검증기



검증기

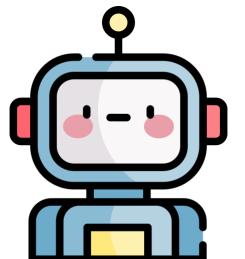
$Ticket(return = 03.11) \Rightarrow ERROR$

캘린더에서 2025.03.11을 선택



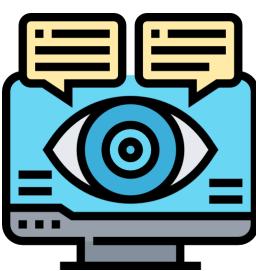
에이전트

캘린더에서 2025.03.09을 선택



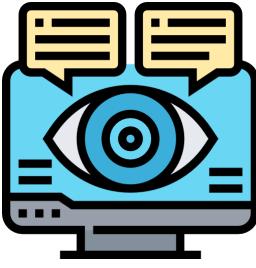
에이전트

예시: 비행기 표 예약하기



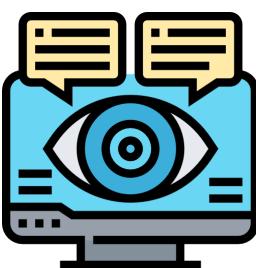
검증기

Ticket(return = 03.11) \Rightarrow ERROR



검증기

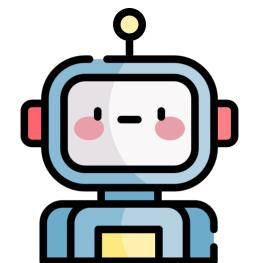
피드백: *Ticket(return = 2025.03.11) \neq Ticket(return = 2025.03.09)*



검증기

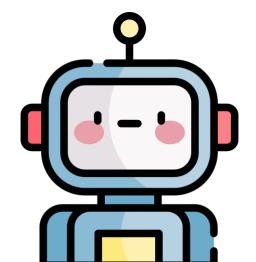
Ticket(return = 03.09) \Rightarrow OK

캘린더에서 2025.03.11을 선택



에이전트

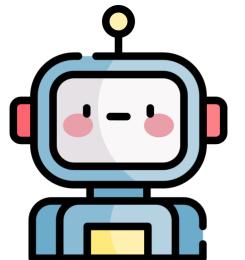
캘린더에서 2025.03.09을 선택



에이전트

예시: 비행기 표 예약하기

비행기 예약 전 조건 확인



에이전트

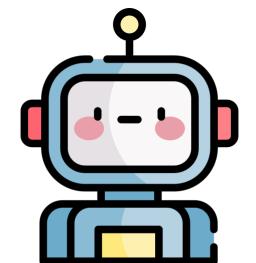
예시: 비행기 표 예약하기



검증기

현재 상태: *Ticket(depart = 03.07, return = 03.09, dest = Japan, price = 200,000)*

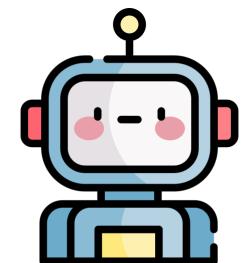
비행기 예약 전 조건 확인



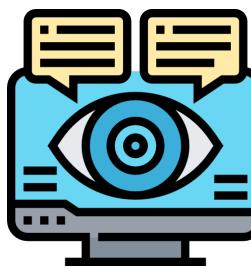
에이전트

예시: 비행기 표 예약하기

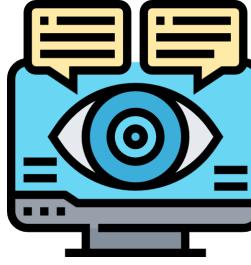
비행기 예약 전 조건 확인



에이전트



검증기



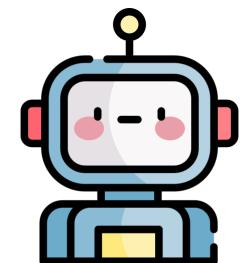
검증기

현재 상태: $Ticket(depart = 03.07, return = 03.09, dest = \text{Japan}, price = 200,000)$

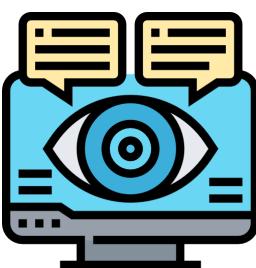
$Ticket(depart = 03.07, return = 03.09, dest = \text{Japan}, price = 200,000) \Rightarrow Book$

예시: 비행기 표 예약하기

비행기 예약 전 조건 확인



에이전트



검증기

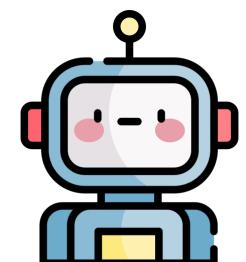
현재 상태: $Ticket(depart = 03.07, return = 03.09, dest = \text{Japan}, price = 200,000)$



검증기

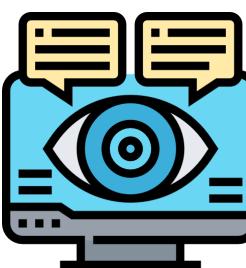
$Ticket(depart = 03.07, return = 03.09, dest = \text{Japan}, price = 200,000) \Rightarrow Book$

모든 조건이 만족되었으므로 예약을 시도합니다



에이전트

예시: 비행기 표 예약하기



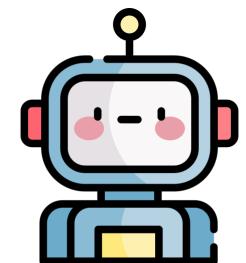
검증기



검증기

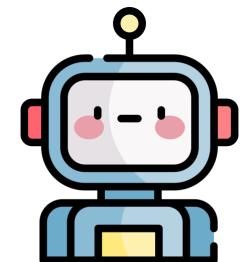
현재 상태: $Ticket(depart = 03.07, return = 03.09, dest = \text{Japan}, price = 200,000)$

비행기 예약 전 조건 확인



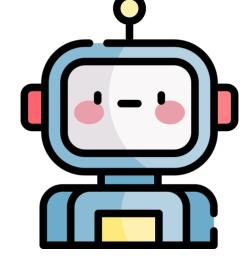
에이전트

모든 조건이 만족되었으므로 예약을 시도합니다



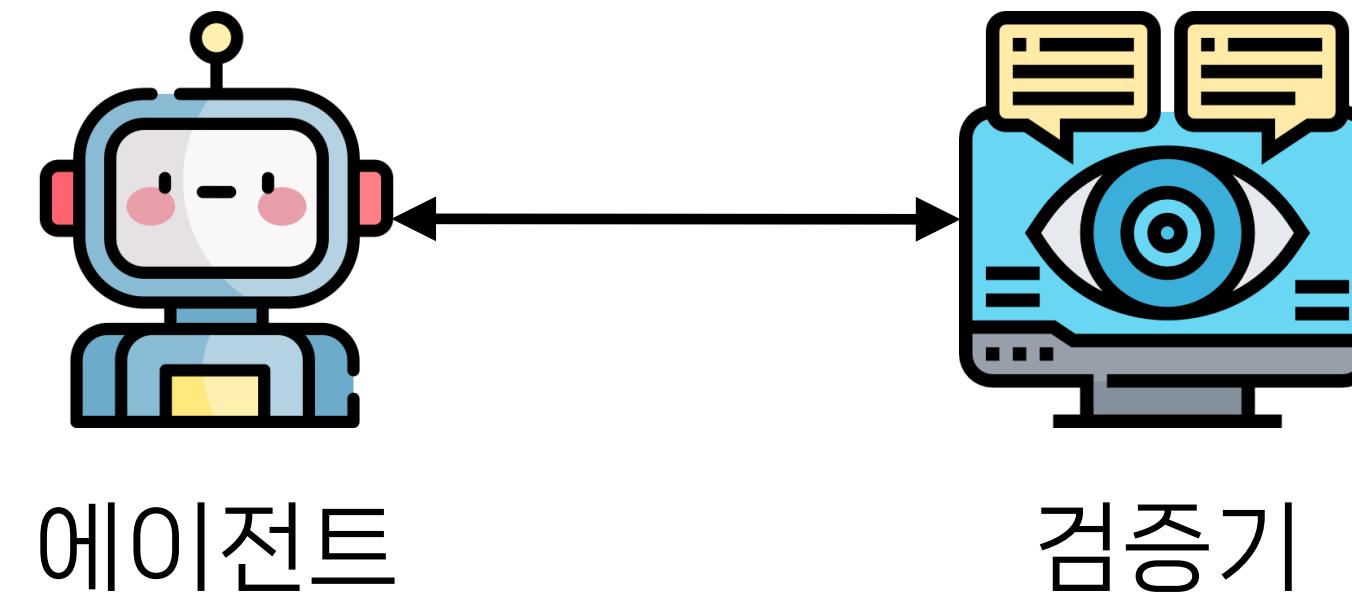
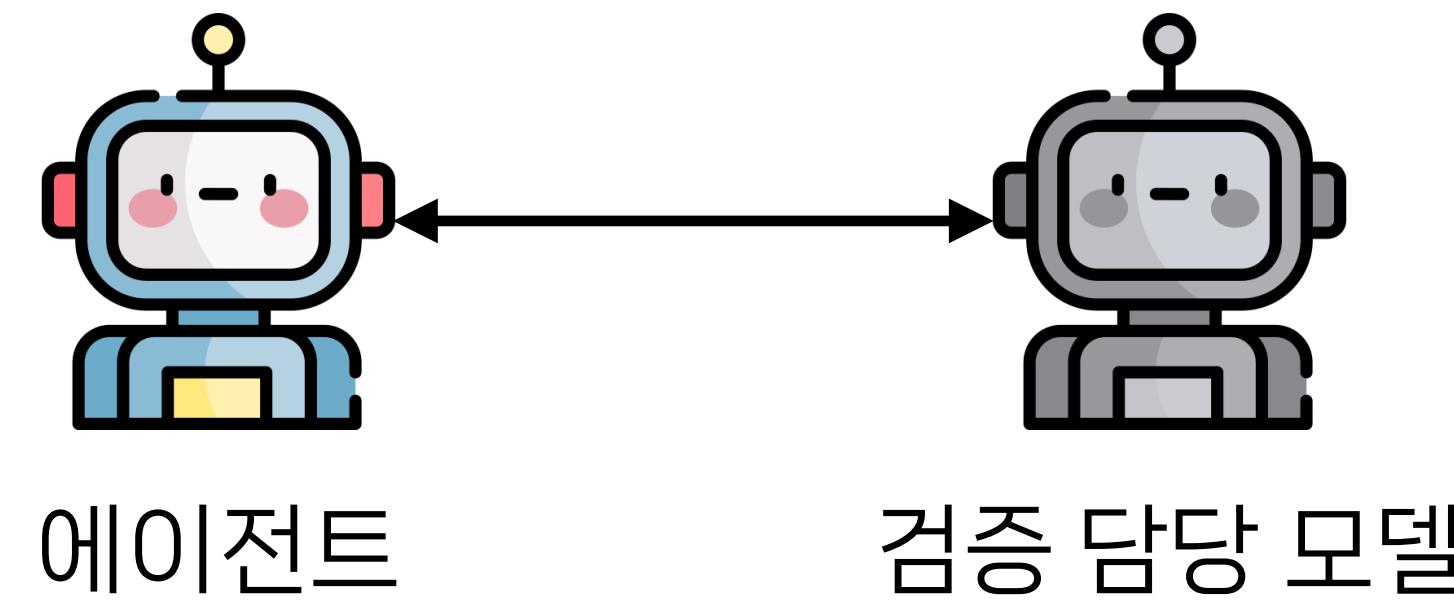
에이전트

완료되었습니다

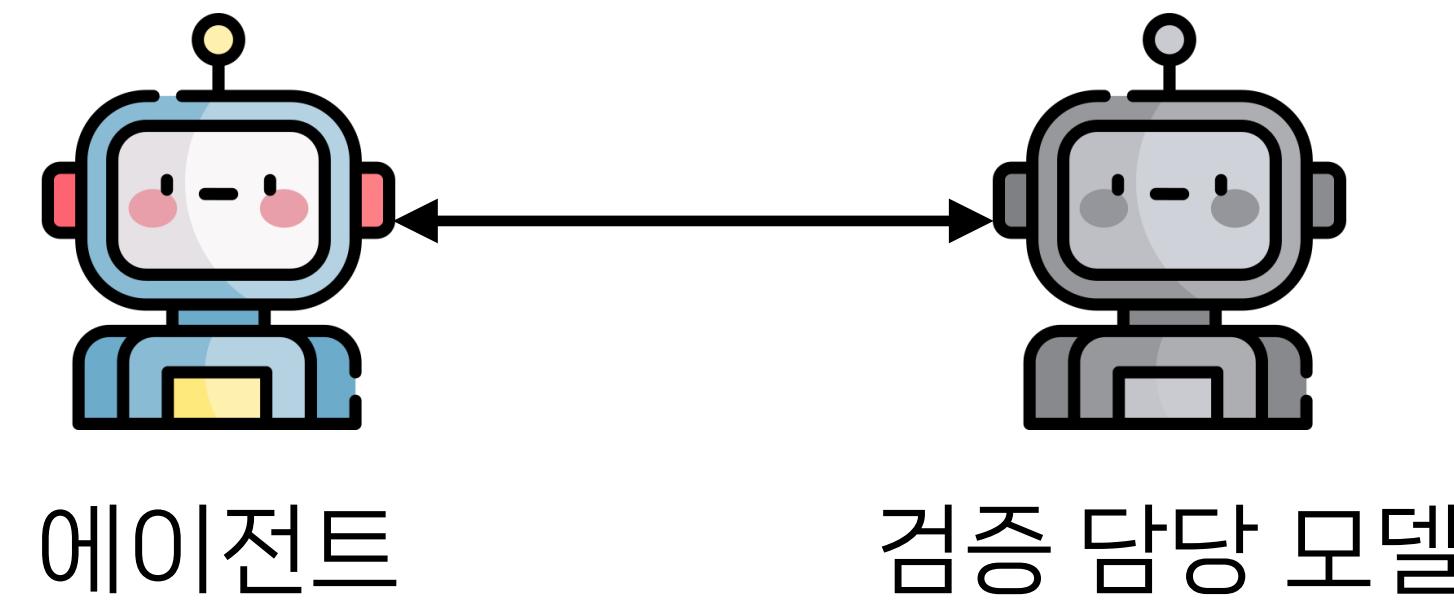


에이전트

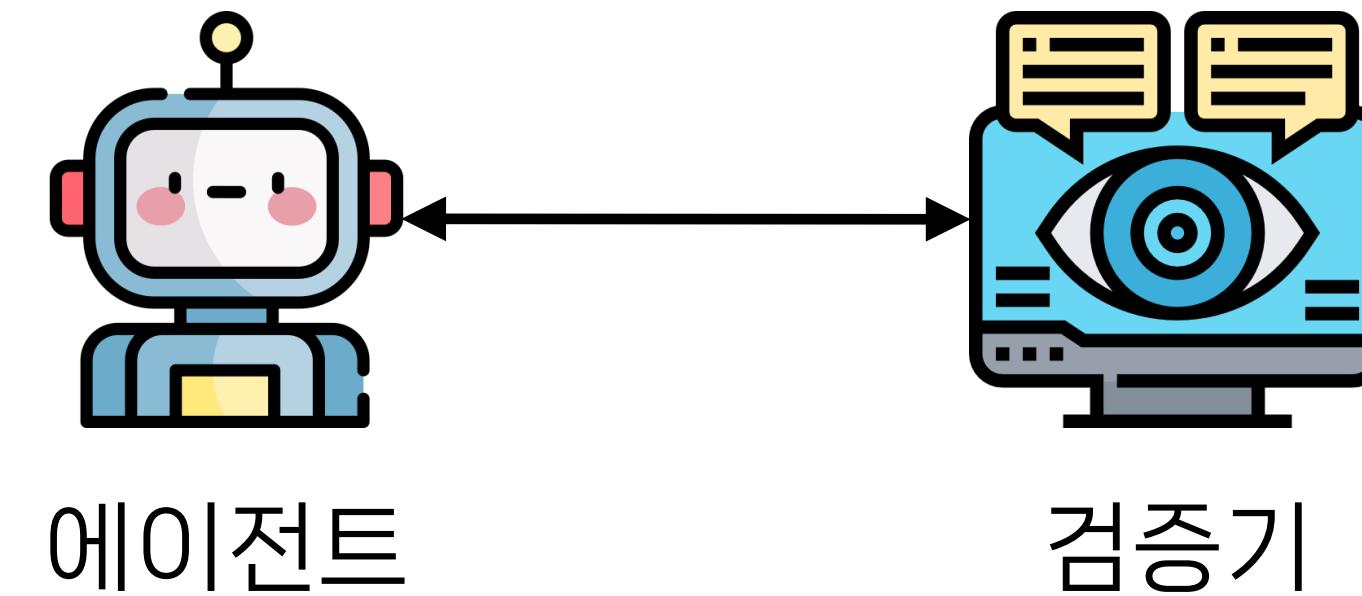
대형 멀티모달 모델 (LMM) 기반 검증 방법과 비교



대형 멀티모달 모델 (LMM) 기반 검증 방법과 비교

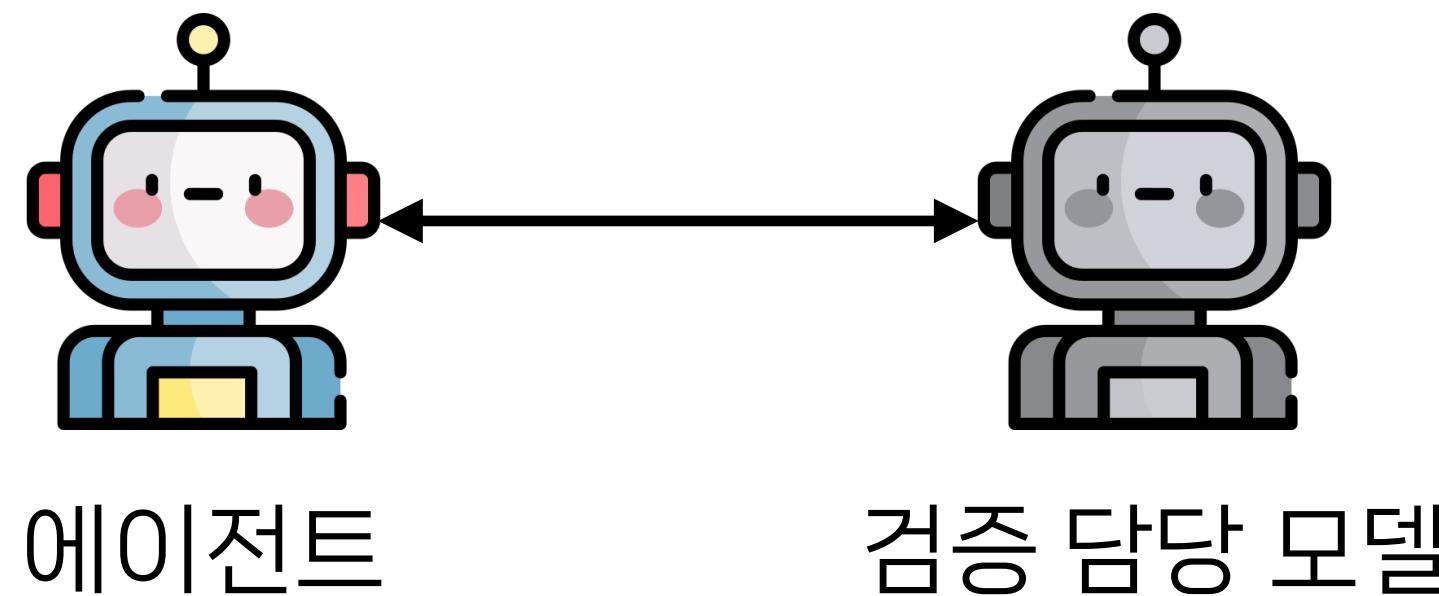


1. 검증 자체에서도 **환각 발생** 가능

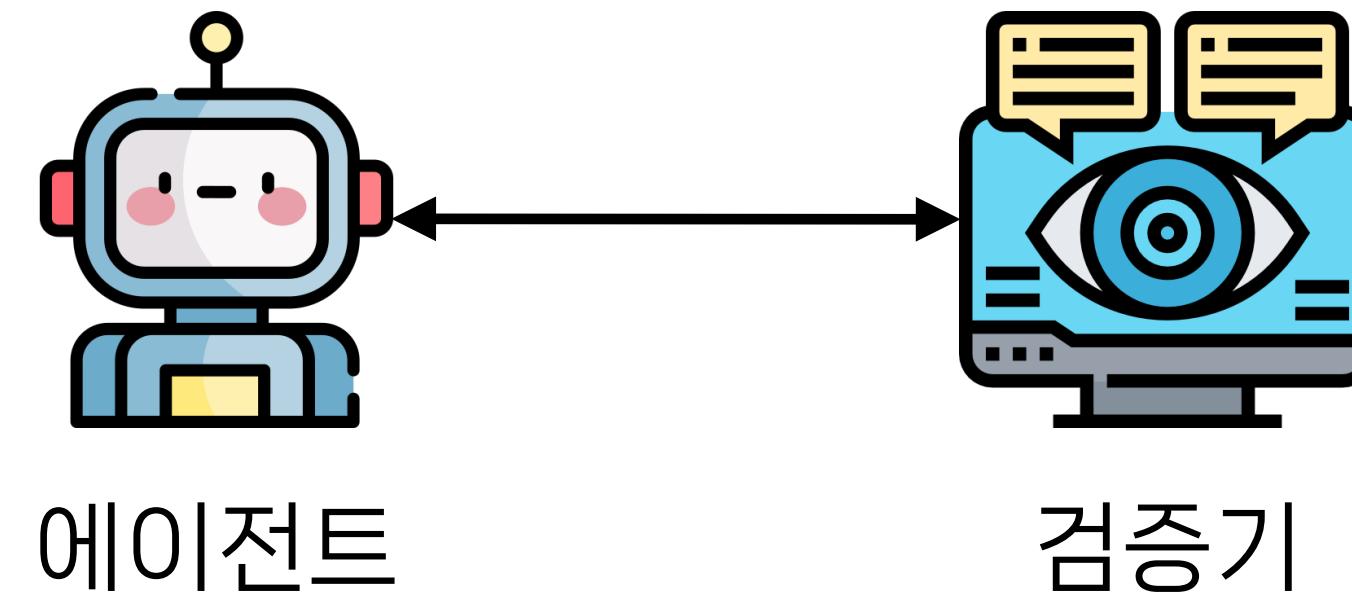


1. 검증에서는 **환각 없음**

대형 멀티모달 모델 (LMM) 기반 검증 방법과 비교

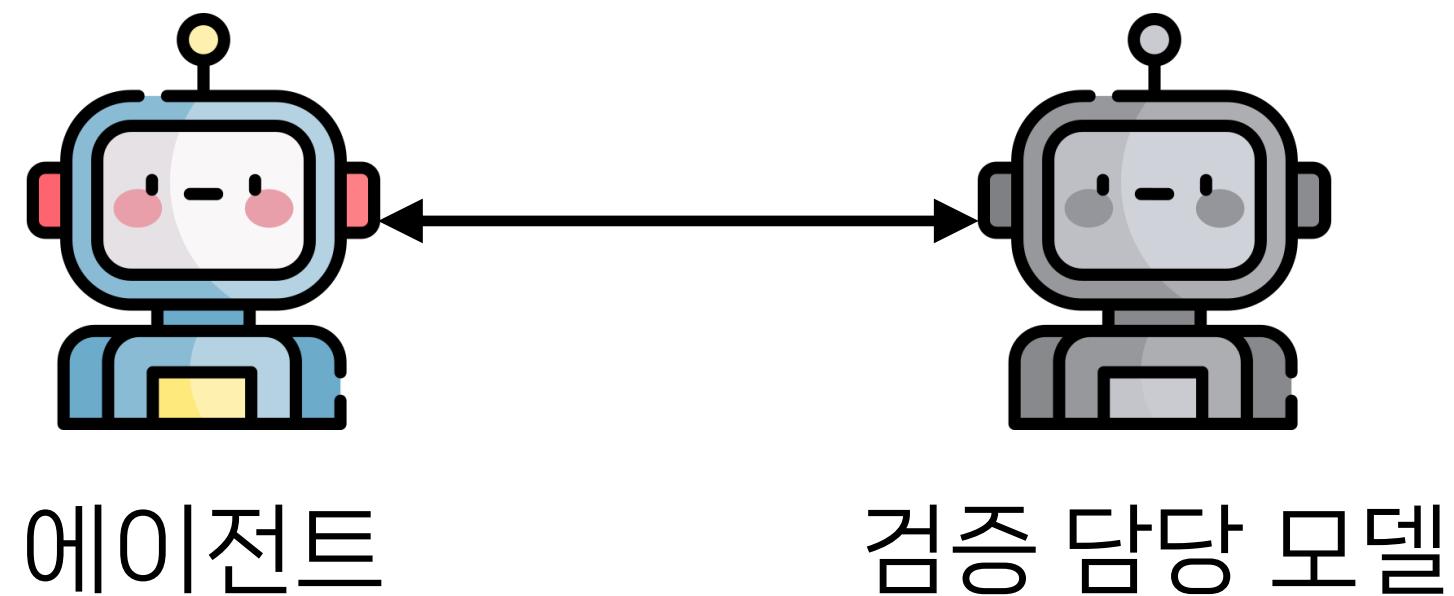


1. 검증 자체에서도 **환각 발생** 가능
2. **일관되지 않은** 검증 결과

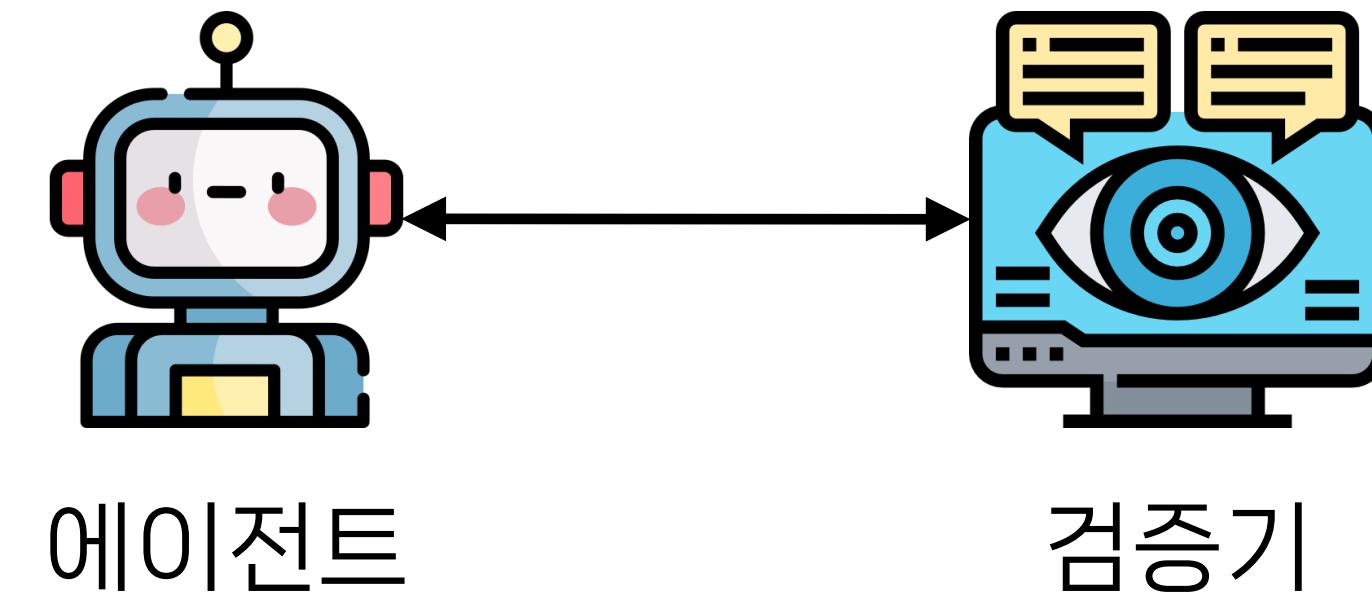


1. 검증에서는 **환각 없음**
2. **일관된** 검증 결과

대형 멀티모달 모델 (LMM) 기반 검증 방법과 비교

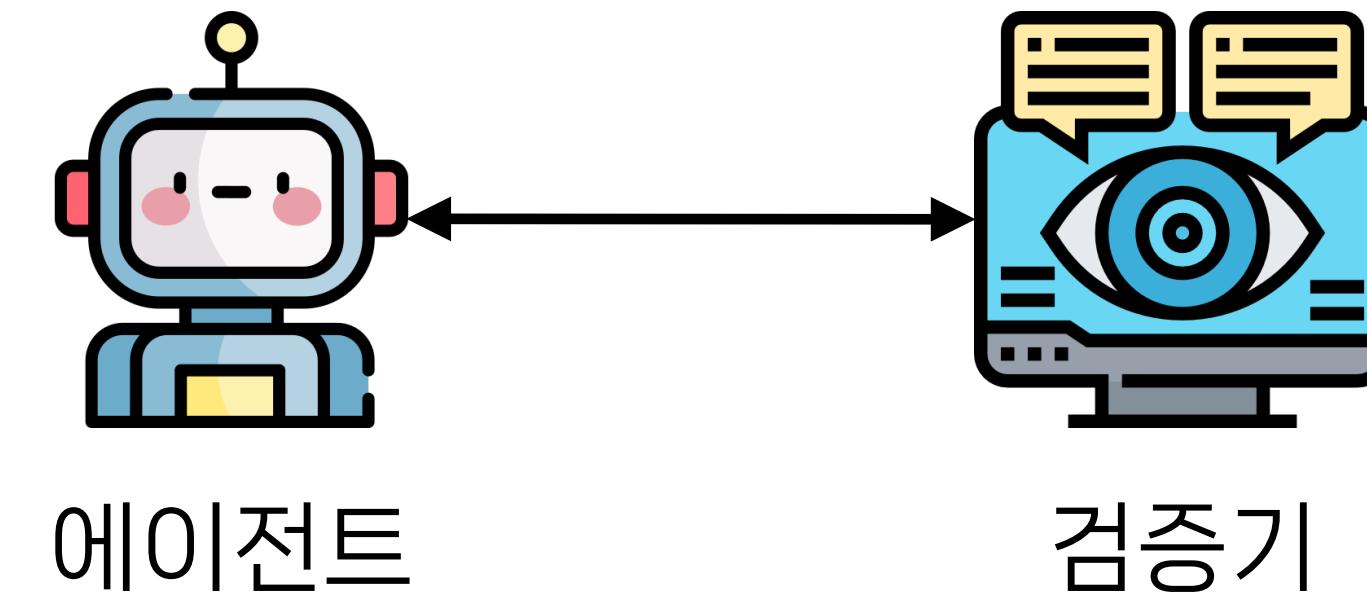
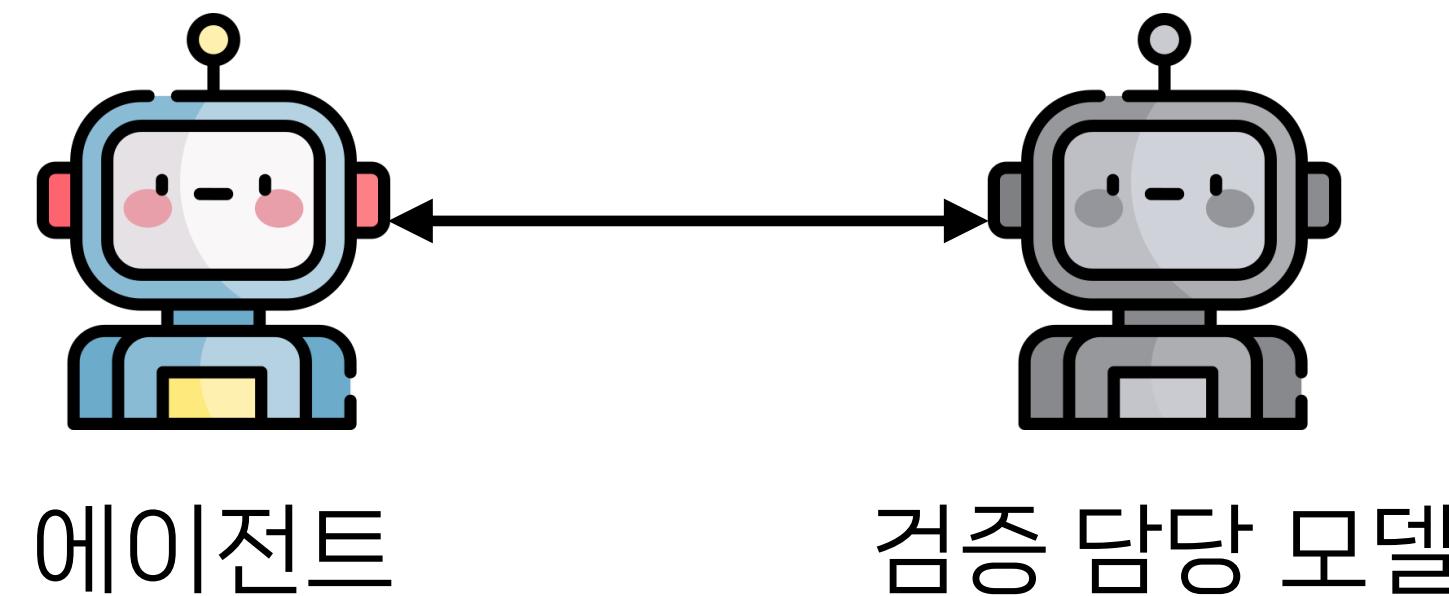


1. 검증 자체에서도 환각 발생 가능
 2. 일관되지 않은 검증 결과
 3. 틀린 이유 설명도 LMM에 의존



1. 검증에서는 **환각 없음**
 2. **일관된** 검증 결과
 3. 틀린 이유도 **수학적으로 추론 가능**

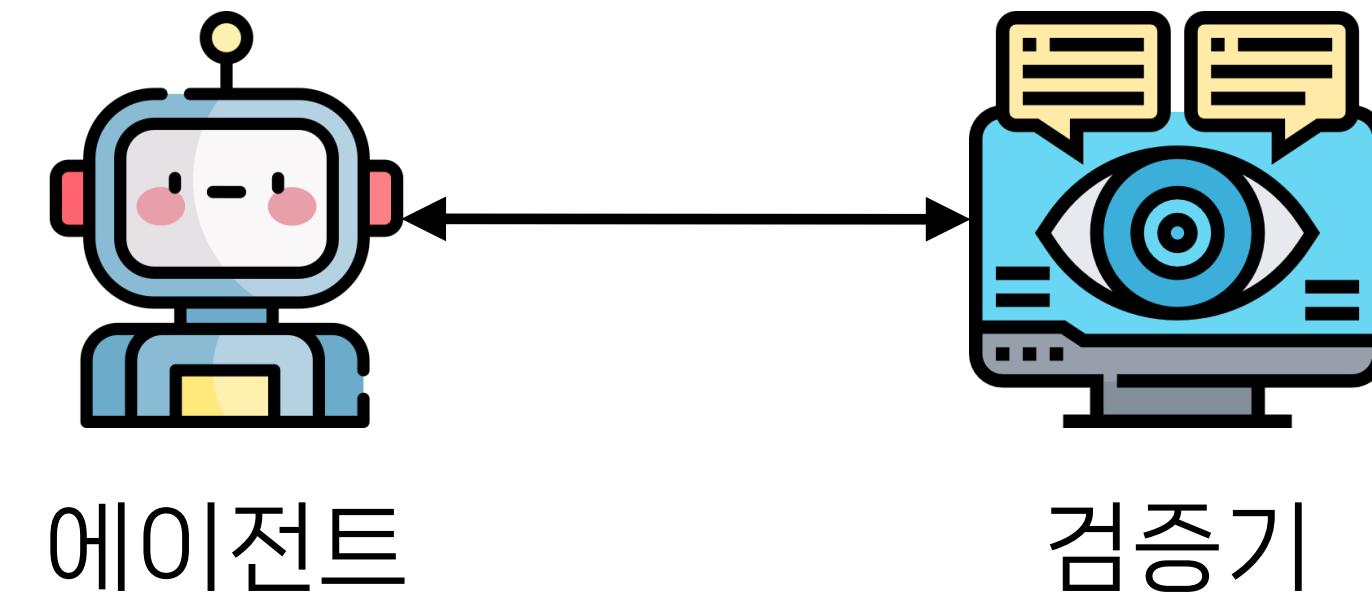
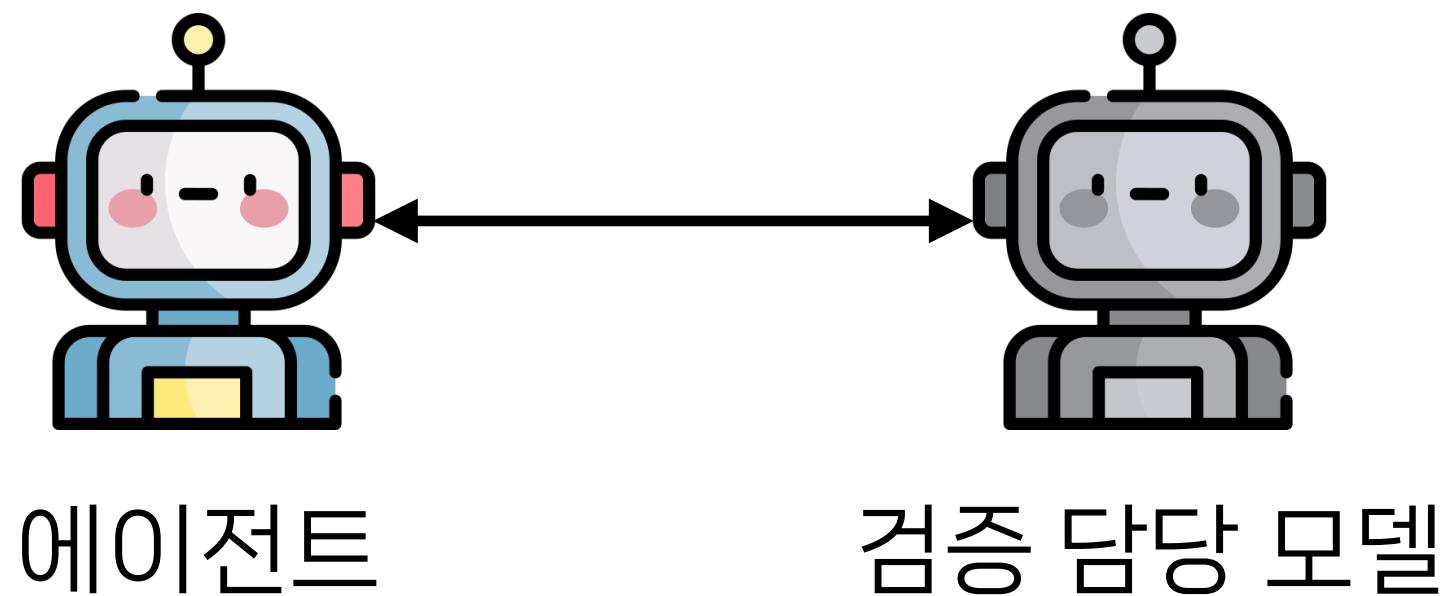
대형 멀티모달 모델 (LMM) 기반 검증 방법과 비교



1. 검증 자체에서도 **환각 발생** 가능
2. **일관되지 않은** 검증 결과
3. 틀린 이유 설명도 **LMM에 의존**
4. 작업 길이가 길어졌을 때 **정확도 저하**

1. 검증에서는 **환각 없음**
2. **일관된** 검증 결과
3. 틀린 이유도 **수학적으로 추론** 가능
4. 작업 길이에 따른 **정확도 저하 없음**

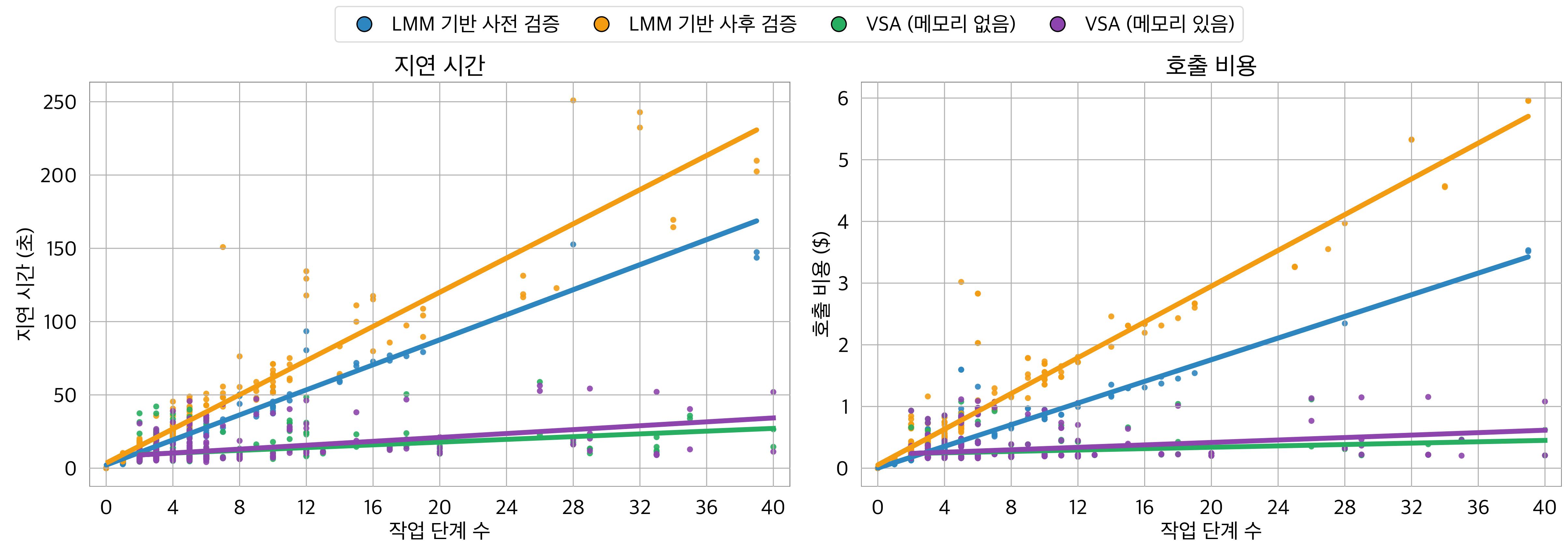
대형 멀티모달 모델 (LMM) 기반 검증 방법과 비교



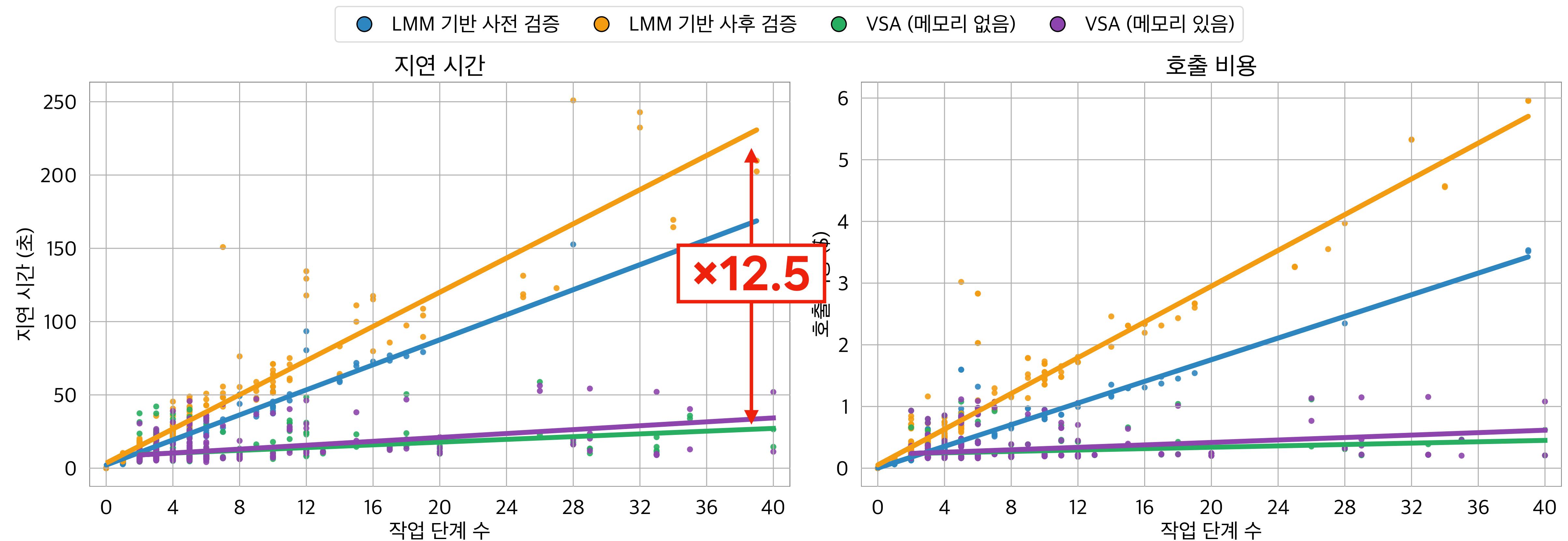
1. 검증 자체에서도 환각 발생 가능
 2. 일관되지 않은 검증 결과
 3. 틀린 이유 설명도 LMM에 의존
 4. 작업 길이가 길어졌을 때 정확도 저하
 5. 짧은 LMM 호출로 높은 비용 발생

1. 검증에서는 환각 없음
 2. 일관된 검증 결과
 3. 틀린 이유도 수학적으로 추론 가능
 4. 작업 길이에 따른 정확도 저하 없음
 5. 검증 과정에서는 LMM 호출 없음

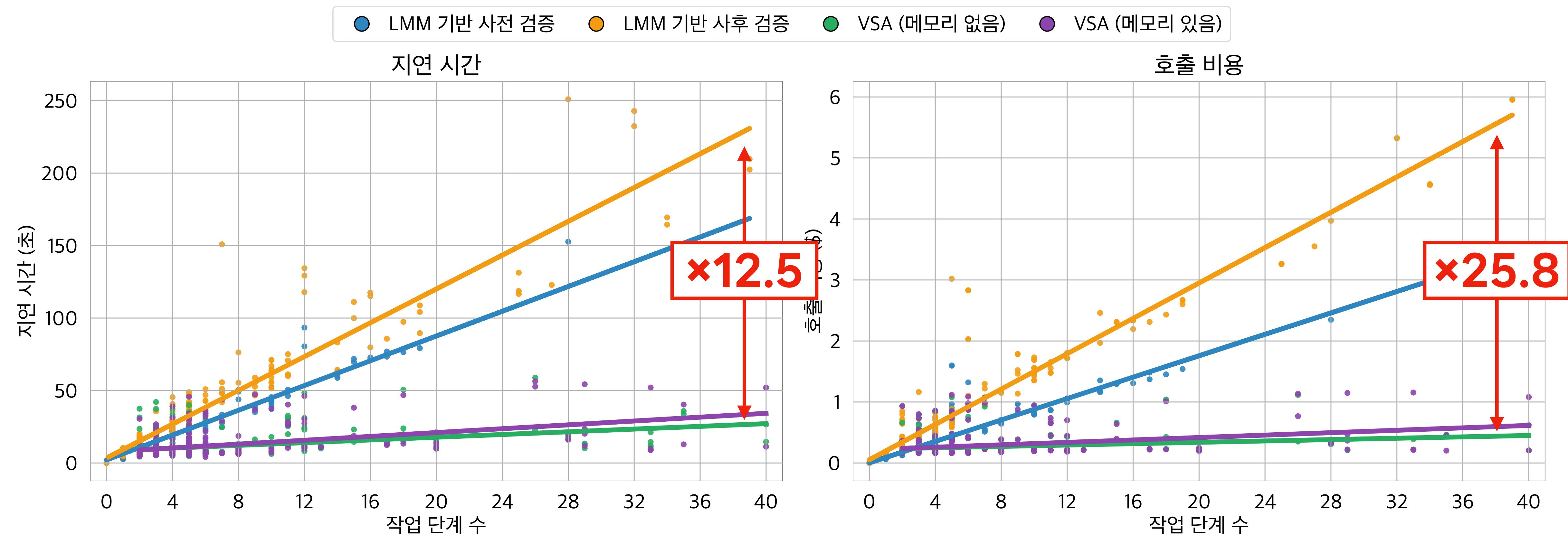
LMM 기반 검증과 비용 차이



LMM 기반 검증과 비용 차이



LMM 기반 검증과 비용 차이



해결해야 할 문제들

해결해야 할 문제들

1. 어떻게 자연어를 논리식으로 바꿀 것인가?

해결해야 할 문제들

1. 어떻게 자연어를 논리식으로 바꿀 것인가?

- ▶ LMM + 구문 분석 + 유형 검사 + 일관성 검사 + 메모리

해결해야 할 문제들

1. 어떻게 자연어를 논리식으로 바꿀 것인가?
 - ▶ LMM + 구문 분석 + 유형 검사 + 일관성 검사 + 메모리
2. 어떤 언어로 작성된 논리식을 사용할 것인가?

해결해야 할 문제들

1. 어떻게 자연어를 논리식으로 바꿀 것인가?

- ▶ LMM + 구문 분석 + 유형 검사 + 일관성 검사 + 메모리

2. 어떤 언어로 작성된 논리식을 사용할 것인가?

- ▶ 모바일 앱에 최적화된 도메인 특화 언어

해결해야 할 문제들

1. 어떻게 자연어를 논리식으로 바꿀 것인가?
 - ▶ LMM + 구문 분석 + 유형 검사 + 일관성 검사 + 메모리
2. 어떤 언어로 작성된 논리식을 사용할 것인가?
 - ▶ 모바일 앱에 최적화된 도메인 특화 언어
3. 실시간으로 바뀌는 앱 상태를 어떻게 논리식에 반영하고 검증하는가?

해결해야 할 문제들

1. 어떻게 자연어를 논리식으로 바꿀 것인가?

- ▶ LMM + 구문 분석 + 유형 검사 + 일관성 검사 + 메모리

2. 어떤 언어로 작성된 논리식을 사용할 것인가?

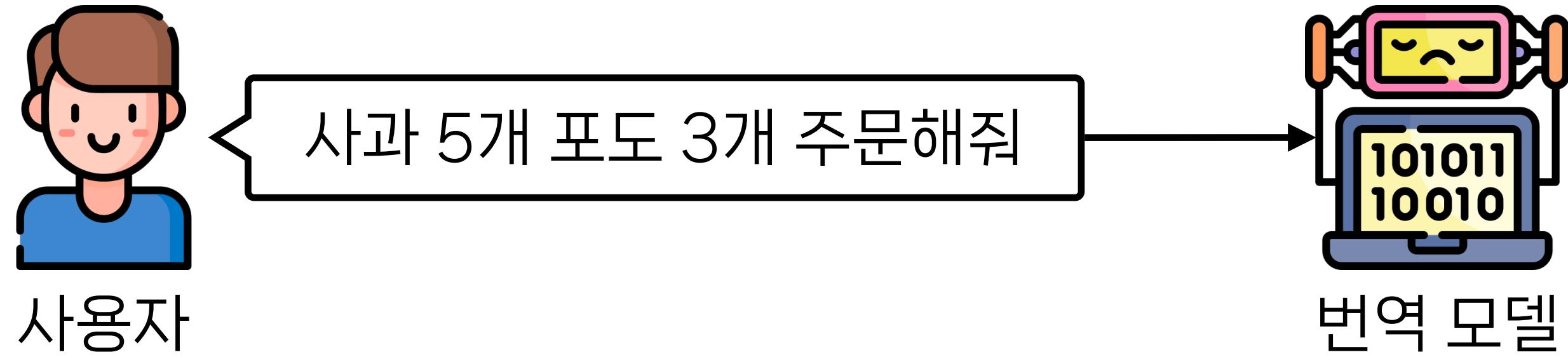
- ▶ 모바일 앱에 최적화된 도메인 특화 언어

3. 실시간으로 바뀌는 앱 상태를 어떻게 논리식에 반영하고 검증하는가?

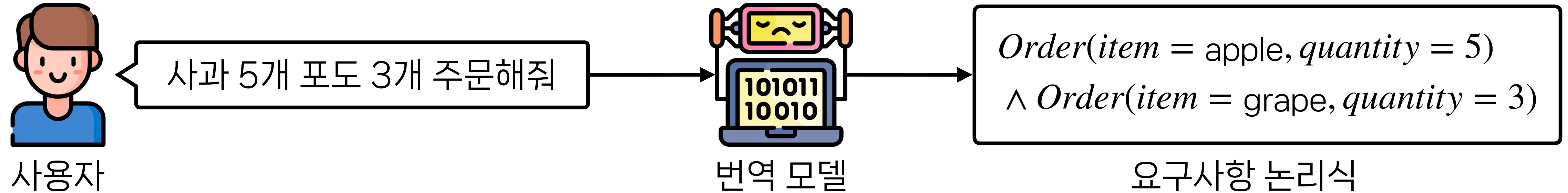
- ▶ 개발자가 기존 이벤트 처리기 (Event Handler)에 상태 업데이트 규칙 추가

문제 1: 어떻게 자연어를 논리식으로 바꿀까?

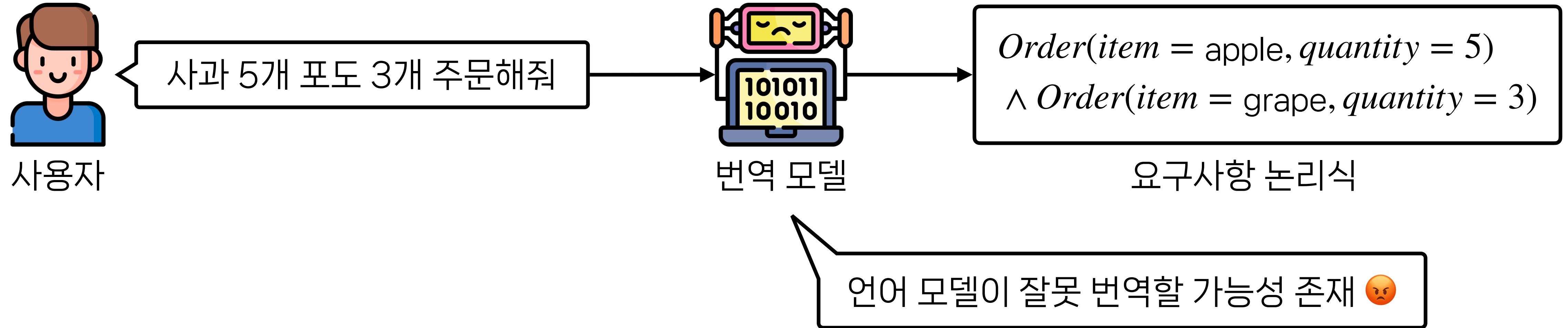
문제 1: 어떻게 자연어를 논리식으로 바꿀까?



문제 1: 어떻게 자연어를 논리식으로 바꿀까?



문제 1: 어떻게 자연어를 논리식으로 바꿀까?

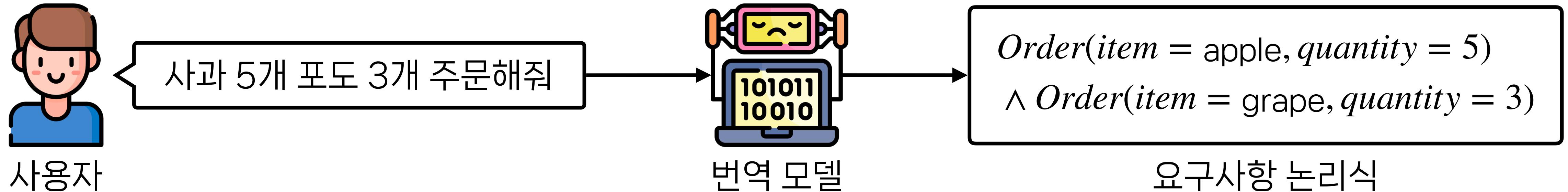


전략 1: 구문 분석 (Parsing)

- 번역 모델이 내놓은 논리식이 올바른 문법을 가지고 있는지 확인

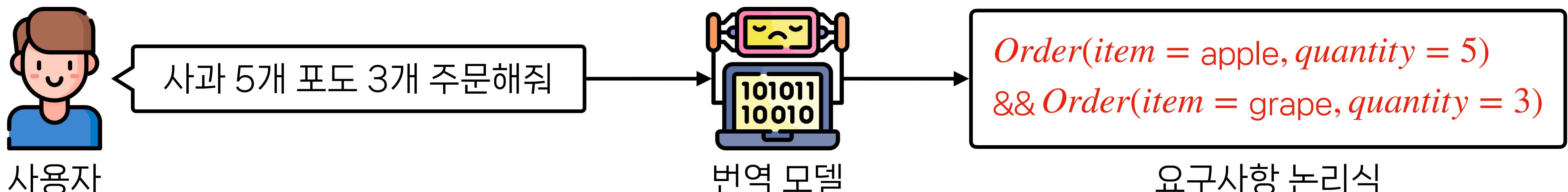
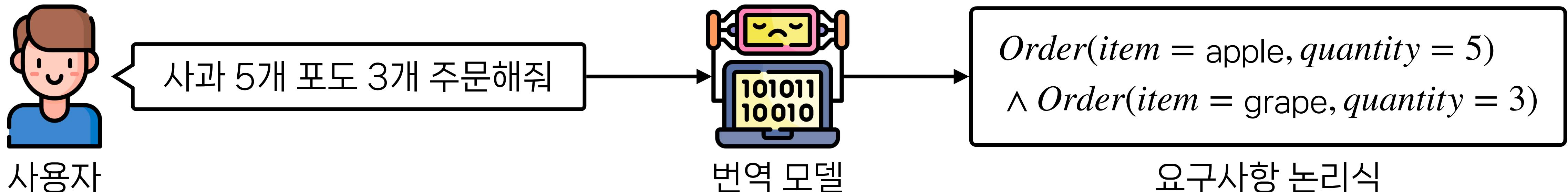
전략 1: 구문 분석 (Parsing)

- 번역 모델이 내놓은 논리식이 올바른 문법을 가지고 있는지 확인



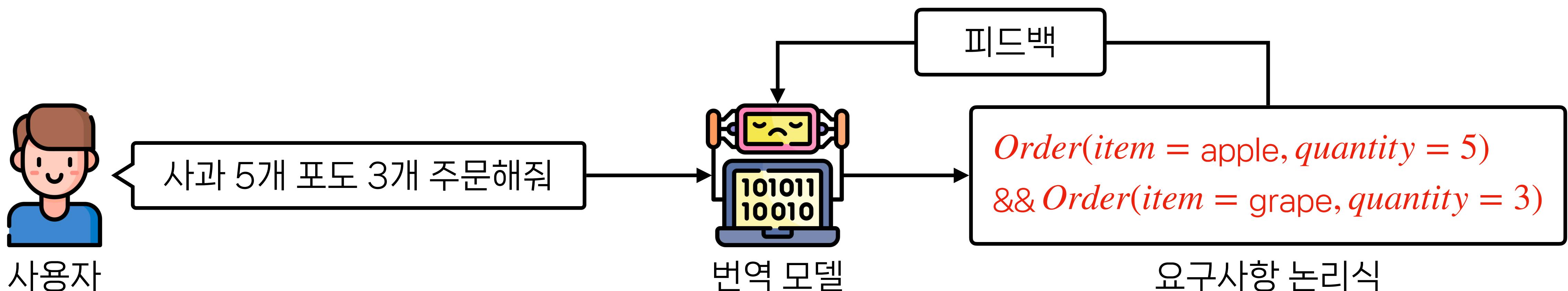
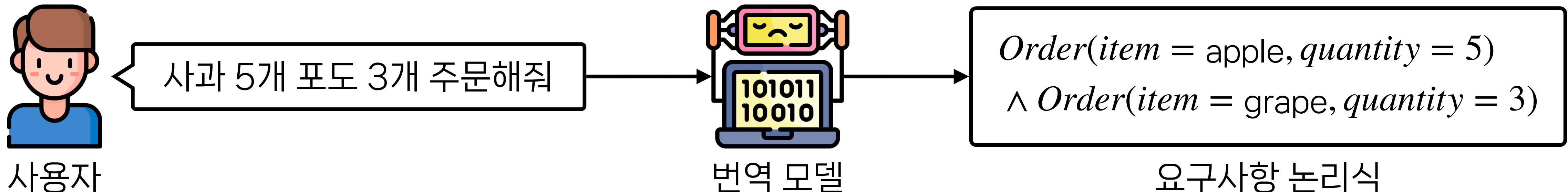
전략 1: 구문 분석 (Parsing)

- 번역 모델이 내놓은 논리식이 올바른 문법을 가지고 있는지 확인



전략 1: 구문 분석 (Parsing)

- 번역 모델이 내놓은 논리식이 올바른 문법을 가지고 있는지 확인

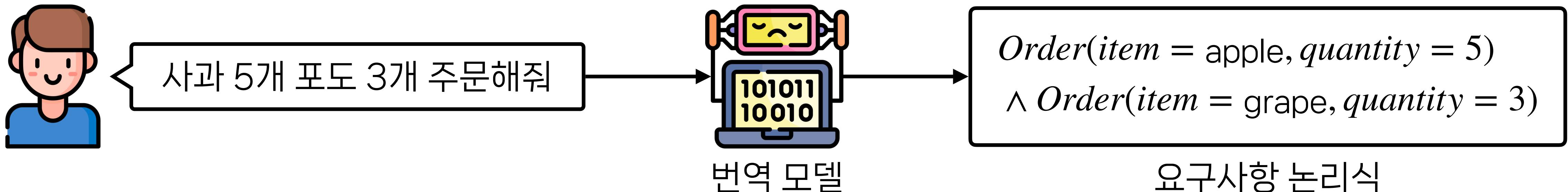


전략 2: 유형 검사 (Type checking)

- LLM이 내놓은 논리식이 올바른 유형을 가지고 있는지 확인

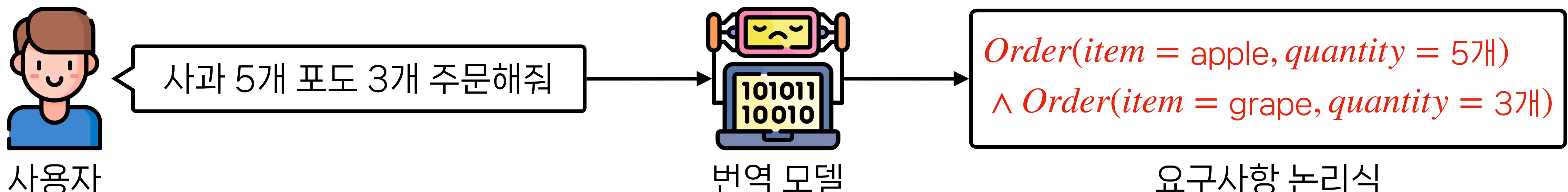
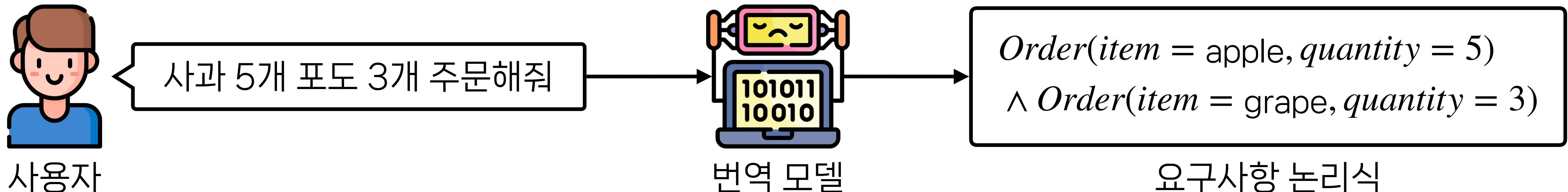
전략 2: 유형 검사 (Type checking)

- LLM이 내놓은 논리식이 올바른 유형을 가지고 있는지 확인



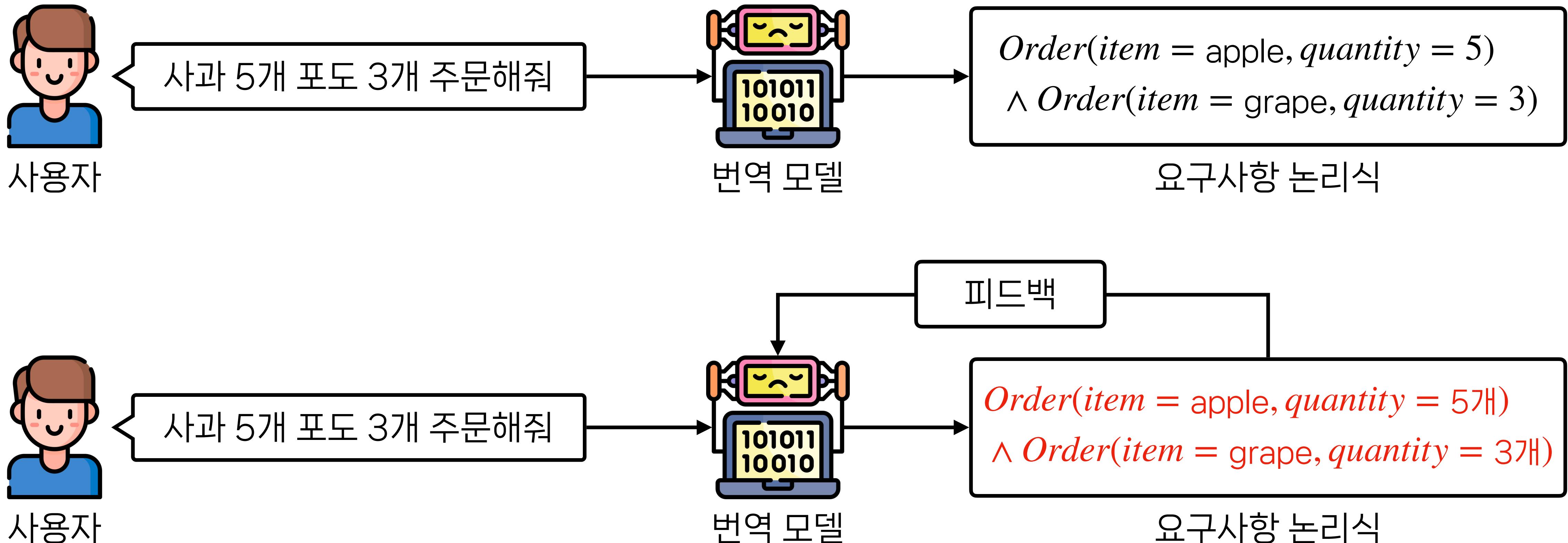
전략 2: 유형 검사 (Type checking)

- LLM이 내놓은 논리식이 올바른 유형을 가지고 있는지 확인



전략 2: 유형 검사 (Type checking)

- LLM이 내놓은 논리식이 올바른 유형을 가지고 있는지 확인

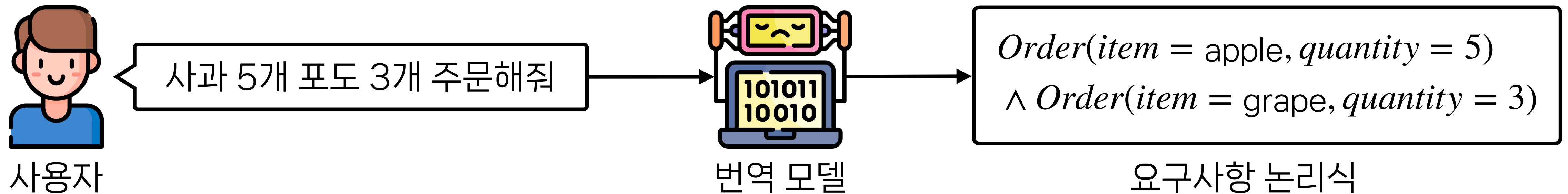


전략 3: 일관성 검사

- 변환된 논리식을 다시 자연어로 바꾼 후 원래 지시사항과 비교하기
- 빠진 내용이나 틀린 내용을 최소화 할 수 있음

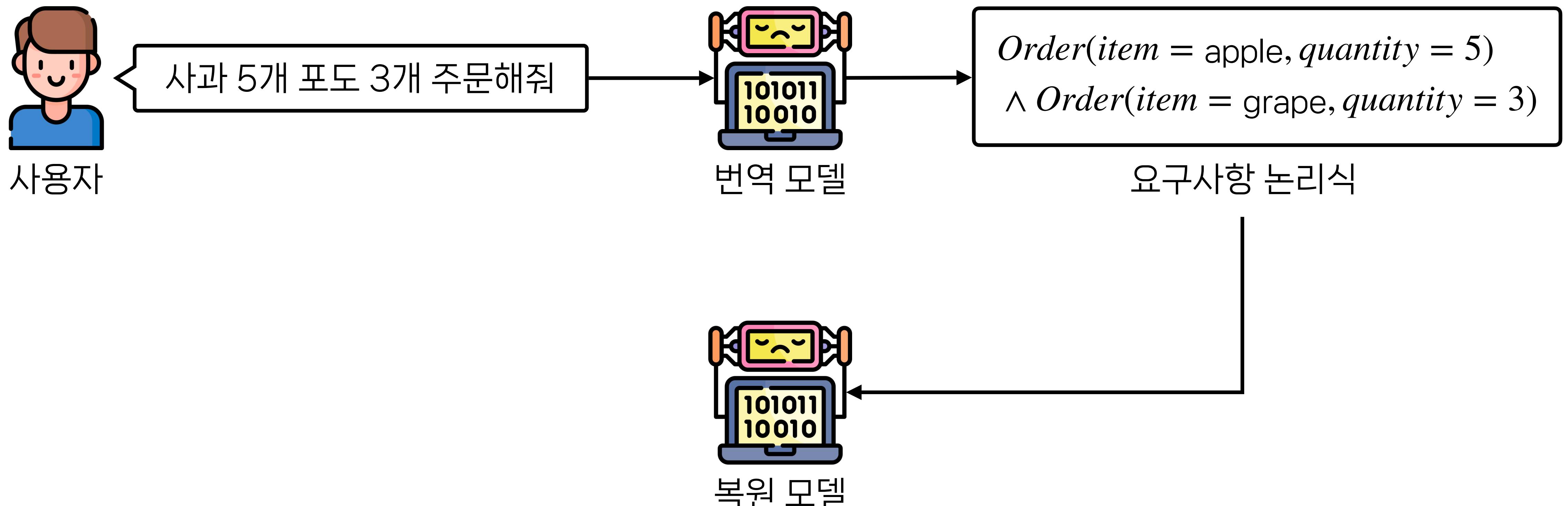
전략 3: 일관성 검사

- 변환된 논리식을 다시 자연어로 바꾼 후 원래 지시사항과 비교하기
- 빠진 내용이나 틀린 내용을 최소화 할 수 있음



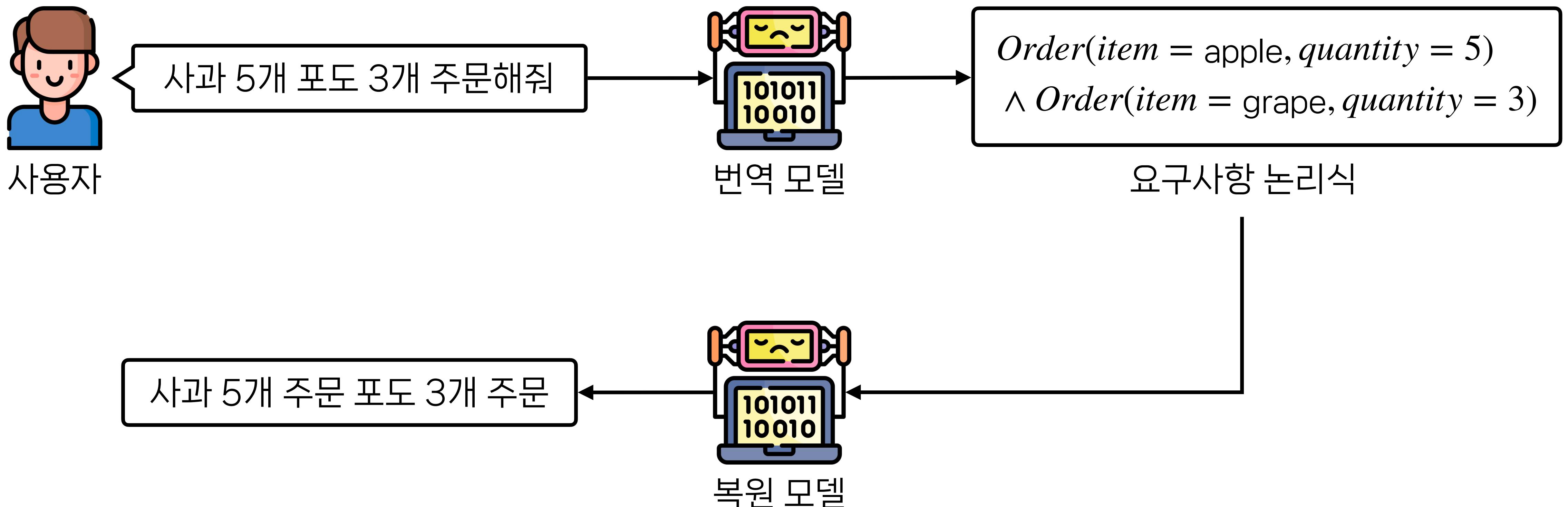
전략 3: 일관성 검사

- 변환된 논리식을 다시 자연어로 바꾼 후 원래 지시사항과 비교하기
- 빠진 내용이나 틀린 내용을 최소화 할 수 있음



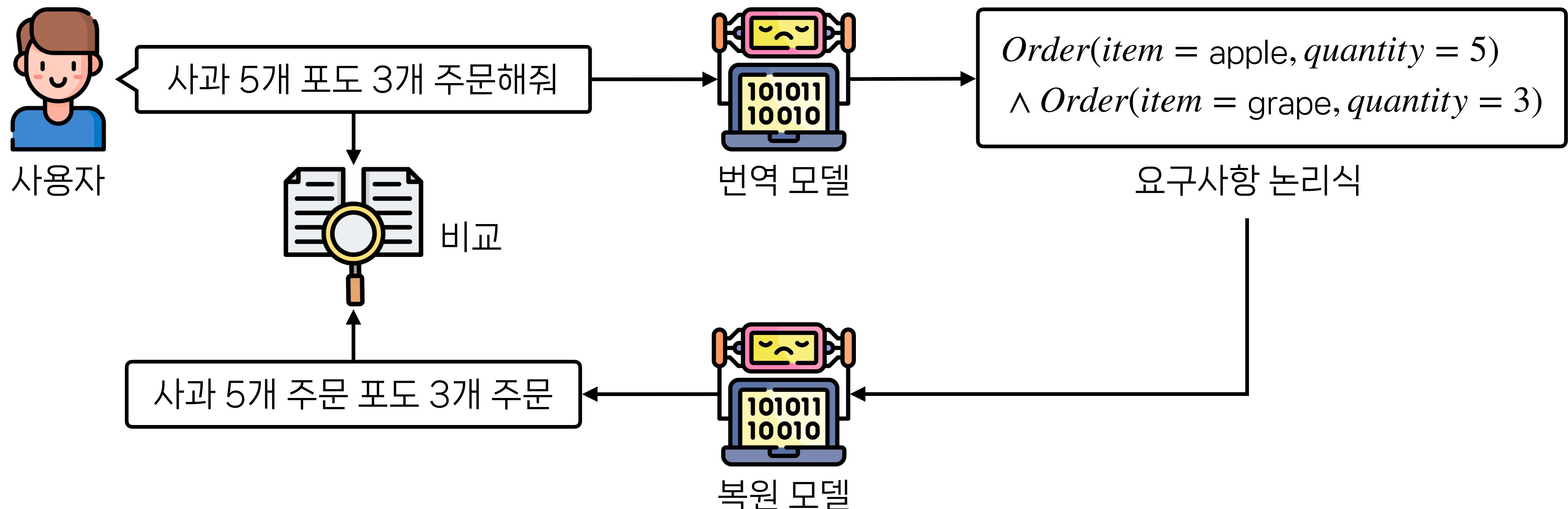
전략 3: 일관성 검사

- 변환된 논리식을 다시 자연어로 바꾼 후 원래 지시사항과 비교하기
- 빠진 내용이나 틀린 내용을 최소화 할 수 있음



전략 3: 일관성 검사

- 변환된 논리식을 다시 자연어로 바꾼 후 원래 지시사항과 비교하기
- 빠진 내용이나 틀린 내용을 최소화 할 수 있음

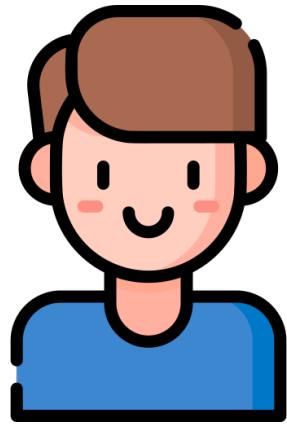


전략 4: 메모리 시스템

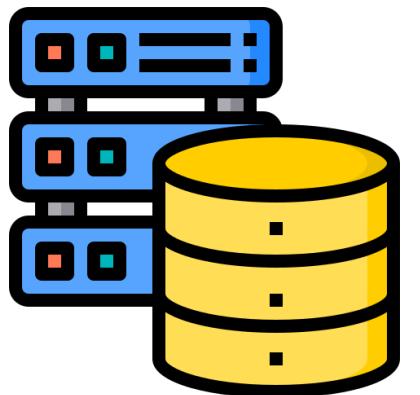
- 과거 번역 데이터를 참고 자료로 활용하기

전략 4: 메모리 시스템

- 과거 번역 데이터를 참고 자료로 활용하기



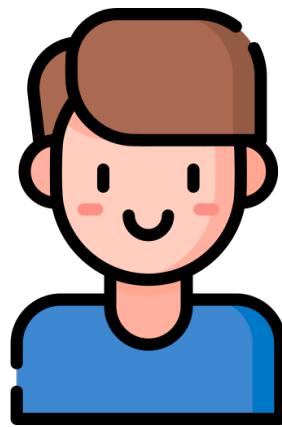
사용자



과거 번역 데이터

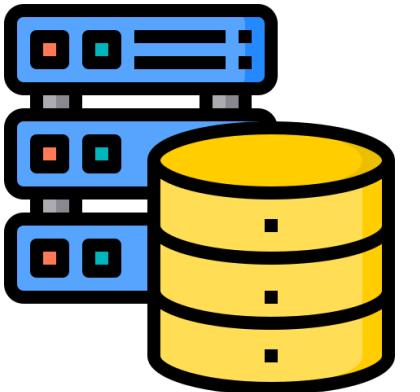
전략 4: 메모리 시스템

- 과거 번역 데이터를 참고 자료로 활용하기



사과 5개 포도 3개 주문해줘

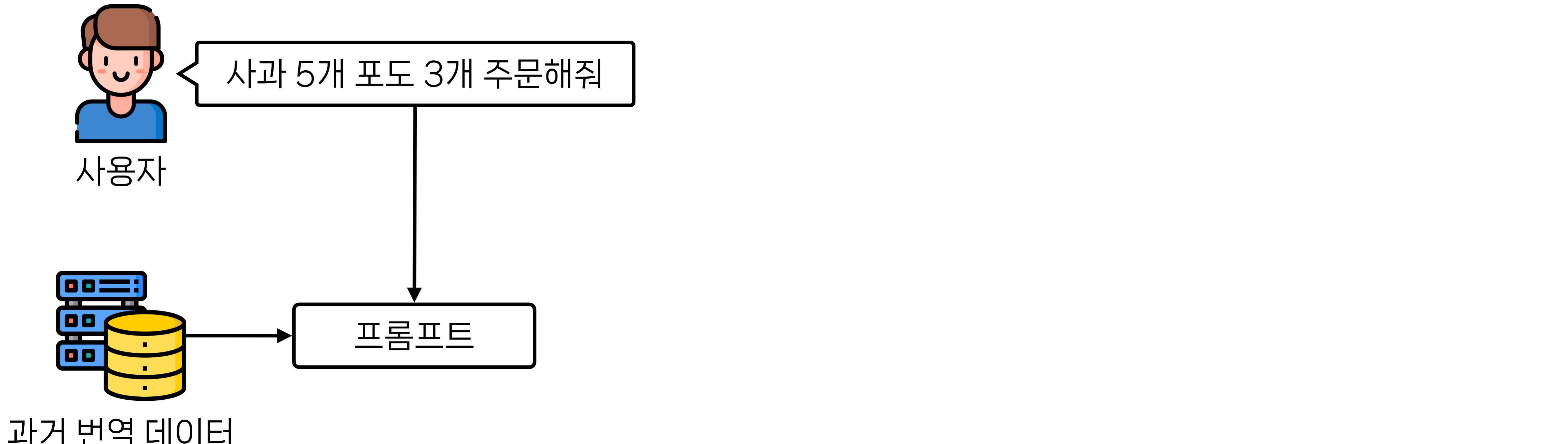
사용자



과거 번역 데이터

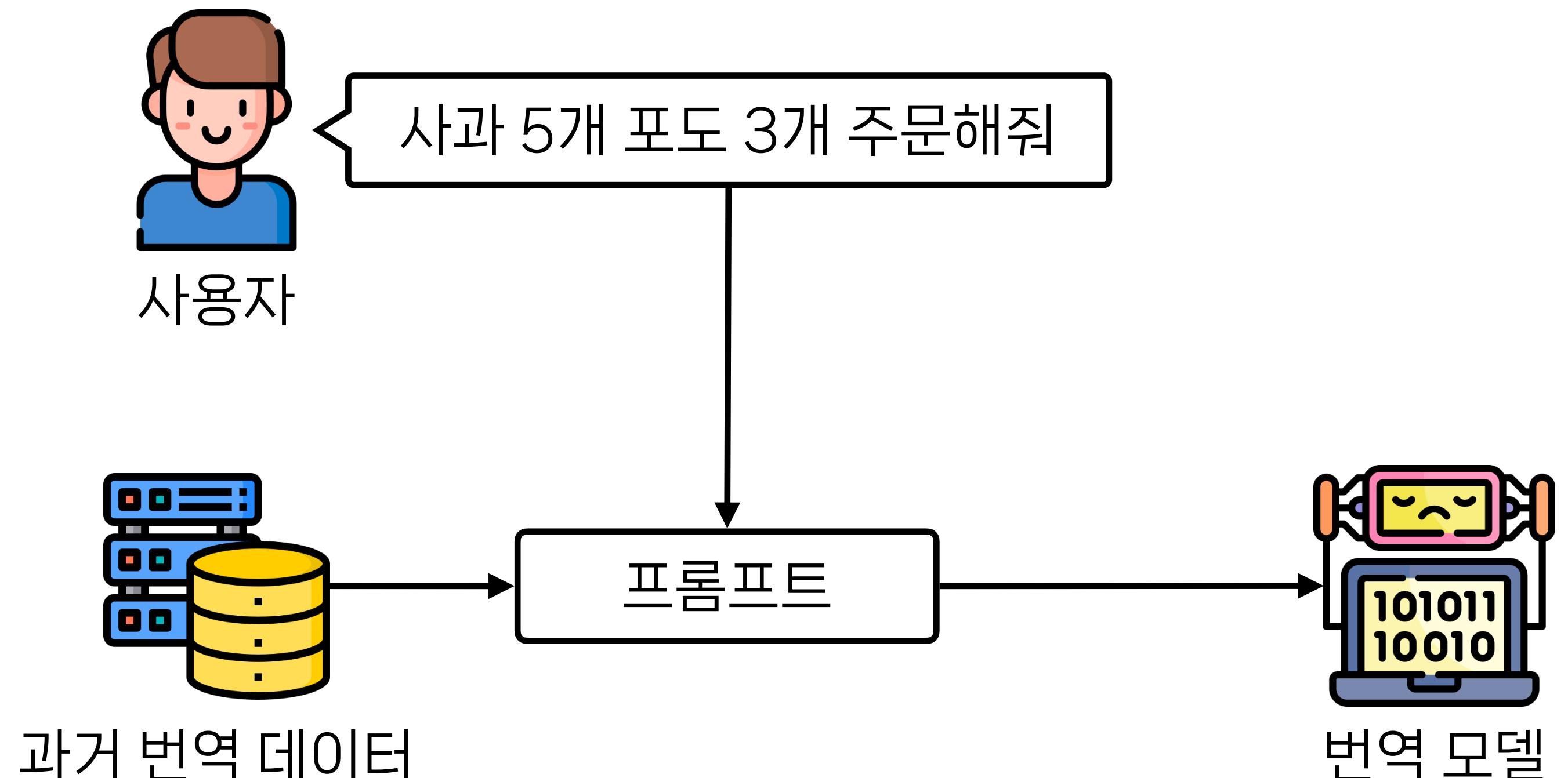
전략 4: 메모리 시스템

- 과거 번역 데이터를 참고 자료로 활용하기



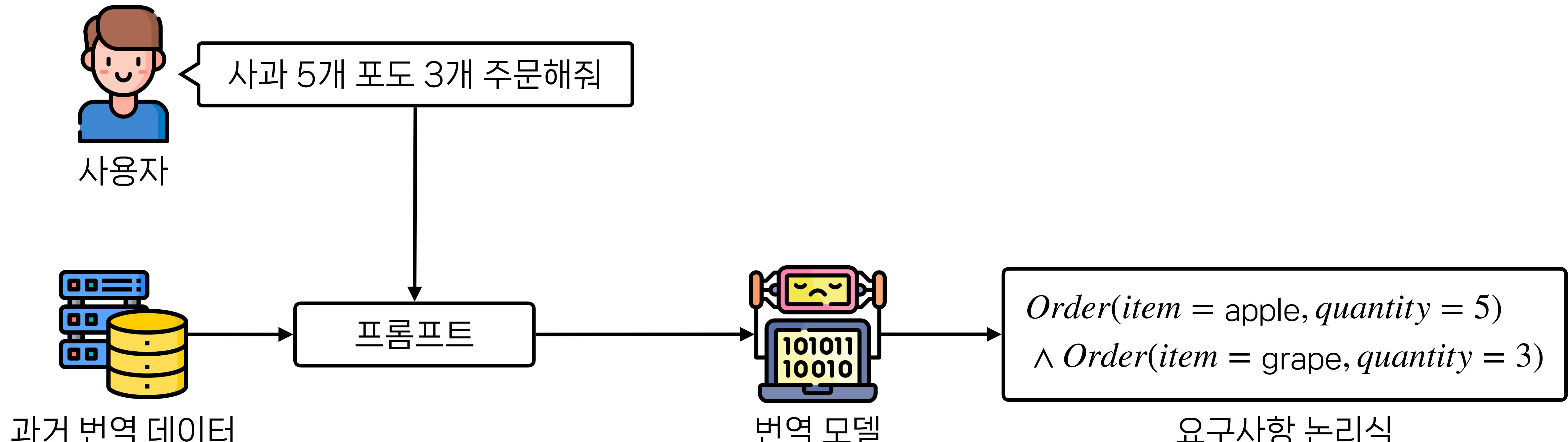
전략 4: 메모리 시스템

- 과거 번역 데이터를 참고 자료로 활용하기



전략 4: 메모리 시스템

- 과거 번역 데이터를 참고 자료로 활용하기



문제 2: 어떤 언어로 논리식을 작성할까?

- 일차 논리는 검증에 필요한 모든 것을 표현할 수 있지만, 최적은 아님

문제 2: 어떤 언어로 논리식을 작성할까?

- 일차 논리는 검증에 필요한 모든 것을 표현할 수 있지만, 최적은 아님
- 앱에 최적화된 언어가 아니기 때문에 번역 난이도가 올라감

문제 2: 어떤 언어로 논리식을 작성할까?

- 일차 논리는 검증에 필요한 모든 것을 표현할 수 있지만, 최적은 아님
 - 앱에 최적화된 언어가 아니기 때문에 번역 난이도가 올라감
- 최적의 논리식 표현 형태가 되려면?

문제 2: 어떤 언어로 논리식을 작성할까?

- 일차 논리는 검증에 필요한 모든 것을 표현할 수 있지만, 최적은 아님
 - 앱에 최적화된 언어가 아니기 때문에 번역 난이도가 올라감
- 최적의 논리식 표현 형태가 되려면?
 1. 사용자의 지시사항이나 앱 상태를 대부분 표현할 수 있어야 함 (높은 표현력)

문제 2: 어떤 언어로 논리식을 작성할까?

- 일차 논리는 검증에 필요한 모든 것을 표현할 수 있지만, 최적은 아님
 - 앱에 최적화된 언어가 아니기 때문에 번역 난이도가 올라감
- 최적의 논리식 표현 형태가 되려면?
 1. 사용자의 지시사항이나 앱 상태를 대부분 표현할 수 있어야 함 (높은 표현력)
 2. 간결하고 직관적으로 사용자의 지시사항을 작성할 수 있어야 함 (쉬운 언어)

문제 2: 어떤 언어로 논리식을 작성할까?

- 일차 논리는 검증에 필요한 모든 것을 표현할 수 있지만, 최적은 아님
 - 앱에 최적화된 언어가 아니기 때문에 번역 난이도가 올라감
 - 최적의 논리식 표현 형태가 되려면?
 1. 사용자의 지시사항이나 앱 상태를 대부분 표현할 수 있어야 함 (높은 표현력)
 2. 간결하고 직관적으로 사용자의 지시사항을 작성할 수 있어야 함 (쉬운 언어)
- ▶ 모바일 에이전트 검증을 위한 **도메인 특화 언어 (DSL)** 정의

도메인 특화 언어의 구조

도메인 특화 언어의 구조

- Horn Clause 기반의 논리 프로그래밍 언어 (Datalog, Prolog와 유사)

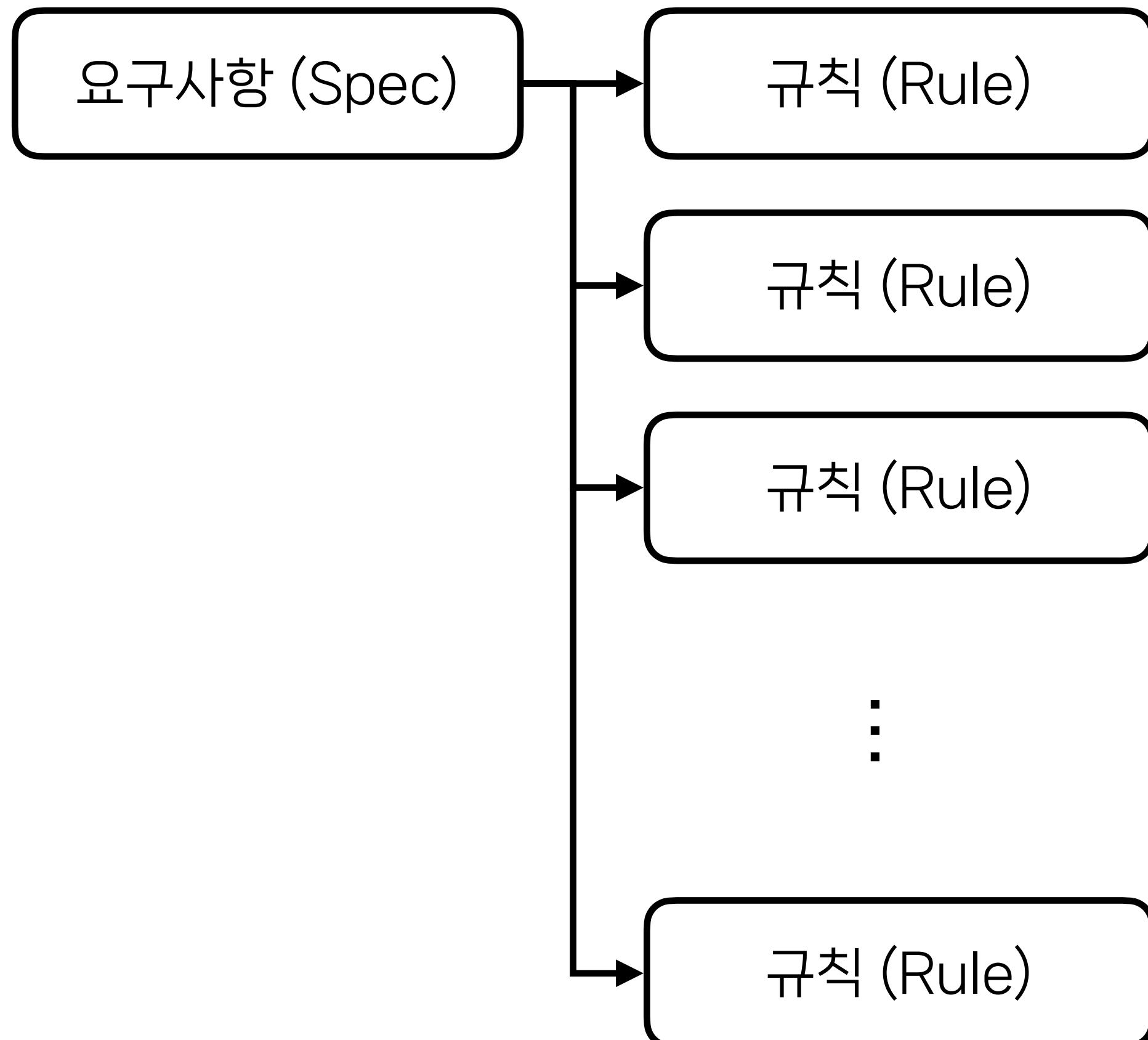
도메인 특화 언어의 구조

- Horn Clause 기반의 논리 프로그래밍 언어 (Datalog, Prolog와 유사)

요구사항 (Spec)

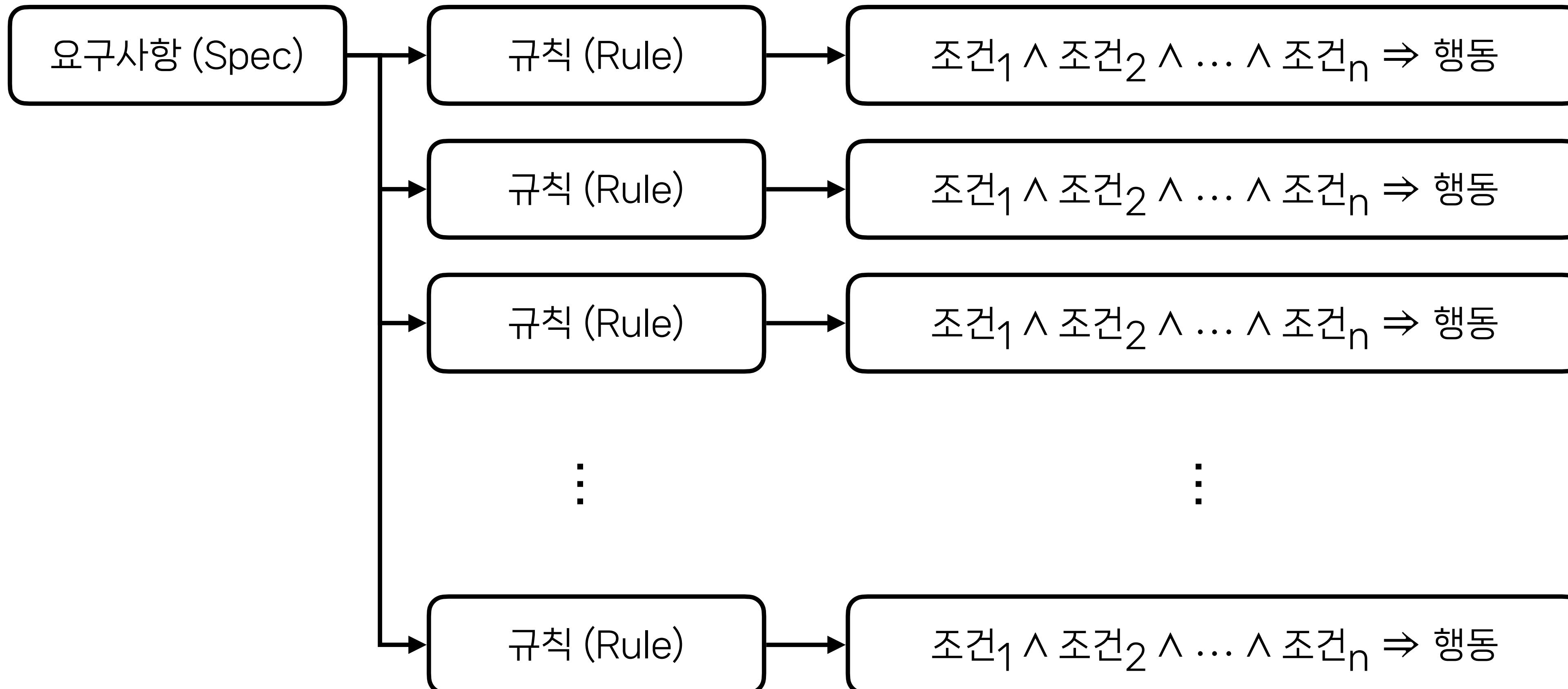
도메인 특화 언어의 구조

- Horn Clause 기반의 논리 프로그래밍 언어 (Datalog, Prolog와 유사)



도메인 특화 언어의 구조

- Horn Clause 기반의 논리 프로그래밍 언어 (Datalog, Prolog와 유사)



도메인 특화 언어 예제

- 예제: 레스토랑 'R'을 오후 7시전으로 예약해. 예약이 불가능하면 아무것도 하지 마

도메인 특화 언어 예제

- 예제: 레스토랑 'R'을 오후 7시전으로 예약해. 예약이 불가능하면 아무것도 하지 마

규칙1: RestaurantInfo(name = R)

\wedge ReserveInfo(date = Today, time < 19:00, available = True)

 → Reserve

도메인 특화 언어 예제

- 예제: 레스토랑 'R'을 오후 7시전으로 예약해. 예약이 불가능하면 아무것도 하지 마

규칙1: $\text{RestaurantInfo}(\text{name} = R)$

$\wedge \text{ReserveInfo}(\text{date} = \text{Today}, \text{time} < 19:00, \text{available} = \text{True})$

$\rightarrow \text{Reserve}$

규칙2: $\text{Reserve} \wedge \text{ReserveResult}(\text{success} = \text{True}) \rightarrow \text{Done}$

도메인 특화 언어 예제

- 예제: 레스토랑 'R'을 오후 7시전으로 예약해. 예약이 불가능하면 아무것도 하지 마

규칙1: $\text{RestaurantInfo}(\text{name} = R)$

$\wedge \text{ReserveInfo}(\text{date} = \text{Today}, \text{time} < 19:00, \text{available} = \text{True})$

$\rightarrow \text{Reserve}$

규칙2: $\text{Reserve} \wedge \text{ReserveResult}(\text{success} = \text{True}) \rightarrow \text{Done}$

규칙3: $\text{RestaurantInfo}(\text{name} = R)$

$\wedge \text{ReserveInfo}(\text{date} = \text{Today}, \text{time} < 19:00, \text{available} \neq \text{True})$

$\rightarrow \text{Done}$

문제 3: 어떻게 현재 앱 상태를 검증에 반영할까?

문제 3: 어떻게 현재 앱 상태를 검증에 반영할까?

- 앱 상태 측정 (Instrumentation) 라이브러리를 개발하여 앱 개발자에게 전권 위임

문제 3: 어떻게 현재 앱 상태를 검증에 반영할까?

- 앱 상태 측정 (Instrumentation) 라이브러리를 개발하여 앱 개발자에게 전권 위임
- 기존 이벤트 처리기 (Event Handler)에 손쉽게 추가 가능

문제 3: 어떻게 현재 앱 상태를 검증에 반영할까?

- 앱 상태 측정 (Instrumentation) 라이브러리를 개발하여 앱 개발자에게 전권 위임
- 기존 이벤트 처리기 (Event Handler)에 손쉽게 추가 가능
- 예제

문제 3: 어떻게 현재 앱 상태를 검증에 반영할까?

- 앱 상태 측정 (Instrumentation) 라이브러리를 개발하여 앱 개발자에게 전권 위임
- 기존 이벤트 처리기 (Event Handler)에 손쉽게 추가 가능
- 예제

```
searchButton.VSAOnClickListener () -> {
    VSA.updateState ("RestaurantInfo", {
        "name": searchTextField.getText () ,
    }) ;
    /* existing code for onClickListener */
} );
```

성능 평가

- 벤치마크: 다양한 모바일 앱에서 사용자 지시사항 수행 (올바른 경우 150개 + 잘못된 경우 150개)
 - LlamaTouch (125개 + 125개)
 - 자체 제작한 Complex Instructions (25개 + 25개)
- 사용 모델
 - GPT-4o (temperature=0.0)
- 기준 (Baseline)
 - LMM을 활용한 자가 점검 (Self-Reflection) 기법
- 측정 지표
 - 검증 정확도 (Accuracy, Precision, Recall, F1 Score)
 - 작업 성공 개수
 - 자연 시간 및 비용

False Positive vs False Negative

False Positive vs False Negative

- False Positive: 올바른 에이전트의 행동을 틀렸다고 잘못 판단
 - 치명적이진 않지만, 에이전트의 효율성을 떨어뜨림
 - 검증을 촘촘하게 할수록 많이 발생

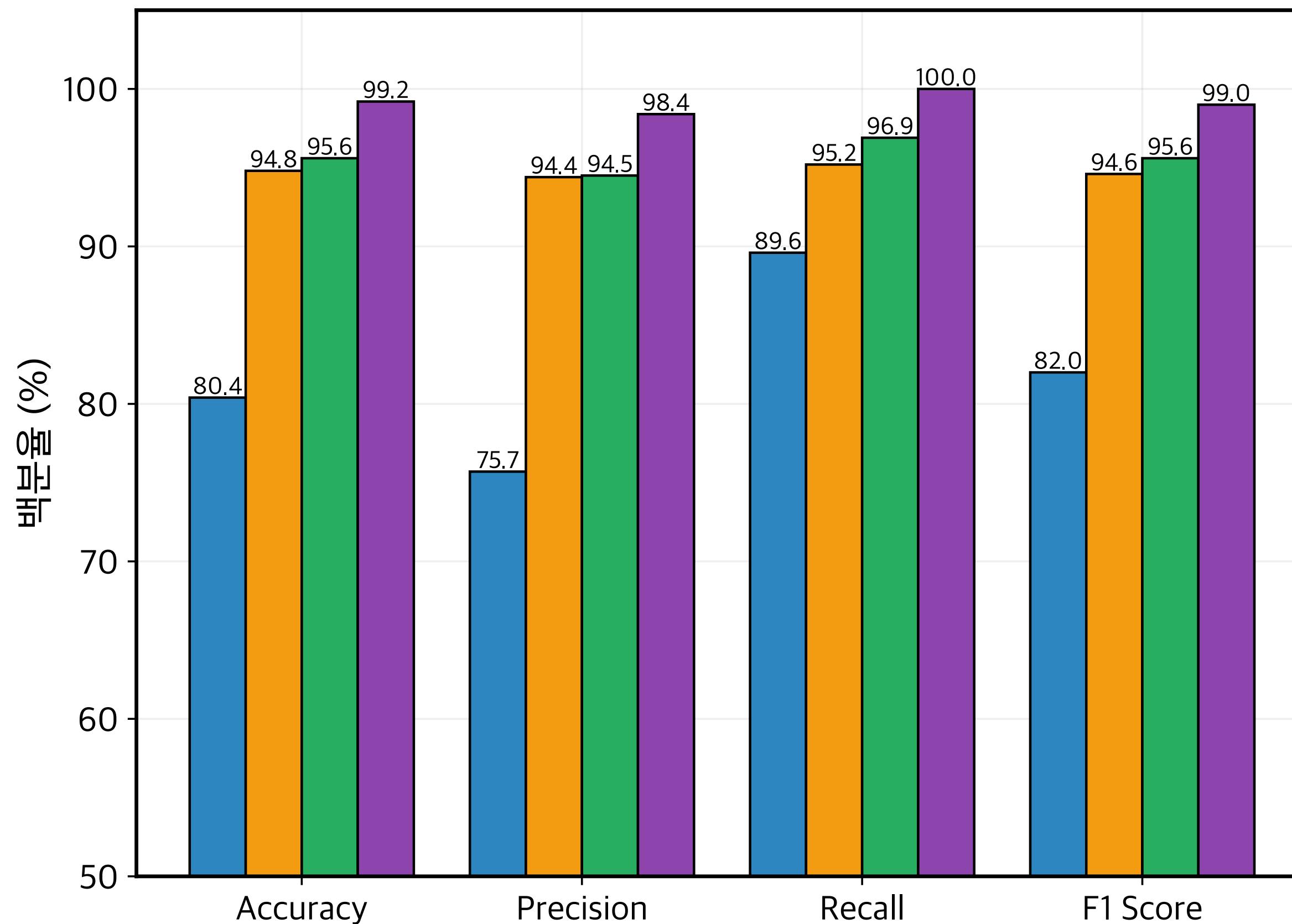
False Positive vs False Negative

- False Positive: 올바른 에이전트의 행동을 틀렸다고 잘못 판단
 - 치명적이진 않지만, 에이전트의 효율성을 떨어뜨림
 - 검증을 촘촘하게 할수록 많이 발생
- False Negative: 잘못된 에이전트의 행동을 올바르다고 판단
 - 치명적인 오류를 유발할 수 있음
 - 예시: 잘못된 계좌로 입금, 다른 날짜로 항공편 예약
 - 검증을 널널하게 할수록 많이 발생

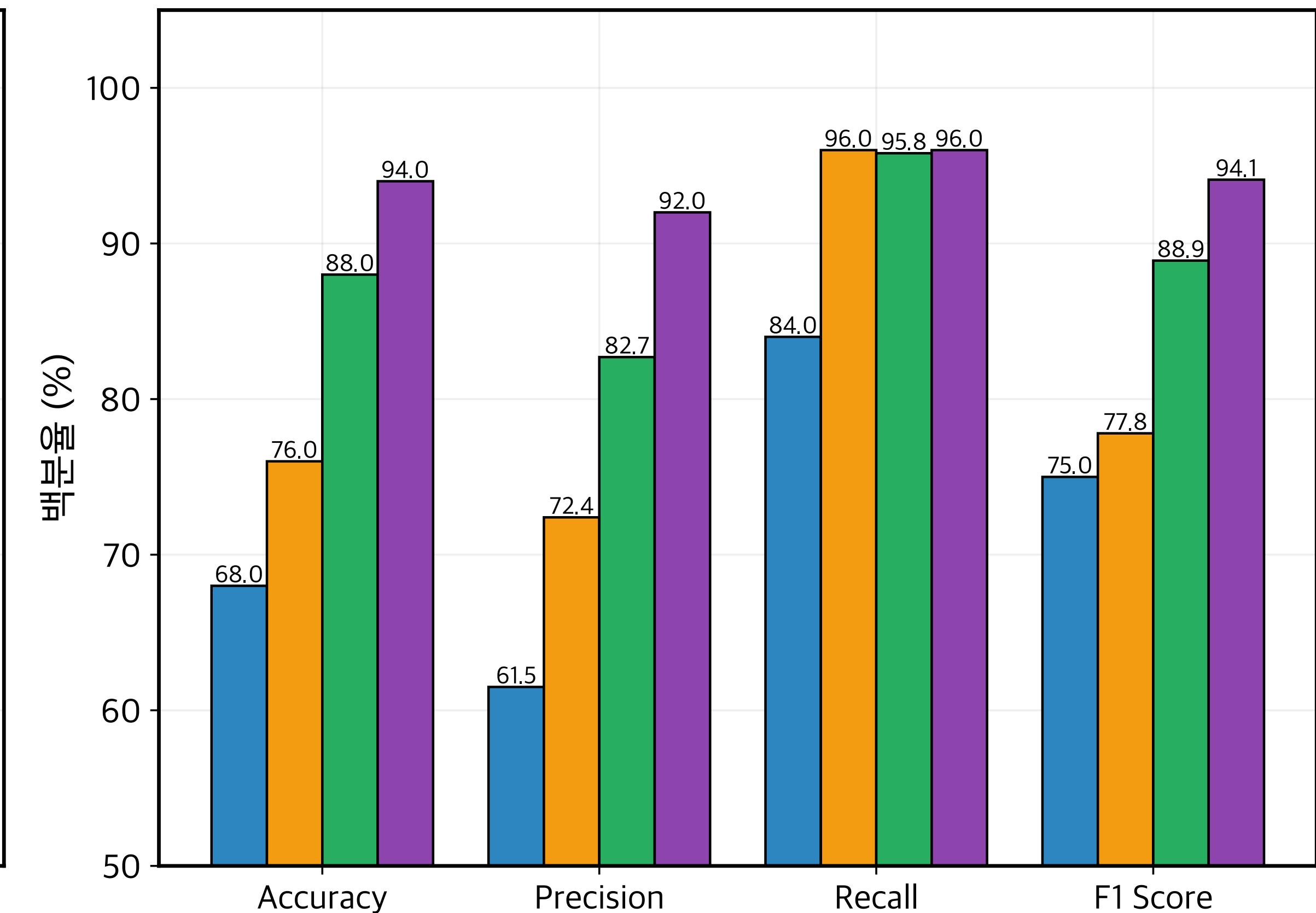
검증 정확도

■ LMM 기반 사전 검증 ■ LMM 기반 사후 검증 ■ VSA (메모리 없음) ■ VSA (메모리 있음)

LlamaTouch 데이터셋



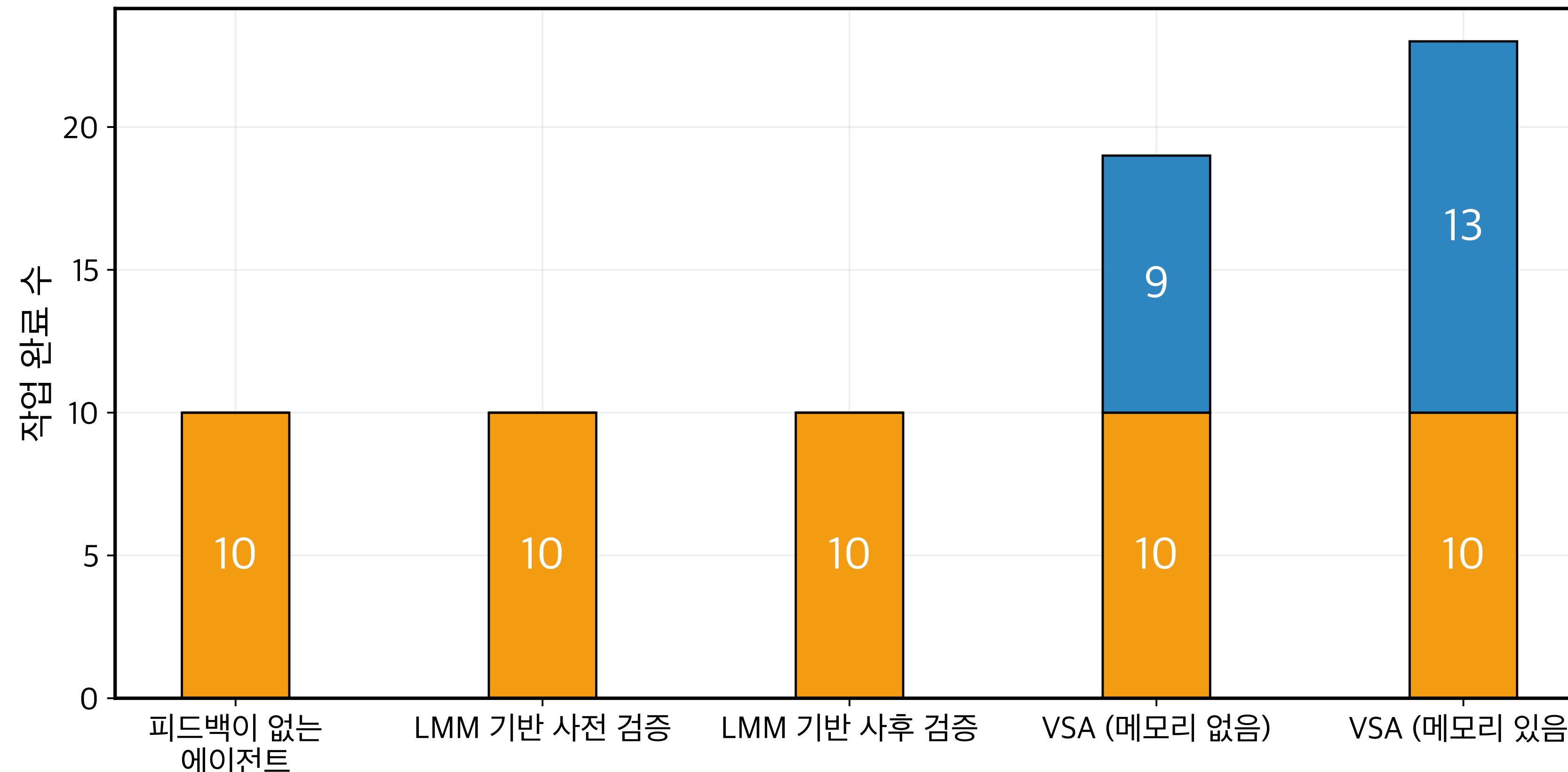
복잡한 데이터셋



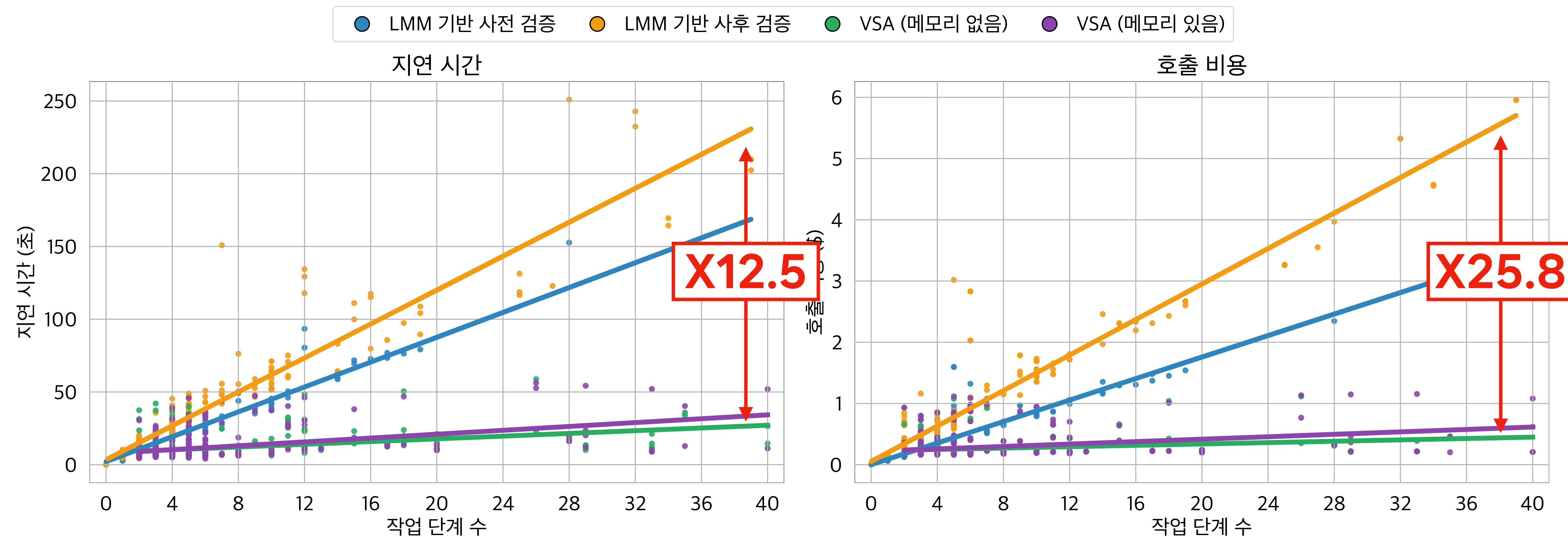
피드백을 통한 작업 성공률 개선

복잡한 데이터셋에서의 피드백 효과

피드백 없음 피드백 있음



지연 시간 및 비용



소프트웨어 재난 재발 방지



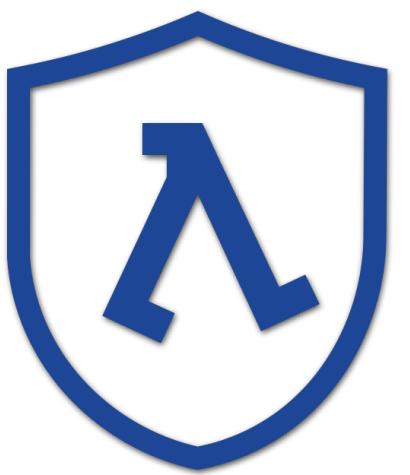
STAAR

Prosys Lab의 목표: 소프트웨어 재난의 재발 방지



"어떤" 소프트웨어?

소프트웨어 재난 재발 방지



STAAR

Prosys Lab의 목표: 소프트웨어 재난의 재발 방지



"어떤" 소프트웨어?



AI 에이전트 소프트웨어!

활용 방안: 에이전트 오류 패턴 DB 구축

활용 방안: 에이전트 오류 패턴 DB 구축

- “어떤” 명령에서 “왜” 에이전트가 오류를 일으켰는가?

활용 방안: 에이전트 오류 패턴 DB 구축

- “어떤” 명령에서 “왜” 에이전트가 오류를 일으켰는가?
- AS-IS: 자연어 명령, 에이전트 행동 기록, 앱 상태, 로그, ...

활용 방안: 에이전트 오류 패턴 DB 구축

- “어떤” 명령에서 “왜” 에이전트가 오류를 일으켰는가?
- AS-IS: 자연어 명령, 에이전트 행동 기록, 앱 상태, 로그, ...
- TO-BE: 논리식 명령, 오류 원인 쌍 (I, E)

활용 방안: 에이전트 오류 패턴 DB 구축

- “어떤” 명령에서 “왜” 에이전트가 오류를 일으켰는가?
 - AS-IS: 자연어 명령, 에이전트 행동 기록, 앱 상태, 로그, ...
 - TO-BE: 논리식 명령, 오류 원인 쌍 (I, E)
- 논리식 명령, 오류 원인 쌍 목록에서 에이전트 오류 패턴을 추출

활용 방안: 에이전트 오류 패턴 DB 구축

- “어떤” 명령에서 “왜” 에이전트가 오류를 일으켰는가?
 - AS-IS: 자연어 명령, 에이전트 행동 기록, 앱 상태, 로그, ...
 - TO-BE: 논리식 명령, 오류 원인 쌍 (I, E)
- 논리식 명령, 오류 원인 쌍 목록에서 에이전트 오류 패턴을 추출
 - 예시: 에이전트가 $Order(item = *, quantity = n > 3)$ 패턴을 자주 틀리더라

활용 방안: 에이전트 오류 패턴 DB 구축

- “어떤” 명령에서 “왜” 에이전트가 오류를 일으켰는가?
 - AS-IS: 자연어 명령, 에이전트 행동 기록, 앱 상태, 로그, ...
 - TO-BE: 논리식 명령, 오류 원인 쌍 (I, E)
- 논리식 명령, 오류 원인 쌍 목록에서 에이전트 오류 패턴을 추출
 - 예시: 에이전트가 $Order(item = *, quantity = n > 3)$ 패턴을 자주 틀리더라
 - 파악된 오류 원인을 에이전트 개선, 모델 추가 훈련에 활용

활용 방안: 에이전트 오류 패턴 DB 구축

- “어떤” 명령에서 “왜” 에이전트가 오류를 일으켰는가?
 - AS-IS: 자연어 명령, 에이전트 행동 기록, 앱 상태, 로그, ...
 - TO-BE: 논리식 명령, 오류 원인 쌍 (I, E)
- 논리식 명령, 오류 원인 쌍 목록에서 에이전트 오류 패턴을 추출
 - 예시: 에이전트가 $Order(item = *, quantity = n > 3)$ 패턴을 자주 틀리더라
 - 파악된 오류 원인을 에이전트 개선, 모델 추가 훈련에 활용
 - 예시: 오류 원인으로부터 합성 데이터 생성

활용 방안: 사전 규칙 지정해주기

활용 방안: 사전 규칙 지정해주기

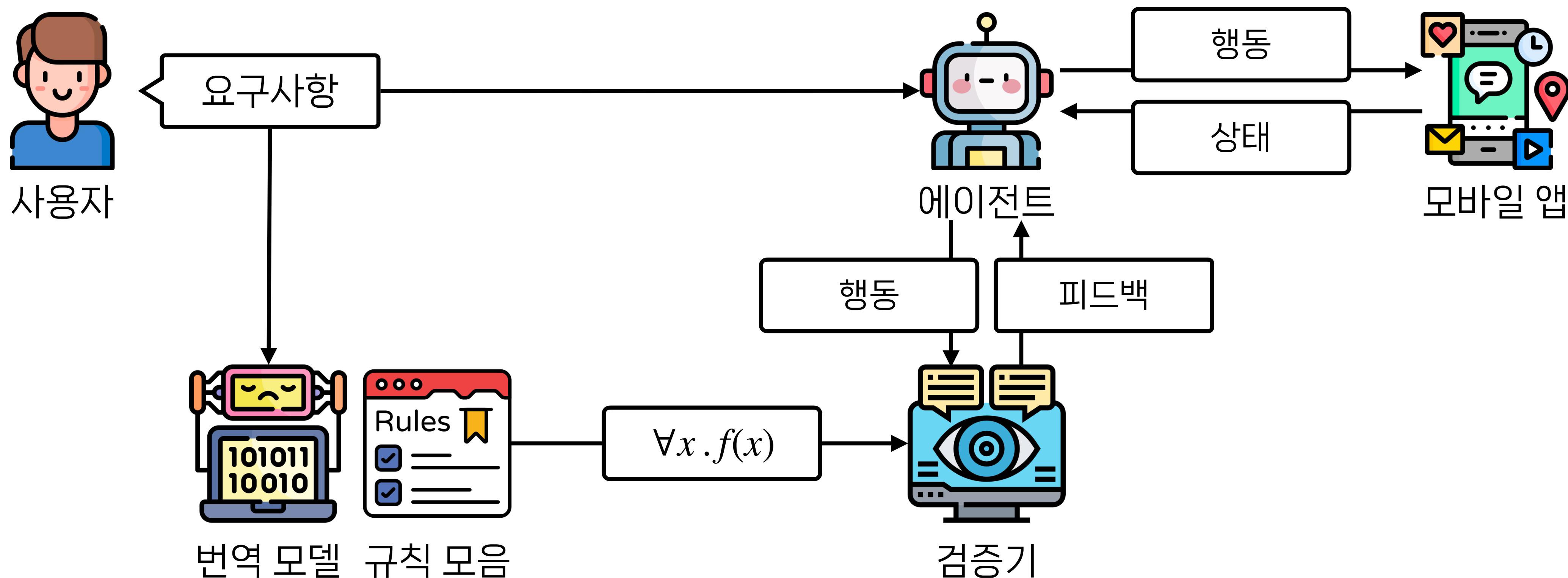
- 사용자의 지시사항 이외에도 에이전트가 지켜야 하는 전역 규칙 지정 가능

활용 방안: 사전 규칙 지정해주기

- 사용자의 지시사항 이외에도 에이전트가 지켜야 하는 전역 규칙 지정 가능
- 예시: 한 번에 5만원 이상 송금 금지

활용 방안: 사전 규칙 지정해주기

- 사용자의 지시사항 이외에도 에이전트가 지켜야 하는 전역 규칙 지정 가능
- 예시: 한 번에 5만원 이상 송금 금지



요약

- AI 에이전트를 위한 논리 기반 안전성 확보 시스템 **VeriSafe Agent** 제안
 - 1. 논리식 기반의 에이전트 행동 검증 및 피드백 시스템 구축
 - 2. 지시사항 번역에 구분 분석, 타입 검사, 일관성 검사, 메모리 시스템 도입
 - 3. 앱에 최적화된 도메인 특화 언어 개발
 - 4. 개발자를 위한 앱 상태 관측 라이브러리 개발
- 기존 LMM 기반 검증 시스템보다 성능, 비용 두 가지 측면에서 모두 크게 앞섬
- 에이전트 오류 패턴 분석 및 재발 방지에 효과적일 것으로 기대
- 자연어에서 엄밀한 명세 추출하기 (포스터)

