2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops

Model, Data and Reward Repair: Trusted Machine Learning for Markov Decision Processes

Shalini Ghosh, Susmit Jha, Ashish Tiwari, Patrick Lincoln, Xiaojin Zhu

shalini.ghosh@gmail.com, {tiwari, jha, lincoln}@csl.sri.com, jerryzhu@cs.wisc.edu

***Abstract*—When machine learning (ML) models are used in safety-critical or mission-critical applications (e.g., self driving cars, cyber security, surgical robotics), it is important to ensure that they provide some high-level guarantees (e.g., safety, liveness). We introduce a paradigm called Trusted Machine Learning (TML) for making ML models more trustworthy. We use Markov Decision Processes (MDPs) as the underlying dynamical model and outline three TML approaches: (1) Model Repair, wherein we modify the learned model directly; (2) Data Repair, wherein we modify the data so that re-learning from the modified data results in a trusted model; and (3) Reward Repair, wherein we modify the reward function of the MDP to satisfy the specified logical constraint. We show how these repairs can be done efficiently for probabilistic models (e.g., MDP) when the desired properties are expressed in some appropriate fragment of logic such as temporal logic (for example PCTL, i.e., Probabilistic Computation Tree Logic), first order logic or propositional logic. We illustrate our approaches on case studies from multiple domains, e.g., car controller for obstacle avoidance, and a query routing controller in a wireless sensor network.**

I. Introduction

When machine learning (ML) algorithms are used in mission-critical domains (e.g., self-driving cars, cyber security) or life-critical domains (e.g., surgical robotics), it is often important to ensure that the learned models satisfy some high-level correctness requirements — these requirements can be instantiated in particular domains via constraints like safety (e.g., a robot arm should not come within five meters of any human operator during any phase of performing an autonomous operation) or liveness (e.g., a car should eventually cross a 4-way intersection). Such constraints can be formally described in propositional logic, first order logic or temporal logics such as Probabilistic Computation Tree Logic (PCTL)[31]. For example, in a lane change controller we can enforce the following PCTL safety property on seeing a slow-moving truck in front: $\Pr_{>0.99}[F(\mathit{changedLane\ or\ reducedSpeed})]$, where F is the *eventually* operator in PCTL logic — this property states that the car should eventually change lanes or reduce speed with high probability (greater than 0.99). *Trusted Machine Learning* (TML) refers to a learning methodology that ensures that the specified properties are satisfied.

In this paper, we show how to ensure that a Markov Decision Process (MDP) model trained on data satisfies logical constraints. Let us consider that a set of safety properties define the *safety envelope* of an ML model. During model training, Model Repair can be used to modify the trained model to ensure that the modified model satisfies the safety properties and stays within the safety envelope. When the data is noisy, the model trained on the corrupt data can potentially go outside the safety envelope — Data Repair can be used to identify and drop the corrupt data points, such that the ML model retrained on the repaired data is now within the safety envelope. When the reward function used in the MDP model is such that the optimal policy of the model does not satisfy the safety constraint, we can use Reward repair to correct the reward such that the corresponding optimal policy satisfies the logical constraint. The key contributions of this paper are:

1. We show how to enforce the satisfaction of logical constraints (e.g., PCTL) by MDP models. The main challenge here is in putting temporal logic constraints into the ML training procedure — we show how to reduce Model and Data Repair to non-linear optimization problems using parametric model checking, so the complex temporal logic properties (e.g., safety, liveness) can be satisfied while keeping the optimization problems feasible and tractable.

2. We propose Reward Repair, which shows how to project an “unsafe” reward function (which generates an optimal policy that violates provided safety constraints) to a safety envelope defined by the provided logical constraints, such that the optimal policy of the MDP model with the modified reward satisfies the safety constraints.

3. We present case-studies showing the actual applications of Model, Data and Reward Repair in the domains of automatic car control and wireless sensor networks.

II. Problem Definition

Notation: Let $\mathcal{M}$ be a class of MDP models with each model being defined as (S, A, R, P, L), where S is the set of states, A is the set of actions, R is the reward function, P is the transition probability between states given an action, and L is a label function [27]. Let $\mathcal{D}$ be the universe of all data sets. Let $\mathtt{ML}$ be a machine learning procedure that takes $D \in \mathcal{D}$ and returns $M \in \mathcal{M}$. Let ϕ be a desired logic property that we want the learned model to possess. We denote the fact that a model M has the property ϕ by $M \models \phi$. If a model $M = \mathtt{ML}(D)$ (trained on some data set D) does not satisfy the required logic property ϕ, then we want to “repair” either the model M or the data set D. We do not consider arbitrary repairs but rather those identified by some given constraints. Given M and D, let $\mathtt{Feas}_{M_p} \subseteq \mathcal{M}$ denote all feasible repairs of M by modifying P, $\mathtt{Feas}_{M_R} \subseteq \mathcal{M}$ denote the feasible repairs of M by modifying R, and $\mathtt{Feas}_D \subseteq \mathcal{D}$ denote all feasible repairs of D. Let $cost(M, M')$ denote the cost (a positive real function) of changing M to M', and $cost(D, D')$ denote the cost of changing D to D'. Based on these notations, we propose the following definitions of `ModelRepair`, `RewardRepair` and `DataRepair` in MDP models.

Definition 1. `ModelRepair`: *Given M, ϕ, $\mathtt{Feas}_{M_P}$, the Model Repair problem seeks to find $M^* \in \mathcal{M}$ by updating P such that $M^* \in \mathtt{Feas}_{M_P}$, $M^* \models \phi$, and M^* minimizes cost(M, M^*).*

Definition 2. `RewardRepair`: *Given M, ϕ, $\mathtt{Feas}_{M_R}$, the Reward Repair problem seeks to find $M^* \in \mathcal{M}$ by updating R such that $M^* \in \mathtt{Feas}_{M_R}$, $M^* \models \phi$, and M^* minimizes cost(M, M^*).*

Definition 3. `DataRepair`: *Given D, ϕ, $\mathtt{Feas}_D$, the Data Repair problem seeks to find $D^* \in \mathcal{D}$ such that $D^* \in \mathtt{Feas}_D$, $\mathtt{ML}(D^*) \models \phi$, and D^* minimizes cost(D, D^*).*

Given a dataset D, we first learn the model $M = \mathtt{ML}(D)$ using a learning procedure `ML` — for MDP this would correspond to using maximum likelihood for learning P or Inverse Reinforcement Learning [34] for learning R. We then check if $M \models \phi$; if it does, we output M. Otherwise, we need to repair the model and hence, we run Model Repair or Reward Repair on M to get M', depending on whether we want to modify P or R. If $M' \models \phi$, we output M'. Otherwise, the Model or Reward Repair formulations don't give a feasible solution — so, we perform Data Repair of the data D using small perturbations on the data to get D' and check if $M'' = \mathtt{ML}(D') \models \phi$. If it does, we output M''. If it doesn't, we report that ϕ cannot be satisfied by the learned model using our formulations of Model, Reward or Data Repair.

III. Background

Let us consider the example of training a probabilistic model for an autonomous car controller — the underlying model we want to learn is an MDP, which is defined as a tuple $M = (S, P, R, A, L)$ where S is a finite set of states with $s_0 \in S$ is the initial state, $P(s'|s, a)$ is a transition function that specifies the probability of the next state given the current state and action, R is the reward function mapping states to real values, A is the set of all possible actions, and L is the labeling function assigning labels to states. Section V-B discusses an example MDP model of an autonomous car controller when confronted with an obstacle in front.

The domain constraint can be provided in terms of temporal logical property ϕ in PCTL, which we want the M to satisfy. In PCTL, properties are specified as $\phi = \Pr_{\sim b}(\psi)$, where $\sim \in \{<, \leq, >, \geq\}$, $0 \leq b \leq 1$, and ψ a *path formula*. A path formula is defined using the temporal operators X (next) and $\cup^{\leq h}$ (bounded/unbounded until), where h is an integer. PCTL also uses the *eventually* operator F (defined as $F\phi = true \cup \phi$), where $F\phi$ means that ϕ is eventually true. A state s of M satisfies $\phi = \Pr_{\sim b}(\psi)$, denoted as $M, s \models \phi$, if $\Pr(Path_M(s, \psi)) \sim b$; i.e., the probability of taking a path in M starting from s that satisfies ψ is $\sim b$, where path is defined as a sequence of states in the model M. The domain constraint can also be provided in first order or other probabilistic extensions of temporal logic [15], [14]. Note that the MDP controller considered here is part of an overall autonomous closed-loop car controller. A real controller would use bounded-time variants of temporal properties, and can be learned from car traces in a vehicle simulator [30].

IV. Repairing Probabilistic Models

In this paper, we consider the class $\mathcal{M}$ to consist of all MDPs with a fixed (graph) structure, but different transition probabilities or reward functions. The property ϕ can be expressed in temporal logic (e.g., PCTL), first order logic or propositional logic.

Our approach of TML has four main steps. First, we learn a model M from data D ignoring the (temporal) safety constraint ϕ. Second, we formally verify if $M \models \phi$, and if $M \not\models \phi$, then we consider a class $\mathtt{Feas} = \{M_\lambda\}$ of models parameterized by λ, where M_λ is a possible repair of M. Third, we find a constraint ψ on λ that is sufficient to guarantee $M_\lambda \models \phi$. Finally, we find a model $M^* = M_{\lambda^*}$ that is closest to M and such that λ^* satisfies the constraint ψ. We concretize our approach below for three different class of repairs: model repair, data repair, and reward repair.

For Model Repair, we consider the subclass $\mathtt{Feas}_{M_P}$ to consist of all models $M' \in \mathcal{M}$ such that M' and M both have nonzero transition probabilities on the *same* set of edges. The user can additionally constrain (say, using lower and upper bounds on) the difference in the transition probabilities on corresponding edges, and thus, only consider "small" perturbations. Note that re-parameterizing the entire transition matrix can be considered as a possibility in Model Repair. How much of the transition matrix is considered repairable depends on the application at hand — it can determine which transition probabilities are perturbable (e.g., which part of the car controller can be modified).

For Data Repair, the subclass $\mathtt{Feas}_D$ consists of all data sets $D' \in \mathcal{D}$ that can be obtained from D by user-specified operations. For example, in our current formulation, we consider data points to be dropped from the original dataset — number of datapoints dropped from the dataset defines a metric that can be used to describe a neighborhood of corrections.

For Reward Repair, the subclass $\mathtt{Feas}_{M_R}$ is defined to be set of all MDP models M' such that M' differs from M only in the reward function — if M' has a trajectory of states/actions that violates the specified constraint then the probability of that trajectory is 0, but if the trajectory respects the specified constraint then its probability is the same as in M.

A. Model Repair

We first present an approach for solving Model Repair for MDPs. Given an MDP M with n states and $n \times n$ transition matrix P, we can get a parametric MDP M_Z by introducing an $n \times n$ matrix Z (of unknowns), such that $P + Z$ is a stochastic matrix and is the transition matrix of M_Z. The unknown parameters in Z may be constrained: if $\exists j Z_{ij} > 0$, then the state i called a controllable state and the transition between states i and j of M is controllable, since its probability can be modified. The matrix Z gives a mechanism for altering or controlling the behavior of M for repair. The parametric

MDP M_Z along with the constraints on Z defines the set $\mathtt{Feas}_{M_P}$, as discussed in Proposition 1.

Proposition 1. [4] *If M is a MDP with transition matrix P and M_Z is a MDP with transition matrix $P + Z$, and $\forall s \sum_{t \in S} Z(s,t) = 0$, then M and M' are ϵ-bisimilar, where ϵ is bounded by the maximum value in Z.*

Note that $M' \in \mathtt{Feas}_{M_P}$ and M are ϵ-bisimilar when there exists an ϵ-bisimulation between them, i.e., any path probability in M' is within ϵ of the corresponding path probability in M. Let us consider the non-zero values in Z to be the vector of variables $\mathbf{v} = v_1 \dots v_k$. We solve the Model Repair problem by solving the following optimization problem:

$$\arg\min_{\mathbf{v}} g(Z) \tag{1}$$

$$\mathtt{s.t.}, M_Z \models \phi, \tag{2}$$

$$P(i,j) + Z(i,j) = 0 \text{ iff } \quad P(i,j) = 0, \quad 1 \le i,j \le n. \tag{3}$$

In Equation 1, $g(Z)$ is a cost function that encodes the cost of making the perturbation to model parameters — a typical function is the sum of squares of the perturbation variables, i.e., $g(Z) = ||Z||_F = v_1^2 + \ldots + v_n^2$, where $||Z||_F$ is the Frobenius norm of the Z matrix. Equation 2 checks if the modified model M_{P^*} (with s_0 as its initial state) satisfies property ϕ. Equation 3 forces that no transitions are added or dropped in M_{P^*} w.r.t. M_P, only the transition probabilities are modified. This condition ensures that Z does not change the structure or stochasticity of the underlying probabilistic model. The main bottleneck in the Model Repair formulation in Equations 1-3 is the constraint in Equation 2 — if it is a temporal logic constraint, it will be difficult to directly handle it in a non-linear optimization problem. Proposition 2 shows how we can transform the above optimization problem with a "non-standard" temporal logic constraint to a standard non-linear optimization problem with non-linear constraints.

Proposition 2. *Consider a probabilistic model M and a probabilistic temporal logic formula ϕ. If M is a parametric Markov Chain (MC) or parametric Markov Decision Process (MDP) and ϕ is expressed in Probabilistic Computational Tree Logic (PCTL), then the* `ModelRepair` *problem, specified in Definition 1 and Equations 1-3, can be reduced to an equivalent set of nonlinear optimization problems with non-linear rational constraints.* (Proof sketch in supplementary material.)

As outlined in Proposition 2, if M is a discrete time Markov Chain (DTMC) or MDP, parametric model checking can convert Equations 1-3 to this constrained optimization problem:

$$\min \quad g(\mathbf{v}), \tag{4}$$

$$\mathtt{s.t.} \quad f(\mathbf{v}) \sim b, \tag{5}$$

$$\forall v_k \in \mathbf{v} : 0 < v_k + P(i,j) < 1. \tag{6}$$

where $P(i,j)$ in Equation 6 corresponds to $Z(i,j)$ matrix entries that have non-zero value v_k. This reparameterization of Equation 2, encoding the satisfiability of ϕ in M to the non-linear equation $f(v)$ in Equation 5, can be obtained using a parametric model checker, e.g., PRISM [17]. Solving the nonlinear objective function in Equation 4 with the non-linear constraints in Equation 5-6 would give us a "local optimum" of Z that transforms M to M^* — we can do that using a non-linear optimization tool, e.g., AMPL [8]. If the nonlinear optimization problem has a feasible solution, it gives us the optimal values of Z that makes the resulting model M^* satisfy the constraints ϕ.

B. Data Repair

In some cases, we try to modify the dataset D to D' so that the model trained on D' satisfies ϕ. For this, we need to solve the Data Repair problem (Definition 3) – a variant of *machine teaching* [33]. Based on the machine teaching formulation [21], the Data Repair problem can be formalized as:

$$\arg \min_{D', \Theta^*} E_T(D, D') \tag{7}$$

$$\mathtt{s.t.} \quad M_{\Theta^*} \models \phi \tag{8}$$

$$\Theta^* \in \arg\min_{\Theta} [R_L(D', \Theta) + \lambda\Omega(\Theta)], \tag{9}$$

$$s.t., g(\Theta) \le 0, h(\Theta) = 0. \tag{10}$$

Here, the inner optimization models the standard machine learning objective of regularized empirical risk minimization, consisting of the empirical risk function R_L and the regularizer Ω. E_T is the teaching "effort" function of modifying the dataset D to D', M_{Θ^*} indicates a model that is parameterized by Θ^*, while g and h are other domain constraints.

Let us consider that the dataset D is transformed to D' using a data perturbation vector p, where entries of p are 0 or 1. In this paper, we consider that a subset of data points need to be dropped from D for the resulting trained model to satisfy ϕ (e.g., those points could have noisy features or labels). So, each datapoint d_i in D is multiplied by p_i, where $p_i = 0$ indicates that the point is dropped — in this case, $p = \{p_1 \dots p_n\}$, where $n = |D|$. Also, let us consider that the effort function is characterized by the magnitude of the data perturbation, i.e., $E_T(D, D') = |D| - ||p||^2$. Using these transforms, Equations 7-10 can be reformulated as:

$$\arg \min_{p, \Theta^*} |D| - ||p||^2 \tag{11}$$

$$\mathtt{s.t.} \quad M_{\Theta^*} \models \phi, \tag{12}$$

$$\Theta^* \in \arg\min_{\Theta} [R'_L(D, p, \Theta) + \lambda\Omega(\Theta)], \tag{13}$$

$$s.t., g(\Theta) \le 0, h(\Theta) = 0. \tag{14}$$

Note that $R'_L(D, p, \Theta)$ is a reparameterization of $R_L(D', \Theta)$, where we use the fact that D' is obtained by perturbing D using p. This formulation of Data Repair can handle the case where we want certain p_i values to be 1, i.e., the case where we want to keep certain data points because we know they are reliable. Proposition 3 shows how we can solve the Data Repair problem.

Proposition 3. *Let us consider a probabilistic model M and a probabilistic temporal logic formula ϕ. If M is a parametric Markov Chain (MC) or parametric Markov Decision Process (MDP) and ϕ is expressed in Probabilistic Computational Tree Logic (PCTL), then the* `DataRepair` *problem in Definition 3, characterized by Equations 11-14, can be reduced to a set of non-linear optimization problems with non-linear rational constraints.* (Proof sketch in supplementary material.)

As outlined in Proposition 3, to solve the non-linear optimization formulation in Equations 11-14, we first solve the inner optimization in Equations 13-14 using maximum likelihood — this gives us a DTMC model $M(p)$, where the transition probabilities are *rational functions* of the data perturbation vector p. The outer optimization in Equations 11-12 can then be reformulated as:

$$\arg\max_{p} ||p||^2 \qquad \texttt{s.t.} M_p \models \phi. \tag{15}$$

This can be solved by using symbolic analysis, specifically parametric model checking, in combination with non-linear optimization. In this case, we are considering data points being removed — we can come up with similar formulations when we consider data points being added or replaced.

C. Reward Repair

The problem of learning reward function in MDPs from data is considered to be the Inverse Reinforcement Learning (IRL) [34] problem. We will consider the probabilistic model [34], where the probability of a trajectory is proportional to the exponential of the total reward along the trajectory in the MDP multiplied with the probability of the transitions along that MDP trajectory:

$$P(U|\Theta, P) = \frac{1}{Z(\Theta)} \exp(\sum_i (\Theta^T f_{s_i})). \prod_i P(s_{i+1}|s_i, a_i) \tag{16}$$

where s are the states, a are the actions, $U = (s_1, a_1) \dots (s_n, a_n)$ is a trajectory of state action pair sequences, i is the index along the path/trajectory, P is the transition distribution of the MDP, and $P_P(U)$ is the probability of observing the transitions in the trajectory U in the MDP. Additionally, f is a feature vector defined over each state, Θ is a weight vector, and the reward in a state is assumed to be a linear function of the features of that state, i.e., $reward(f_U) = \Theta^T f_U$, where $||\Theta||^2 \leq 1$.

In order to learn the reward function, we have to learn Θ (assuming f is known). Given a trace U, we maximize the log likelihood $L = P(U|\Theta)$ using the probability formulation given above, i.e., we find: $\Theta^* = P(U|\Theta)$, which gives the optimal reward. In the Reward Repair formulation, we additionally enforce that a given set of rules are satisfied along with likelihood maximization, i.e., we enforce that [13]: $E_Q[\phi_{l,g_l}(U)] = 1$, where g_l is the grounding of l-th rule $\phi_l(U)$. The rules ϕ are defined over the trajectory, and hence can be in any logic that can be interpreted over a trajectory, such as propositional, first-order, or linear temporal logic.

Q is the projection of P using updated reward function R' that satisfies the rule ϕ_l (where R' has the same state features f but a different weight vector Θ). We will use the projection approach [13] to project the MDP probability P to the subspace that enforces the satisfaction of the given rules to get Q. This will involve solving the following optimization problem:

$$\arg \min_{Q,\boldsymbol{\zeta}} KL(Q||P) + C||\boldsymbol{\zeta}||_1 \tag{17}$$

$$s.t. \quad \lambda_l[1 - E_Q[\phi_{l,g_l}(U)]] \leq \zeta_l, \zeta_l \geq 0, l = 1 \dots L \tag{18}$$

where the constraint enforces satisfaction of the rules, KL divergence minimizes the amount of difference between Q (defined over corrected reward R') and P (defined over optimal reward R^*) caused by correcting the optimal reward function R^* to R', λ_l and ζ_l are the importance weight and slack variable associated with satisfying the l^{th} rule, and C is the regularization parameter. Proposition 3 shows how we can solve the Reward Repair problem.

Proposition 4. *Consider a MDP M and a formula ϕ in propositional, first order or linear temporal logic. The* `RewardRepair` *problem in Definition 2, characterized by Equations 17-18, is optimized by choosing* $Q(U) = \frac{1}{Z}P(U)\exp(-\sum_{l,g_l} \lambda_l[1 - \phi_{l,g_l}(U)])$.

Equation 17 follows from the Posterior Regularizer formulation (Proposition 2.1 in [9]) and gives us the path probabilities Q using the corrected reward function R'. Note that this repair is intuitive. If for a set of groundings the rules are satisfied, then the argument of the exponent is 0. Hence, the corresponding Q for the path U is same as P. If the groundings do not satisfy a formula, then for large values of λ_l the argument of the exponent is $-\infty$ and consequently, the probability of that path is 0. Once we have the repaired Q function, we use it to estimate R' by using the trajectory probabilities corresponding to every transition in the model. For propositional rules ϕ, the groundings are provided by the values of the states and actions in the traces. For first order logic rules ϕ, we will have to consider groundings based on all possible trajectories U drawn from the MDP — this can be approximated by samples of trajectories drawn from the MDP using Gibbs sampling. For linear temporal logic, we pass the constraints through a parametric model checker like PRISM — that gives us the set of propositionalized constraints, which can then be used to estimate Q.

V. Case Studies

We outline two case studies —- Model/Data Repair in a MDP controller for query routing in a wireless sensor network, and Reward Repair in a MDP for automatic car control.

A. Query Routing Controller in Wireless Sensor Network: Model and Data Repair

We consider a wireless sensor network (WSN) arranged in a $n \times n$ grid topology (in our case-study $n = 3$). The $n = 3$ row corresponds to "field" nodes that are closer to the field of deployment, while $n = 1$ corresponds to "station" nodes that are closer to the base station — the goal is to route any message originating from a node to n_{11} via peer-to-peer routing in the minimum number of attempts, so that n_{11} can forward the message directly to the base station hub [10]. We model the network as a Markov Decision Process (MDP). Different node MDPs are connected through shared actions, e.g., the MDPs of nodes n_{21} and n_{22} are connected through the shared action f_{11_22} of forwarding a message from node n_{22} to node n_{11}. A node has a fixed probability f of forwarding a message to a neighboring node in the network, and a node-dependent probability of ignoring the message. A node can be in one of 3 states — (a) $S = 0$: on being forwarded a message, the node has decided to ignore the message; (b) $S = 1$: node has not ignored the message and is considering whether or not to forward it; (c) $S = 2$: node has forwarded the message. On the action of message forwarding from a neighbor, a node processes and forwards it to its neighbors probabilistically.

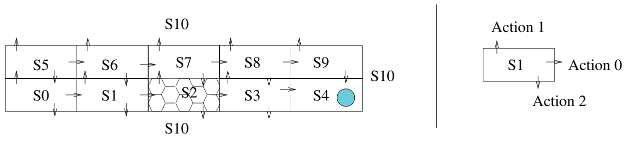

Fig. 1. Car safely overtaking a van: states S0-S10 and possible actions for states S0-S3 and S5-S9. S4 is target sink state, S10 is unsafe sink state, S2 is unsafe state (collision with van).

1) Model Repair in Wireless Sensor Network: The reward function of the WSN MDP is used to estimate the number of forwarding attempts to route a message from one end of the network to another. We assume that the reward corresponding to every forward attempt is 1.0 — the total reward counts the number of forwarding attempts necessary to route the message across the network. The structure of the MDP is decided by the grid structure of the network. The transition probabilities in the MDP model are learned using maximum likelihood estimation from message routing traces. We work through 3 different cases — in each case, we assume that the message (i.e., query) is initiated at the field node n_{33}, and the goal is to get the message to the station node n_{11}. In each case, we check if the learned MDP model satisfies the property $R\{attempts\} \leq X[F\ S_{n_{11}} = 2]$, where $R\{attempts\}$ is the cumulative reward function value for message forwarding attempts, and X indicates a particular number of message forwarding attempts.

Model satisfies property: Consider $X = 100$, i.e., we want to ensure that the MDP model can route a message from field node n_{33} to station node n_{11} under 100 attempts. PRISM indicates that the initial MDP model satisfies this property without any modifications.

Model Repair gives feasible solution: Consider $X = 40$: the original MDP model does not satisfy this property. We subsequently run parametric model checking of the model with the two parameters p and q, which are correction variables added to the ignore probabilities of field/station nodes and other nodes respectively (which are considered controllable in this formulation), and plug in the resulting non-linear equation into AMPL to get the solution $p = -0.045, q = -0.04$. So, the property is satisfied by the model if the node ignore probabilities are lowered, since in this case there is a higher chance of a node forwarding a message and hence the number of routing attempts is less.

Model Repair gives infeasible solution: Consider $X = 19$: in this case parametric model checking and non-linear optimization states this to be a "infeasible problem", which indicates that Model Repair cannot perturb the model in order to satisfy the property.

2) Data Repair in Wireless Sensor Network: We consider data traces of message forwarding and traces of query dropping (ignoring) in n_{11} and a node near the message source, viz., n_{32}. Let us consider that 40% of the traces involving message forwarding have a successful forward, while 60% do not. If we assign probabilities p_1 and p_2 of dropping those 2 trace types respectively, we get that the maximum likelihood forwarding probability = $0.4/(0.4 + 0.6p)$, where $p = p_2/p_1$. Using a similar approach, we get that the ignore probabilities for $n_{11} = 0.5/(0.5 + 0.5q)$, and for node $n_{32} = 0.5/(0.5 + 0.5r)$. When we run parametric model checking in PRISM for the model with these Data Repair transition values, we get a non-linear equation for the property $R\{attempts\} \leq 19[F\ S_{n_{11}} = 2]$, which are solved in AMPL to get the values $p = 0.00001, q = 18.8129, r = 18.813$ — with these data corrections, the model learned on the corrected data satisfies the property.

B. Obstacle Avoidance Controller in Autonomous Vehicle: Reward Repair

In this example, we consider a scenario where a car needs to avoid an obstacle by switching to left lane and eventually return back to the right lane at a safe distance ahead of the van. Figure 1 illustrates the scenario. The current location of the car is S0. The states S0-S4 correspond to the right lane and S5-S9 correspond to left lane. The state S2 corresponds to collision with the obstacle, and hence it is unsafe. The state S10 corresponds to going off road or not returning to the right lane by S4, and hence S4 is also marked as unsafe. Each state in S0-S3 and S5-S9 has three possible actions: action 0 corresponds to moving forward, action 1 corresponds to changing lane to left and action 2 corresponds to changing lane to right. Once the car reaches state the unsafe state S10, it remains there. The state S4 is a sink state since it marks the end of the maneuver.

We can represent each state using three features: ϕ_1 corresponding to the lane to which the state belongs, ϕ_2 corresponding to the distance from the nearest unsafe state and ϕ_3 corresponding to whether the state is the target sink state S4. We are given the following expert policy: (S0, 0),(S1,1),(S6,0),(S7,0),(S8,2),(S3,0),(S4,0). Using max entropy inverse reinforcement learning, we learn the reward: $reward(Si) = 0.38\phi_1(Si) + 0.29\phi_2(Si) + 1.21\phi_3(Si)$. The corresponding optimum deterministic policy is: (S0, 1), (S1, 0), (S2, 0), (S3, 0), (S4, 0), (S5,0), (S6, 0), (S7, 0), (S8, 0), (S9, 2), (S10, 0). This policy is unsafe since the action 0 in state S1 would lead the car to state S2, that is, collide with the van. Next, we try to repair the reward function to eliminate reaching the unsafe state S2. In order to do so, we solve the following optimization problem where Q is the state-action value function corresponding to the reward: $\min |\theta^* - \theta|, s.t., Q(S1, 1) > Q(S1, 0)$. We obtain the corresponding repaired reward functions as: $reward(Si) = 0.38\phi_1(Si) + 0.39\phi_2(Si) + 1.21\phi_3(Si)$. The corresponding optimal policy is: $(S0, 1), (S1, 1), (S2, 0), (S3, 0), (S4, 0), (S5, 0), (S6, 0),$ $(S7, 0), (S8, 2), (S9, 2), (S10, 0)$. This policy avoids going to unsafe states — the repaired reward function ensures safety.

VI. Related Work

Machine learning (ML) has a rich history of learning under constraints [6], [22] — different types of constrained learning algorithms have been proposed. Propositional constraints on size [3], monotonicity [16], time and ordering [18], probabilities [25], etc. have been incorporated into learning algorithms using constrained optimization [5] or constraint programming [28], while first order logic constraints have also been introduced into ML models [20], [29]. However, to the best of our knowledge, temporal logic constraints have not been incorporated into ML models before. There has been work on training models that capture dynamical/temporal behavior, e.g., DBNs [23], LSTMs [12], and also efforts in learning temporal logic relations [7], [19]. Sadigh et al. [30] study the problem of human driver behavior using Convex Markov Chains, and show how we can verify PCTL properties for these models. [26] show how Convex MDPs can be modified to satisfy PCTL formulas. However, these methods follow techniques different from Model and Data Repair. We would also like to explore connections between TML and probabilistic CEGAR and CEGIS algorithms [11].

The closest related work to Reward Repair is reward shaping – that's the formulation where intermediate rewards are specified for subgoals, to help the MDP learn reward functions for a complex task while keeping the optimal policy unchanged [24]. Reward shaping has been shown to be related to Q-value initialization [32], and is also a way to incorporate background knowledge into model-free RL algorithms for MDP [2]. The reward shaping work provides direct suggestions about the reward function to the model. In contrast, the Reward Repair formulation enforces that certain domain-level logical constraints are satisfied by the RL algorithm. Another related work is Constrained Policy Optimization [1] — this is defined over Constrained Markov Decision Processes, where constraints are defined on expectations of auxilliary costs (as compared to our formulation, which can handle logical constraints).

VII. Conclusions and Future Work

We have developed TML techniques of Model Repair, Data Repair and Reward Repair in MDP models, which ensure that the repaired models satisfy specified logical properties. In this paper, we focused on MDPs since they are commonly used to model controllers. Other types of dynamic models (e.g., probabilistic timed automata) can also be handled by our approach. For other probabilistic models that have hidden states (e.g., Hidden Markov Models, Dynamic Bayes Nets), we can incorporate the temporal constraints into the E-step of an EM algorithm for parameter learning. In the future, we would also like o extend TML to other types of logical properties (e.g., Linear Temporal Logic), other mission-critical domains (e.g., cyber security), and non-probabilistic models (e.g., SVM). We will also focus on more scalable repair algorithms, e.g., using efficient localized changes, or as the underlying verification techniques and optimization procedures improve with time.

VIII. Acknowledgment

This work was funded in part by the US National Science Foundation (NSF) under award numbers CNS-1750009, CNS-1740079, CNS-1314956 and US ARL Cooperative Agreement W911NF-17-2-0196. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF and ARL.

References

[1] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *ICML*, 2017.

[2] J. Asmuth, M. L. Littman, and R. Zinkov. Potential-based shaping in model-based reinforcement learning. In *AAAI*, 2008.

[3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *JMLR*, 6, 2005.

[4] E. Bartocci, R. Grosu, P. Katsaros, C. R. Ramakrishnan, and S. A. Smolka. Model repair for probabilistic systems. In *Tools and Algos. for Construction and Analysis of Systems*, 2011.

[5] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Opt. and Neural Comp. Series)*. Athena Scientific, 1996.

[6] T. G. Dietterich. *Constraint Propagation Techniques for Theory-driven Data Interpretation (AI, ML)*. PhD thesis, Stanford Univ., 1985.

[7] A. Fern, R. Givan, and J. M. Siskind. Specific-to-general learning for temporal events with application to learning event definitions from video. *CoRR*, abs/1106.4572, 2011.

[8] R. Fourer, D. Gay, and B. Kernighan. Algos. and model formulations in mathematical programming. chapter AMPL: A Mathematical Programming Language. 1989.

[9] K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *JMLR*, 11, 2010.

[10] S. Ghosh and P. D. Lincoln. Query routing in wireless sensor networks: A novel application of social query models. Technical Report SRI-CSL-12-01, SRI Intl., 2012.

[11] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *CAV*, 2008.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8), 1997.

[13] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. In *ACL*, 2016.

[14] S. Jha and V. Raman. Automated synthesis of safe autonomous vehicle control under perception uncertainty. In *NFM*, 2016.

[15] S. Jha, V. Raman, D. Sadigh, and S. A. Seshia. Safe autonomy under perception uncertainty using chance-constrained temporal logic. *Journal of Automated Reasoning*, 60(1), 2018.

[16] W. Kotlowski and R. Słowiński. Rule learning with monotonicity constraints. In *ICML*, 2009.

[17] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, 2011.

[18] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *CVPR*, 2007.

[19] F. M. Maggi, A. Burattin, M. Cimitile, and A. Sperduti. Online process discovery to detect concept drifts in LTL-based declarative process models. In *OTM*, 2013.

[20] S. Mei, J. Zhu, and X. Zhu. Robust RegBayes: Selectively incorporating first-order logic domain knowledge into Bayesian models. In *ICML*, 2014.

[21] S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, 2015.

[22] K. D. Miller and D. J. C. MacKay. The role of constraints in Hebbian learning. *Neural Comp.*, 6(1), 1994.

[23] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice-Hall, 2003.

[24] A. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and appln. to reward shaping. In *ICML*, 1999.

[25] T. Papai, S. Ghosh, and H. A. Kautz. Combining subjective probabilities and data in training Markov logic networks. In *ECML*, 2012.

[26] A. Puggelli, W. Li, A. Sangiovanni-Vincentelli, and S. A. Seshia. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *CAV*, 2013.

[27] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1st edition, 1994.

[28] L. D. Raedt, T. Guns, and S. Nijssen. Constraint programming for data mining and machine learning. In *AAAI*, 2010.

[29] M. Richardson and P. Domingos. Markov logic networks. In *Machine Learning*, 2006.

[30] D. Sadigh, K. Driggs-Campbell, A. Puggelli, W. Li, V. Shia, R. Bajcsy, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia. Data-driven probabilistic modeling and verification of human driver behavior. In *Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symp.*, 2014.

[31] K. Sen, M. Viswanathan, and G. Agha. *Tools and Algos. for the Construction and Analysis of Systems*, chapter Model-Checking Markov Chains in the Presence of Uncertainties. 2006.

[32] E. Wiewiora. Potential-based shaping and q-value initialization are equivalent. *CoRR*, abs/1106.5267, 2011.

[33] X. Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, 2015.

[34] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.