Learning A Risk-Aware Trajectory Planner From Demonstrations Using Logic Monitor Supplementary Material

A. Past Time Signal Temporal Logic (ptSTL)

Let $x^{t_0:t_1} = \{x_{t_0}, ..., x_{t_1}\}$ denote the state trajectory from time t_0 to t_1. $x^{t_0:t_1} \models \phi$ denotes that trajectory $x^{t_0:t_1}$ satisfies ϕ. The Boolean semantics of ptSTL is defined recursively in Equation (A1) as

$$\begin{aligned}
x^{t_0:t_1} &\models (p(x) < \epsilon) && \Leftrightarrow p\left(x_{t_0}\right) < \epsilon \\
x^{t_0:t_1} &\models \neg(p(x) < \epsilon) && \Leftrightarrow \neg(x^{t_0:t_1} \models (p(x) < \epsilon)) \\
x^{t_0:t_1} &\models \phi_1 \wedge \phi_2 && \Leftrightarrow x^{t_0:t_1} \models \phi_1 \wedge x^{t_0:t_1} \models \phi_2 \\
x^{t_0:t_1} &\models \phi_1 \vee \phi_2 && \Leftrightarrow x^{t_0:t_1} \models \phi_1 \vee x^{t_0:t_1} \models \phi_2 \\
x^{t_0:t_1} &\models G^{-}_{[a,b]}\phi && \Leftrightarrow x_{t',t_1} \models \phi \ \forall t' \in (-b, a] \\
x^{t_0:t_1} &\models F^{-}_{[a,b]}\phi && \Leftrightarrow \exists t' \in (-b, a] \text{ s.t. } x^{t':t_1} \models \phi
\end{aligned} \tag{A1}$$

ptSTL also admits quantitative semantics in the form of a robustness degree $r(x^{t_0,t_1}, \phi)$ given in Equation (A2) that quantifies satisfaction and violation of specification ϕ by signal x^{t_0,t_1}.

$$\begin{aligned}
r(x^{t_0:t_1}, (p(x) < \epsilon)) &= \epsilon - p\left(x_{t_0}\right) \\
r(x^{t_0:t_1}, \neg\phi) &= -r(x^{t_0:t_1}, \phi) \\
r\left(x^{t_0:t_1}, \phi_1 \wedge \phi_2\right) &= \min(r(x^{t_0:t_1}, \phi_1), r(x^{t_0:t_1}, \phi_2)) \\
r\left(x^{t_0:t_1}, \phi_1 \vee \rho_2\right) &= \max(r(x^{t_0:t_1}, \phi_1), r(x^{t_0:t_1}, \phi_2)) \\
r\left(x^{t_0:t_1}, G^{-}_{[a,b]}\phi\right) &= \inf_{t' \in (-b,a]} r\left(x^{t':t_1}, \phi\right) \\
r\left(x^{t_0:t_1}, F^{-}_{[a,b]}\phi\right) &= \sup_{t' \in (-b,a]} r\left(x^{t':t_1}, \phi\right)
\end{aligned} \tag{A2}$$

The robustness degree is sound, meaning that positive values of r imply satisfaction, while negative values imply violation.

A ptSTL formula ϕ is said to be time-bounded if there exists a finite duration d, the formula's horizon, such that the all signals of duration at least d can be checked for satisfaction or violation of the formula. The horizon of the STL formula can be computed recursively based on the time bounds of the temporal operators [1].

In the definition of semantics, we assume that the lengths of the time intervals $[t_0, t_1]$ associated with a signals are always greater than the horizon of the considered ptSTL formulas. Thus, satisfaction and robustness degrees can be computed.

Example 1. *Assume a robot is navigating in the 2D plane with its coordinates as states i.e.* $s = (x, y)$. *Let* A *and* B *be two disjoint circular regions with* c_A, r_A *and* c_B, r_B *be the center and radius of* A *and* B *respectively. Define predicates* $\psi_A = ||s - c_A||_2 < r_A$ *and* $\psi_B = ||s - c_B||_2 < r_B$

where $||\cdot||_2$ is the L2-norm. ψ_A and ψ_B tracks whether the robot has entered A and B. We can define a required past time property using ptSTL as

$$\phi = F^-_{[0,H]}\psi_A \wedge G^-_{[0,H]}\neg\psi_B, \tag{A3}$$

where H is length of past signal required to evaluate the formula (also referred to as the horizon). ϕ describes that within the previous H time-steps the robot needs to "eventually visit region A and always avoid region B for all times from $-T$ to 0". Time 0 is the current time-step. Given a history of the robot $x^{-T:0}$, we can evaluate the robustness of the trajectory with respective to ϕ by applying Equation (A2) recursive which gives us

$$r(x^{-T:0}, \phi) = \min\big[\max_{t\in(-T,0]}(r_A - ||x_t - c_A||_2), \min_{t\in(-T,0]}(||s_t - c_B||_2 - r_B)\big]. \tag{A4}$$

$r(x^{-T:0}, \phi) > 0$ implies that the robot has satisfied ϕ.

ptSTL is very similar to STL [2]. The difference been the way time bounds are handled. STL is forward-looking and requires future signals whereas ptSTL is backward-looking and requires past signals. In the case where signal length is smaller than the required horizon of the formula, then the full signal will be used to for evaluation. Otherwise, the most recent H steps of the signal will be used.

B. Predicate Definitions

Recall that the rules we used in the experiments are

$$\phi_1 = G^-_{[0,H)}\left(\text{driveNearLane} \wedge \text{keepSafeCarDistFromEgo}\right) \tag{A5}$$

$$\phi_2 = G^-_{[0,H)}\left(\text{egoInIntersection} \rightarrow (\text{farFromIntersection} \vee \text{driveSlowly})\right) \tag{A6}$$

where H is the tracking horizon (we use $H = 4$). ϕ_1 describes that a safe ado vehicle should "*always* drive near the center lane *and* keep a safe distance from the ego vehicle." ϕ_2 expresses that at intersections "*always* if the ego vehicle is in the intersection *implies* that a safe ado vehicle should either be far from the intersection *or* be driving slowly." The final rule $\phi = \phi_1 \wedge \phi_2$ takes the form of a conjunction of a set of sub-rules. The predicates are defined as

- "driveNearLane": denoted as $dist(\boldsymbol{p}_a, \boldsymbol{p}_{lane}) < \epsilon_1$ as a thresholded distance between the ado vehicle's position and a nominal point on the center lane;
- "farFromIntersection": denoted as $dist(\boldsymbol{p}_a, \boldsymbol{p}_{\text{int. center}}) > \epsilon_2$ is a thresholded distance between the ado vehicle's position and position of the intersection center;
- "egoInIntersection": denoted as $dist(\boldsymbol{p}_e, \boldsymbol{p}_{\text{int. center}}) < \epsilon_3$ is a thresholded distance between the ego vehicle's position and the position of the intersection center;
- "driveSlowly": denoted as $v_a < \epsilon_4$ as the thresholded ado vehicle's speed;
- "keepSafeCarDistFromEgo": denoted as $dist(\boldsymbol{p}_a, \boldsymbol{p}_e) < \epsilon_5$ as a thresholded distance between the ado vehicle's position and the ego vehicle's position.

In the above definitions, $\epsilon_i,\ i \in \{1, 2, 3, 4, 5\}$ are learnable parameters.

C. Implementation Details

In our experiments, the behavior cloning agent (*BC* agent) is implemented similarly to [3]. We use a convolutional neural network (CNN) that takes as input a rasterized image (similar to [4]) and a feature vector is extracted using 5 CNN layers then 3 fully connected layers. The network outputs the speed and steering commands which is then passed through a unicycle model to obtain the future position. The hyperparameters of the network are listed in Table A1. In the table, FC denotes fully

Table A1: Behavior cloning agent hyperparameters.

Layer	Type	Hyperparameters
1	Conv2d	F: 24, K:5, S:2
2	Conv2d	F: 36, K:5, S:2
3	Conv2d	F: 48, K:3, S:2
4	Conv2d	F: 64, K:3, S:1
5	Conv2d	F: 64, K:3, S:1
6	FC	U: 300, ReLU
7	FC	U: 150, ReLU
8	FC	U: 2, ReLU

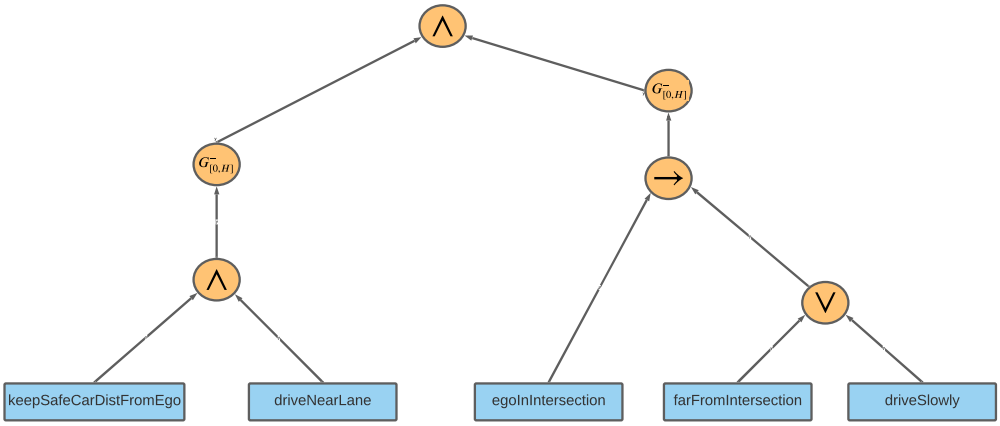

Figure A1: Syntax tree for $\phi = \phi_1 \wedge \phi_2$.

connected layer, F denotes number of filters, K denotes kernel size, S denotes stride, U denotes number of units in the fully connected layer.

The *TrajOpt* agent is different from the *LogicRiskNet-IOC* agent only in the risk feature f^{ϕ}. Instead of having a scaled collision feature as with the *LogicRiskNet-IOC* agent, the *TrajOpt* agent is simply implemented with a collision feature.

$$f(\boldsymbol{\omega}_T) = \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \frac{1}{||\boldsymbol{p}_e^t - \boldsymbol{p}_{a_i}^t||^2} \tag{A7}$$

TrajOpt agent allows us to study and contrast the influence LogicRiskNet.

D. RiskLogicNet Structure And Monitoring Traces

The formula $\phi = \phi_1 \wedge \phi_2$ where ϕ_1 and ϕ_2 are defined in Equations (A9) and (A6) can be translated into a syntax tree where the leaves are the predicates and all intermediate nodes are Boolean or temporal operators. Figure A1 shows the syntax tree for ϕ. This syntax tree is the basis for LogicRiskNet as it can transform a ptSTL formula into a computation graph for calculating risk values (refer to [5] for treatment in calculating robustness values). During operation, the leaf nodes (blue) calculate the risk-based predicate function values α^{ρ} and this value is propagated through the computation graph. The root node provides the final risk value for the entire formula.

By outputting intermediate risk values we are able to obtain risk values for sub-formulas which can provide insights into why an agent is risky or safe. For the agent highlighted by the yellow dashed circle in Figure A4, the monitor traces are illustrated by Figure A3 where each plot is the output of the corresponding node in Figure A2. We can from the root plot of Figure A3 the risk of the monitored ado agent is increasing (becoming increasingly negative). Tracing down the tree

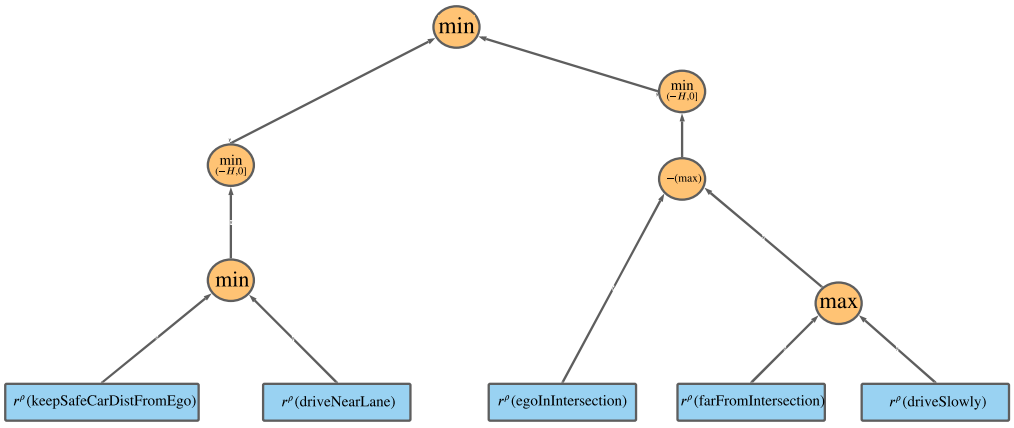

Figure A2: LogicRiskNet computation graph for $\phi = \phi_1 \wedge \phi_2$.

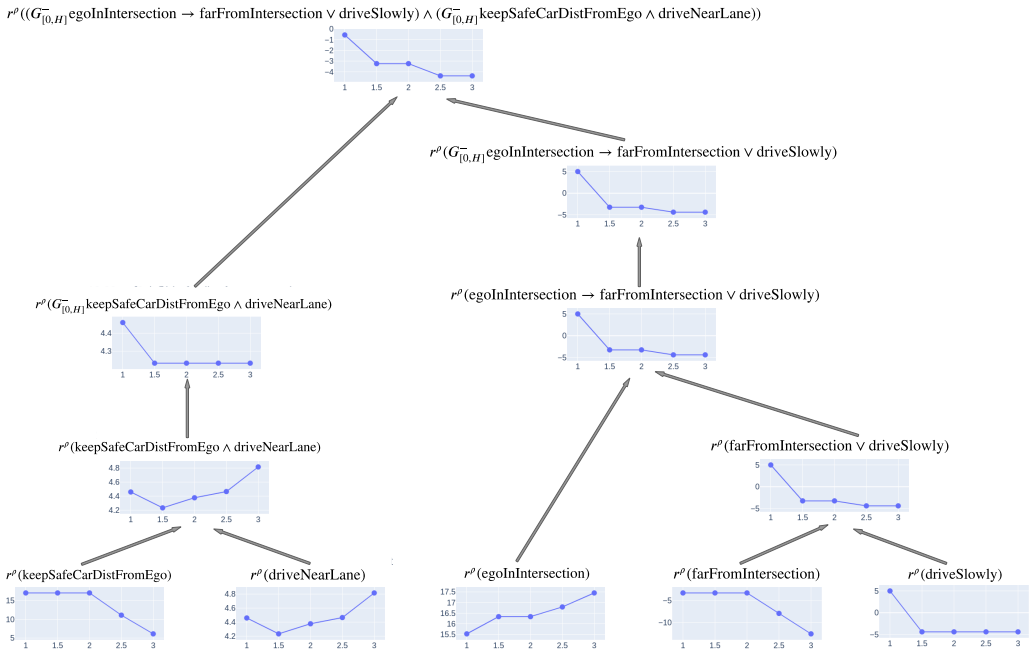

Figure A3: LogicRiskNet monitor traces for $\phi = \phi_1 \wedge \phi_2$. For all plots, the x-axis is time-step and y-axis is risk value.

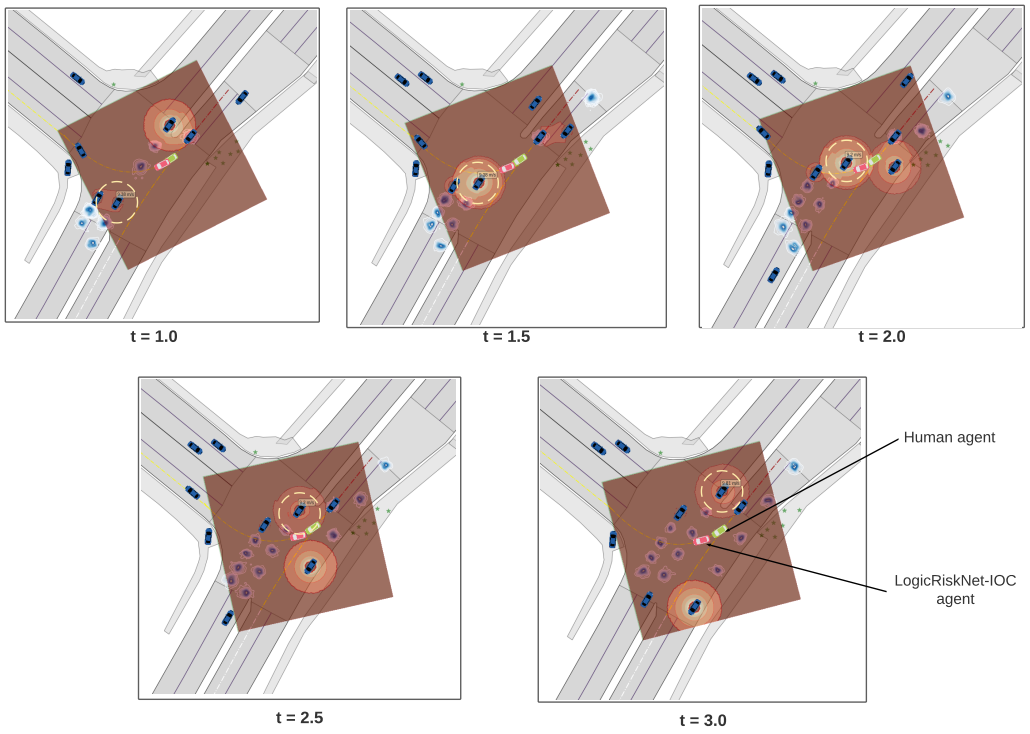

Figure A4: Scene execution trace. The ado vehicle highlighted by the yellow dashed circle produces the monitor traces in Figure A3.

we can find that the main cause is that the monitored ado agent is getting closer to the intersection center (deceasing $r^{\rho}(\text{farFromIntersection})$) and that it is speeding up (decreasing $r^{\rho}(\text{driveSlowly})$). In Figure A4 we can see that the human agent is stopped to yield for the ado agent. Our agent (LogicRiskNet-IOC agent) is a little more aggressive in that it tried to make the turn but eventually also stops to yield. Given a ptSTL formula, the computation graph in Figure A2 can be automatically generated and the monitor traces for all ado agents can be efficiently computed.

E. Additional Discussions

The proposed approach can be applied to domains other than autonomous driving, we expect that it can apply equally well to other robotic and cyber-physical domains calling for interactions with humans. One modification to our approach is that different features will be required to form the IOC cost function in Equation (5), which, in our examples, are tailored to the autonomous driving domain. The generality of our approach is that, as long as the domain is (partly) governed by a set of rules and a set of demonstration data with naturalistic human behavior can be obtained, the LogicRiskNet and IOC weights can be trained to output trajectory plans that resemble the demonstrators' risk management behaviors.

In our experiments, we use the past 2 seconds of observations to calculate the risk factor and use it to generate a trajectory plan 3 seconds into the future. During this process, we assume the risk of the agent remains constant for the next 3 seconds. During deployment, we execute this plan for only 1 timestep and then re-plan (in a receding horizon fashion), therefore, the risk of the agent is in fact re-evaluated at each timestep. It is helpful to investigate if historical data provides a good indicator of future risks. Therefore, for each agent i in the dataset, we calculate the risk values along its trajectory which gives us its risk trajectory $R^i_{0:T}$ (T is the trajectory length of agent i). at every timestep t along the risk trajectory, we take 3 seconds of its future $R^i_{t+1:t+4}$ and calculate the max

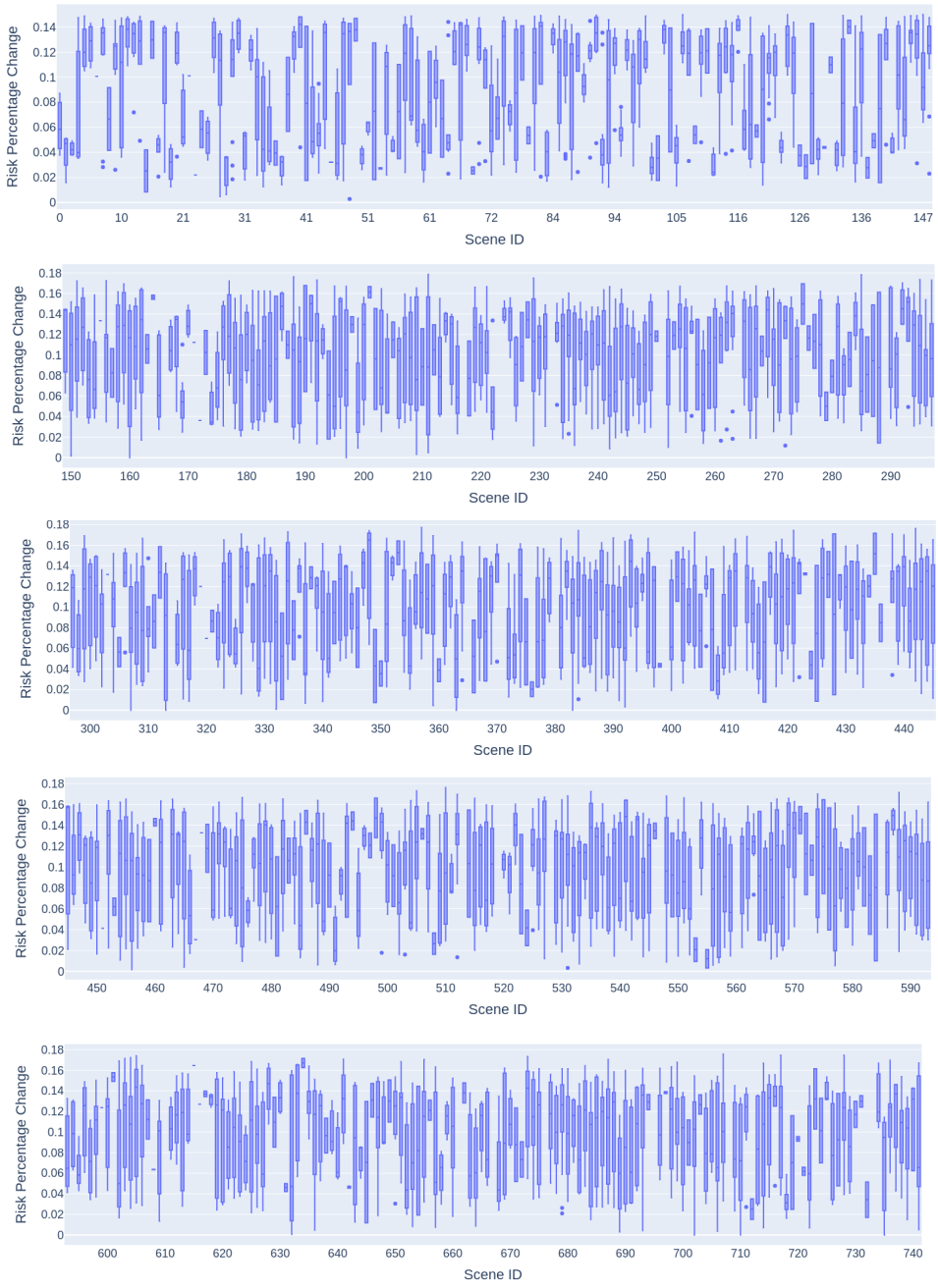

Figure A5: Study of future risk change

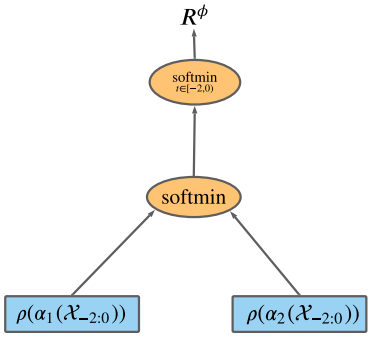

Figure A6: An example LogicRiskNet

percentage change with respect to the current risk value given by the following equation

$$\text{percent risk change} = \max_{t' \in [t+1, t+4]} \frac{|R^i_{t'} - R^i_t|}{|R^i_t|} \tag{A8}$$

We calculate this percent change for all times on the risk trajectory and for all agents and report the distribution aggregated by scenes. The results in Figure A5 show that max future risk change is around 18 percent for all vehicles in the dataset. Given the fact that we re-evaluate risks and re-plan at every step, the assumption that past behavior is indicative of short term future risk should not affect the overall performance.

F. Example Construction of The LogicRiskNet

In this section, we will provide a concrete example of how the LogicRiskNet is constructed from an STL formula. This is to supplement the explanations in Section 4 of the paper.

For clarity, we will use the simple rule with a horizon of 2 seconds,

$$\phi = G^-_{[0,2)} \left(\text{driveNearLane} \wedge \text{keepSafeCarDistFromEgo}\right) \tag{A9}$$

where the predicates are defined in Section B. Suppose we have a tracking history of 2 timesteps - $\boldsymbol{h} = \{\boldsymbol{p}_{-2}, \Sigma_{-2}, \boldsymbol{p}_{-1}, \Sigma_{-1}\}$ where $\boldsymbol{p}_t = (x_t, y_t)$ is the mean position coordinates and Σ_t is the covariance at time t (negative t indicates past timesteps). Suppose also that we choose the expectation risk measure i.e. $\rho(\boldsymbol{p}, \Sigma) = \boldsymbol{p}$. Then, at each timestep, we can calculate the risk-based predicate value $\rho(\alpha(\mathcal{X}_t))$ ($\mathcal{X}_t = (\boldsymbol{p}_t, \Sigma_t)$ is the Gaussian represented random variable at t) for each of the predicates driveNearLane and keepSafeCarDistFromEgo.

Following the definition of robustness in Equation (3), the robustness risk of ϕ is

$$R^\phi = \min_{t \in [-2,0)} \left(\min[\rho(\alpha_1(\mathcal{X}_t)), \rho(\alpha_2(\mathcal{X}_t))] \right) \tag{A10}$$

where α_1 is the predicate robustness for driveNearLane and α_2 is that for keepSafeCarDistFromEgo. Specifically

$$\alpha_1(\mathcal{X}_t) = \epsilon_1 - dist(\mathcal{X}_t, \boldsymbol{p}_{lane}), \quad \alpha_2(\mathcal{X}_t) = \epsilon_5 - dist(\mathcal{X}_t, \boldsymbol{p}_e) \tag{A11}$$

The computation graph that represents Equation (A10) is shown in Figure A6. Here, we use the softmin function instead of the min function such that we can differentiate through this computation graph and adjust the predicate parameters. We refer to this computation graph as the LogicRiskNet.

Table A2: Performance comparison results with simple reactive agents.

	Time to goal				Safety distance				Goal%
	min	**mean**	**max**	**90th**	**min**	**mean**	**max**	**90th**	
BC	13.20	19.67	20.0	20.0	0.56	3.67	9.56	8.32	32%
TrajOpt	17.12	18.72	20.0	19.53	3.60	6.35	12.01	8.03	85%
RiskLogicNet-IOC	15.78	17.69	20.0	19.20	2.64	4.02	8.93	5.98	92%
Human	19.0	19.75	20.0	20.0	3.03	4.62	9.45	6.84	100%

G. Additional Results With Reactive Agents

In this section, we provide additional results where reactive agents are inserted into the environment during evaluation. For each evaluation scene, we insert a random number (from 1 - 3) of controlled ado agents at the beginning of the scene (positions randomly distributed on driveways near the ego agent) and that lives throughout the scene. The reactive agents exhibit lane following behavior and their speeds are controlled by the intelligent driver's model (IDM). This results in the simple behavior that if there are no vehicles in the vicinity of the controlled ado agents, they will follow the target lane at a constant speed, otherwise, they will speed up or slow down to avoid collision. Our implementation of IDM references that of [6]. All the agents in the dataset still remain. The evaluation results for the four comparison cases are presented in Table A2.

In Table A2, *Human* results are the same as that in Table 1, as the ego agent in the dataset is not aware of our inserted agents. The overall effect of inserting controlled ado agents is that the evaluation environment is more crowded than the original. Hence safety distance and goal reaching percentage decreased whereas the time to goal increased. However, this decrease in performance is minimal for our method due to the fact that IDM agents are good at giving way to others and hence are in general safe agents. The IDM agents have a more significant influence on the success rate of the *BC* agent. This is most likely due to the fact that *BC* agent is prone to distribution shift and inserting a new set of agents with behaviors that differ from those in the training set deteriorates this problem.

H. Additional Related Works on Safe and Risk-aware Formal Synthesis

Safe and risk-aware formal synthesis is an area that focuses on incorporating uncertainty and risks in planning with TL constraints. We have taken much inspiration from [7, 8] in their techniques of incorporating belief states into STL risk definitions. The resulting controls are obtained by solving a mixed-integer linear program with the STL risk as constraints. This can be difficult to scale with the complexity of the STL risk formulas. In [9, 10], the authors consider the uncertainties in the agent trajectories via probabilistic STL. Methods of computing specification-satisfying controllers under probabilistic STL constraints are also introduced. In these works, the probability of events happening at each timestep is required, which is difficult to obtain and does not reflect how humans reason about risks in the world. Our approach, in contrast, is data-driven and our logic risk metric is compatible with existing model-based/model-free policy learning paradigms.

References

[1] E. A. Gol. Efficient online monitoring and formula synthesis with past stl. 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT), pages 916–921, 2018.

[2] A. Donzé, T. Ferrere, and O. Maler. Efficient robust monitoring for STL. In Computer Aided Verification, pages 264–279. Springer, 2013.

[3] W. Farag and Z. Saleh. Behavior cloning for autonomous driving using convolutional neural networks. 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), pages 1–7, 2018.

[4] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional

networks. 2019 International Conference on Robotics and Automation (ICRA), pages 2090–2096, 2019.

[5] K. Leung, N. Aréchiga, and M. Pavone. Back-propagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. In Workshop on Algorithmic Foundations of Robotics, 2020.

[6] E. Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

[7] S. Safaoui, L. Lindemann, D. Dimarogonas, I. Shames, and T. Summers. Control design for risk-based signal temporal logic specifications. IEEE Control Systems Letters, 4:1000–1005, 2020.

[8] L. Lindemann, N. Matni, and G. J. Pappas. Stl robustness risk over discrete-time stochastic processes. ArXiv, abs/2104.01503, 2021.

[9] D. Sadigh and A. Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In Robotics: Science and Systems, 2016.

[10] C. Yoo and C. Belta. Control with probabilistic signal temporal logic. ArXiv, abs/1510.08474, 2015.