Fine-Tuning Language Models Using Formal Methods Feedback

Yunhao Yang*, Neel P. Bhatt*, Tyler Ingebrand*, William Ward, Steven Carr, Zhangyang Wang, and Ufuk Topcu

* Equal Contribution

Paper available at: https://arxiv.org/abs/2310.18239

Paper

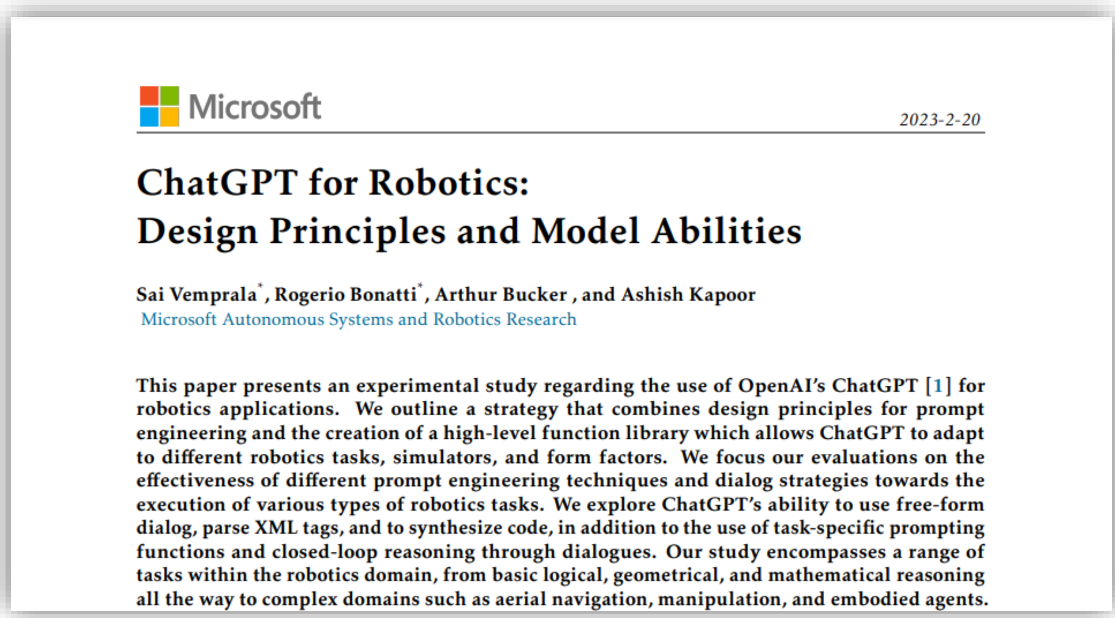
TEXAS
The University of Texas at Austin

VITA

CENTER FOR
autonomy

* Equal Contribution

Pre-Trained Language Models for Autonomous Systems

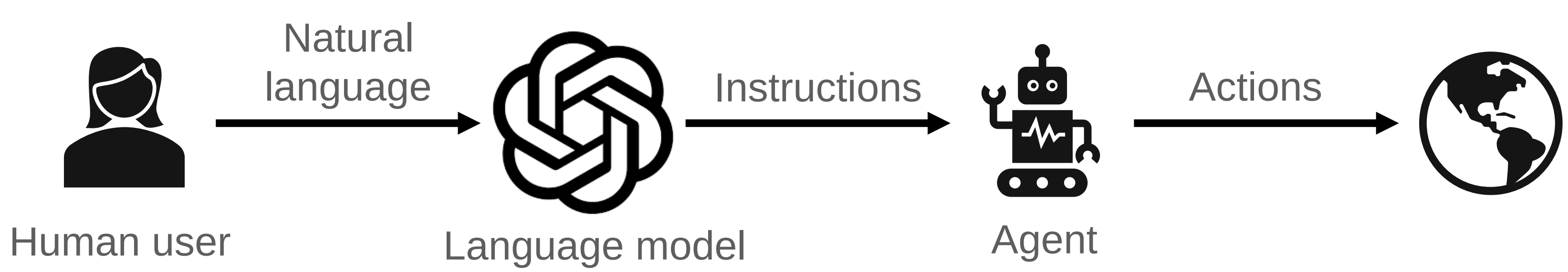

Microsoft 2023-2-20

ChatGPT for Robotics:
Design Principles and Model Abilities

Sai Vemprala*, Rogerio Bonatti*, Arthur Bucker , and Ashish Kapoor
Microsoft Autonomous Systems and Robotics Research

This paper presents an experimental study regarding the use of OpenAI's ChatGPT [1] for robotics applications. We outline a strategy that combines design principles for prompt engineering and the creation of a high-level function library which allows ChatGPT to adapt to different robotics tasks, simulators, and form factors. We focus our evaluations on the effectiveness of different prompt engineering techniques and dialog strategies towards the execution of various types of robotics tasks. We explore ChatGPT's ability to use free-form dialog, parse XML tags, and to synthesize code, in addition to the use of task-specific prompting functions and closed-loop reasoning through dialogues. Our study encompasses a range of tasks within the robotics domain, from basic logical, geometrical, and mathematical reasoning all the way to complex domains such as aerial navigation, manipulation, and embodied agents.

Pre-trained language models encode **rich world knowledge** and provide a **new interface** between humans and machines.



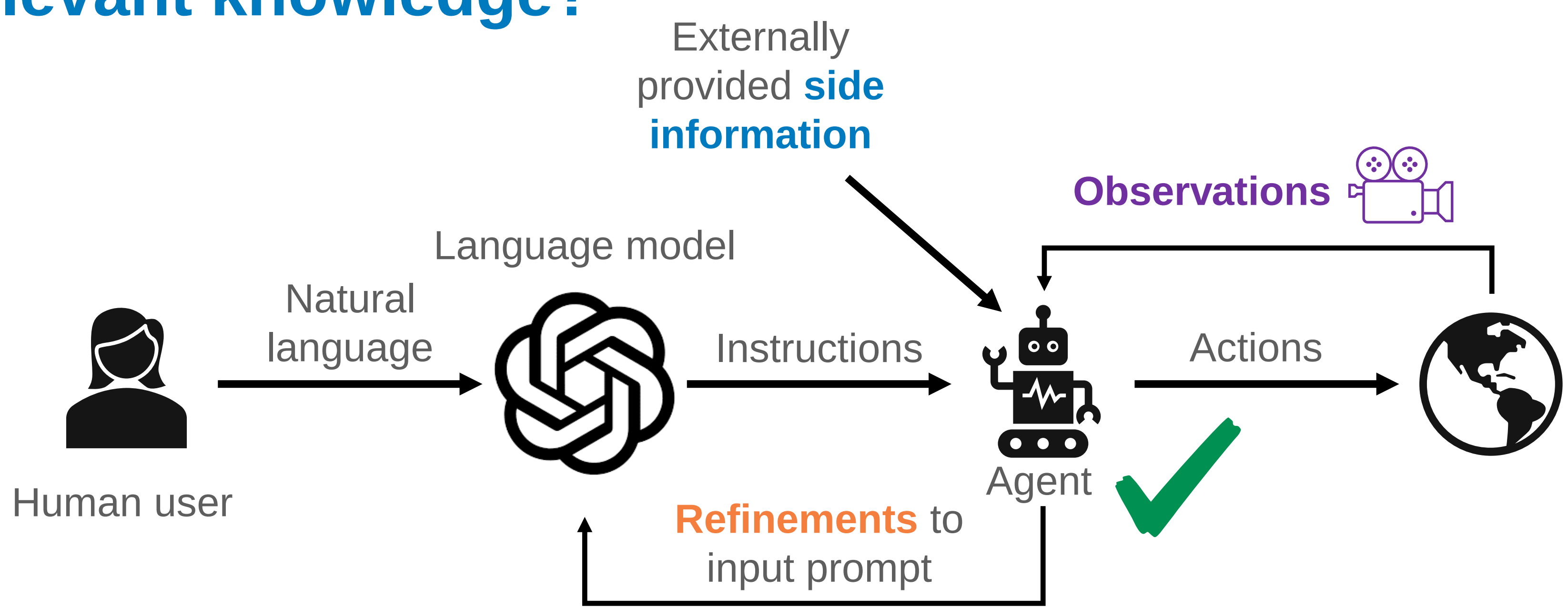
How can we incorporate language models to generate reliable high-level plans or control policies for autonomous systems?

Generative pre-trained models for autonomous systems

How can we …

Leverage additional sources of task-relevant knowledge?

Close the decision-making loop?

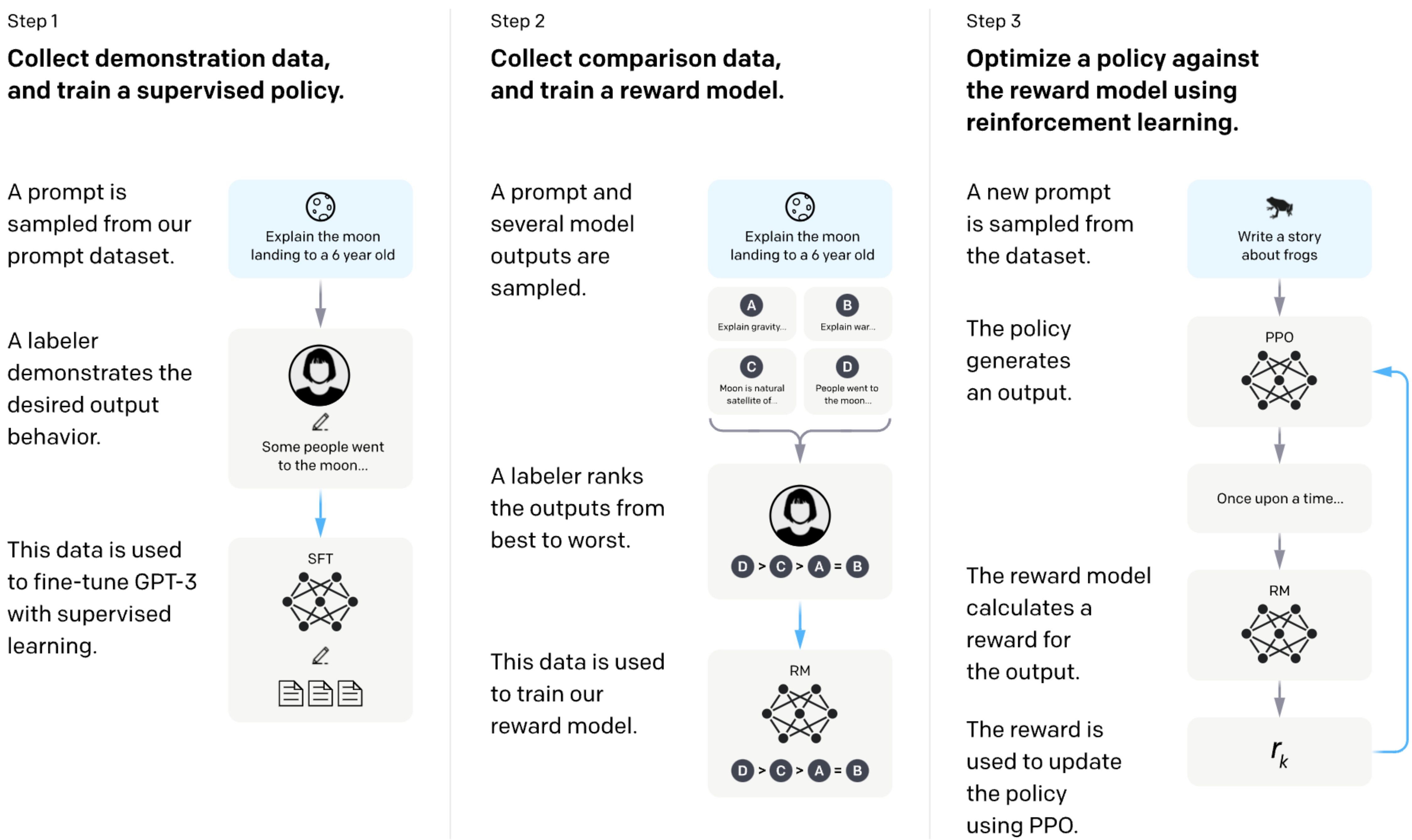


Automatically refine the generated behaviors?

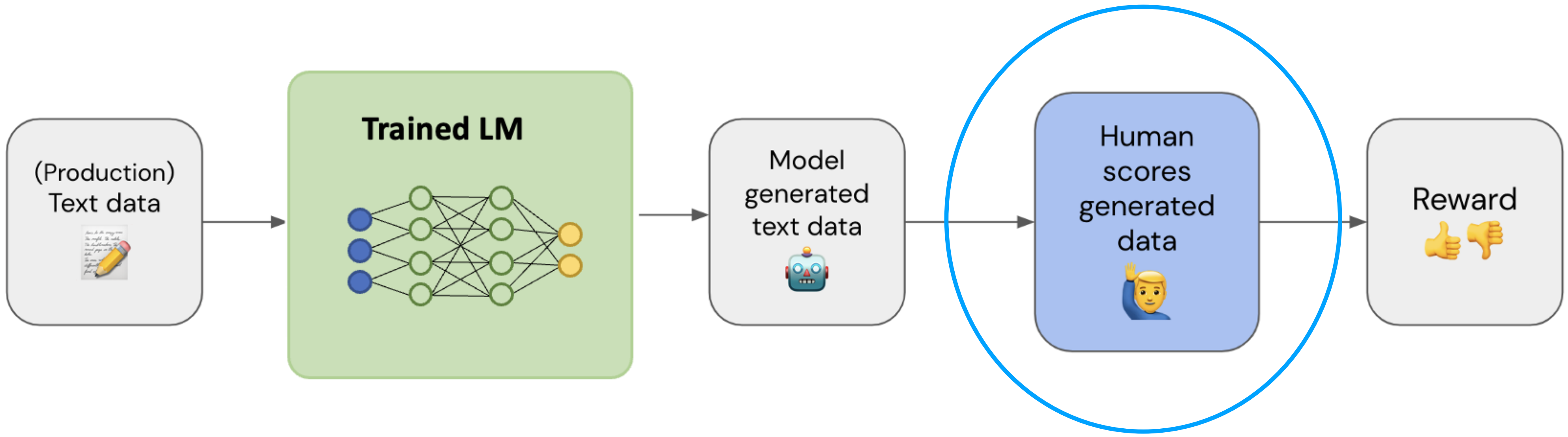
Verify that the generated behavior will satisfy critical requirements?

Fine-Tuning Language Models Using Human Feedback

Example: OpenAI Scheme for Instruct GPT



Fine-Tuning Language Models Using Human Feedback

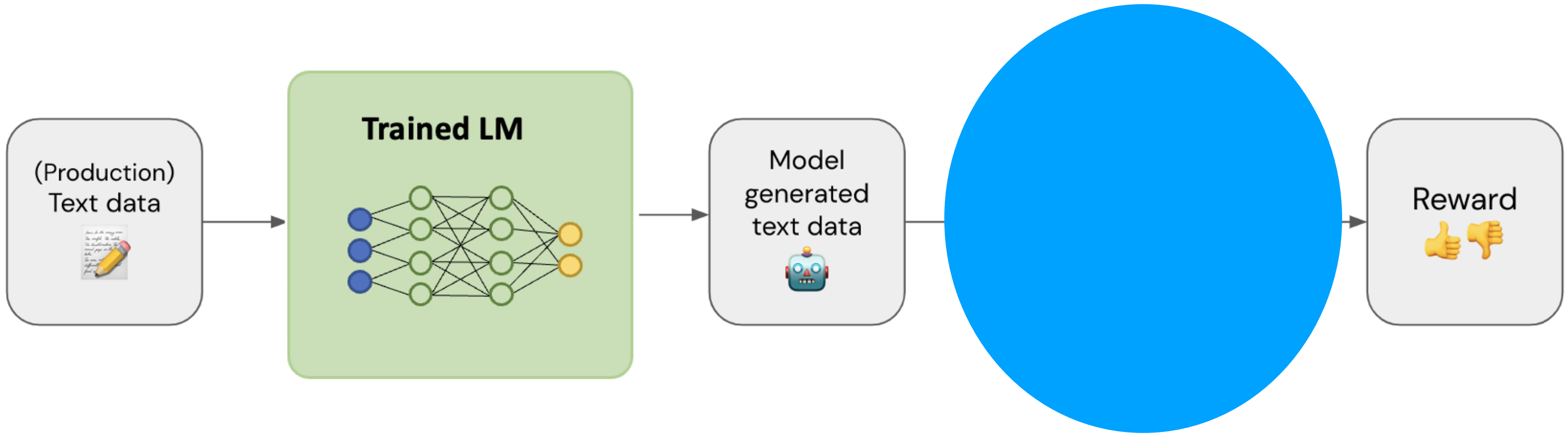


Labor-Intensive

Subjective/Inconsistent Feedback

Fine-Tuning Language Models Using Formal Feedback?

Methods

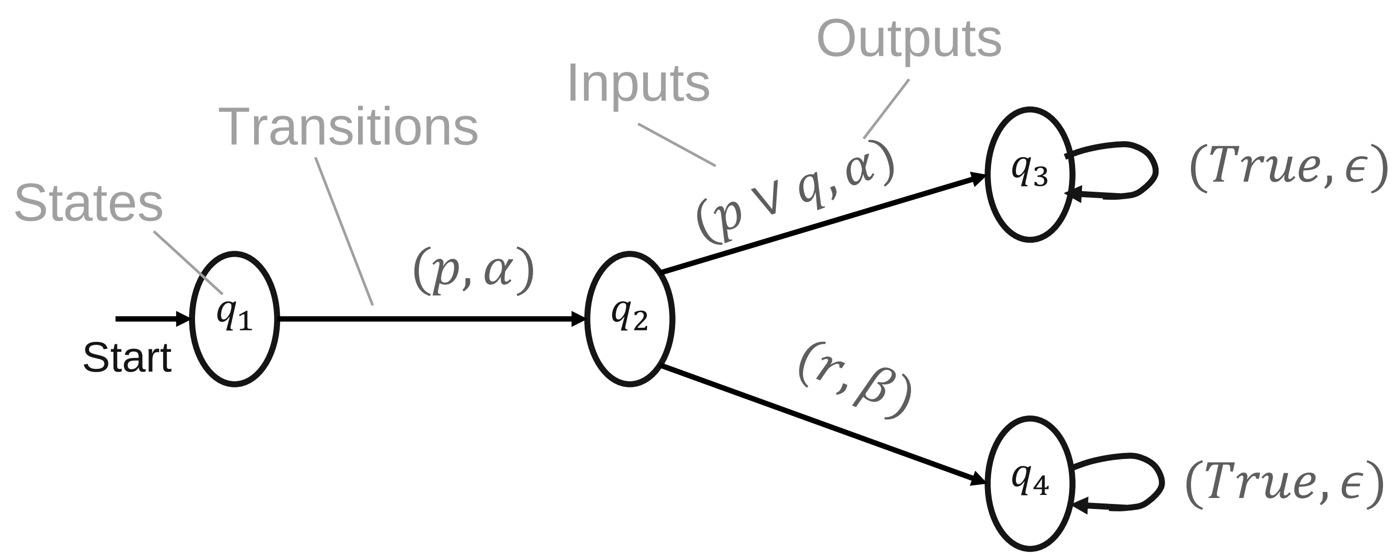


Formal Methods:
Automaton-based Representation, Model Checking, Temporal Logic Specification, etc.

Labor-Intensive

Subjective/Inconsistent Feedbac

Background: Automaton-based Representations

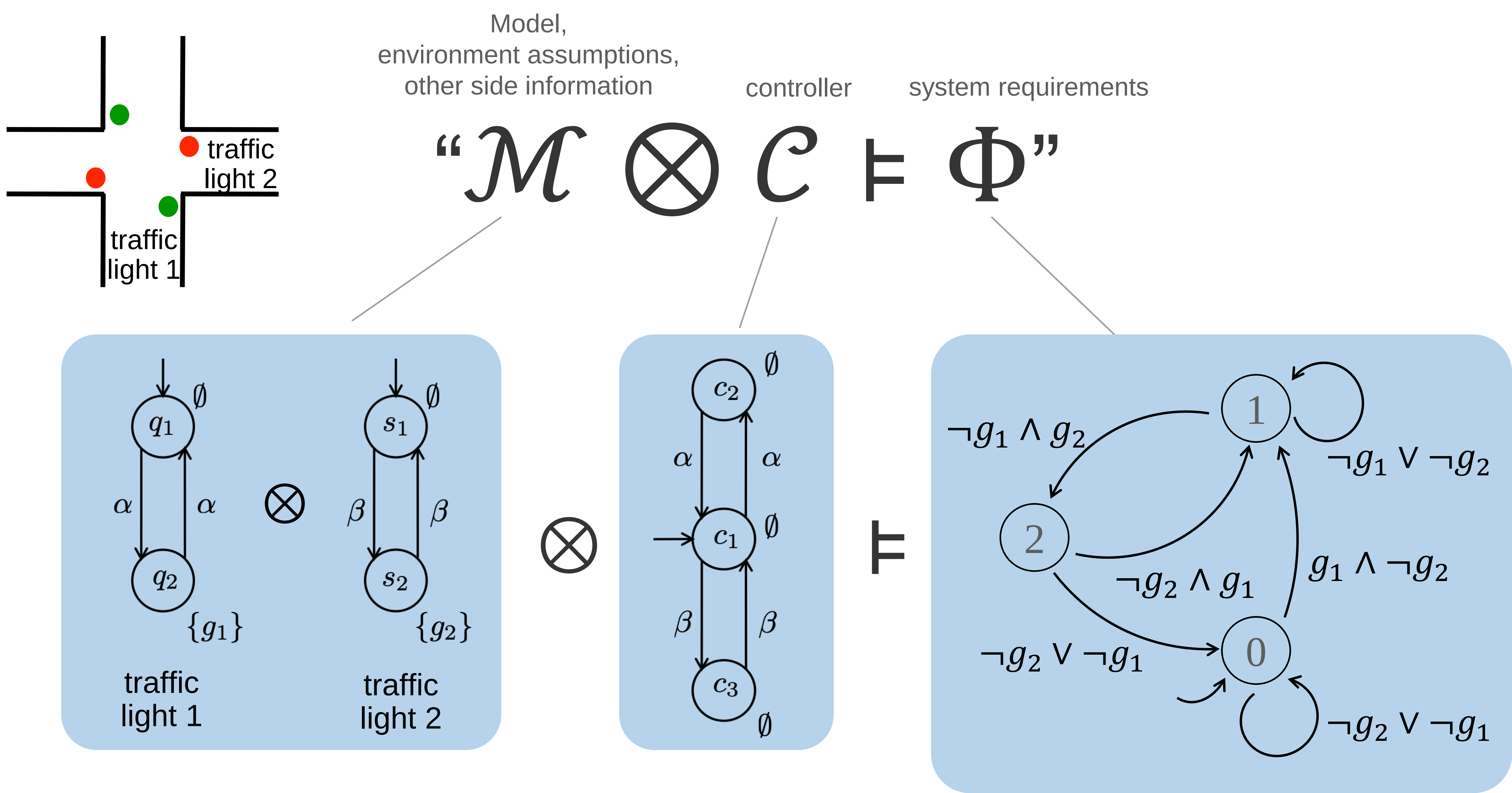


Why automaton-based representations? They are used for

- model checking, planning,…
- reactive synthesis, games on graphs, …
- probabilistic verification and synthesis, and
- reinforcement learning.

A (Very) Brief Introduction to Model Checking

Are the controller's outcomes **guaranteed** to satisfy user-specified requirements when implemented against a system model?

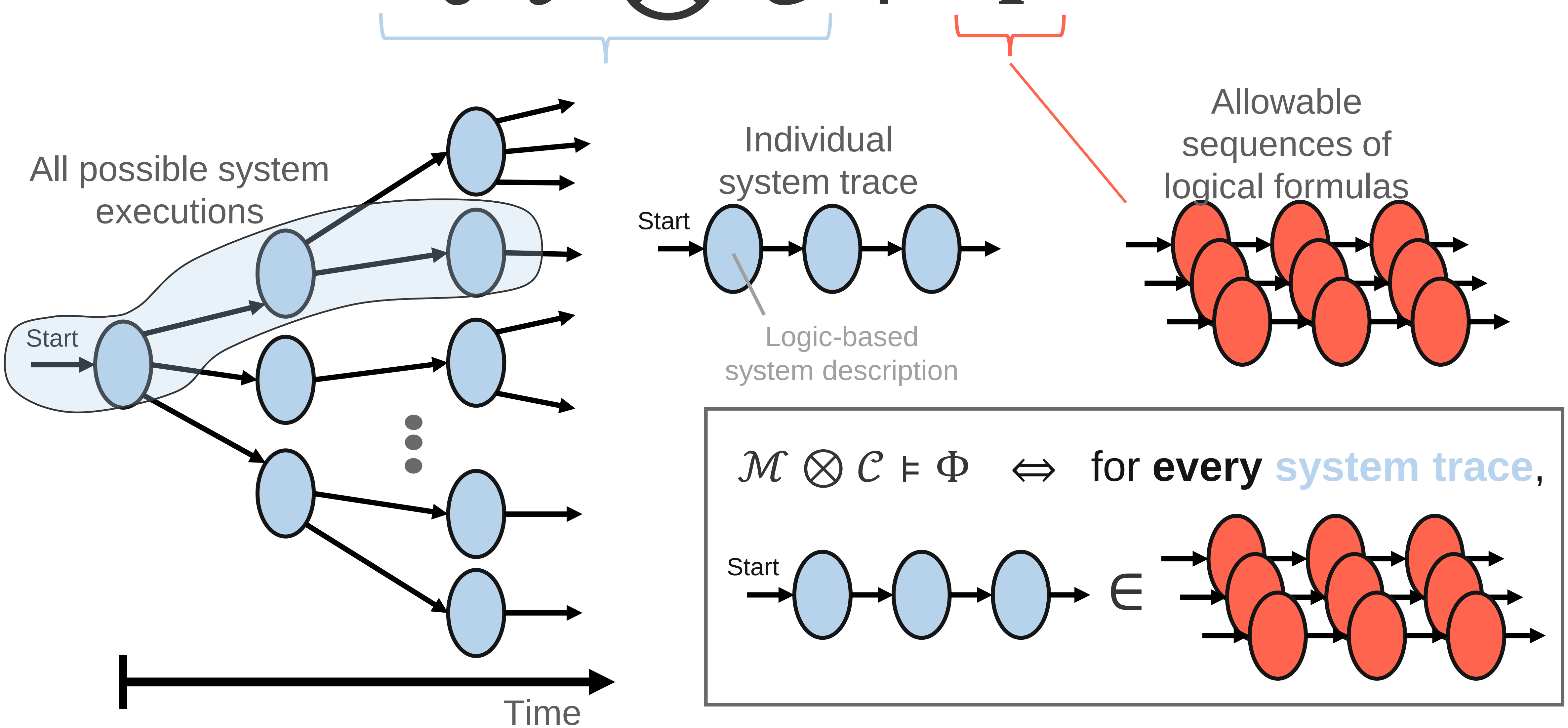


A (Very) Brief Introduction to Model Checking

Are the controller's outcomes **guaranteed** to satisfy user-specified requirements when implemented against a system model?

Model, environment assumptions, other side information | Controller | System requirements

$$\text{“}\mathcal{M} \otimes \mathcal{C} \models \Phi\text{”}$$

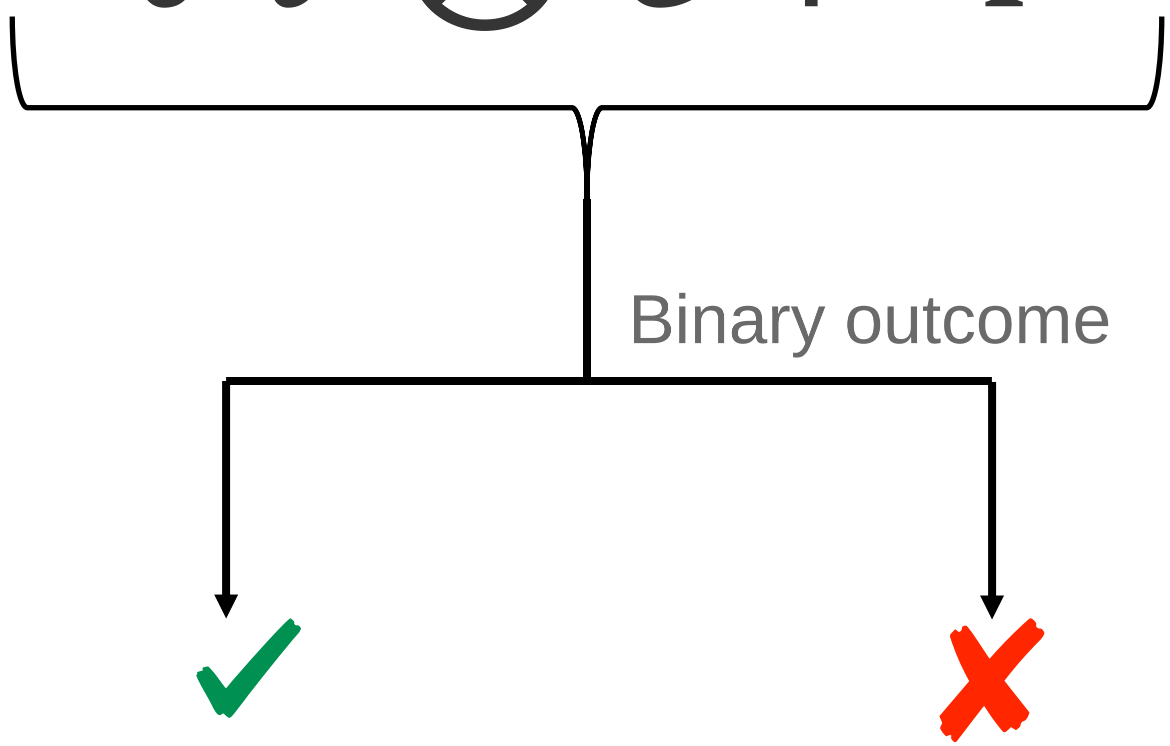


A (Very) Brief Introduction to Model Checking

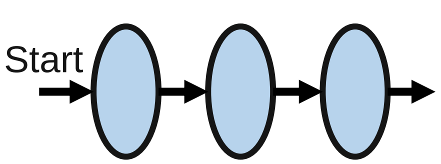
Are the controller's outcomes **guaranteed** to satisfy user-specified requirements when implemented against a system model?

Model, environment assumptions, other side information | Controller | System requirements

$$\text{“}\mathcal{M} \otimes \mathcal{C} \models \Phi\text{”}$$



Byproduct: Counterexample trace that **violates** the specification.



How to Connect Generative Models to Automata?

(GLM2FSA: Generative Language Model to Finite-State Automaton)

Input : "Cross the road"

Query GLM for steps and substeps

```
Steps for: Cross the road
[1] Look both ways before crossing the road.
[2] If there are no cars coming, proceed to cross the road.
[3] If there are cars coming, wait for them to pass before crossing the road.

Substeps for: [1] Look both ways before crossing the road.
[1.1] Face the direction you want to cross the road in.
[1.2] Look to the left.
[1.3] Look to the right.
[1.4] If there are no cars coming, proceed to [2]. If there are cars coming, proceed to [3].
...
```

query

response

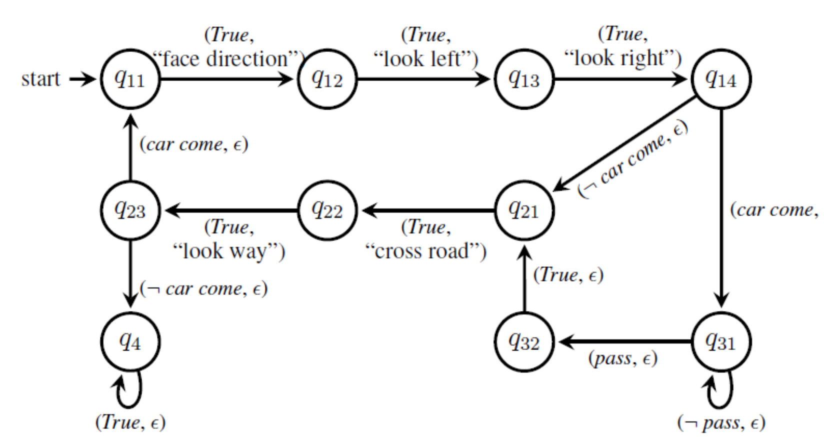
Parse sentences, extract keywords

[1] <look way> <before> <cross road>
[2] <if> <no car come> <proceed cross road>
[3] <if> <car come> <wait> <pass> <before> <cross road>
[1.1] <face direction want cross road>
[1.2] <look left>
[1.3] <look right>
[1.4] <if> <no car come> <proceed> <if> <car come> <proceed>
...

Build finite state automata from substeps

$Q = \{q_{11}, q_{12}, q_{13}, q_{14}, \ldots\}$
P = {car come, pass, ...}
A = {face direction, look left, look right, ...}

Visualize



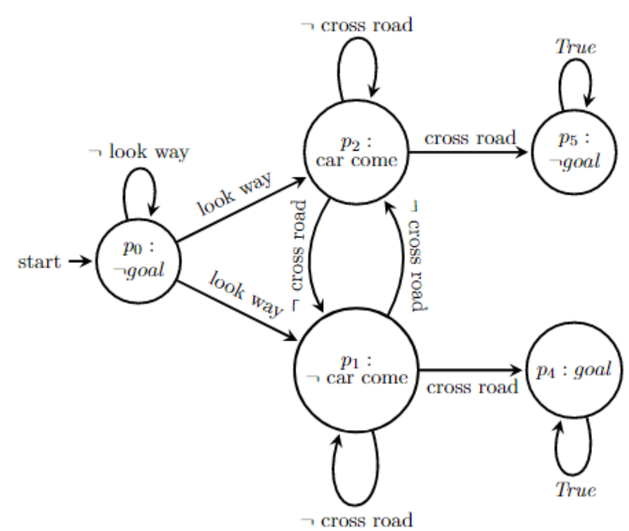
Verifying Whether the Generated Behaviors Satisfy User-Specified Requirements

$$\mathcal{M} \otimes \mathcal{C} \models \Phi$$

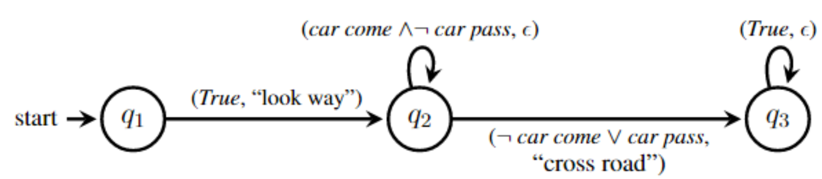
$\mathcal{M}$ Additional available information, e.g., a model

$\mathcal{C}$ Controller constructed using GLM2FSA

Φ Specification

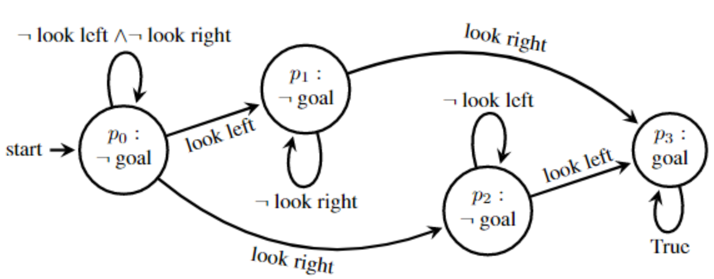


$\otimes$

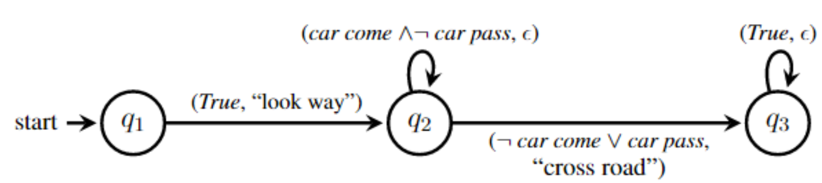


$\models \neg traffic\ light \rightarrow \Diamond\ goal$ ✔

"If not at a traffic light, eventually complete the task as specified by the model"

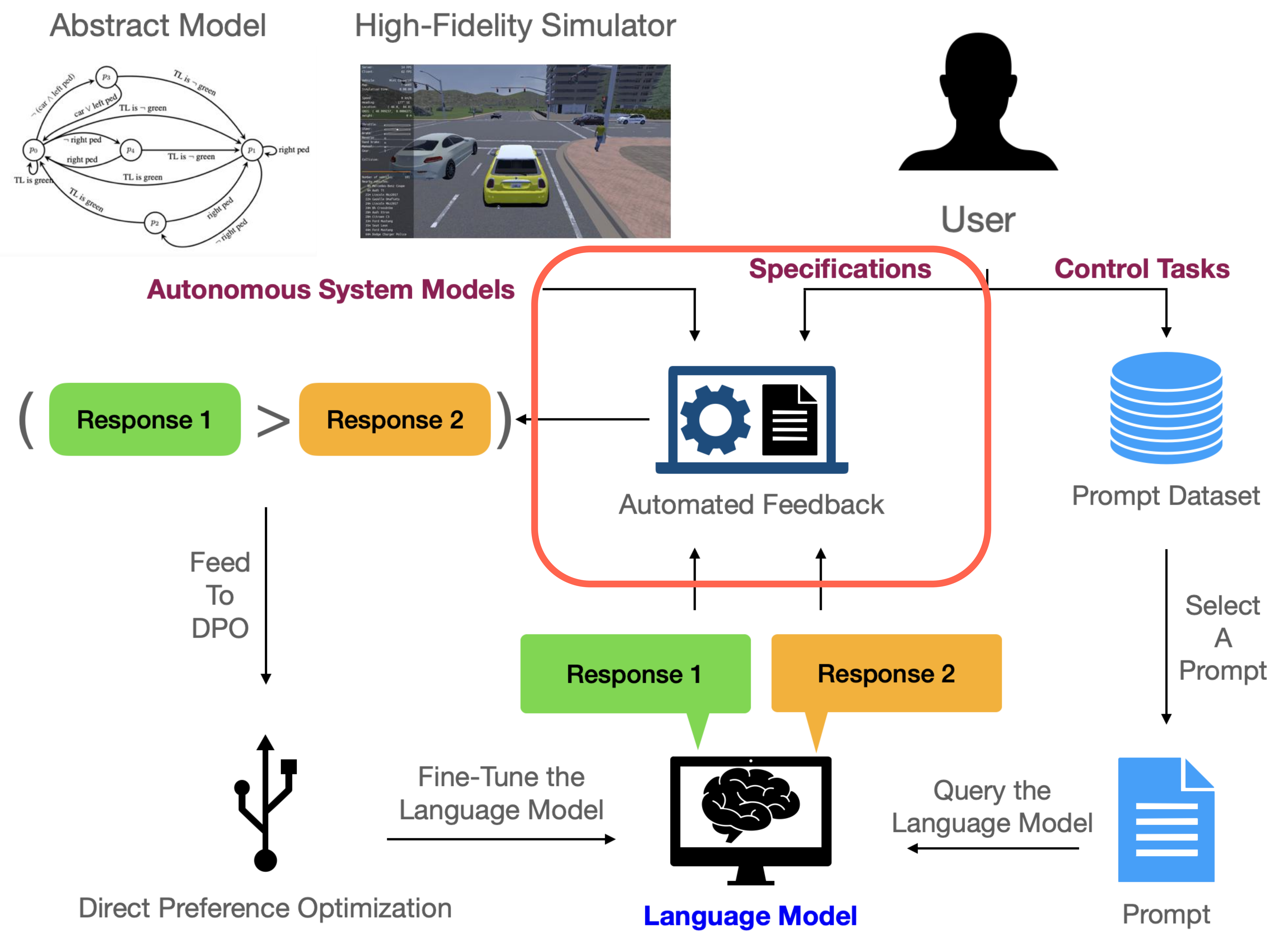


$\otimes$

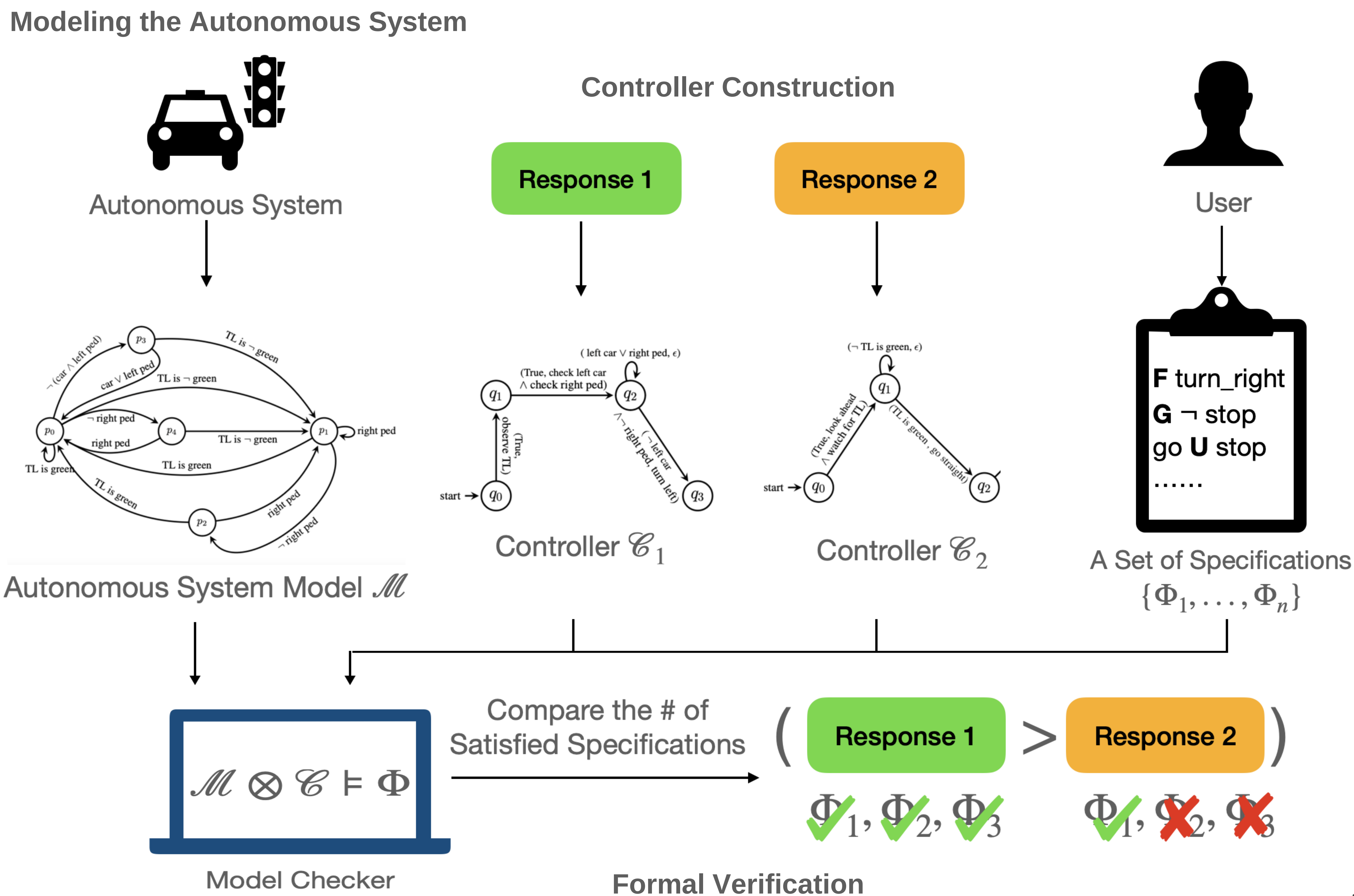


$\models \neg traffic\ light \rightarrow \Diamond\ goal$ ✘

Fine-Tuning Language Models Using Formal Methods Feedback

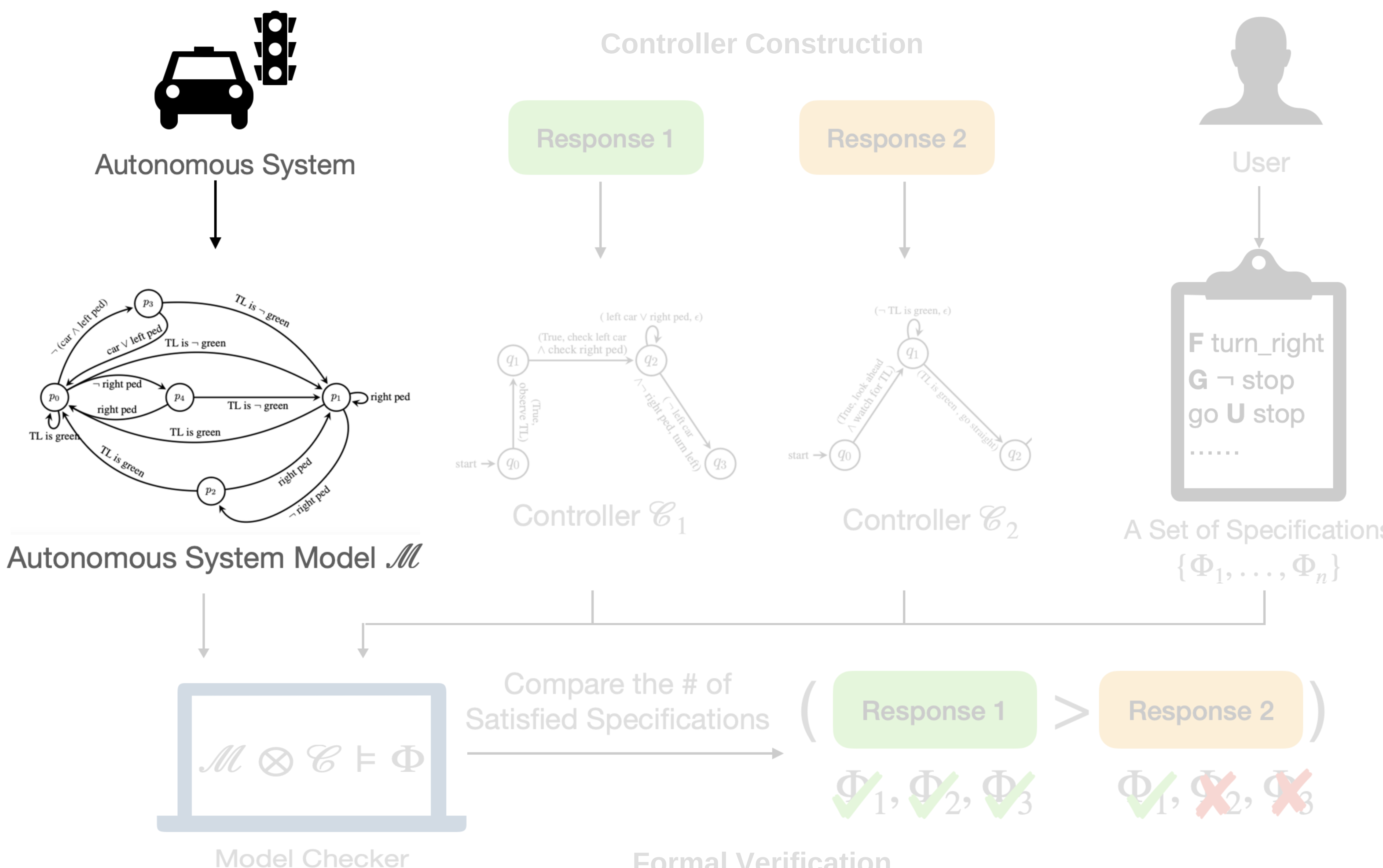


Fine-Tuning Language Models Using Formal Methods Feedback



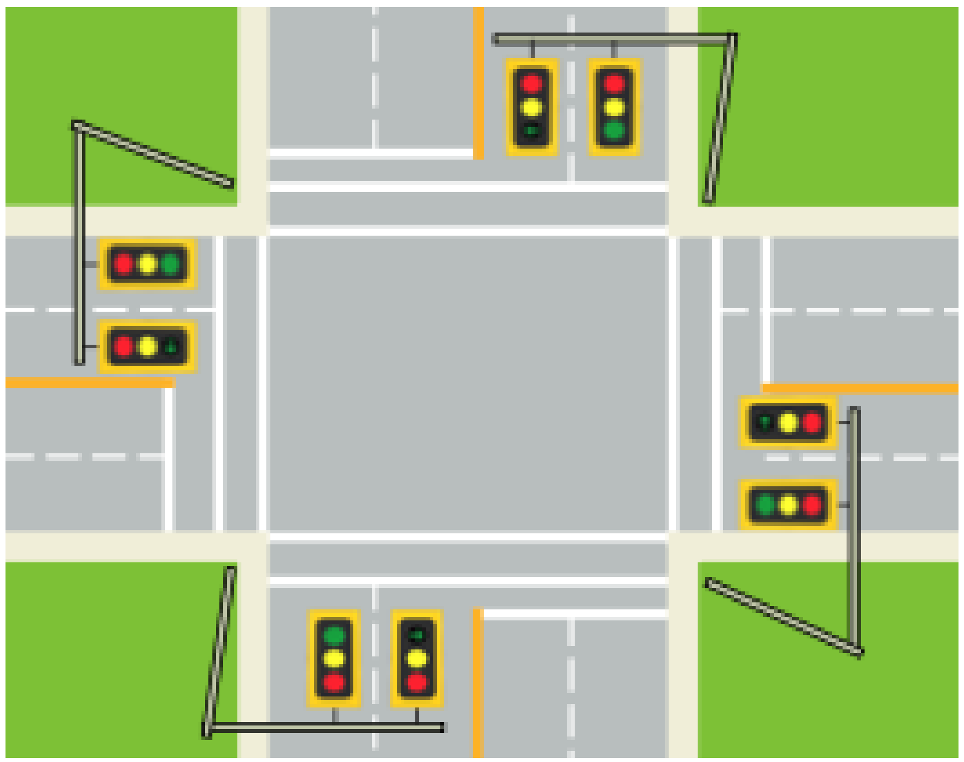
Fine-Tuning Language Models Using Formal Methods Feedback

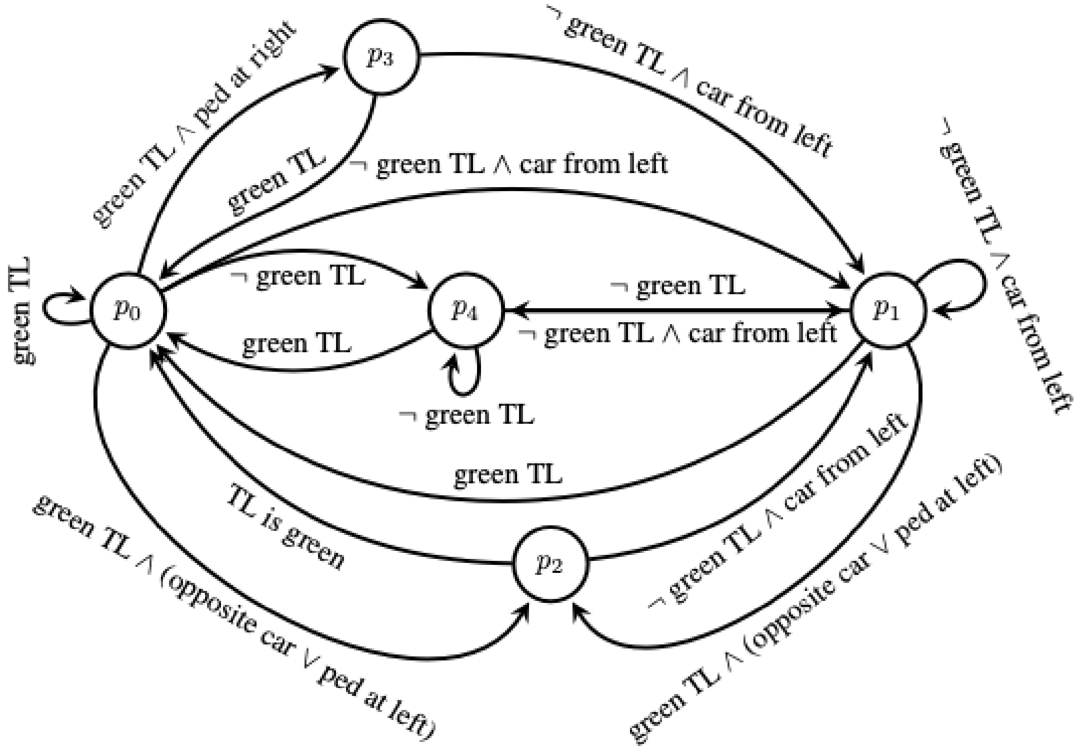
Modeling the Autonomous System

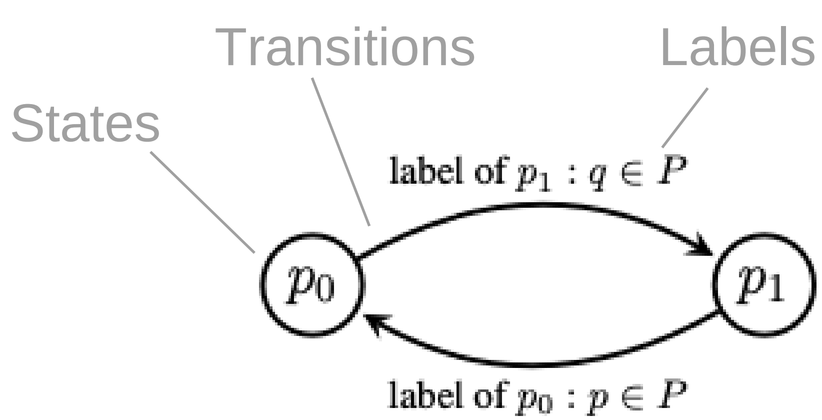


Formal Methods Feedback in an Autonomous Driving System

Modeling the Autonomous System





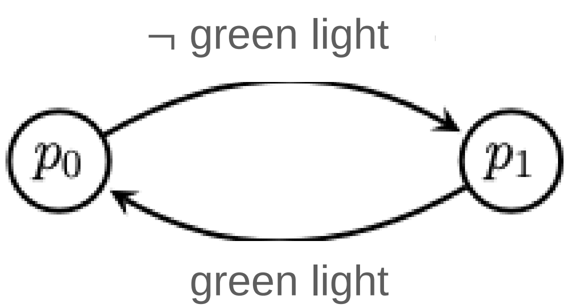


Formal Methods Feedback in an Autonomous Driving System

Modeling the Autonomous System

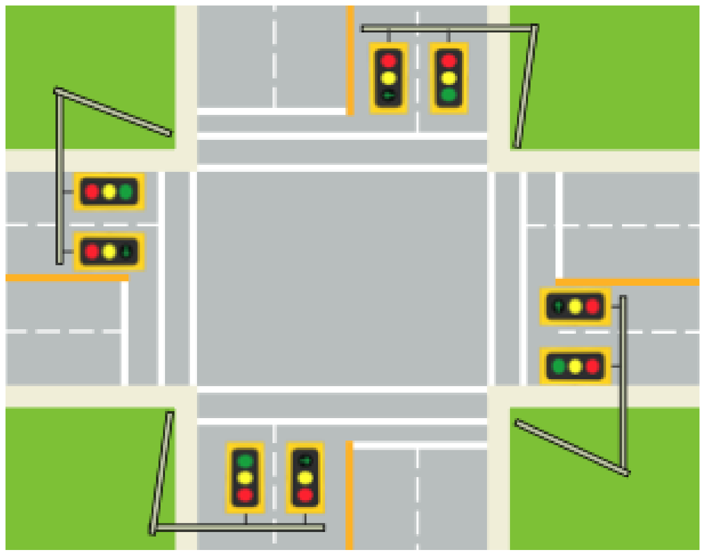
$P = \{green\ light\}$
$S \sim Traffic\ Light$

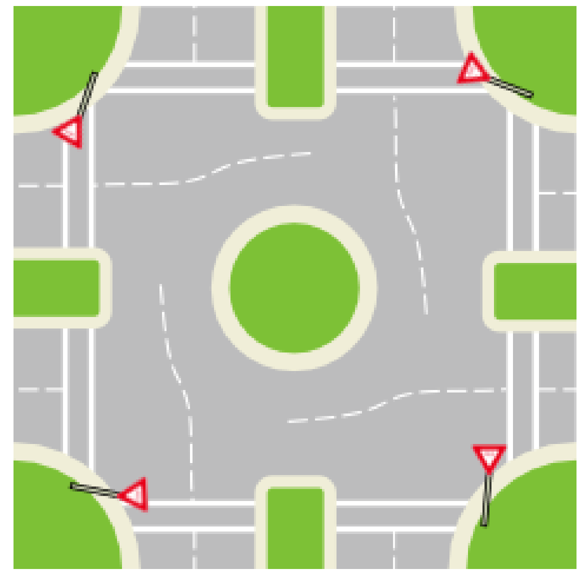
$Q_M = \{p_0\ p_1\}$
$\lambda_M(p_0) = green\ light$
$\lambda_M(p_1) = \neg\ green\ light$
$\delta_M(p_0, p_1) = 1$
$\delta_M(p_1, p_0) = 1$

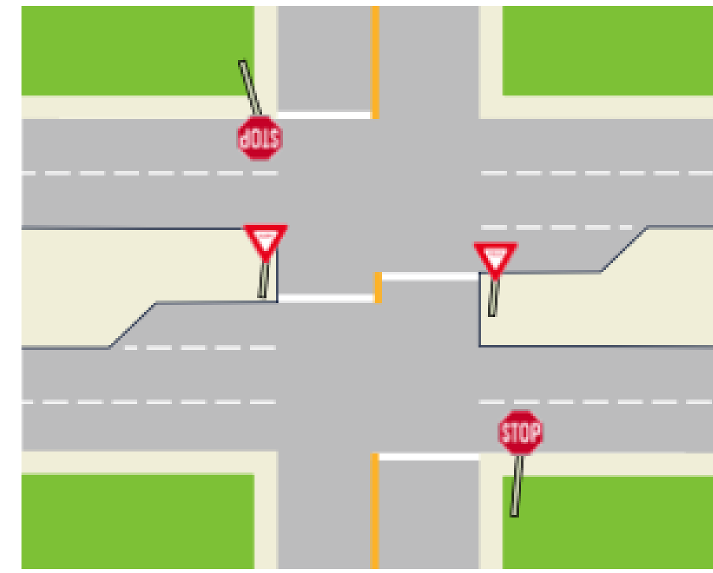


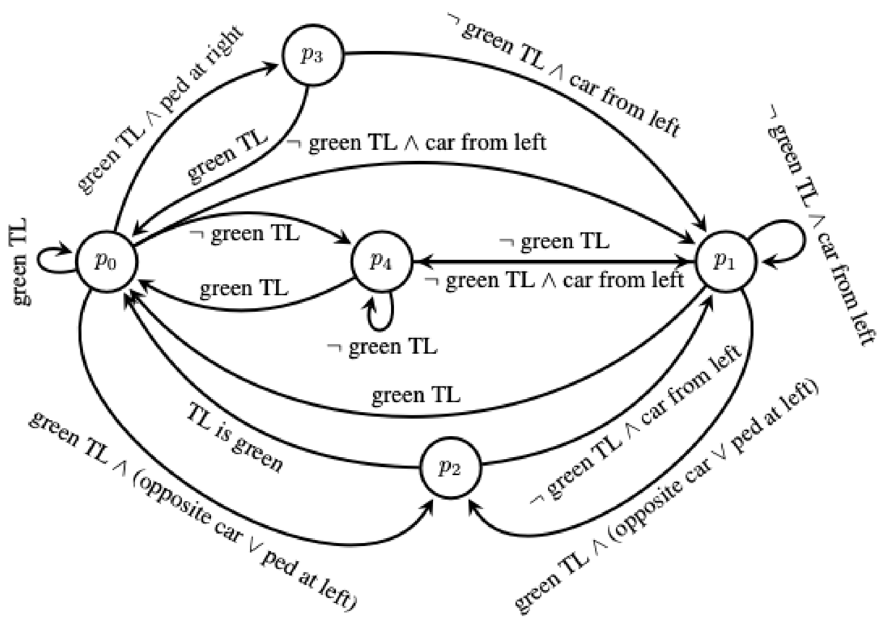
Formal Methods Feedback in an Autonomous Driving System

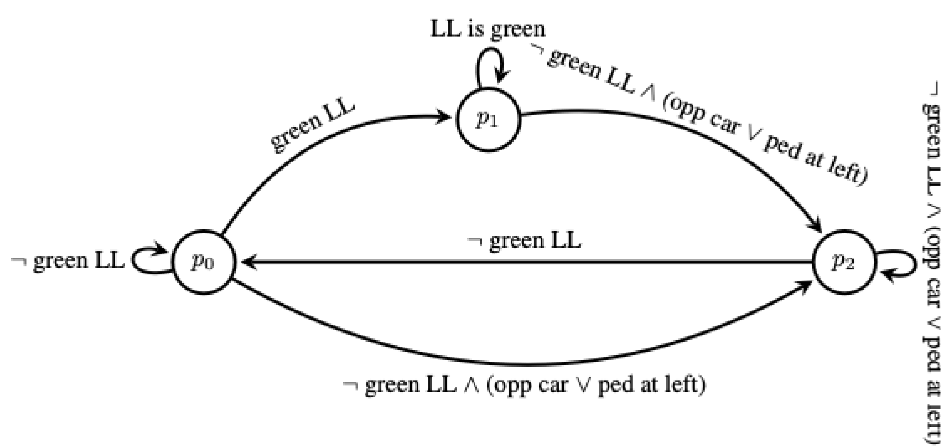
Modeling the Autonomous System

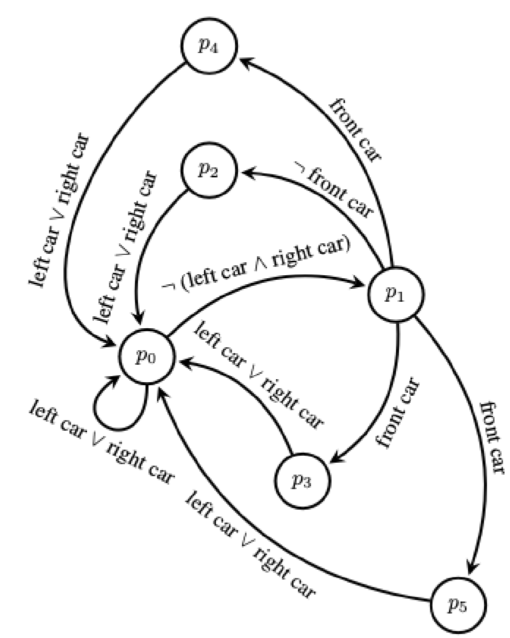




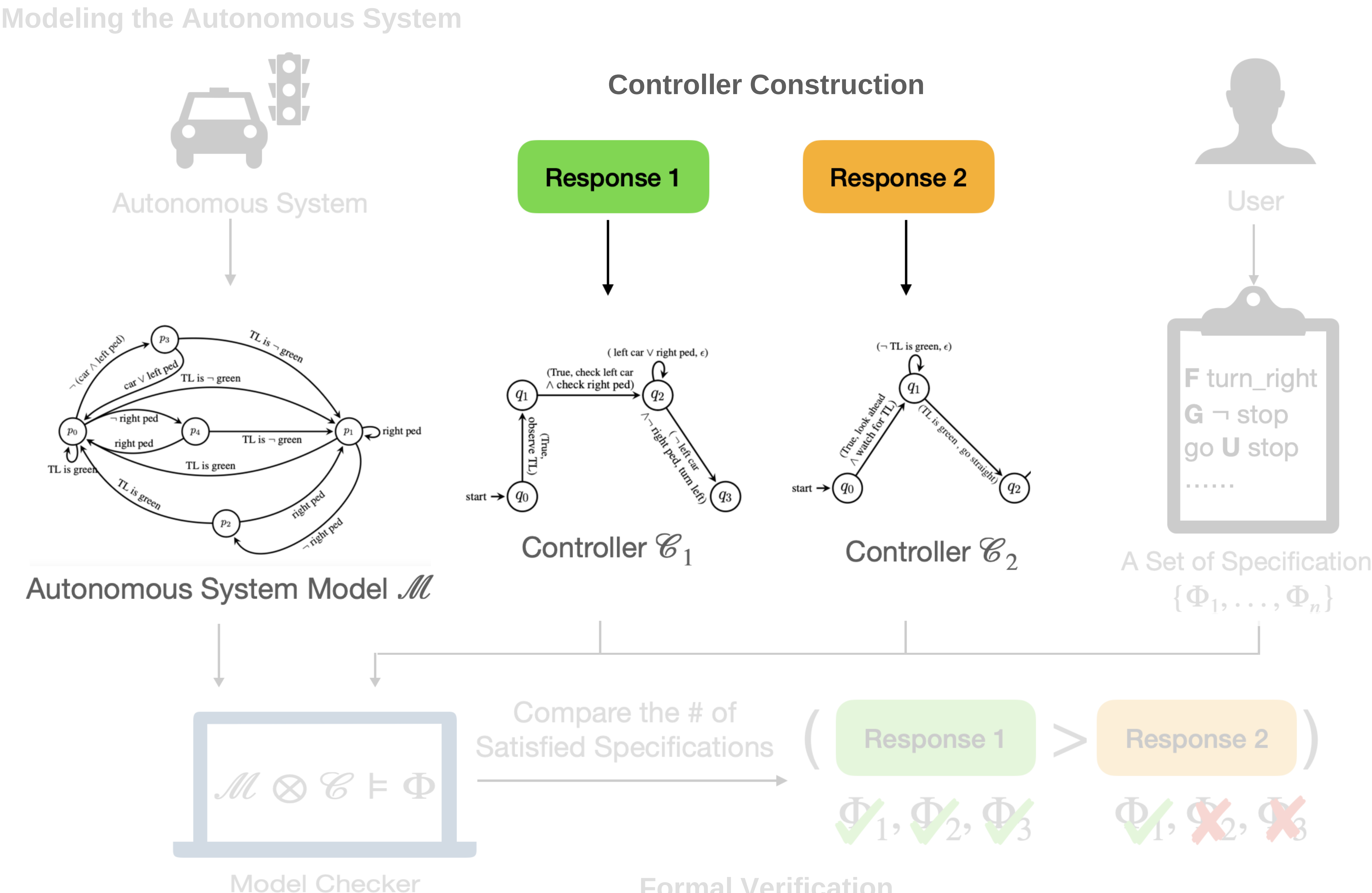








Fine-Tuning Language Models Using Formal Methods Feedback



Yang, Bhatt, Ingebrand, Ward, Carr, Wang, and Topcu (The University of Texas at Austin)

Formal Methods Feedback in an Autonomous Driving System

Controller Construction

User

Steps for turning right at the traffic light

1. Look straight ahead and watch for traffic light.
2. If the traffic light turns green, start moving forward.
3. As you approach the intersection, look to your left for oncoming traffic.
4. If there is no traffic coming from your left, check pedestrians on your right.
5. If it is safe, turn your vehicle right.

Language Model

1. Observe the traffic light in front of you.
2. Check for the left approaching car and right side pedestrian.
3. If no car from the left is approaching and no pedestrian on the right, proceed to turn right.

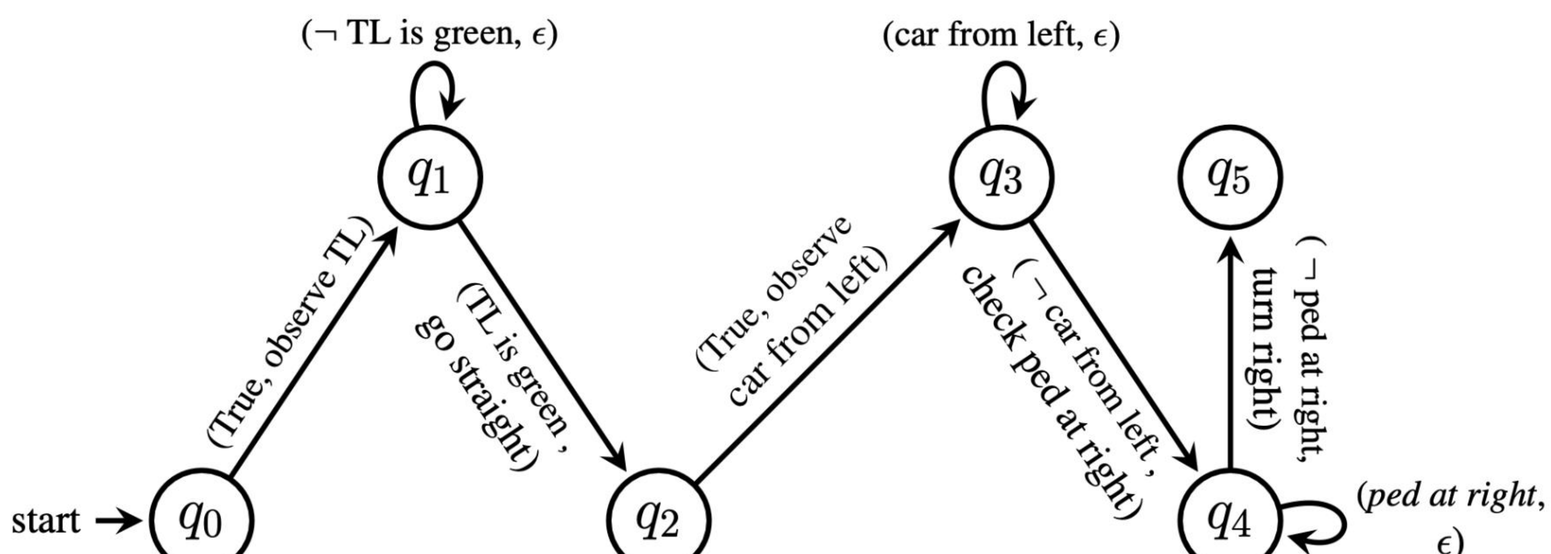
Formal Methods Feedback in an Autonomous Driving System

Controller Construction

1. Look straight ahead and watch for traffic light.
2. If the traffic light turns green, start moving forward.
3. As you approach the intersection, look to your left for oncoming traffic.
4. If there is no traffic coming from your left, check pedestrians on your right.
5. If it is safe, turn your vehicle right.

Parse the sentence
Align the vocabulary to *P* and *A*

```
1. <observe traffic light>.
2. <if> <green traffic light>, <go straight>.
3. <observe car from left>.
4. <if> <no car from left>, <check pedestrian
     at right>.
5. <if> <no pedestrian at right>, <turn
    right>.
```

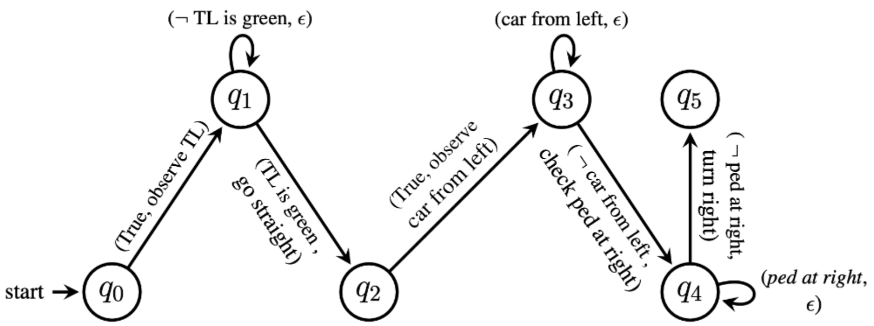


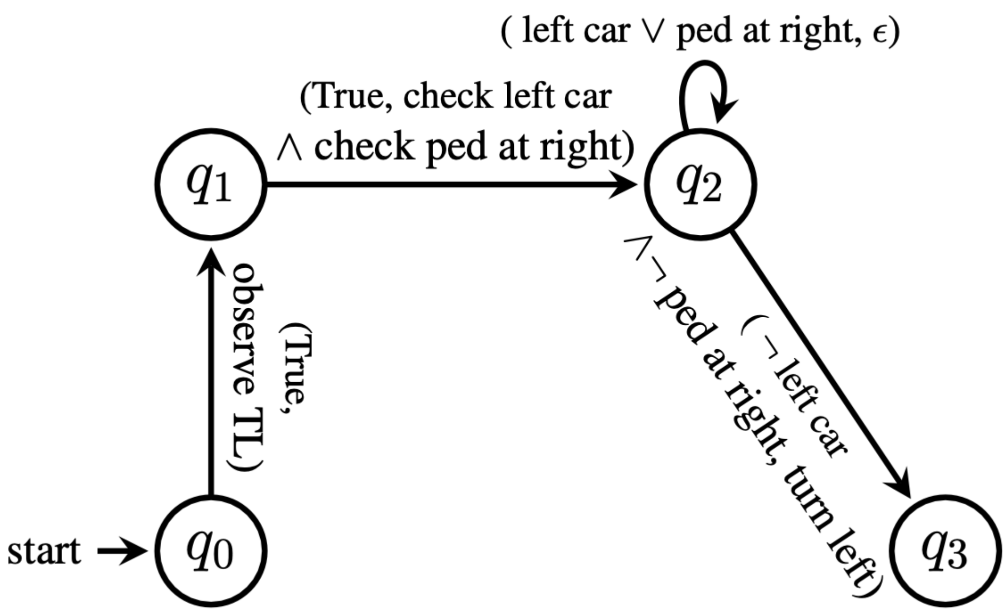
One state per step
Transition input: condition
Transition output: action

Formal Methods Feedback in an Autonomous Driving System

Controller Construction

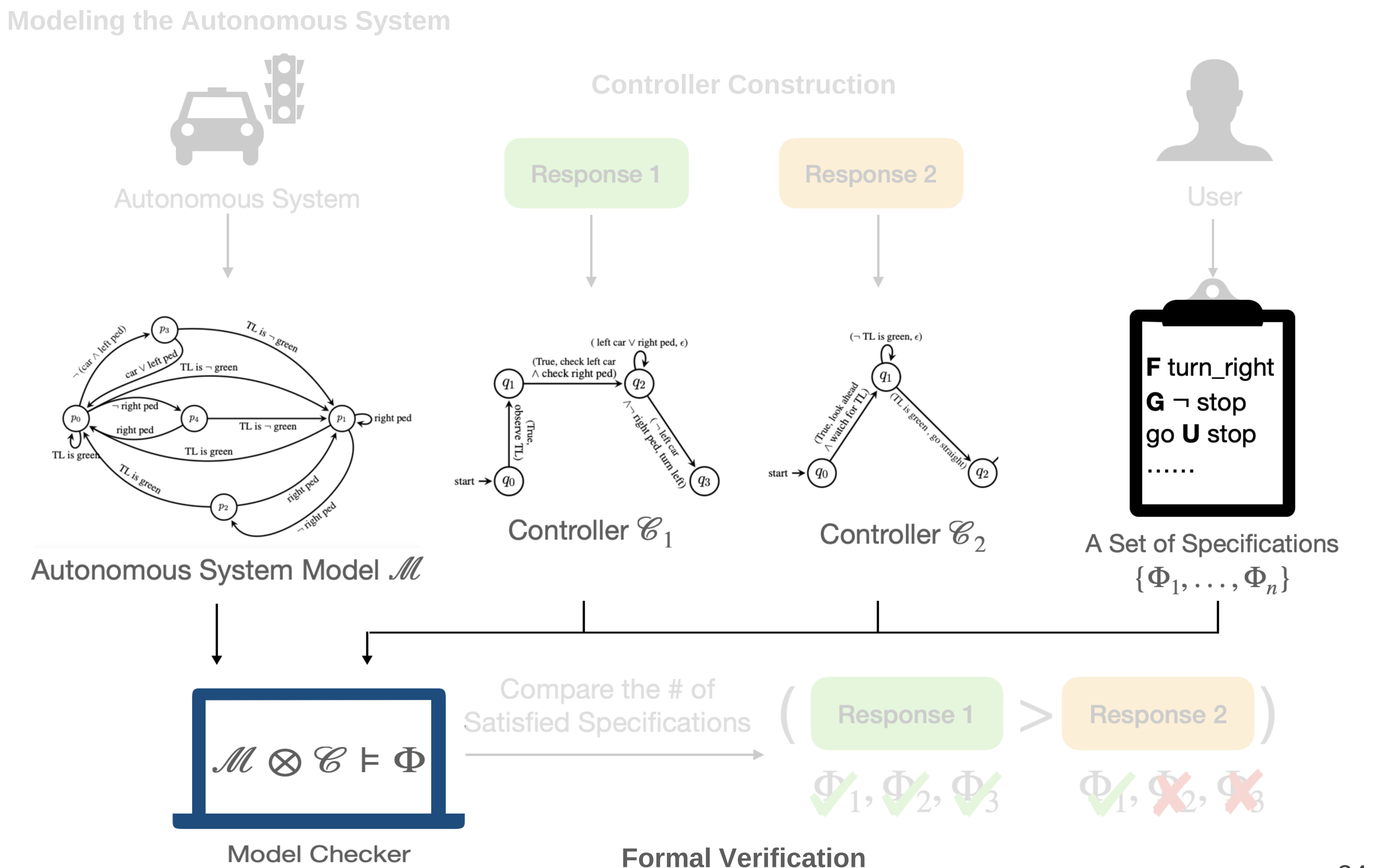
1. Look straight ahead and watch for traffic light.
2. If the traffic light turns green, start moving forward.
3. As you approach the intersection, look to your left for oncoming traffic.
4. If there is no traffic coming from your left, check pedestrians on your right.
5. If it is safe, turn your vehicle right.





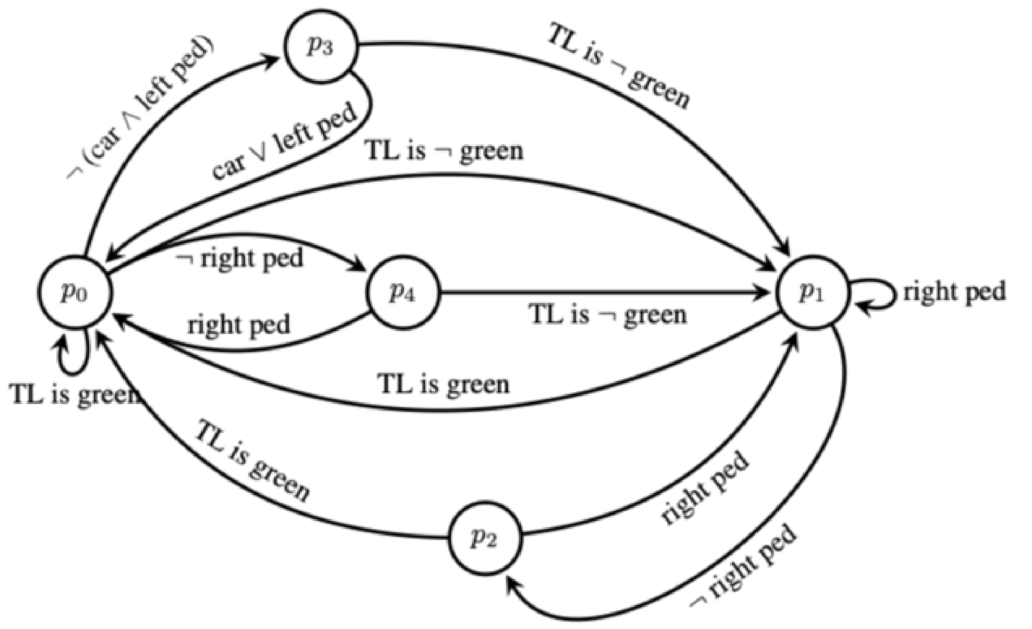
1. Observe the traffic light in front of you.
2. Check for the left approaching car and right side pedestrian.
3. If no car from the left is approaching and no pedestrian on the right, proceed to turn right.

Fine-Tuning Language Models Using Formal Methods Feedback

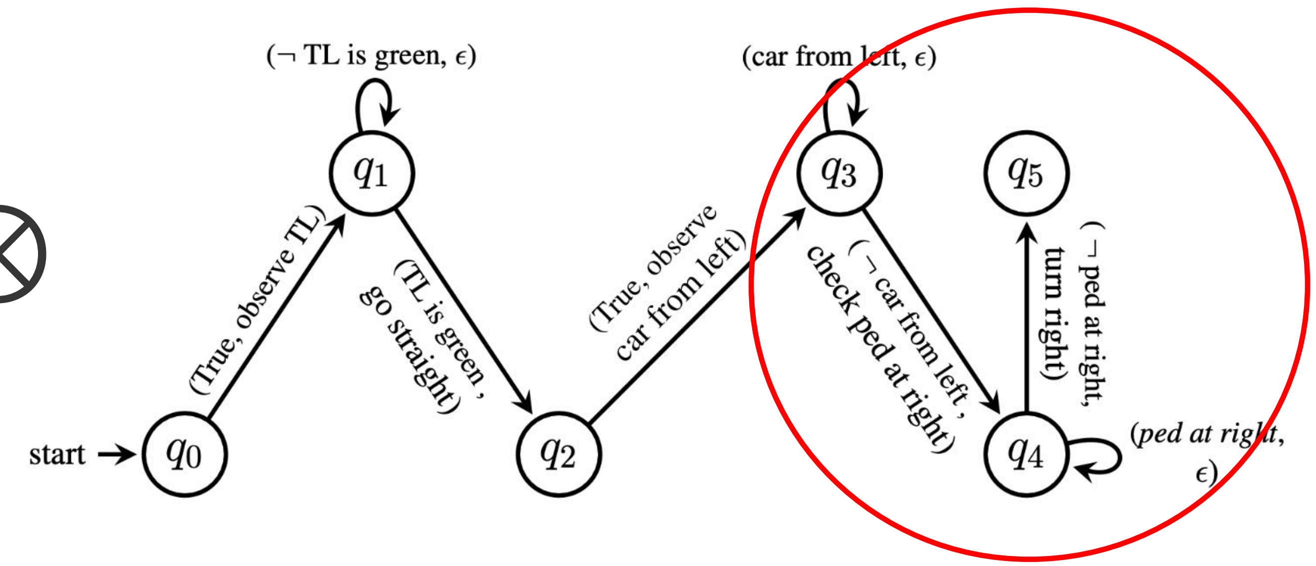


Formal Methods Feedback in an Autonomous Driving System

Formal Verification



$\otimes$



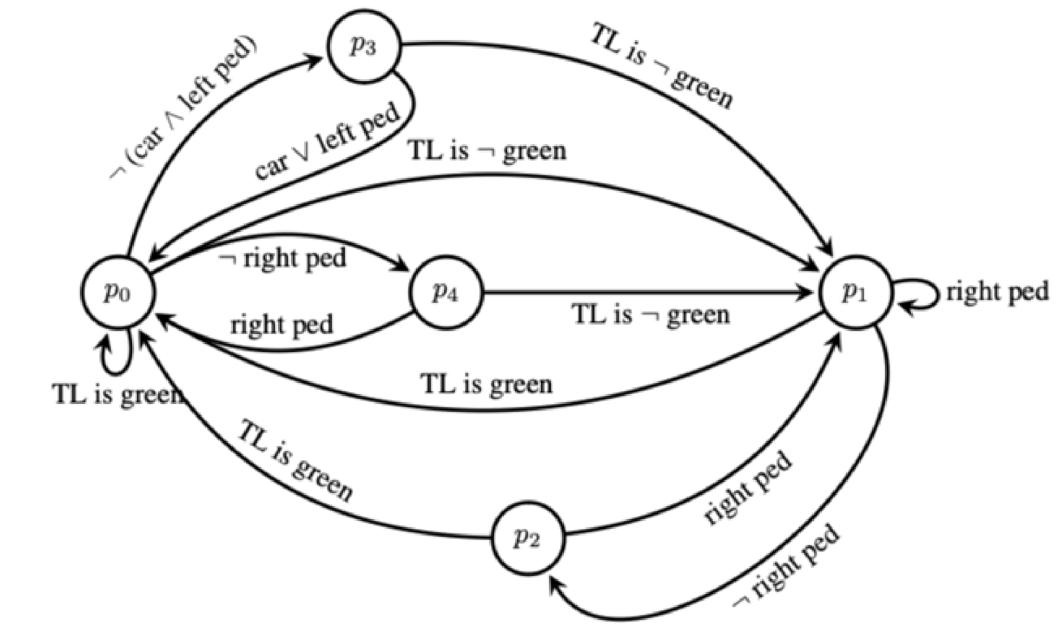
$\square(\neg\text{green traffic light} \to \neg\text{go straight})$, ✓

$\square(\text{stop sign} \to \Diamond\,\text{stop})$, ✓

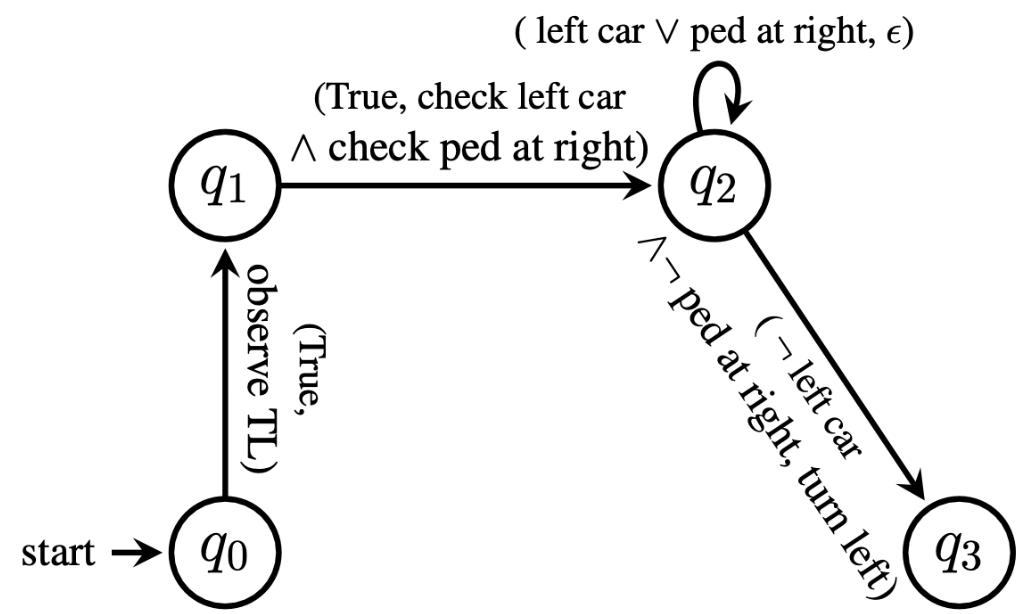
$\square\,\neg\text{turn right} \vee \neg(\text{car from left} \vee \text{pedestrian at right})$, ✗

Formal Methods Feedback in an Autonomous Driving System

Formal Verification



$$\otimes$$

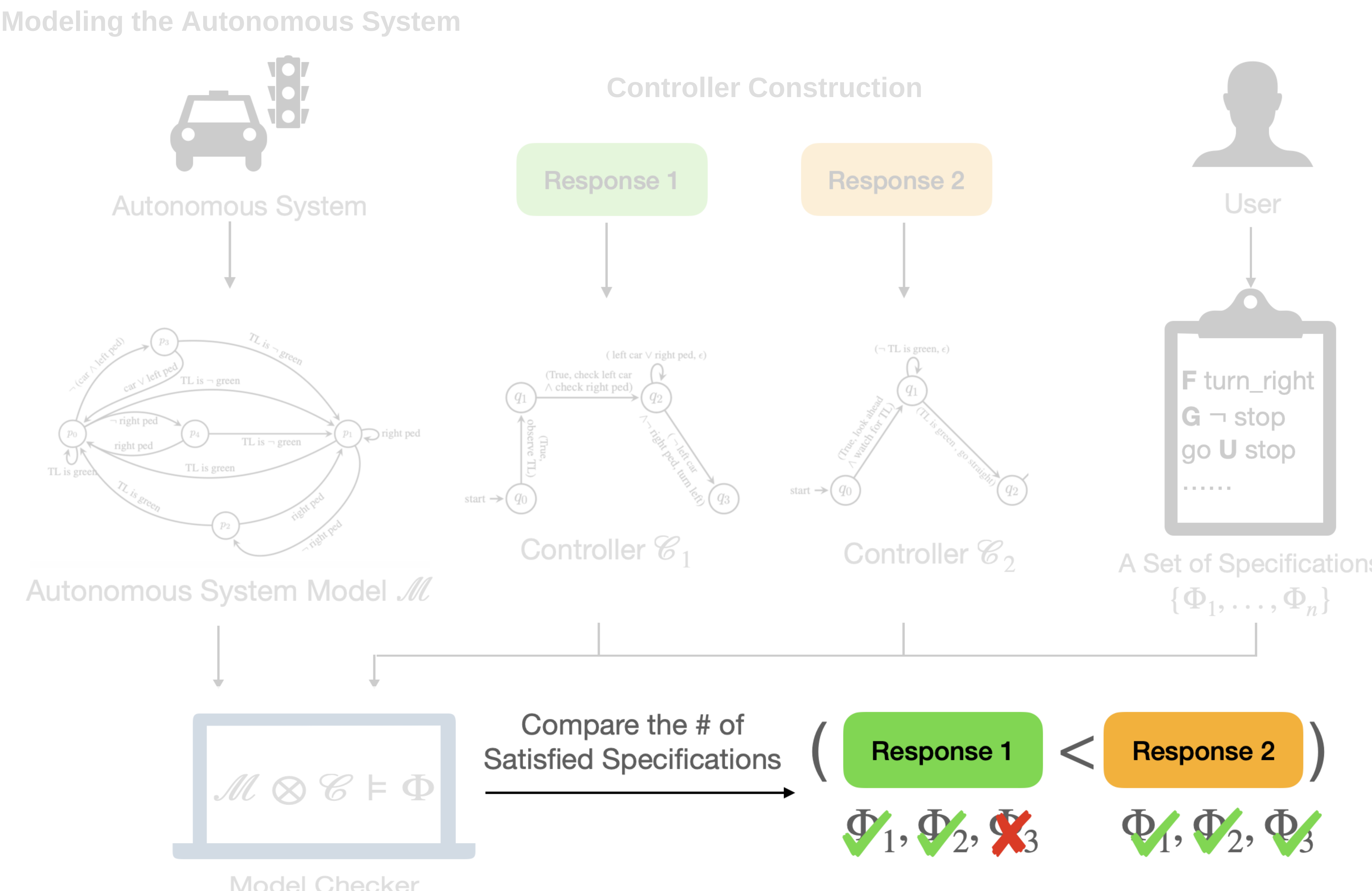


$\square(\neg\text{green traffic light} \rightarrow \neg\text{go straight}),$

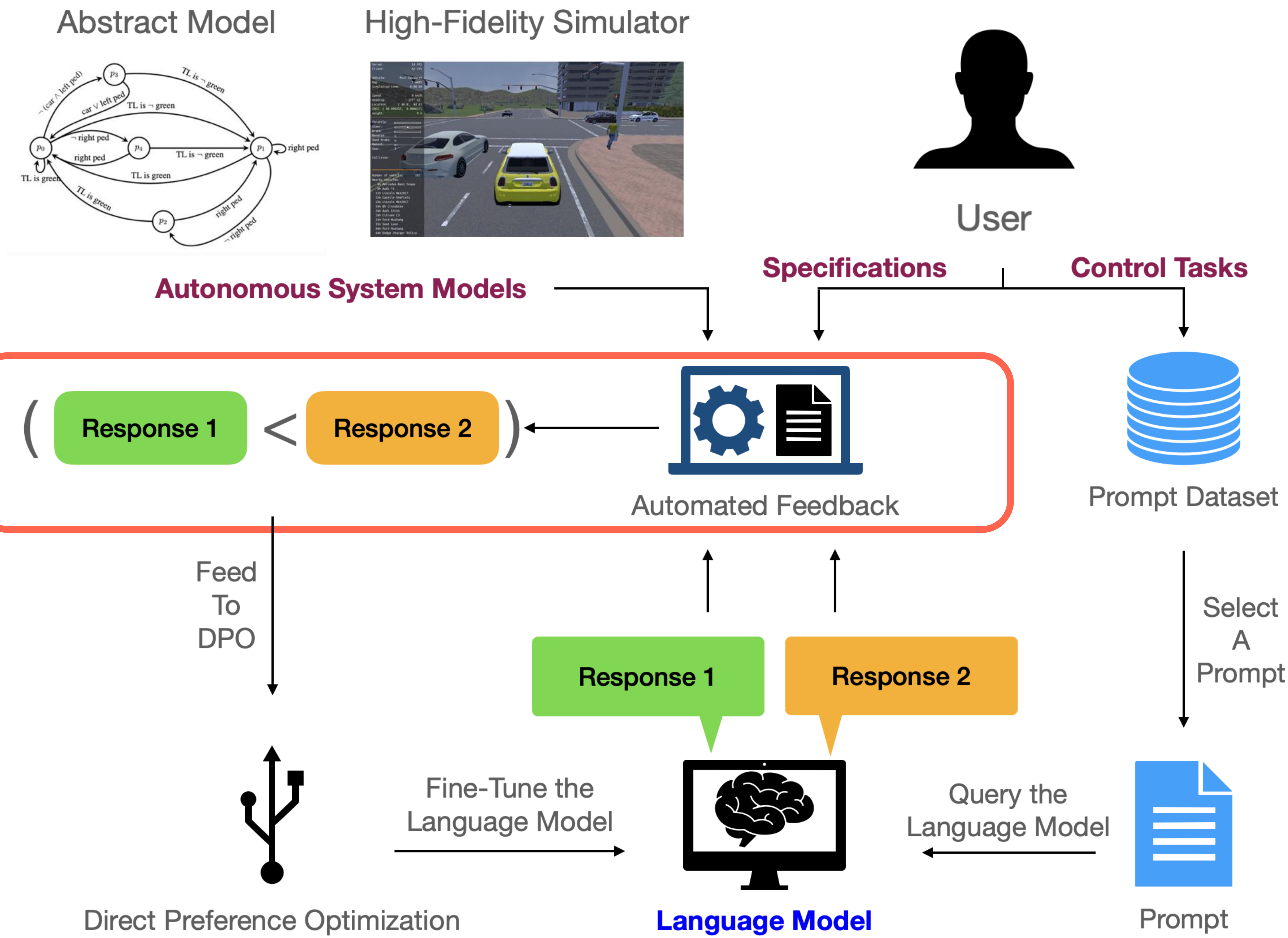
$\square(\text{stop sign} \rightarrow \Diamond\, \text{stop}),$

$\square\ \neg\text{turn right} \vee \neg(\text{car from left} \vee \text{pedestrian at right}),$

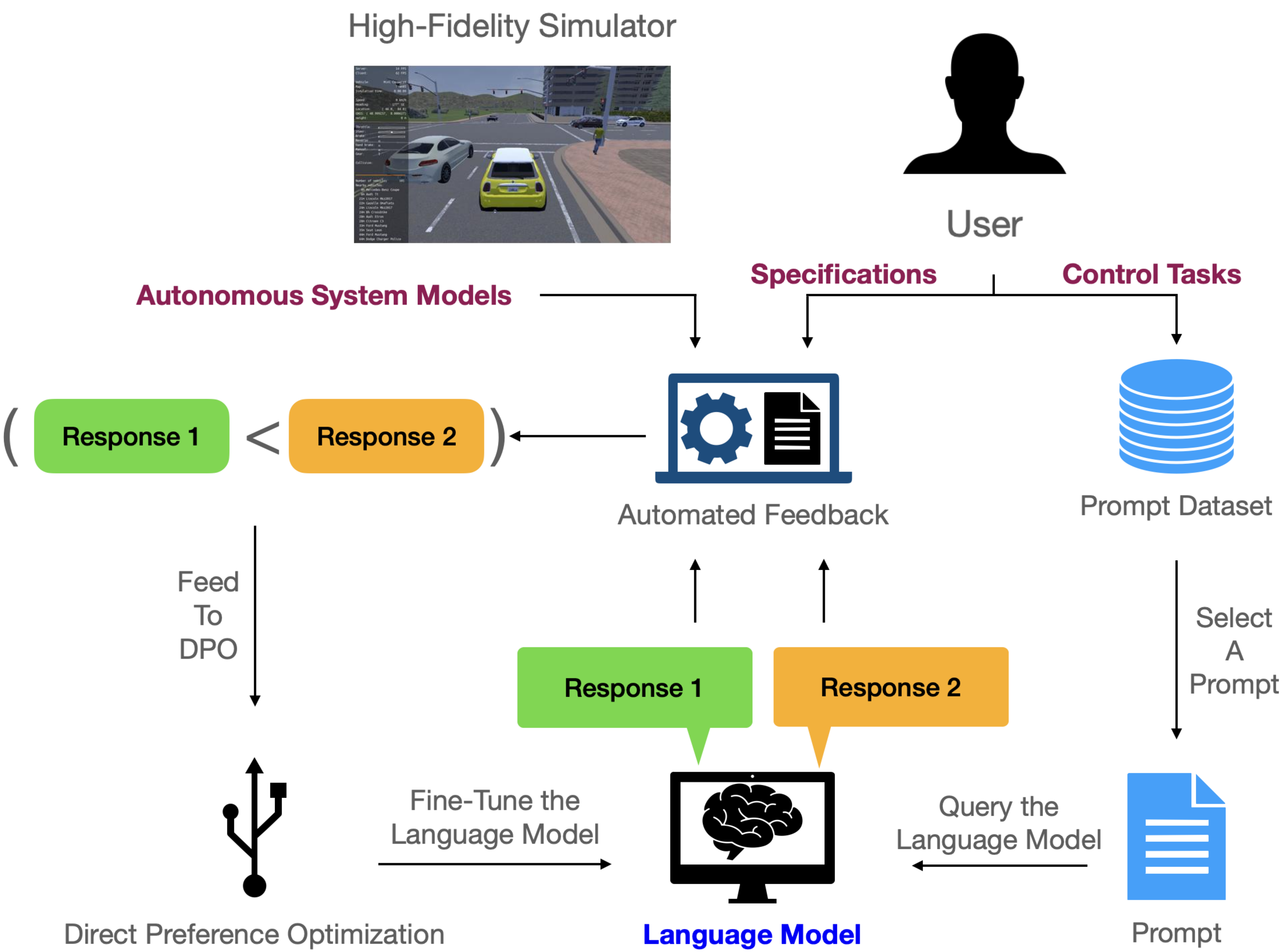
Fine-Tuning Language Models Using Formal Methods Feedback



Fine-Tuning Language Models Using Formal Methods Feedback



Fine-Tuning Language Models Using Formal Methods Feedback

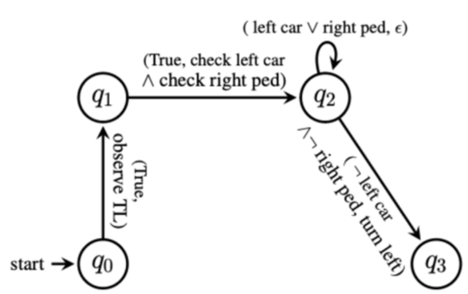


Fine-Tuning Language Models Using Formal Methods Feedback

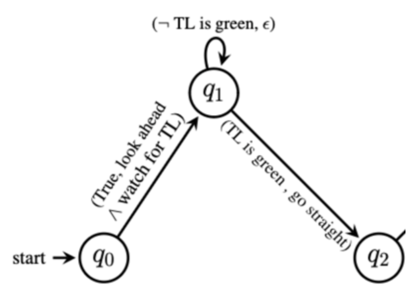
Empirical Evaluation via Simulation

Response 1

Response 2



Controller $\mathscr{C}_1$



Controller $\mathscr{C}_2$

High-Fidelity Simulator

Information from the Simulator

```
-----------------------------------------
ector3D(x=10.495529, y=5.662697, z=2.781272) Class: TL or TS
Vector3D(x=2.850960, y=18.144859, z=2.781269) Class: TL or TS
Vector3D(x=-9.272949, y=29.014732, z=2.781277) Class: TL or TS
-----------------------------------------


-----------------------------------------
Vector3D(x=3.002785, y=17.807884, z=2.781323) Class: TL or TS
Vector3D(x=-9.121124, y=28.677757, z=2.781330) Class: TL or TS
-----------------------------------------


-----------------------------------------
ector3D(x=3.161827, y=17.457993, z=2.782099) Class: TL or TS
ctor3D(x=-8.962082, y=28.327866, z=2.782107) Class: TL or T
```

Execution Trace

$(left\_car \wedge go\_straight, pedestrian \wedge stop, \neg pedestrian \wedge go\_straight)$

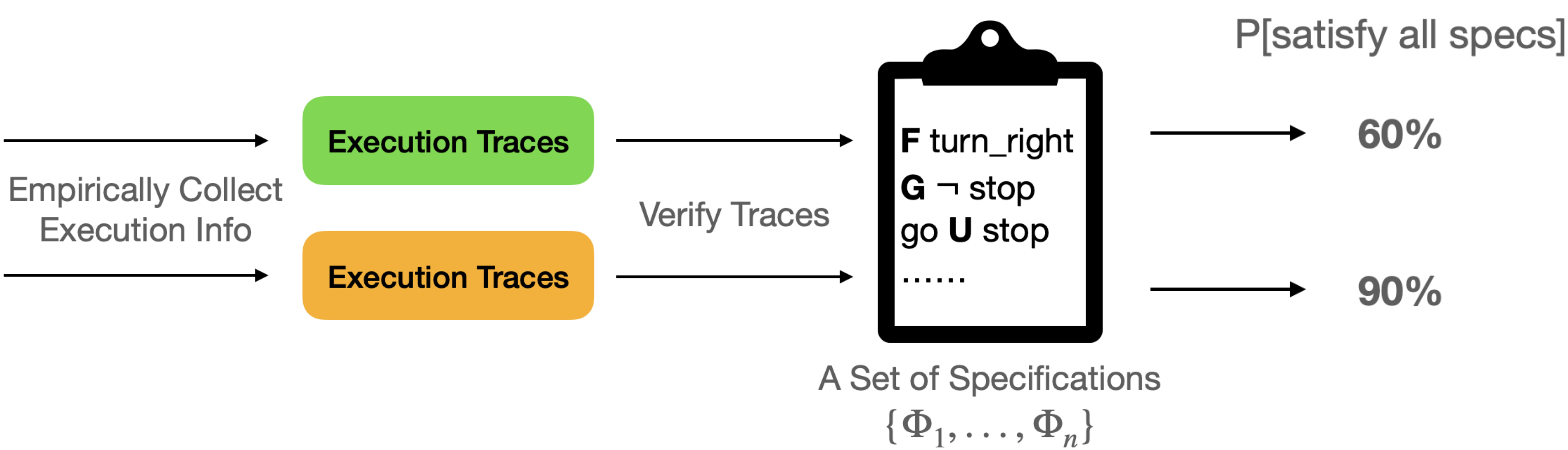
Empirically Collect Execution Info

Execution Traces

Execution Traces

Fine-Tuning Language Models Using Formal Methods Feedback

Empirical Evaluation via Simulation



Quantitative Analysis

Empirical Evaluation via Simulation

Carla Simulator: Extract execution traces.

Carla Simulation Video

Object and Position Information

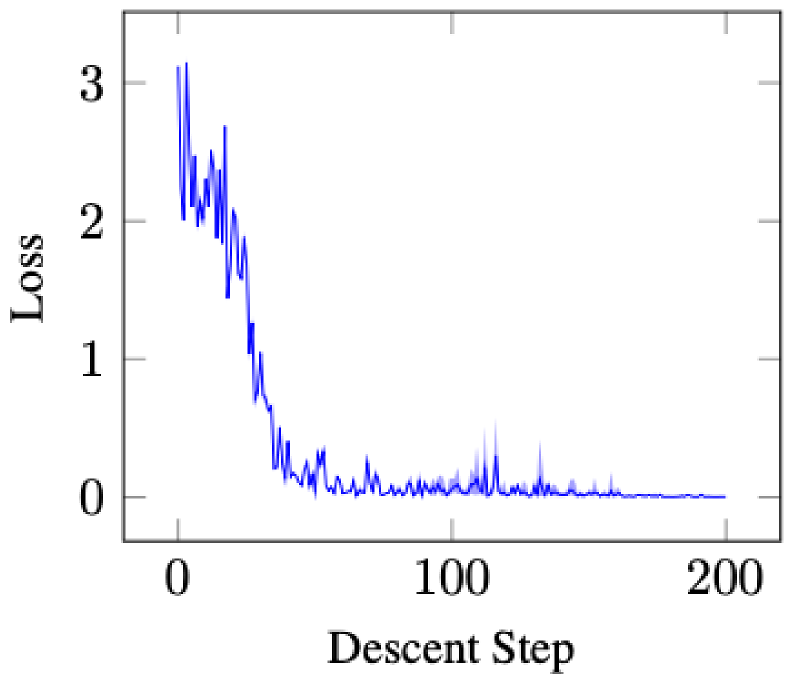
```
------------------------------------------


------------------------------------------
Vector3D(x=7.464157, y=-23.339211, z=2.899784) Class: TL or TS
Vector3D(x=7.464157, y=-23.339211, z=2.232456) Class: TL or TS
Vector3D(x=7.627457, y=-43.336159, z=2.806130) Class: TL or TS
Vector3D(x=7.699051, y=-43.277988, z=1.985428) Class: TL or TS
Vector3D(x=-12.249878, y=-28.675850, z=2.482181) Class: TL or TS
Vector3D(x=-12.471230, y=-8.660095, z=3.196321) Class: TL or TS
Vector3D(x=-12.468399, y=-8.708710, z=-1.387742) Class: TL or TS
Vector3D(x=-7.435089, y=-44.920258, z=0.036881) Vehicle, ID:  91
Vector3D(x=2.802956, y=-17.326714, z=0.022928) Vehicle, ID:  89
Vector3D(x=-7.692642, y=-21.904179, z=-0.002782) Vehicle, ID:  87
------------------------------------------


------------------------------------------
Vector3D(x=7.518684, y=-22.961929, z=2.900478) Class: TL or TS
Vector3D(x=7.518684, y=-22.961929, z=2.233150) Class: TL or TS
Vector3D(x=7.681984, y=-42.958878, z=2.806824) Class: TL or TS
Vector3D(x=7.753578, y=-42.900707, z=1.986122) Class: TL or TS
Vector3D(x=-12.210609, y=-48.282810, z=2.822465) Class: TL or TS
Vector3D(x=-12.270897, y=-48.344036, z=1.984991) Class: TL or TS
Vector3D(x=-12.195351, y=-28.298569, z=2.482875) Class: TL or TS
Vector3D(x=-12.416702, y=-8.282814, z=3.197015) Class: TL or TS
Vector3D(x=-12.413872, y=-8.331429, z=-1.387048) Class: TL or TS
Vector3D(x=-7.385361, y=-44.102268, z=0.036914) Vehicle, ID:  91
Vector3D(x=2.859573, y=-17.434631, z=0.023687) Vehicle, ID:  89
Vector3D(x=-7.644753, y=-20.938622, z=-0.002182) Vehicle, ID:  87
------------------------------------------
```

Execution Trace: (desired objects with positions, action),……

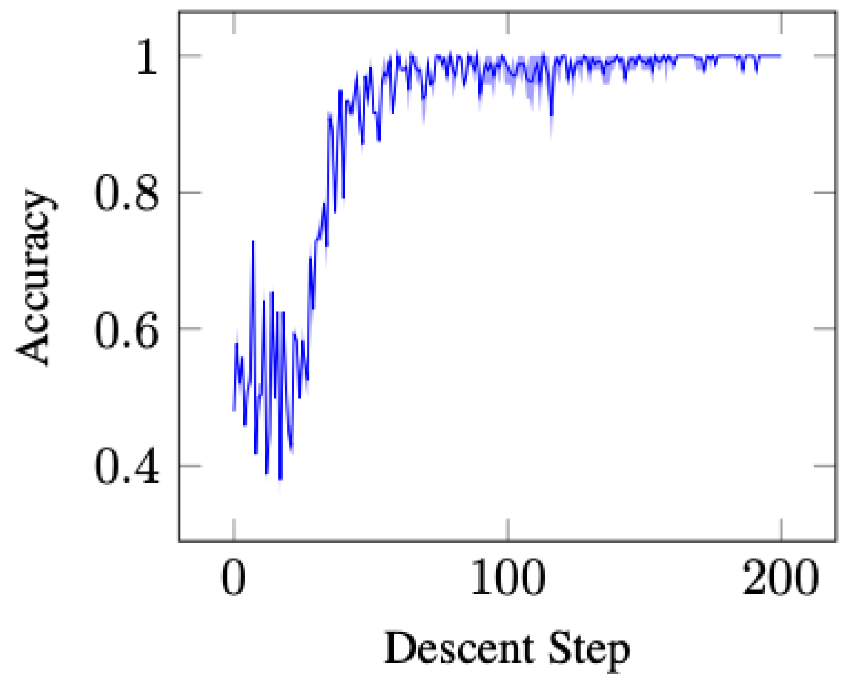
Quantitative Analysis

Formal Verification

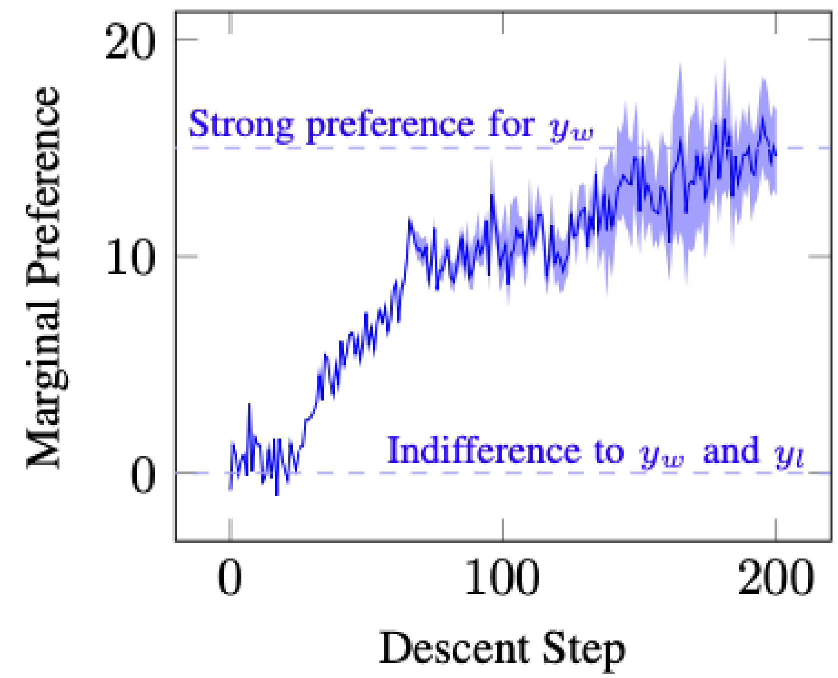
Training Loss



The percentage of the LLM outputs with preferred response rather than the unpreferred response.



How strong the LLM intends to output the preferred response (y_w) rather than the unpreferred response (y_l).

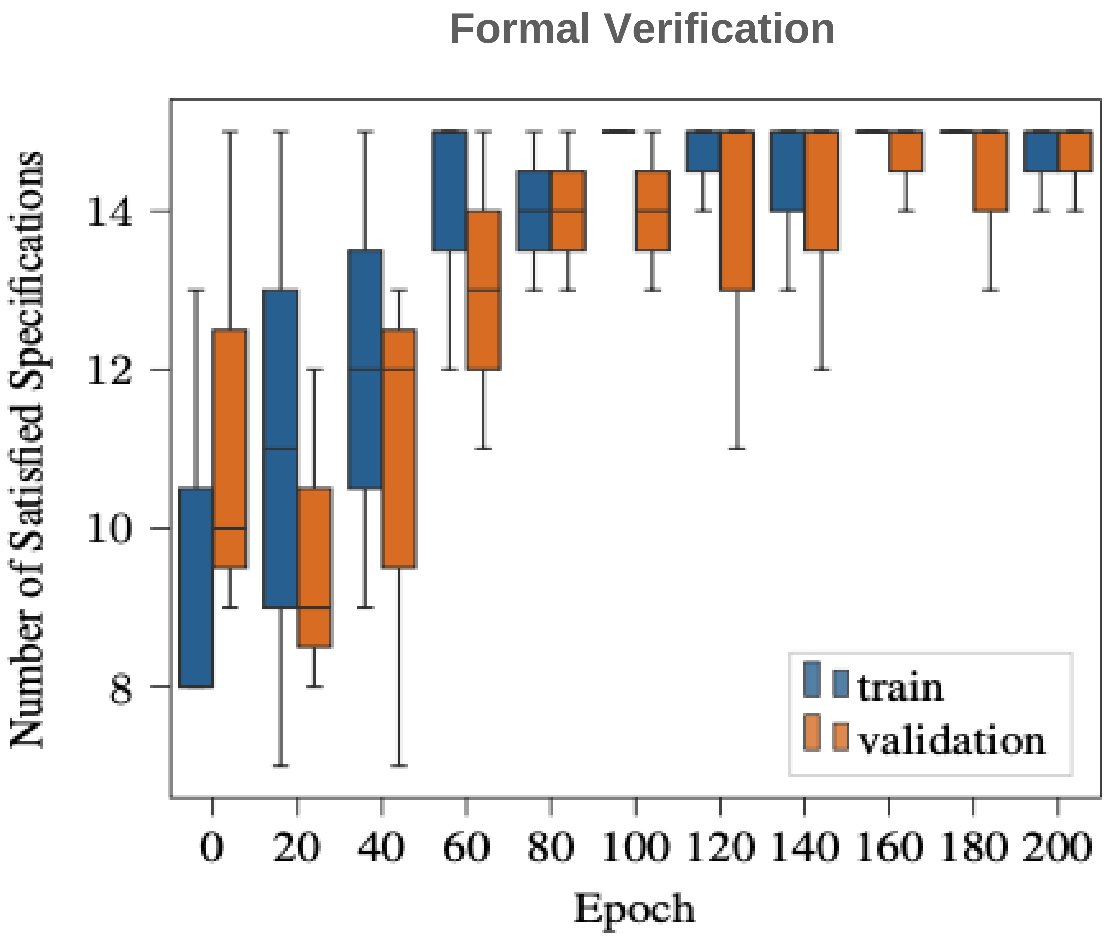


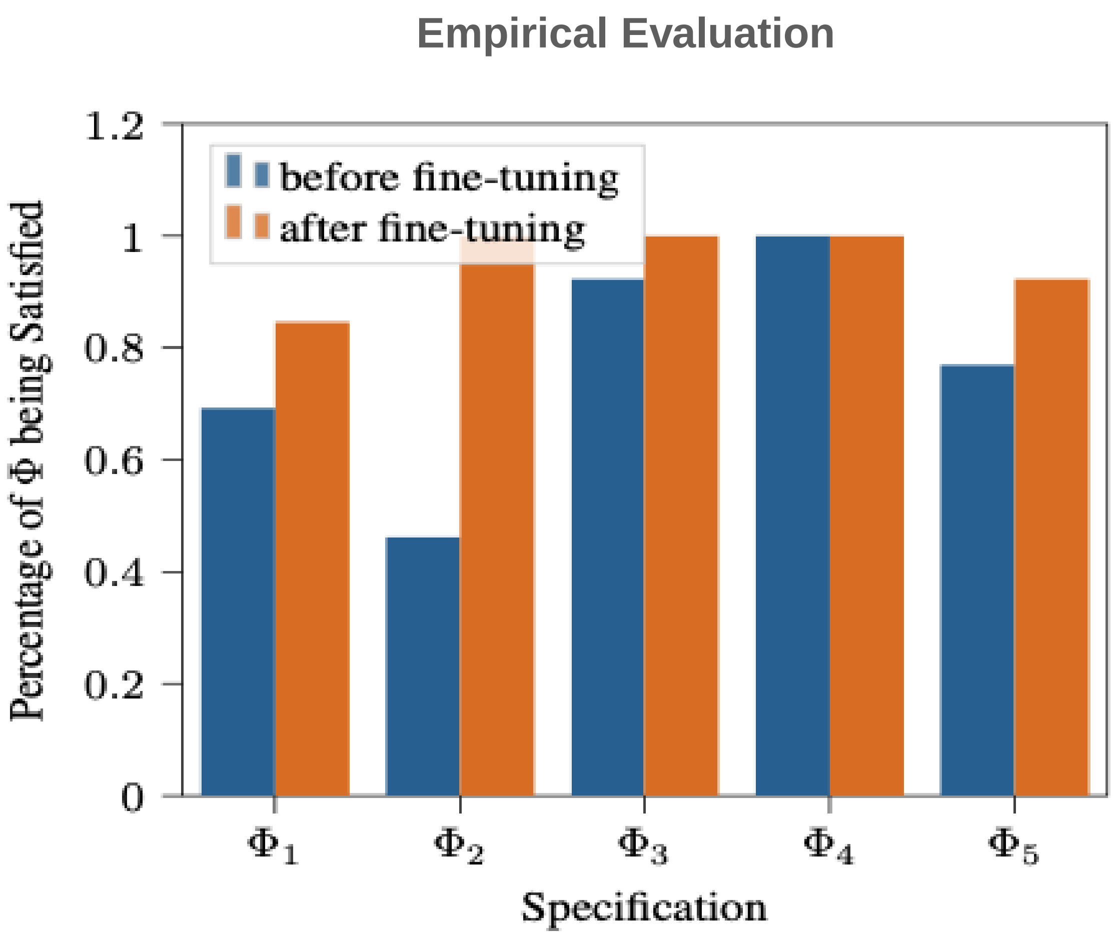
Training Performance

1. **Training loss converges after 100 epochs.**
2. **Nearly 100% preference accuracy.**
3. **The fine-tuned language model strongly prefers the "correct" responses.**

Quantitative Analysis

Empirical Evaluation via Simulation





The results indicate that our approach can improve the language model's ability to satisfy critical requirements.

Our approach can act as a starting point to guide the design process for real-world implementations of autonomous driving systems (60% → 90%).

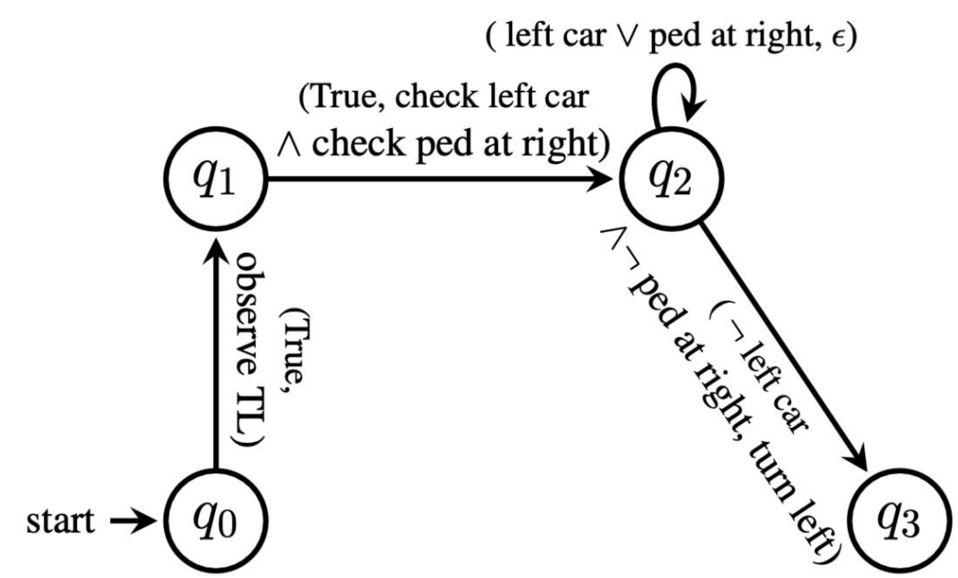
A Step Toward Real-World Grounding

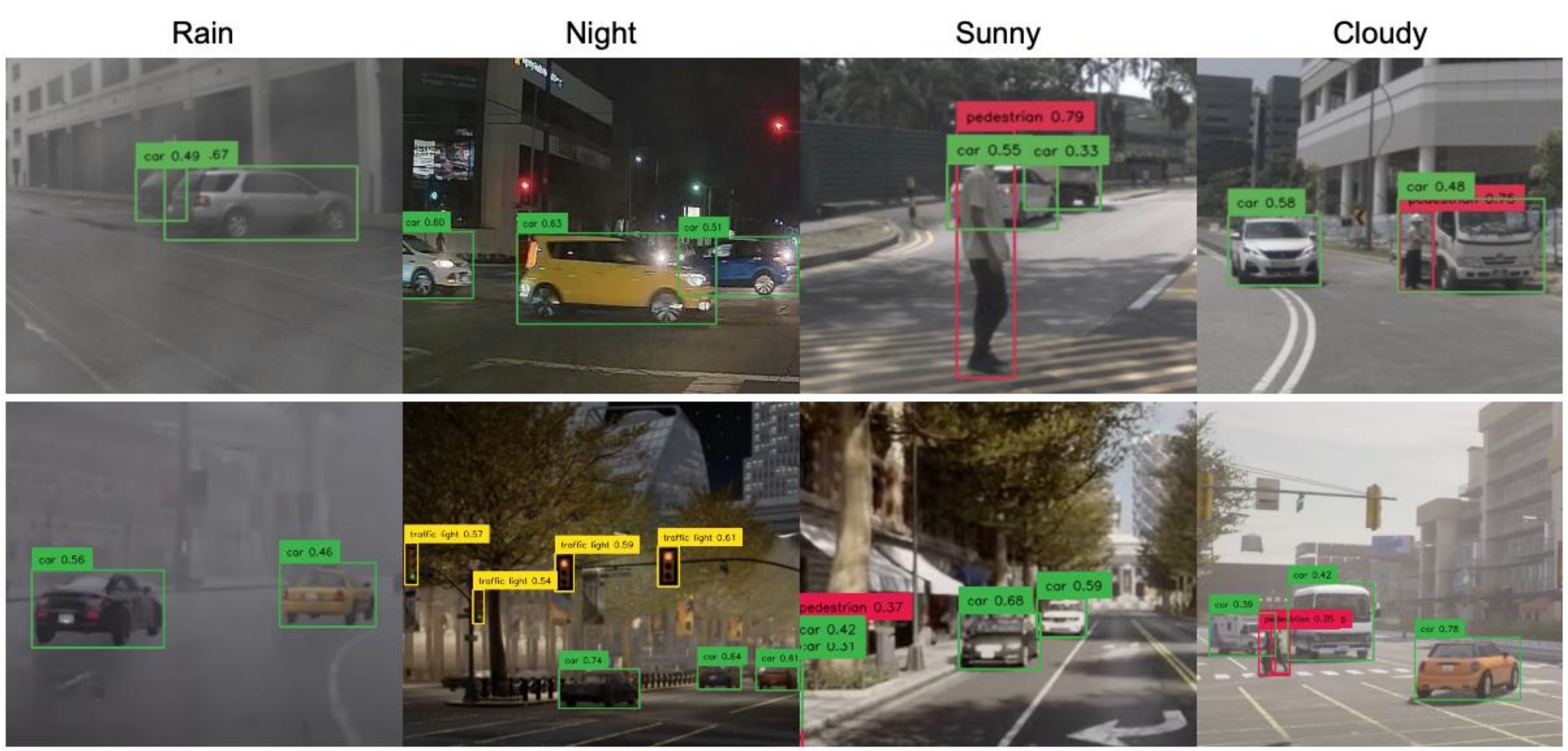
Empirical Evaluation via Simulation → Real-World Performance

Statements

1. The controller's decisions are solely based on visual observations collected from the environment.

2. The vision model performs consistently in simulation and reality.





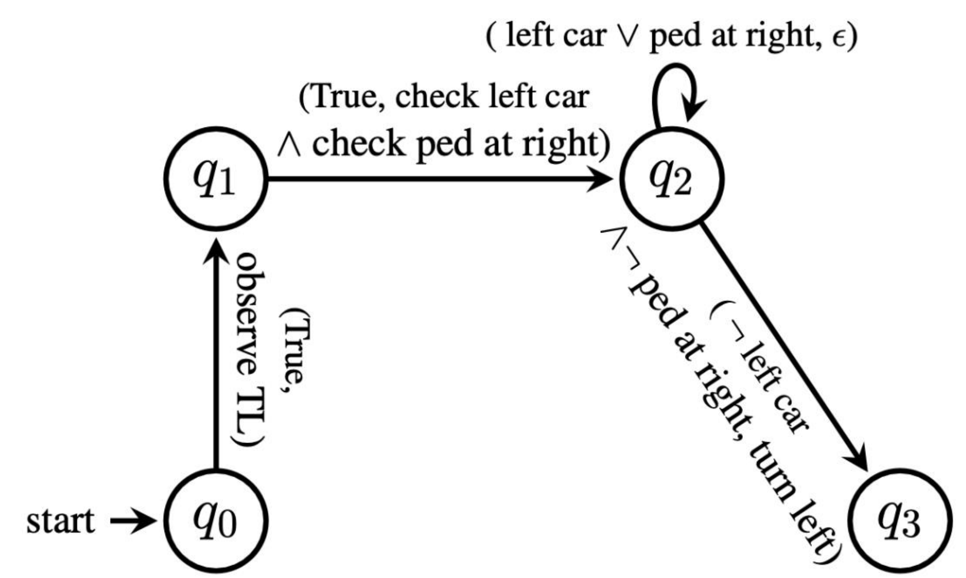
A Step Toward Real-World Grounding

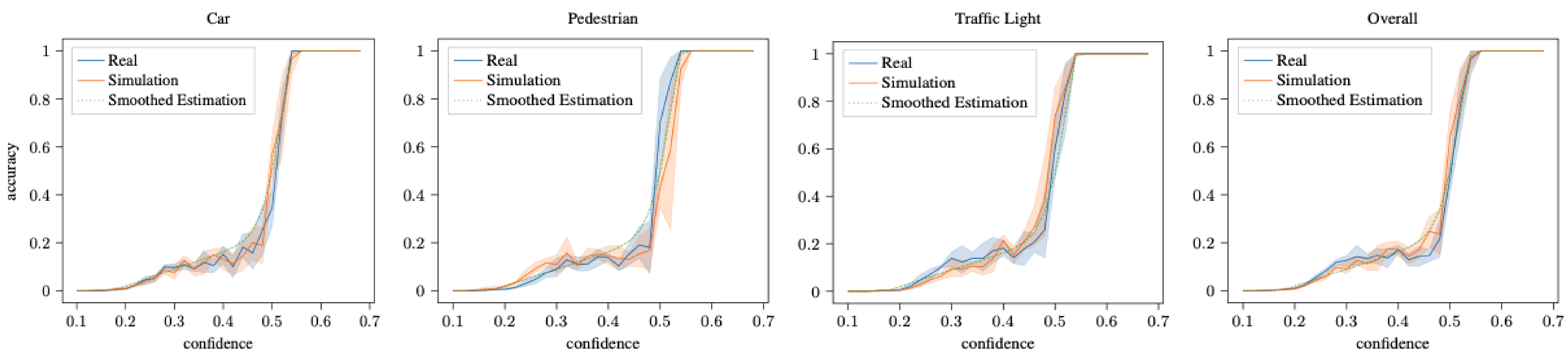
Empirical Evaluation via Simulation → Real-World Performance

Statements

1. The controller's decisions are solely based on visual observations collected from the environment.

2. The vision model performs consistently in simulation and reality.

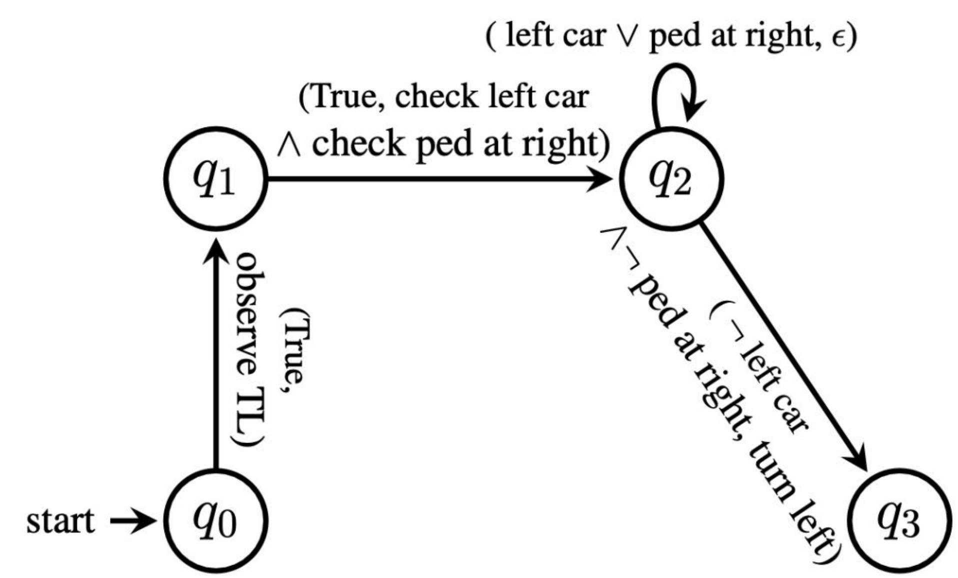




Extension: A Step Toward Real-World Grounding

Empirical Evaluation via Simulation → Real-World Performance

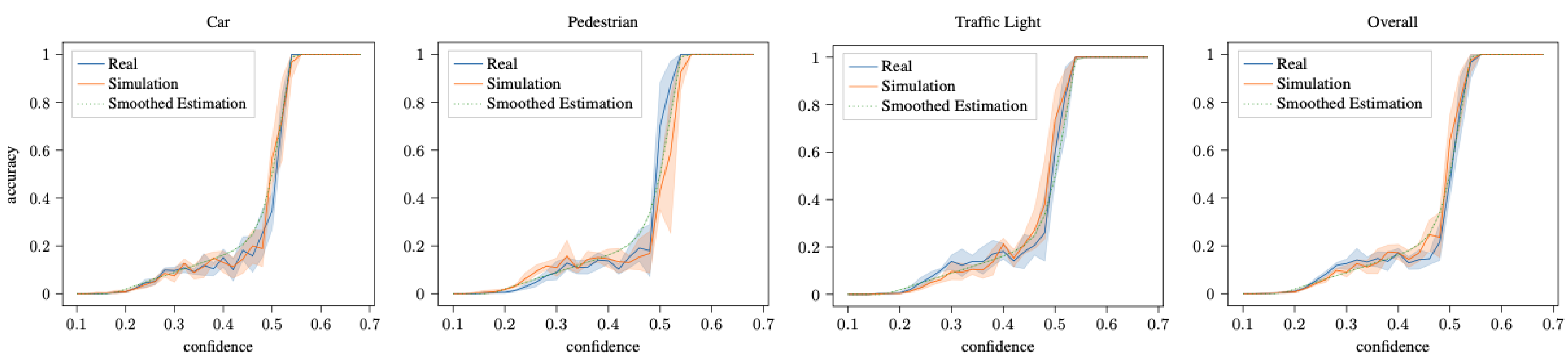
Statements



1. The controller's decisions are solely based on visual observations collected from the environment.

2. The vision model performs consistently in simulation and reality.

3. If Statement 2 holds and if the controllers satisfy the critical specifications in simulation, then the controllers also satisfy the specifications in reality.



Extension: A Step Toward Real-World Grounding

```python
class AutonomousVehicle():
    def __init__(self):
        self.pedestrian = False
        self.car = False

    @abstractmethod
    def move_forward(self):
        # vehicle starts moving forward
        pass

    @abstractmethod
    def turn_left(self):
        # vehicle turns left
        pass

    @abstractmethod
    def turn_right(self):
        # vehicle turns right
        pass

    @abstractmethod
    def stop(self):
        # vehicle slows down and stops
        pass
```

$\otimes$

(left car $\vee$ ped at right, ϵ)

(True, check left car

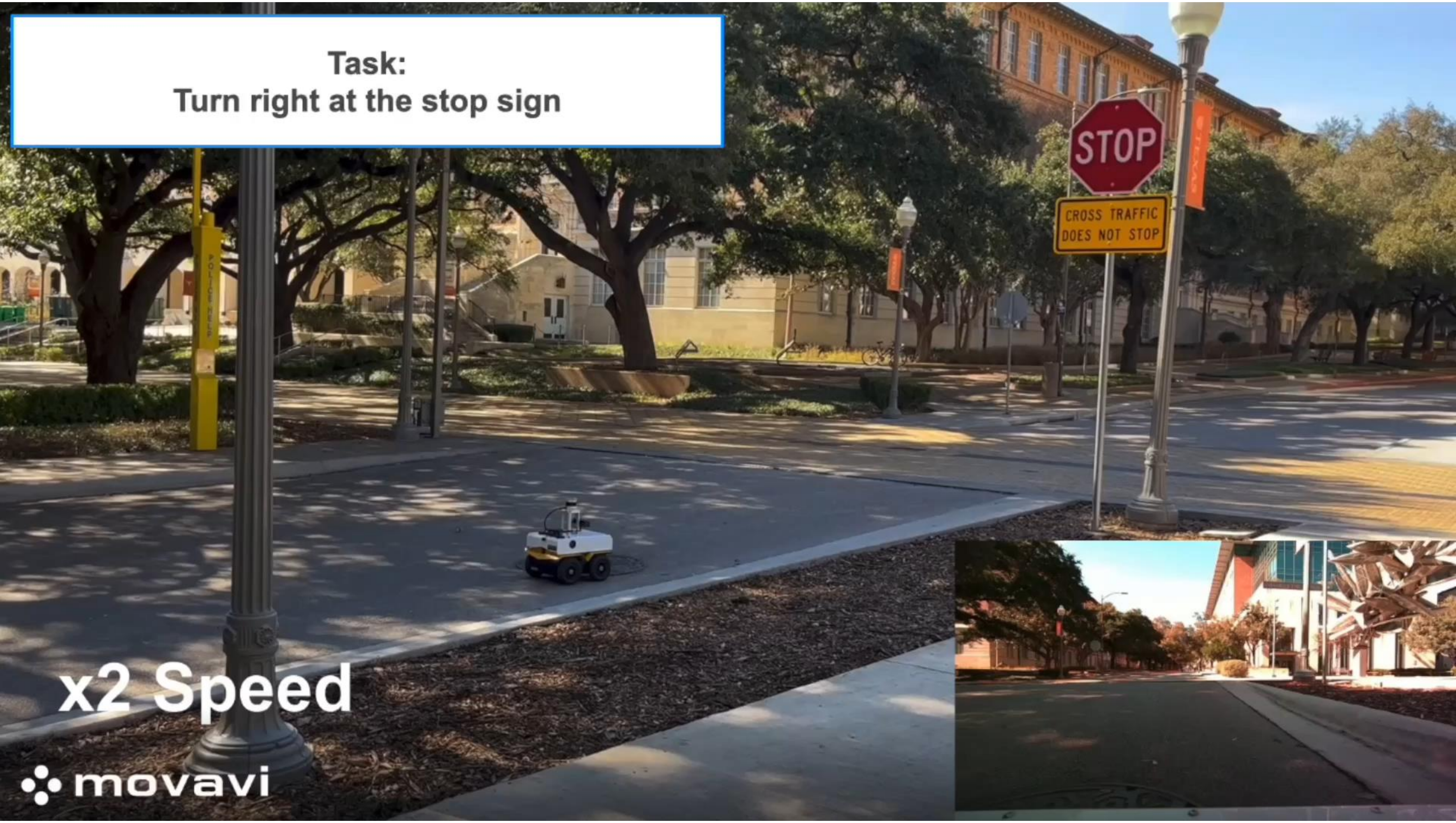
```python
def CrossRoad(vehicle):
    vehicle.observe()  # observe the environment first
    if vehicle.pedestrian_observed():  # Step 1
        vehicle.stop()  # stop if pedestrian is observed
        return "Stop"
    elif vehicle.car_observed():  # Step 2
        vehicle.stop()  # stop if car is observed
        return "Stop"
    else:  # Step 3
        vehicle.move_forward()
        return "Move_forward"
```

$\square(\neg$green traffic light $\rightarrow \neg$go straight), ✓

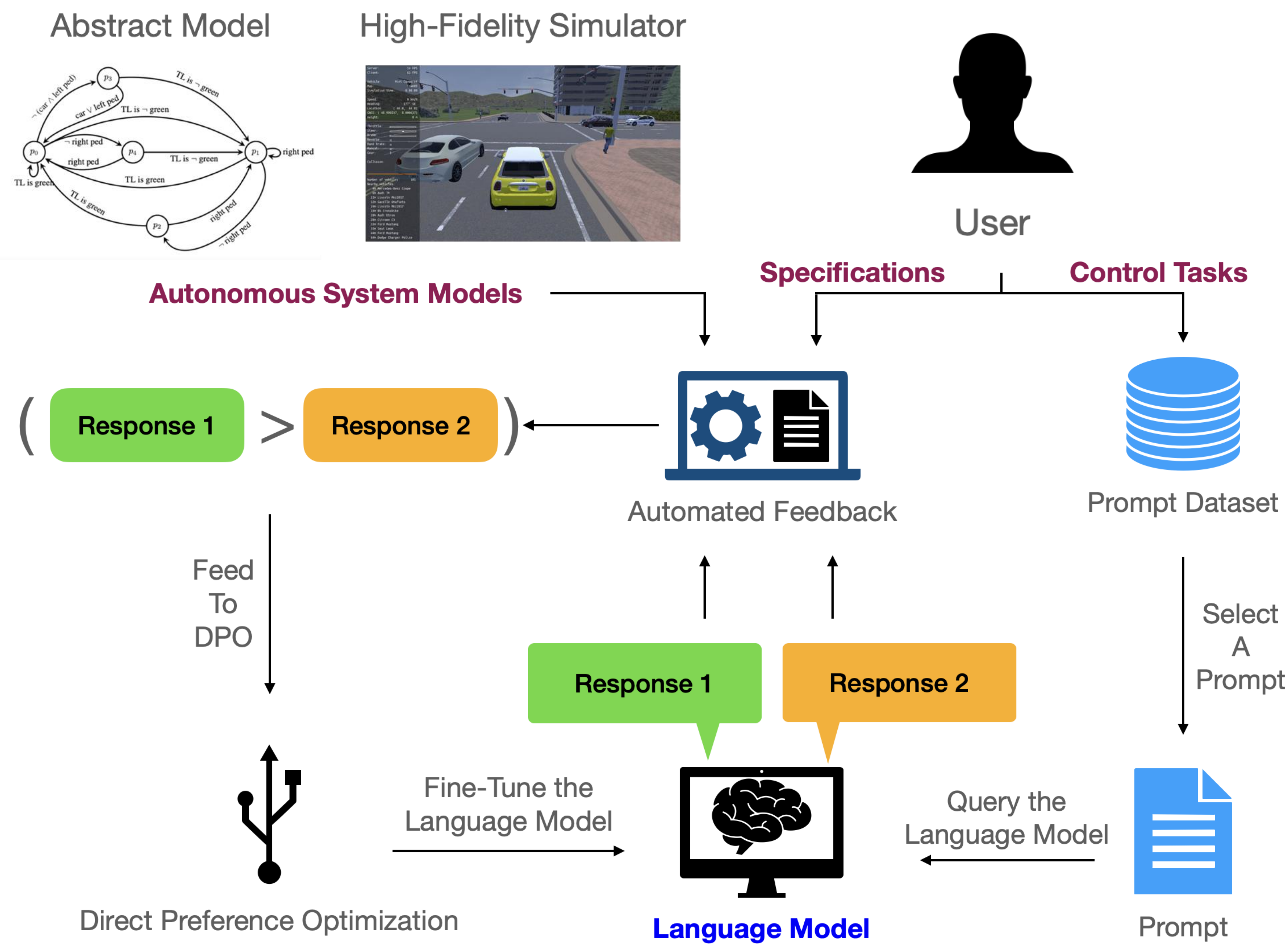
$\square$(stop sign $\rightarrow \Diamond$ stop), ✓

$\square\ \neg$turn right $\vee\ \neg$(car from left $\vee$ pedestrian at right), ✓

Experimental Evaluation



Fine-Tuning Language Models Using Formal Methods Feedback



Thank you!

Paper

Paper available at: https://arxiv.org/abs/2310.18239