A ADDITIONAL BACKGROUND

Confidence Scores A deep learning model produces an output with a confidence score $c \in [0, 1]$. The resulting confidence score represents the likelihood of the output being "correct" with respect to the ground truth labels in the training data. Foundation models such as GPT-series and LLaMA-series produce a distribution of outputs, sample one output from the distribution, and return the sampled output to users. Each output within the distribution is associated with a softmax confidence score indicating its probability of being sampled.

B DEMONSTRATION OF ALGORITHM 1

Given a set of atomic propositions $AP = \{$ green light, red light, car, pedestrian, wait, turn left,... $\}$. The robot agent sends an image as presented in Fig. 9 with objects $Y = \{$ car, truck $\}$ and a task description "turn left" as inputs to the foundation model and obtains the instruction T:

```
    Wait at the red light.
    Wait for opposite cars.
    Turn left at the green light.
```

The algorithm takes T, Y, and AP as inputs and returns a Kripke structure, as presented in Fig. 9. In detail, the algorithm first parses T into phrases

Then, it creates an initial state with a label $Y = \{car\}$. Next, it adds one state per phrase, where the state label is the parsed verbs and nouns in the phrase. Last, it adds a final state and connects all the states sequentially. The resulting Kripke structure is in Fig. 9.



Figure 9: A sample Kripke structure constructed from an image $I \in \mathcal{I}$ with observed objects Y and a textual instruction T using Algorithm 1.

We use a model checker named NuSMV (Cimatti et al., 2002) to verify the Kripke structure against the specifications. The Kripke structure in Fig. 9 and the temporal logic specifications can be expressed as:

```
MODULE Plan_Kripke_Structure
VAR
green_light : boolean;
car : boolean;
pedestrian : boolean;
stop_sign : boolean;
action : {wait, turn_left, turn_right, move_forward};
```

```
10 ASSIGN
    init(car) := TRUE;
    init(pedestrian) := FALSE;
12
    init(green_light) := FALSE;
13
    init(action) := wait;
14
15
    next(action) :=
16
      case
        green_traffic_light & !car : {turn_left};
17
        green_traffic_light & car : wait;
18
19
        !green_traffic_light : wait;
20
      esac;
  LTLSPEC NAME sample_phi_1 :=
22
      G( !green_traffic_light -> ! action=move_forward );
23
24
  LTLSPEC NAME sample_phi_2 :=
      G( pedestrian -> action=wait );
26
```

Next, we can run the following commands in the terminal to obtain the verification results:

```
1 #!NuSMV -source
2 read_model -i plan.smv # replace it with your file name
3 go
4
5 check_ltlspec -P "sample_phi_1" -o result1.txt
6
7 check_ltlspec -P "sample_phi_2" -o result2.txt
8
9 guit
```

Note that these commands will save the verification results into text files with the result in the following format:

-- specification G (!green_traffic_light -> ! action=move_forward) is true

If the plan violates the specification, the result shows a counter-example, which is a sequence of states that violate the specification. For example,

```
-- specification G ((car | pedestrian) -> X action = wait) is false
  -- as demonstrated by the following execution sequence
  Trace Description: LTL Counterexample
  Trace Type: Counterexample
    -> State: 1.1 <-
     pedestrian = FALSE
      car= FALSE
     stop_sign = FALSE
     action = move_forward
9
    -> State: 1.2 <-
10
11
     pedestrian = TRUE
    -> State: 1.3 <-
12
     pedestrian = FALSE
     action = turn_left
14
15
    -- Loop starts here
    -> State: 1.4 <-
16
     action = move_forward
    -> State: 1.5 <-
18
```

C ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS

C.1 Task Description and Sample Response

During plan generation, we provide a set of examples to the foundation model to enforce its output structure. An example is presented below, where the blue texts represent the inputs to the foundation model, and the red texts represent the outputs generated from the foundation model.

```
We: You are a robot driving a vehicle.
2
  User: <Image 1>
3
  User: Turn right at the traffic light.
4
  Assistant:
5
  1. There is no traffic light visible, move forward.
6
8 User: <Image 2>
9 User: Turn right at the traffic light.
10 Assistant:
11 1. Wait for light turn green.
12 2. Turn right.
14 User: <Image 3>
15 User: Go straight at the traffic light.
16 Assistant:
17|1. There is no traffic light visible, but there is a stop sign.
18 2. Stop at the stop sign.
19 3. Move forward.
20
21 User: <Image 4>
22
  User: Turn left at the traffic light.
23
  Assistant:
24 1. Wait for light turn green.
25 2. Wait for car.
26 3. Move forward.
27
28 User: <Image 5>
29 User: Go straight at the stop sign.
30 Assistant:
31 1. Stop at the stop sign.
  2. Wait for pedestrian.
32
33 3. Move forward.
34
35 User: <Image 6>
36 User: Turn left at the stop sign.
37 Assistant:
38 1. There is no stop sign visible, but there is a traffic light.
39 2. Wait for light turn green.
40 3. Turn left.
41
42 User: <Image 7>
43 User: Turn left at the stop sign.
44 Assistant:
45 1. Stop at the stop sign.
46 2. Wait for pedestrian.
47 3. Turn left.
48
49 User: <path to image>
50 User: Turn right at the traffic light.
51 You:
52 1. There is a traffic light ahead.
53 2. Wait for the light to turn green.
54 3. Turn right if there are no pedestrians.
55
  We: does the plan satisfy the following rules:
56
      a. Yield for pedestrian
57
      b. Do not move forward or turn left at the red light
58
59
      Answer 'Y' or 'N' indicating yes or no.
60
61 You:
  Υ
62
```



Figure 10: Q-Q plot for the distributions of the calibration set from the Carla simulator and the images from the real-world driving environment. This plot shows that the vision encoder performs consistently on the calibration set and the real-world images, i.e., two datasets are identically distributed.

Note that the foundation model will return a single letter to indicate its choices.

C.2 Autonomous Driving Task Specifications

To get the nonconformity distribution for decision uncertainty, we manually generate a set Φ of specifications regarding the driving rules. Next, we transform the textual decisions outputted from the foundation model to a Kripke structure through Algorithm 1. Then, we use a model checker to verify the Kripke structure against each specification $\phi \in \Phi$. In the experiment, the set of specifications Φ contains

 $\phi_1 = \Box$ (red light $\rightarrow \neg$ move forward),

$$\phi_2 = \Box$$
(pedestrian \rightarrow wait),

 $\phi_3 = \Box(\neg \text{stop sign} \land \neg \text{traffic light} \rightarrow \text{move forward}),$

 $\phi_4 = \Box(\text{green light} \land \neg \text{pedestrian} \to \bigcirc \neg \text{wait}),$

 $\phi_5 = \Box((\text{stop sign} \land \neg \text{car} \land \neg \text{pedestrian}) \rightarrow \bigcirc \neg \text{wait})$

- $\phi_6 = \Box$ (opposite car $\rightarrow \neg$ turn left),
- $\phi_7 = \Box$ (red light $\rightarrow \neg$ turn left),
- $\phi_8 = \Diamond \neg$ wait,
- $\Phi_9 = \Box$ (wait \lor move forward \lor turn left \lor turn right),

 $\Phi_{10} = \Box(\text{green light} \land \text{opposite car} \rightarrow (\text{wait} \lor \text{move forward} \lor \text{turn right})).$

C.3 Sim2Real Transfer

Our uncertainty estimation framework assumes that the distribution of the calibration set is identically distributed with the test data distribution (*Assumption 1*).

We empirically demonstrate that the data (images) collected from the Carla simulator and the images collected from the real world are identically distributed. Hence *Assumption 1* holds. Therefore, the uncertainty score from the nonconformity distribution bounds the probability of correctly classifying objects in the environment.

We collect the confidence scores of the correct predictions from CLIP for the two sets of images and use a Q-Q plot that shows that the two sets of images are identically distributed. The Q-Q plot (Gnanadesikan & Wilk, 1968) compares the distribution of the confidence scores for the two datasets by plotting their quantiles against each other. The closer the scatter



Figure 11: The left and right figures depict nonconformity distributions for perception and plan generation in table-top manipulation.



Figure 12: Illustration of our strategy in Fig. 1 solving table-top manipulation tasks.

points are to the orange line (y = x), the higher the similarity between the two distributions.

This confirms that the assumption made for our Sim2Real transfer holds. Hence, we can apply the foundation model fine-tuned in the Carla simulator directly to the real-world driving environment.

The outcomes of both the fine-tuning experiments (Fig. 7) and the planning demonstrations (Fig. 5) indicate that a foundation model, when fine-tuned and calibrated on simulation data, exhibit the capacity to be effectively transferred to real-world tasks without the need for domain-specific adjustments.

C.4 Out-of-Domain Application: Table-top Manipulation

Beyond autonomous driving, we demonstrate our planning strategy with uncertainty disentanglement through a set of table-top manipulation tasks.

For table-top manipulation, we obtain nonconformity distributions (presented in Fig. 3). We collect 622 images from three camera views for calibration and supply the images to LLaVA with task descriptions such as "[pick/place] [an object] on the [table]"). We define a set of specifications and present the nonconformity distributions in Fig. 11.

Fig. 12 shows a use case where we successfully complete a task "drop the block into a bowl" while satisfying all the defined specifications. The specifications include

 $\phi_1 = \Box(\neg \operatorname{grab} \operatorname{the cup} \land \neg \operatorname{grab} \operatorname{the bowl}),$

- $\phi_2 = \Box(\neg \text{block} \rightarrow (\neg \text{grab the block} \land \text{locate the block})),$
- $\phi_3 = \Box(\neg \text{mouse} \rightarrow \neg \text{grab} \text{ the mouse} \land \text{locate the mouse}),$

| Threshold $(t_d = t_p)$ | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|-------------------------|-------|-------|-------|-------|-------|
| Accuracy | 0.632 | 0.789 | 0.842 | 0.891 | 0.947 |
| AS Frequency | 0.294 | 0.529 | 1.176 | 2.117 | 3.647 |
| Satisfy Prob | 0.857 | 0.904 | 0.959 | 0.967 | 0.981 |

Table 3: Effect of different thresholds on the fine-tuned model performance.

 $\phi_4 = \Box(\neg \operatorname{cup} \rightarrow \operatorname{locate} \operatorname{the} \operatorname{cup}),$

 $\phi_5 = \Box(\neg \text{bowl} \rightarrow \text{locate the bowl}).$

This use case indicates the generalizability of our planning strategy across multiple domains.

C.5 Effect of User-specified Thresholds for Uncertainty Levels

The thresholds— t_p and t_d —are user-defined theoretical lower bounds on perception and decision errors, indicating the user's error tolerance. We note that different thresholds may lead to different performance. To investigate this further, we use our fine-tuned model to present the effect of different thresholds when deployed in the Carla simulations.

Note that in Tab. 3, threshold refers to the value for both t_p and t_d , e.g., $t_p = t_d = 0.5$. Accuracy refers to the average probability of correctly classifying objects, and AS frequency refers to the average number of active sensing interventions triggered to meet the threshold, i.e., the number of iterations. Lastly, Satisfy Prob refers to the average probability of satisfying the specifications.

There is a trade-off between the perception/decision accuracy and the frequency of triggering active sensing (i.e., computation overhead).