Interpretable Policies from Formally-Specified Temporal Properties

Jonathan DeCastro[1], Karen Leung[2], Nikos Aréchiga[3], Marco Pavone[2]

***Abstract*— We present an approach for interpreting parameterized policies based on a formally-specified abstract description of the importance of certain behaviors or observed outcomes of a policy. The standard way to deploy data-driven policies usually involves sampling from the set of outcomes produced by the policy. Our approach leverages parametric signal temporal logic (pSTL) formulas to construct an interpretable view on the modeling parameters via a sequence of variational inference problems; one to solve for the pSTL parameters and another to construct a new parameterization satisfying the specification. We perform clustering using a finite set of examples, either real or simulated, and combine computational graph learning and normalizing flows to form a relationship between these parameters and pSTL formulas either derived by hand or inferred from data. We illustrate the utility of our approach to model selection for validation of the safety properties of an autonomous driving system, using a learned generative model of the surrounding agents.**

I. Introduction

Parameterized policies, widely prevalent in robot decision-making, motion prediction, and scenario-generation, are often black-box, having been learned from data. However, for a general data-driven policy, it is unclear how the choice of parameters will affect the outcome of the policy when it is deployed as an agent in an environment. A user of a policy, for instance, may wish to produce a style of behavior or evoke a certain outcome upon interacting with the environment. More specifically for the context of autonomous driving, the user may want to generate a suite of challenging driving scenarios in a simulation environment in order to stress test their newly developed autonomous driving policy and evaluate its performance. Thus, a challenge is in quantifying a relationship between parameters of data-driven policies and the emergent behaviors from deploying that policy.

For safety-critical systems, such as for autonomous cars, it is important to quantify the relationship between the parameters of a policy and the resulting behaviors of using that policy because it can (i) help with the verification of data-driven policies [1], [2] which is currently a bottleneck in the wide-spread adoption of learning-based components in safety-critical systems, (ii) provide interpretability and thus transparency which can potentially improve performance of downstream applications such as decision-making and control, and (iii) provide a quantitative handle on the types of behaviors that can emerge and use it to "tune" desired behaviors which are useful for scenario generation when validating an autonomous agent [3], [4].

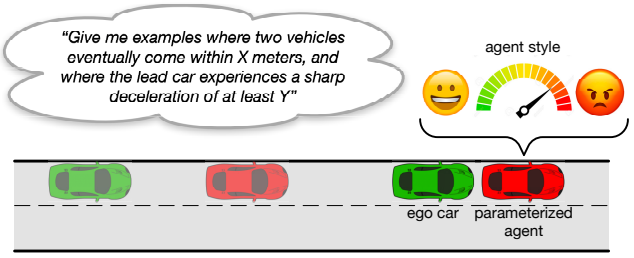

Fig. 1: Tailoring the learned driver behavior model according to outcomes as experienced by a vehicle behind it.

Consider, for example, a driving scenario shown in Fig. 1, where we assume the car in front (in red) is running a data-driven parameterized policy whose behavior style may be adjusted via a vector of bounded parameters, and the ego car (in green) is running a fixed policy which we would like to evaluate its performance when interacting with other vehicles. To characterize the parameters of the parameterized policy, one must observe the interactions that play out and obtain a mapping between the parameters and a measure of such outcomes. This is a straightforward task when the outcomes are simply defined. However, the problem becomes more challenging when outcomes are described by a complex logical specification with incomplete knowledge. For instance, if we request parameterizations that produce "worst-case" drivers as defined by a template specification shown in Fig. 1, the concrete specification may not be known until placed in the context of a dataset of driver interactions. The process of finding such mapping, i.e., executing the policy and measuring the outcome, is typically expensive, even for simulation environments. Though having this mapping can widen the bottleneck in the verification and validation of robotic systems such as autonomous cars, because more challenging, or rare, scenarios can now be generated automatically.

To quantitatively characterize a relationship between parameters of a parameterized policy and resulting behaviors from running that policy in a computationally efficient way, we adopt variational inference to connect parameters from parameterized policies to parameters from parametric signal temporal logic (pSTL) formulas [5], [6]. Signal temporal logic (STL) is a temporal logic that is specified over dense-time real-valued signals, such as time-series data produced from continuous and hybrid systems. STL provides a con-

[1]Toyota Research Institute, Cambridge, MA 02139, USA. jonathan.decastro@tri.global
[2]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA. {karenl7, pavone}@stanford.edu
[3]Toyota Research Institute, Los Altos, CA 94022, USA. nikos.arechiga@tri.global

cise language to construct specifications (i.e., formulas) that describe relationships between the spatial (e.g., states of a robotic system) and temporal properties of a signal. pSTL is a parametric extension of STL where the parameters of the STL formula are unknown and need to be determined from input signals, and can be used as a form of feature extraction for time-series data [7].

Related work: The problem of constructing interpretable views and formally-specifiable models has ties to ongoing work within machine learning and formal methods. In creating policy models that achieve a given objective, there are several learning-based approaches exist. For instance, variational autoencoders (VAE) and conditional VAE (cVAE) have proven successful of multi-agent imitation learning with a compact parameter representation exist, e.g. [8], [9]. Where there exists a well-defined terminal goal or outcome, reinforcement learning techniques have been employed to configure agents that achieve certain goals [10]. Such goals can be made more expressive by leveraging temporal logics [11], and symbolic abstractions that can be learned [12]. However, it is in general difficult to configure policies that generate certain time-series properties through rewards and, moreover, goal adherence often comes at the expense of reconstruction (e.g., imitation learning).

Within the context of latent models, the concept of disentangelement [13], [14] has become a popular means to enforce a direct relationship between metrics and latent variable models. Techniques such as [15] provide semantic meaning in the latent structure using unsupervised learning. The compactness of the representation and ability to semantically cluster the behavior of a deep network model is useful when a metric exists, but inferring such metrics from data while respecting a user-defined logical structure remains a challenge.

Bridging metrics with an expressive set of logical specifications is a well-studied area. In particular, [16] studies computation graph learning over fixed predicates, and use formula robustness, describing the degree to which a property is satisfied, to guide the search of a decision tree that fits features of a set of measured traces. The work of [17] exploits logical structure of a problem to inform testing of a given cyber-physical system. In [18], the authors propose an approach to infer temporal logic formulas and impose a tightness requirement to learned predicates that precisely fit features of a corpus of time-series data. [19] introduces a technique that allows construction of sparse formulas in a semi-supervised manner where it is assumed an oracle can provide positive or negative labels. The work of [20] use time-series analysis to perform logical clustering over a set of observed time traces. While the above works set the stage for using logical structure in the data to learn representations, there are often some simplifying assumptions on the types of abstractions used to perform clustering (i.e. hyperboxes or ellipsoids) and fall short of enhancing interpretability of policies.

Perhaps the closest work to ours is that of [7], where the authors learn the parameters of a temporal logic specification to perform clustering. Our work differs in two respects. First, rather than learning formulas under strict semantic interpretations, we focus on the problem of interpreting policies under a variational inference scheme to yield more expressive probabilistic interpretations of the policies. Second, our approach uses backpropagation to train the predicates directly, rather than training box-constraints on the satisfying predicates, which may be limiting.

Statement of contributions: In this paper, we address the problem of policy interpretability. We introduce a means for clustering a given dataset using pSTL formulas with parameters under a given fixed policy. Using our approach, one can generate specific outcomes of a policy where such examples exist and are supported by actual observed outcomes of the policy, either from data or simulation. The specific contributions are the following.

1) A variational inference approach for learning parameters of pSTL formulas.
2) Proposals for several optimization criteria for specifying the desired semantics for the learned predicates.
3) A demonstration of the approach in the setting of policy interpretation for parameterized policies for behavior modeling in driving scenarios, in which the goal is to expose parameters that allow configuration of agents to achieve some formally-specifiable outcome.

We envision utility in a wide array of contexts. In motion planning, an interpretable view may be useful in providing a mechanism to select learned policies; in social contexts, it may be used to determine legibility of a given motion plan, or inferring the intent of multiple agents.

II. Background

We provide background on the policies we consider and the underpinning logical formalism we adopt; parametric signal temporal logic.

A. Parameterized Policies

In this section, we outline a general definition of a parameterized policy. We define a dynamics model of a system to be a mapping from state-action pairs to next states, i.e. $\mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is a set of states and $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ is a set of actions. We further define $\mathcal{Y} \subseteq \mathbb{R}^{n_y}$ to be a set of observations of the environment (external to the system). Given a value $s \in \mathcal{X} \times \mathcal{Y}$, we define a time trace (or trajectory) as $\xi^t = s_0 s_1 s_2 ... s_t$, where t is the current time step. Letting $z \in \mathcal{Z} \subset \mathbb{R}^{n_z}$ denote a vector of parameters and $\xi^t = (\xi_x^t, \xi_y^t) \in \mathcal{X}^t \times \mathcal{Y}^t$ a trajectory of states and observations, we define a *parameterized policy* as the distribution over actions at timestep t; $p(a_t \mid \xi_x^t, \xi_y^t, z)$, with $a_t \in \mathcal{U}$.

A popular choice for policy model is the conditional variational autoencoder (cVAE) [8], [21], in which both the policy $p(a_t \mid \xi_x^t, \xi_y^t, z)$, referred to as a *decoder*, and the posterior $q(z \mid \xi_x^t, \xi_y^t)$, referred to as an *encoder*, are trained simultaneously to ensure a compact representation in which z can be inferred from a history of traces ξ_x^t and ξ_y^t. In a cVAE, $q(z \mid \xi_x^t, \xi_y^t)$ provides a measure of

the likelihood of a parameter given the prediction and the context of its environment is expected to be encountered, while the likelihood $p(a_t \mid \xi_x^t, \xi_y^t, z)$ informs the confidence of the prediction under the parameterization and environment. Often, we assume the conditional distributions to be Gaussian, with the mean and variance represented by the nonlinear mappings $\mathcal{X}^t \times \mathcal{Y}^t \mapsto_{\theta_\mu} \mathcal{U}$ and $\mathcal{X}^t \times \mathcal{Y}^t \mapsto_{\theta_\sigma} \mathcal{U}$, respectively, where θ_μ and θ_σ are neural network weights. The latent representation is useful for sample efficiency and lends to interpretability when the dimension of z is small. While interpretation of z can often be found by clustering, e.g. according to semantics [9], the objective of this work is to make such policies more guided, more principled, and human-understandable.

As we detail in Sec. IV, while our formulation supports $q(z \mid \xi_x^t, \xi_y^t)$ when available, it is not strictly required; e.g. analytically-derived policies such as ordinary differential equations parameterized by lookup tables. Also, note that policies on actions are not limiting. Without loss of generality, we may subsume dynamics and adopt the same approach to policies of the form $p(x_{t+1} \mid \xi_x^t, \xi_y^t, z)$, with $x_{t+1} \in \mathcal{X}$, the state at timestep $t+1$.

B. Parameterizable Temporal Logics

To make the policies introduced above interpretable by a human, we introduce the parametric extension of the formal specification language signal temporal logic.

Definition II.1 (Parametric Signal Temporal Logic)**.** *Parametric STL (pSTL) formulas are defined recursively as follows:*

$$\varphi ::= \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \square_{[a,b]}\varphi \mid \varphi \, \mathcal{U}_{[a,b]}\psi$$

where μ is an atomic predicate, whose truth value is determined by the inequality $\mu(s) < c$, for some parameter c, and φ is an STL formula. We write φ_c to denote a pSTL formula parameterized by some vector c. A trace $\xi^T = s_0 s_1 s_2 ... s_T$ satisfies φ if ξ^T satisfies φ_c at time $t = 0$. In other words, ξ^T satisfies $\square_{[a,b]}\varphi$ if φ holds at every time step between a and b, and ξ^T satisfies $\varphi \mathcal{U}_{[a,b]}\psi$ if, between time steps a and b, φ is true at every time step up to some step when ψ holds. Additionally, let $\Diamond_{[a,b]}\varphi := \top \mathcal{U}_{[a,b]}\varphi$, such that ξ^T satisfies $\Diamond_{[a,b]}\varphi$ if φ is true at some time step between a and b.

The predicates are assumed to be of the form $\mu(\xi^T) \bowtie c$, with $\bowtie \in \{\leq, \geq, =\}$. In this work, we adopt the *quantitative semantics* of pSTL, wherein, for a given formula φ_c, each trace of length T admits a robustness value $\rho_{\varphi_c} : \mathbb{R}^T \times \mathbb{R}_{\geq 0} \mapsto \mathbb{R}$, as follows.

- $\rho_{\varphi_c}(\xi^T, t) \geq 0$ if ξ^T satisfies φ_c at time t, and
- $\rho_{\varphi_c}(\xi^T, t) < 0$ if ξ^T does not satisfy φ_c at time t.

For a given STL formula, the robustness value is calculated by recursion on the parse tree of the formula as follows:

$$\begin{aligned}
\rho_\top(\xi^T, t) &= \rho_{\max} \\
\rho_{\mu(s_t)\leq c}(\xi^T, t) &= c - \mu(s_t) \\
\rho_{\varphi_c \wedge \psi_{c'}}(\xi^T, t) &= \min(\rho_{\varphi_c}(\xi^T, t), \rho_{\psi_{c'}}(\xi^T, t)) \\
\rho_{\varphi_c \vee \psi_{c'}}(\xi^T, t) &= \max(\rho_{\varphi_c}(\xi^T, t), \rho_{\psi_{c'}}(\xi^T, t)) \\
\rho_{\Diamond_{[a,b]}\varphi_c}(\xi^T, t) &= \max_{t' \in [t+a, t+b]} \rho_{\varphi_c}(\xi^T, t') \\
\rho_{\square_{[a,b]}\varphi_c}(\xi^T, t) &= \min_{t' \in [t+a, t+b]} \rho_{\varphi_c}(\xi^T, t') \\
\rho_{\varphi_c \mathcal{U}_{[a,b]} \psi_{c'}}(\xi^T, t) &= \\
&\max_{t' \in [t+a, t+b]} (\min(\rho_{\psi_{c'}}(\xi^T, t'), \min_{t'' \in [0, t']} \rho_{\varphi_c}(\xi^T, t'')))
\end{aligned}$$

Definitions corresponding to the operators "$=$", "$\geq$" and "$\Rightarrow$" can be constructed accordingly. For brevity, we use the shorthand $\rho_c(\xi^T) := \rho_{\varphi_c}(\xi^T, 0)$.

III. Problem Statement

We are concerned with instantiating parameterized policies that interact with an environment involving possibly many different agents. Due to the causal nature of the execution of a policy, it is not always clear what the outcome of a given parameterization will be. For instance, in the running example of Fig. 1, different behaviors and driving styles may emerge with different parameterizations of a given driving policy. However, there may be no easy way to select these parameters allowing for exploration of the spectrum between "easy" and "challenging" cases from the perspective of the vehicle behind. However, we may be able to observe interactions in data and infer the parameters of a formal specification to best match the observed outcomes in data corresponding to a user's interpretation of "easy" or "challenging" cases.

More precisely, we posit that a pSTL specification φ_c, referred to as a *pSTL template*, is both human-interpretable and able to encode desirable characteristics of a signal. Our goal is to: (1) learn parameters of φ_c yielding clusters of z that best fit the data, and (2) use the learned specification to construct an interpretable parameterization $\hat{z} \in \hat{\mathcal{Z}}$ whose parameters are commensurate with satisfaction of φ_c. A fixed policy can have several interpretations. As such, we call this approach an *interpretable view* on a parameterized policy.

IV. Interpretable Views on Policies

Given a pSTL formula parameterized by c, we infer c under a given policy and data corpus. Consider a dataset of trajectories $\mathcal{D} = \{\xi_0^{T_0}, \ldots, \xi_N^{T_N}\}$ produced by a policy in a particular multi-agent environment. Our goal is to find a set of parameters for a given pSTL template that best characterizes $\mathcal{D}$ with respect to the logical structure provided by the template. The approach is outlined in Fig. 2.

A. Variational Inference

Since the problem of inferring c is intractable to solve in general, we draw upon variational inference [22]. We discuss the case where a policy exists in latent variable form, i.e. where both $p(a_t \mid \xi^t, z)$ and $q(z \mid \xi^t)$ are given (e.g. cVAE), then discuss the standard form where only $p(a_t \mid \xi^t, z)$ is given (e.g. z are hyperparameters).

Suppose we are given a distribution of a batch (vector) of modeling parameters $\mathbf{z}$ conditioned on a batch of trajectories $\boldsymbol{\xi}$, $q(\mathbf{z} \mid \boldsymbol{\xi})$. In our derivation, we are instead interested in

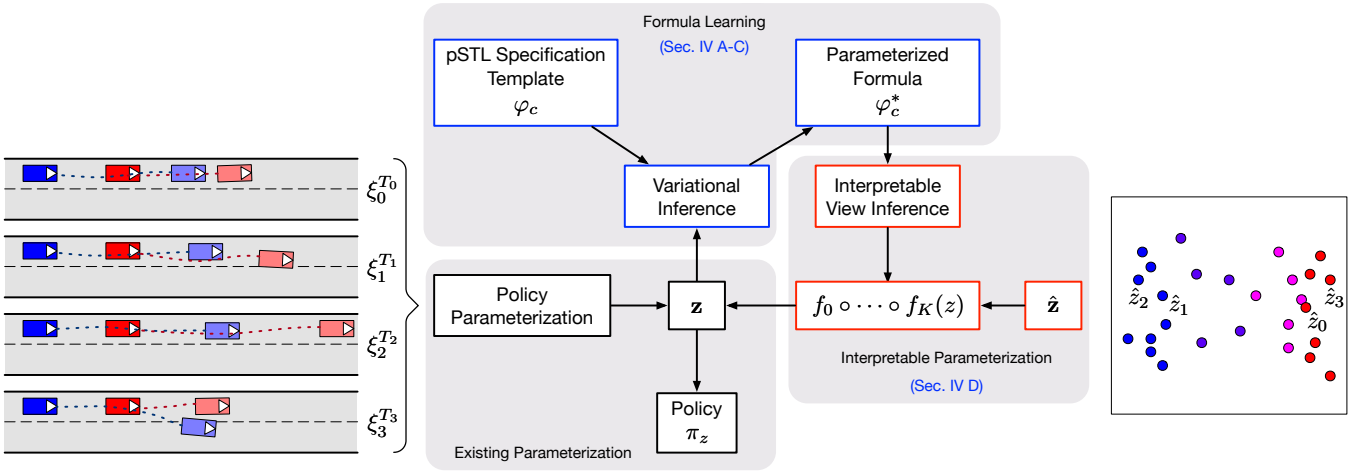

Fig. 2: Overview of the interpretable policy parameterization scheme.

the joint distribution covering the parameterizations across the entire dataset, i.e. a batch vector $Z \in \mathbb{R}^{n_z \times |\mathcal{D}|}$. We may exploit independence so that $q(Z \mid \mathcal{D}) = \prod_i q(Z \mid \xi_i^{T_i})$. Our objective is to find an estimate $q(c \mid \mathcal{D})$ of the true posterior $p(c \mid \mathcal{D})$ of pSTL parameters c via

$$\begin{aligned}
&\mathbb{D}_{KL}\left[q(c \mid \mathcal{D}) \parallel p(c \mid \mathcal{D})\right] \\
&\quad = \int q(c \mid \mathcal{D}) \log \frac{q(c \mid \mathcal{D}) p(\mathcal{D})}{p(c, \mathcal{D})} dc \\
&\quad = \int \mathbb{E}_{q(\mathbf{Z}|\mathcal{D})} q(c \mid \mathbf{Z}) \log \frac{\mathbb{E}_{q(\mathbf{Z}|\mathcal{D})} q(c \mid \mathbf{Z}) p(\mathcal{D})}{p(\mathcal{D} \mid c) p(c)} dc \\
&\quad \geq \mathbb{E}_{q(c|\mathcal{D})} \left[\log \frac{\mathbb{E}_{q(\mathbf{Z}|\mathcal{D})} q(c \mid \mathbf{Z})}{p(c)} - \log p(\mathcal{D} \mid c)\right] \\
&\quad = \mathbb{D}_{KL}\left[\mathbb{E}_{q(\mathbf{Z}|\mathcal{D})} q(c \mid \mathbf{Z}) \parallel p(c)\right] \\
&\qquad\qquad - \mathbb{E}_{q(c|\mathcal{D})} \log p(\mathcal{D} \mid c) \qquad (1)
\end{aligned}$$

where $p(\mathcal{D})$ is the true (unknown) distribution of the data, hence the introduction of the inequality.

The expression on the right hand side of the inequality is commonly referred to as the *expectation lower bound*, or ELBO. We assume the conditional distributions are normally distributed with mean and variance given by neural networks. The expectation appearing within the KL-divergence is approximated via Monte Carlo sampling of the batch-vectorized parameters $\mathbf{Z}$. Hence, $q(\mathbf{Z} \mid \mathcal{D})$ consists of a Monte Carlo approximation to the mean and variance. We generate the samples via the reparameterization trick using the provided mean and variance. We found that a standard Normal distribution for the prior $p(c)$ yielded good results. The likelihood $p(\mathcal{D} \mid c)$ is discussed in the next section.

It is well-known that the evidence ELBO in (1) suffers from approximation errors, due to the Gaussian distribution assumed in the conditional distributions, we adopt normalizing flows to enrich the distributions. Normalizing flows are invertible, distribution-preserving transformations that enable representation of a rich set of distributions. In similar fashion to [23], we let c_i denote the output of the ith layer of the flow f_i, and treat c_0 as the input and $c = c_K$ as the final layer of a sequence of K flows.

Equation (1) may then then re-written as:

$$\begin{aligned}
&\mathbb{D}_{KL}\left[\mathbb{E}_{q(\mathbf{Z}|\mathcal{D})} q(c_0 \mid \mathbf{Z}) \parallel p(c_K)\right] - \mathbb{E}_{q(c_0|\mathcal{D})} \log p(\mathcal{D} \mid c_K) \\
&\qquad + \mathbb{E}_{q(c_0|\mathcal{D})} \left[\log \prod_{t=1}^{K} \left|\det \frac{\partial f_t}{\partial f_{t-1}}\right|\right] \qquad (2)
\end{aligned}$$

which gives an arbitrarily rich distributional representation of the ELBO. We choose inverse autoregressive flows (IAF) [24] within the inference scheme to improve the expressive power of each introduced transformation layer.

For policies not in variational form and lacking an encoder, one may train (2) directly on traces by replacing $\mathbb{E}_{q(\mathbf{Z}|\mathcal{D})} q(c_0 \mid \mathbf{Z})$ with $q(c_0 \mid \mathcal{D})$ and using a recurrent structure similarly to standard trajectory-based cVAEs.

B. pSTL Semantics via Likelihood Tailoring

The likelihood term $p(\mathcal{D} \mid c)$ describes a criterion we wish to assert in order to establish concrete semantics. As with existing data clustering problems, there exist a number of possible approaches. Below we outline two approaches, which have commonly-used analogs in standard data analysis, for interpreting a given dataset.

- **Discriminative Clustering.** A discriminative clustering model seeks a description that best distinguishes between two types of data traces. In similar fashion to binary classification, the objective is to obtain a pSTL formula whose parameterization minimizes the mean square error of the robustness values across the population. To cast this problem into the probabilistic setting, we use a softmax likelihood, i.e.

$$p(\xi_i \mid c) = \frac{\exp(-\rho_c^2(\xi_i)/\tau)}{\sum_i \exp(-\rho_c^2(\xi_i)/\tau)}$$

  where τ is a softness parameter. The choice provides a semantic for *best discriminator between satisfying and non-satisfying examples* under the template.
- **Absolute Clustering.** This type of model seeks to cluster together common traits shared among a set of

traces. The likelihood is modeled by

$$p(\xi_i \mid c) = \frac{\exp(L(\xi_i)/\tau)}{\sum_i \exp(L(\xi_i)/\tau)}$$

where $L(\xi) = (1 + \rho_c(\xi))\,|\rho_c(\xi)| + \rho_c(\xi)^2$ is a modified hinge loss. That is, those data that satisfy the formula are given priority over those that do not, and those that do are made to be only marginally satisfied. The loss improves as formulas are discovered that collect more marginally-satisfiable data. Such models can give a semantic for *best marginal fit of satisfying examples* under the template.

In both cases, we assume that likelihoods are independent for each trace, such that jointly we have $p(\mathcal{D} \mid c) = \prod_i p(\xi_i \mid c)$. Notice that, given the same dataset, the semantics of each are different. The purpose of the discriminative model is to form a logical relationship to the data that minimizes the differences in robustness values over the population of traces. The absolute clustering model, on the other hand, is a regression problem; it constructs a logical relationship that maximizes the number of marginally-robust examples.

C. Training Procedure

To capture the discrete nature of the formula learning problem, we solve the variational inference approach using backpropagation. As in previous works [25], [26], we use the softmax/softmin trick in order to cast the pSTL formulas as smooth, differentiable functions over the parameters c. During training, we anneal both the softness parameter and the weighting between reconstruction and KL divergence in the loss function to prevent degeneracy.

D. Interpretable Views on Latent Encodings

Transforming the existing policy parameter space $\mathcal{Z}$ into an interpretable space under the learned formula requires solving an additional inference problem. Similar to [13], we use distribution-preserving normalizing flows to construct a mapping between z and $\hat{z}$. We learn flows that map the modeling parameters $\mathbf{z}$ to some space $\hat{\mathbf{z}}$, constrained to have a linear mapping to the robustness value $\rho_c(\boldsymbol{\xi})$ (see Fig. 2).

We choose the following objective:

$$\begin{aligned}\mathbb{D}_{KL}\left[q(\mathbf{z}_0 \mid \mathbf{z}, \mathbf{m}) \parallel p(\mathbf{z}_{K'})\right] - \mathbb{E}_{q(\mathbf{z}_0|\mathbf{z})} \log p(\mathbf{z} \mid \mathbf{z}_{K'}) \\ - \mathbb{E}_{q(\mathbf{z}_0|\mathbf{z})} \log p(\mathbf{m} \mid \mathbf{z}_{K'}) + \mathbb{E}_{q(\mathbf{z}_0|\mathbf{z})} \left[\log \prod_{t=1}^{K'} \left|\det \frac{\partial f_t}{\partial f_{t-1}}\right|\right]\end{aligned} \tag{3}$$

where we define the interpretability metric $\mathbf{m} = \tanh(\rho_c(\boldsymbol{\xi}))$, and where the mean and variance of $p(\mathbf{m} \mid \mathbf{z}_{K'})$ are fully-connected linear layers. We find planar flows to be sufficient to performing inference. For vector-valued metrics, we wish to form linear relationships between each element of the vector and individual elements of the new parameter space by training individual, hence independent, linear decoder layers for each metric.

It is useful to point out that solving (2) provides a clustering of the policy parameter space according to the formula, while the interpretable view in (3) provides a selection mechanism for parameters, as depicted in Fig. 2. Note that since the data we source is incomplete and possibly changes with different interactions with the environment, the parameterization is only locally valid in the neighborhood of the data we train on.

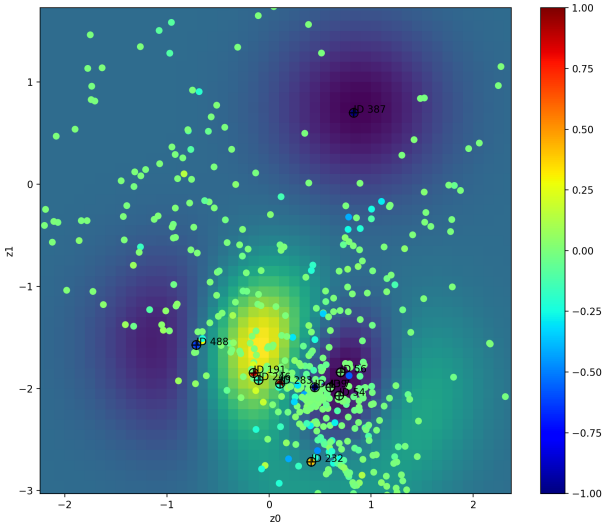

Fig. 3: Latent space with interpretability metric corresponding to the learned formula in Table I. The greatest- and least-robust examples are labeled.

V. Case Study: Agent Behavior Modeling

We examine the case of simulating driver behavior, where the use case can be intent prediction, simulation, and validation of an autonomous ego car interacting with different styles of drivers. We consider these agents to be human-driven vehicles driving in tandem, as pictured in Fig. 1.

A. Agent Policy

We use the Argoverse dataset [27] to build a behavior model of longitudinal behaviors, a policy taking in features and conditioning variables and producing acceleration as an output. We train a long short-term memory (LSTM) encoder and decoder as a cVAE with the position and velocity of the car ahead treated as features x, and the position and velocity of the car behind treated as the conditioning variables y. We model the driver's style using a two-dimensional latent vector z. The decoder LSTM transforms x, y, z into deterministic acceleration commands. The future states are determined by a point-mass model approximating the car's physics.

In training the cVAE, we harvest data that only produces in-lane examples where there exists a vehicle ahead. Since the formula propositions require position, velocity and accelerations of both cars, we use a Kalman smoother to compute the derivatives from positional data, before transforming each trace into the lane-relative reference frame of the ego car.

B. Construction of a Policy Selection Mechanism

We next synthesize an interpretable view from a pSTL formula to construct a new, compact parameterization where

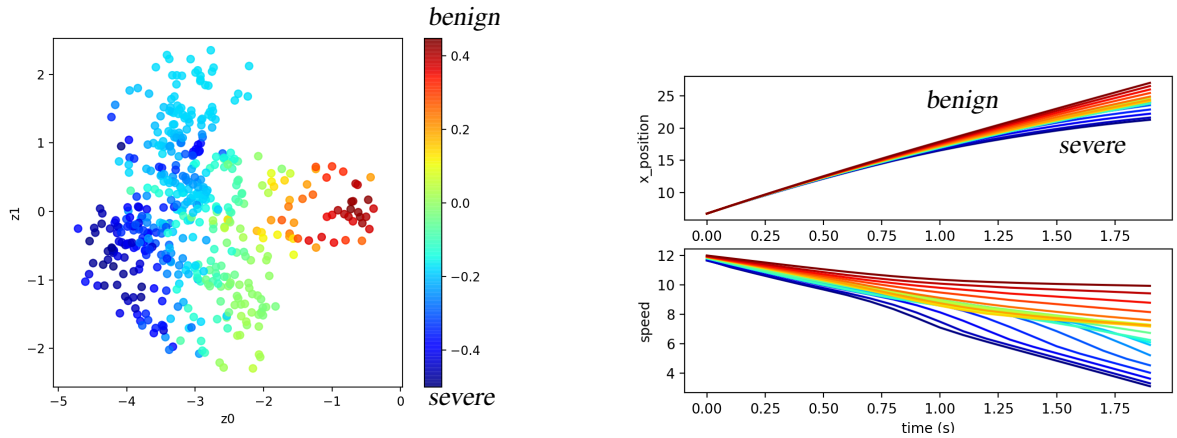

Fig. 4: Left: the interpretable parameterization, where the parameter $\hat{z}^0$ clearly can be used to adjust the style of the behavior according to the severe-benign pair of formulas. Right: The progression of trajectories realized when adjusting $\hat{z}^0$.

TABLE I: Learned pSTL parameters for the cVAE.

$c_{\Delta x}$	$c_{\Delta \dot{x}}$	$c_{\ddot{x}}$	$\hat{c}_{\Delta x}$	$\hat{c}_{\Delta \dot{x}}$	$\hat{c}_{\ddot{x}}$
91.36	-2.31	-3.08	-3.75	-10.63	0.1981

individual parameters are made to allow for adjustment of increasing or decreasing satisfaction of the formula. This can be useful, for instance, in validation of an autonomous driving system, where it is often important to stress-test the autonomy against more- or less-difficult agents. Our metric is constructed from a pair of STL declarative specifications defining outcomes where the interaction with the vehicle behind are deemed most or least severe:

$$\textit{Severe:} \quad \Diamond(\Delta x \leq c_{\Delta x}) \wedge \Diamond(\Delta \dot{x} \leq c_{\Delta \dot{x}}) \wedge \Diamond(\ddot{x}_{lead} \leq c_{\ddot{x}})$$
$$\textit{Benign:} \quad \Box(\Delta x \geq \hat{c}_{\Delta x}) \wedge \Box(\Delta \dot{x} \geq \hat{c}_{\Delta \dot{x}}) \wedge \Box(\ddot{x}_{lead} \geq \hat{c}_{\ddot{x}})$$

where $\Delta x = x_{lead} - x_{trailing}$ is the distance between vehicles, $\Delta \dot{x} = \dot{x}_{lead} - \dot{x}_{trailing}$ is the relative speed, and $\ddot{x}_{lead}$ is the lead vehicle's acceleration. The first formula describes a preference for obtaining collisions and near-collisions that the driver in back could possibly have avoided. That is, at some point, the lead car had experienced sharp deceleration but at some other point there was a significant difference in speeds. The second formula describes the opposite behavior; those behaviors which we consider benign; i.e. cases where the distance, relative speeds, and accelerations are always kept high. Note that the *benign* formula is a stronger requirement than the negation of the *severe* formula, and hence will represent a well-defined, compact cluster of behaviors.

pSTL parameter learning: We train the pSTL parameters over the same dataset used to train the cVAE. To be able to form a regression from one formula to the other, we adopt the metric $\mathbf{m} = \exp(\rho_{severe}(\boldsymbol{\xi})) - \exp(-\rho_{benign}(\boldsymbol{\xi}))$, and assign $\hat{z}^0$ to this metric, reserving $\hat{z}^1$ to represent behaviors that have not been expressed by the formula.

We use the tool STLCG for differentiable pSTL computation graph learning described in [25]. For the parameter inference model, we use a 16-dimensional encoder with three fully-connected layers and $\tanh$ activations. We use five IAF layers for the clustering model. To estimate the log-likelihood, we use 50 Monte Carlo samples within the inference network.

The learned pSTL parameters are shown in Table I. These are chosen as the empirical maximum likelihood values. In the plot of the original parameterization in Fig. 3, it is clear that there exists pockets of greater and lower severity of the model throughout the latent space.

Interpretable view learning: We construct an interpretable view using a composition of 15 planar flows to provide a two-dimensional mapping $\hat{z} = (\hat{z}^0, \hat{z}^1)^T \mapsto z$. For both the encoder and decoder for the latent variables, three 16-dimensional fully-connected $\tanh$ layers are used. In Fig. 4, the metric associated with the pair of formulas is seen to align with the $\hat{z}^0$-axis, producing a new parameterization that offers the progression of trajectories that span from *benign* to *severe* as the principal axis $\hat{z}^0$ is adjusted.

C. Discriminative Clustering

To illustrate the utility in discriminating between types of behavior of a given policy, we introduce a formula that is able to discriminate between different types of agent behavior that are interpreted as *benign*, while preserving the ability to easily progress from any of the benign cases to more severe behavior.

We base our discriminative model on acceleration:

$$\Box(\ddot{x}_{lead} \leq \tilde{c}_{accel})$$

That is, we wish to separate examples that have demonstrated the consistent deceleration from examples that may have accelerated at some point along the trace. We form an interpretable view using both the metric from Sec. V-B and the one learned here to form a parameter axis allowing progression from *severe* to *benign* with $\Box(\ddot{x}_{lead} \leq \tilde{c}_{accel})$ and, on the other axis, a progression from *severe* to *benign* with $\neg\Box(\ddot{x}_{lead} \leq \tilde{c}_{accel}) = \Diamond(\ddot{x}_{lead} > \tilde{c}_{accel})$. The resulting parameter set is two-dimensional $\tilde{z} = (\tilde{z}^0, \tilde{z}^1)^T$.

The corresponding behaviors shown in Fig. 5 illustrate the resulting trajectories for each case. Clearly the two cases

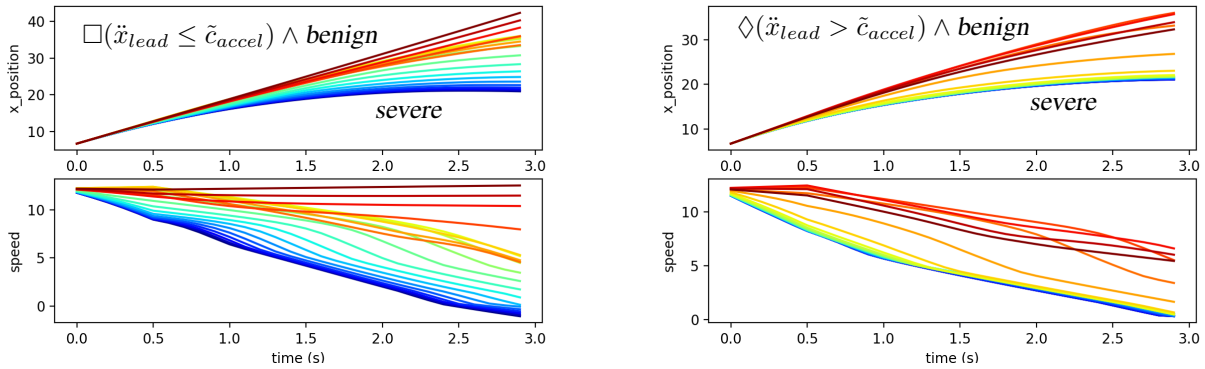

Fig. 5: A discriminative view constructed only on benign styles using the acceleration-discriminating formula, with the left-hand pane showing trajectories were created by adjusting interpretable parameter $\tilde{z}^0$ and right-hand pane trajectories obtained by adjusting $\tilde{z}^1$.

show different speed profiles, and these can be configured independently with $\tilde{z}^0$ and $\tilde{z}^1$.

VI. Conclusions

We address the problem of making policies more transparent and configurable to a user by constructing an interpretable view that follows a user's formal specification, and learning a new parameterization that best matches outcomes revealed in a corpus of data. Our proposed approach is applicable to general data-driven policies where upon interacting with an environment, we are interested in the resulting spatial and temporal properties. As demonstrated with our case study, we find that this approach is particularly beneficial for scenario generation in the testing and validation of autonomous driving policies in simulation, but may also be useful for on-line intent prediction and social awareness.

Future work includes performing inference over the logical structure of formulas, and learning over integer-valued parameters. Currently, the challenge of learning formulas from data that are sufficiently expressive while remaining interpretable to humans is an interesting topic that should be further explored. Such tradeoffs are practically interesting in multi-agent settings, especially as the number of agents in the environment grows.

References

[1] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," 2019, Available at https://arxiv.org/abs/1903.06758.

[2] K. Julian and M. J. Kochenderfer. (2019) A reachability method for verifying dynamical systems with deep neural network controllers. Available at https://arxiv.org/abs/1903.00520.

[3] T. A. Wheeler and M. Kochenderfer, "Factor graph scene distributions for automotive safety analysis," in *IEEE International Conference on Intelligent Transportation Systems*, 2016.

[4] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *IEEE Intelligent Vehicles Symposium*, 2018.

[5] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. Int. Symp. Formal Techniques in Real-Time and Fault-Tolerant Systems, Formal Modeling and Analysis of Timed Systems*, 2004.

[6] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Int. Conf, on Runtime Verification*, 2012.

[7] M. Vazquez-Chanlatte, S. Ghosh, J. V. Deshmukh, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Time series learning using monotonic logical properties," in *International Conference on Runtime Verification*, 2018.

[8] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3399–3406.

[9] X. Huang, S. G. McGill, J. A. DeCastro, B. C. Williams, L. Fletcher, J. J. Leonard, and G. Rosman, "Diversity-Aware Vehicle Motion Prediction via Latent Semantic Sampling," *arXiv:1911.12736 [cs]*, Nov. 2019, arXiv: 1911.12736.

[10] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement Learning from Imperfect Demonstrations," *arXiv:1802.05313 [cs, stat]*, May 2019, arXiv: 1802.05313.

[11] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3834–3839.

[12] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[13] T. Adel, Z. Ghahramani, and A. Weller, "Discovering interpretable representations for both deep generative and discriminative models," in *International Conference on Machine Learning*, 2018.

[14] Y. Hristov, D. Angelov, M. Burke, A. Lascarides, and S. Ramamoorthy, "Disentangled relational representations for explaining and learning from demonstration," in *Conference on Robot Learning*, 2019.

[15] Y. Li, J. Song, and S. Ermon, "InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3812–3822.

[16] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A Decision Tree Approach to Data Classification using Signal Temporal Logic," in *Hybrid Systems: Computation and Control (HSCC)*, Vienna, Austria, April 2016, pp. 1–10.

[17] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient Guiding Strategies for Testing of Temporal Properties of Hybrid Systems," in *NASA Formal Methods*, K. Havelund, G. Holzmann, and R. Joshi, Eds., Cham, 2015, pp. 127–142.

[18] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, "TeLEx: Learning signal temporal logic from positive examples using tightness metric," *Formal Methods in System Design*, vol. 54, no. 3, pp. 364–387, 2019.

[19] S. Jha, T. Sahai, V. Raman, A. Pinto, and M. Francis, "Explaining AI Decisions Using Efficient Methods for Learning Sparse Boolean Formulae," *Journal of Automated Reasoning*, vol. 63, no. 4, pp. 1055–1075, Dec. 2019.

[20] M. Vazquez-Chanlatte, J. V. Deshmukh, X. Jin, and S. A. Seshia, "Logic-based Clustering and Learning for Time-Series Data," *arXiv:1612.07823 [cs]*, May 2017, arXiv: 1612.07823.

[21] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence,

D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3483–3491.

[22] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, May 2014, arXiv: 1312.6114.

[23] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, Lille, France, Jul. 2015, pp. 1530–1538.

[24] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved Variational Inference with Inverse Autoregressive Flow," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 4743–4751.

[25] K. Leung, N. Arechiga, and M. Pavone, "Backpropagation for parametric STL," in *IEEE Intelligent Vehicles Symposium: Workshop on Unsupervised Learning for Automated Driving*, 2019.

[26] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic." IEEE, Aug. 2017, pp. 1235–1240.

[27] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Conference on Computer Vision and Pattern Recognition*, 2019.