# Trusted Machine Learning: Model Repair and Data Repair for Probabilistic Models

**Shalini Ghosh, Patrick Lincoln, Ashish Tiwari, Xiaojin Zhu**
{shalini,lincoln,tiwari}@csl.sri.com jerryzhu@cs.wisc.edu

**Abstract**

When machine learning algorithms are used in life-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 correctness guarantees. We introduce a paradigm called Trusted Machine Learning (TML) with the goal of making learning techniques more trustworthy. We outline methods that show how symbolic analysis (specifically parametric model checking) can be used to learn the dynamical model of a system where the learned model satisfies correctness requirements specified in the form of temporal logic properties (e.g., safety, liveness). When a learned model does not satisfy the desired guarantees, we try two approaches: (1) Model Repair, wherein we modify a learned model directly, and (2) Data Repair, wherein we modify the data so that re-learning from the modified data will result in a trusted model. Model Repair tries to make the minimal changes to the trained model while satisfying the properties, whereas Data Repair tries to make the minimal changes to the dataset used to train the model for ensuring satisfaction of the properties. We show how the Model Repair and Data Repair problems can be solved for the case of probabilistic models, specifically Discrete-Time Markov Chains (DTMC) or Markov Decision Processes (MDP), when the desired properties are expressed in Probabilistic Computation Tree Logic (PCTL). Specifically, we outline how the parameter learning problem in the probabilistic Markov models under temporal logic constraints can be equivalently expressed as a non-linear optimization with non-linear rational constraints, by performing symbolic transformations using a parametric model checker. We illustrate the approach on two case studies: a controller for automobile lane changing, and query router for a wireless sensor network.

## 1 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). Such constraints are often defined over temporal sequences (or trajectories) generated by the model, and can be formally described in so-called temporal logics, such as the Probabilistic Computation Tree Logic (PCTL)(Sen, Viswanathan, and Agha 2006). *Trusted Machine Learning* (TML) refers to a learning methodology that ensures that the specified properties are satisfied.



We illustrate the need for TML by considering an automatic lane change controller trained to handle a slow-moving truck in front, which is part of an overall closed-loop autonomous car controller. We provide training data ($D$) in the form of example traces of driving in this situation in a simulated environment, and learn a Markov Chain model $M$ of the controller from $D$ (Sen, Viswanathan, and Agha 2006). In our application, we want the lane change controller to respect certain safety properties. For example, we want the controller to ensure that detecting a slow-moving truck in front causes the car to either change lanes or reduce speed with very high probability. This can be stated as a PCTL property $\Pr_{>0.99}[F(changedLane\ or\ reducedSpeed)]$, where $F$ is the *eventually* operator in PCTL logic (Section 3 discusses how in real applications we would use bounded-time variants of such a property). Note that PCTL is a probabilistic temporal logic used widely for formally specifying temporal properties like safety (a property should always be true) and liveness (a property should eventually become true) and their combinations — such temporal logic properties are useful for specifying important requirements of models used in mission-critical cyber-physical system (CPS) domains, e.g., self-driving cars, cyber security.

In this paper, we show how to ensure that a ML model trained on data (e.g., the one used in the car controller) satisfies temporal logic constraints. Putting temporal logic properties directly into learning algorithms (e.g., maximum likelihood) may not be feasible in the general case. So, we propose two approaches for TML, Model Repair and Data Repair — Model Repair modifies a trained model to satisfy a property, while Data Repair modifies the data so that re-learning from the modified data results in a trusted model. We consider two aspects of a trusted ML model — safety and security. Let us consider that a set of safety properties define the *safety envelope* of a ML model. During model training, Model Repair can be used to ensure that the trained model satisfies the safety properties and stays within the safety envelope. When the model undergoes data attack, 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 still within the safety envelope. Thus, Data Repair can help us detect violations of the safety properties and ensure that the model is resilient to security attacks.

The key contributions of this paper are:

1. We present Model Repair and Data Repair as two possible approaches for Trusted Machine Learning (TML). Model Repair is adapted from (Bartocci et al. 2011), but it generalizes the original approach (that considered only small repairs) to handle larger model corrections. We also propose a novel Data Repair approach for TML, and show how it can be converted to a standard constrained non-linear optimization problem.
2. In the particular case when the learned model is either a Markov Chain (MC) or a Markov Decision Process (MDP), and the property is specified in PCTL, we theoretically characterize the Model Repair and Data Repair problems and show how they can be solved efficiently using symbolic analysis (in particular parametric model checking), specifically by expressing parameter learning under temporal logic constraints as non-linear optimization with non-linear rational constraints.
3. We present case-studies showing actual applications of Model Repair and Data Repair in the domains of automatic car control (using MC) and wireless sensor networks (using MDP).

As we will show in this paper, the main challenge in putting PCTL constraints into the ML training procedure is in reducing the Model and Data Repair problems to an optimization framework where complex temporal logic properties (e.g., safety, liveness) can be satisfied while keeping the optimization problems feasible and tractable.

## 2 Problem Definition

**Notation:** Let $\mathcal{M}$ be a class of models (concept class) and $\mathcal{D}$ be the universe of all data sets. Let $\texttt{ML}$ be a machine learning procedure that takes $D \in \mathcal{D}$ and returns $M \in \mathcal{M}$. Let $\phi$ be a desired (temporal logic) property that we want the learned model to possess. We denote the fact that a model $M$ has the property $\phi$ by $M \models \phi$. If a model $M = \texttt{ML}(D)$ (trained on some data set $D$) does not satisfy the required (temporal logic) property $\phi$, then we want to either repair the model $M$ or the data set $D$. We do not consider arbitrary "repairs" but only certain ones that are identified by some given constraints. Given $M$, let $\texttt{Feas}_M \subseteq \mathcal{M}$ denote all "feasible repairs" of $M$. Given $D$, let $\texttt{Feas}_D \subseteq \mathcal{D}$ denote all "feasible repairs" of $D$. Let $c(M, M')$ denote the cost (a positive real function) of changing $M$ to $M'$, and $c(D, D')$ denote the cost of changing $D$ to $D'$. Based on these notations, we propose the following two definitions of `ModelRepair` and `DataRepair`.

**Definition 1.** `ModelRepair`: *Given $M$, $\phi$, $\texttt{Feas}_M$, and the function $c$, the Model Repair problem seeks to find $M^* \in \mathcal{M}$ such that $M^* \in \texttt{Feas}_M$ and $M^* \models \phi$, and $M^*$ minimizes the cost $c(M, M^*)$.*

**Definition 2.** `DataRepair`: *Given $D$, $\phi$, $\texttt{Feas}_D$, and the function $c$, the Data Repair problem seeks to find $D^* \in \mathcal{D}$ such that $D^* \in \texttt{Feas}_D$, $\texttt{ML}(D^*) \models \phi$, and $D^*$ minimizes the cost $c(D, D^*)$.*

Given a dataset $D$, we first learn a model $M = \texttt{ML}(D)$ using a learning procedure $\texttt{ML}$ (e.g., maximum likelihood training). We then check if $M \models \phi$; if it does, we output $M$. Otherwise, if we want to modify the model, we run Model Repair on $M$ to get $M'$, where Model Repair makes small perturbations to $M$. If $M' \models \phi$, we output $M'$. Else, if the Model Repair formulation we consider has no feasible solution, we relax the "small perturbations" constraint on Model Repair (and thereby consider a larger feasible set $\mathcal{M}$), to see if that enables the modified model to satisfy the property. If we want to modify the data, we perform Data Repair of the data $D$ using small perturbations on the data to get $D'$ and check if $M'' = \texttt{ML}(D') \models \phi$. If it does, we output $M''$. Otherwise, we relax the constraint of "small perturbations" on the data, retrain the model and check if the retrained model satisfies the constraint. If it doesn't, we report that $\phi$ cannot be satisfied by the learned model using our formulations of Model Repair or Data Repair.

In this paper, we will solve the two problems outlined above for specific choices of the concept class ($\mathcal{M}$), the possible modifications ($\texttt{Feas}_M, \texttt{Feas}_D$), and the property language (used to specify $\phi$). In particular, we will consider $\mathcal{M}$ to be probabilistic models like Discrete-Time Markov Chains (DTMC) or Markov Decision Processes (MDP), and $\phi$ to be PCTL. The particular types of modifications we will consider (corresponding to $\texttt{Feas}_M$ and $\texttt{Feas}_D$) are outlined in detail in Section 4.

## 3 Motivating Example

Let us consider the example of training a probabilistic model for an autonomous car controller — the underlying model we want to learn is a DTMC, which is defined as a tuple $M = (S, s_0, P, L)$ where $S$ is a finite set of states, $s_0 \in S$ is the initial state, $P : S \times S \to [0, 1]$ is a transition matrix such that $\forall s \in S, \sum_{s' \in S} \Pr(s, s') = 1$, and $L : S \to 2^{AP}$ is a labeling function assigning labels to states, where $AP$ is a set of atomic propositions. Figure 1 shows a DTMC that determines the action of an autonomous car controller when confronted with a slow-moving truck in front, while Figure 1 outlines the semantics of the different states and labels. We will later perform Model Repair by finding instantiations for the parameters $p$ and $q$ in the model (details in Section 4).

Let us consider that we have some property $\phi$ in PCTL that we want the model $M$ to satisfy. In PCTL, properties are specified as $\phi = \Pr_{\sim b}(\psi)$, where $\sim \in \{<, \leq, >, \geq\}$, $0 \leq b \leq 1$, and $\psi$ 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 $\phi$ 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 $\psi$ is $\sim b$, where path is defined as a sequence of states in the model $M$. For example, let us consider a DTMC $M$ in Figure 1 with start state $S_0$. The "reduced-Speed" predicate is true in the state $S_4$, while "changed-

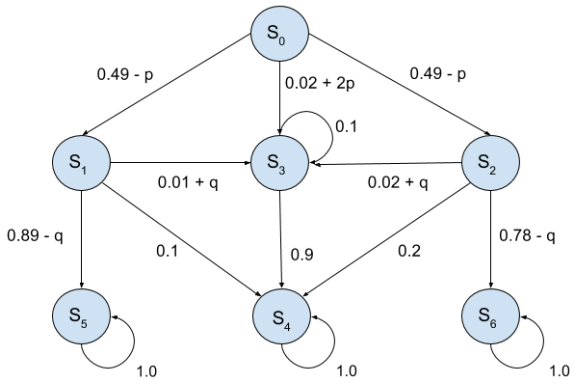


Figure 1: Car controller DTMC.

| S | Description | Labels |
|---|---|---|
| 0 | Initial state | keepSpeed, keepLane |
| 1 | Moving to left lane | keepSpeed, changingLane |
| 2 | Moving to right lane | keepSpeed, changingLane |
| 3 | Remain in same lane with same speed | keepSpeed, keepLane |
| 4 | Remain in same lane with reduced speed | reducedSpeed, keepLane |
| 5 | Moved to left lane | changedLane |
| 6 | Moved to right lane | changedLane |

Table 1: DTMC: states and labels.

Lane" predicate is true in states $S_5$ and $S_6$. Now, a probabilistic safety property can be specified as: $\phi = \Pr_{>0.99}[F(changedLane\ or\ reducedSpeed)]$. $M$ will satisfy the property $\phi$ only if any path starting from $S_0$ eventually reaches $S_4, S_5$ or $S_6$ with probability $> 0.99$.

Note that the DTMC lane change controller considered here is part of an overall autonomous closed-loop car controller. The lane changing DTMC module is triggered when the car gets too close to a vehicle in front, and hence it is a feedback controller. These controllers can be learned from car traces in a vehicle simulator (Sadigh et al. 2014). The "eventually change lanes or slowdown" property here illustrates a "liveness" property — a real controller would use bounded-time variants of such a property.

## 4 Repairing Probabilistic Models

In this paper, we consider the model $M$ to be a probabilistic model — specifically, we consider DTMC and MDP. Specifically, the class $\mathcal{M}$ consists of all DTMCs (or MDPs) with a fixed (graph) structure, but different transition probabilities. The property $\phi$ is a temporal logic property in PCTL. Efficient solvability of the two problems defined above depends also on the choice of the "allowed repairs", given by the subclass $\texttt{Feas}_M$ or $\texttt{Feas}_D$.

For Model Repair, we consider the subclass $\texttt{Feas}_M$ 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 — the domain helps determine which transition probabilities are perturbable and which are not (e.g., which part of the car controller can be modified).

For Data Repair, the subclass $\texttt{Feas}_D$ 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 — the number of datapoints dropped from the total dataset defines the neighborhood of corrections. When we run Model Repair or Data Repair, we first consider a small correction (to the model or data, respectively) — if that does not succeed, we relax the restriction and consider larger corrections.

### Model Repair

We first present an approach for solving Model Repair for probabilistic models. We use the DTMC in Section 3 as our running example. Given a DTMC $M$ with $n$ states and $n \times n$ transition matrix $P$, we can get a parametric DTMC $M_Z$ 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 DTMC $M_Z$ along with the constraints on $Z$ defines the set $\texttt{Feas}_M$, as discussed in Proposition 1.

**Proposition 1.** (Bartocci et al. 2011) *If $M$ is a DTMC with transition matrix $P$ and $M_Z$ is a DTMC with transition matrix $P + Z$, and $\forall s \sum_{t \in S} Z(s,t) = 0$, then $M$ and $M'$ are $\epsilon$-bisimilar, where $\epsilon$ is bounded by the maximum value in $Z$.*

Note that $M' \in \texttt{Feas}_M$ and $M$ are $\epsilon$-bisimilar when there exists an $\epsilon$-bisimulation between them, i.e., any path probability in $M'$ is within $\epsilon$ of the corresponding path probability in $M$. Figure 1 shows an example of a parametric DTMC.

Let us consider the non-zero values in $Z$ to be the vector of variables $\mathbf{v} = v_1 \ldots v_k$. We solve the Model Repair problem by solving the following optimization problem:

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

$$\texttt{s.t.}, M_Z \models \phi, \quad (2)$$

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

In Equation 1, $g(\mathbf{v})$ 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., $g(\mathbf{v}) = ||\mathbf{v}||^2 = v_1^2 + \ldots + v_n^2$. The main bottleneck in the Model Repair formulation in Equations 1-3 is the constraint in Equation 2 — it is a temporal logical constraint, which is difficult to directly handle in a non-linear optimization problem. Proposition 2 shows how we can transform the

above optimization problem with a "non-standard" temporal logical constraint to a standard non-linear optimization problem with non-linear constraints.

**Proposition 2.** *Let us consider a probabilistic model $M$ and a probabilistic temporal logic formula $\phi$. If $M$ is a parametric Discrete-Time Markov Chain (DTMC) or parametric Markov Decision Process (MDP) and $\phi$ 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:* We give a brief proof sketch of Proposition 2.

**Markov Chain:** Given a parametric DTMC, properties that are specified in PCTL, either using intrinsic relations like $X$ (next) and $U$ (until) or operators like $F$ (eventually) or $G$ (always)[1], can be mapped to the parametric reachability problem in the parametric DTMC (Ogawa, Nakagawa, and Tsuchiya 2015). So, any property specified in PCTL logic can be equivalently expressed as a reachability problem to a set of target states. The probability of reaching a set of target states in a parametric DTMC can be computed in the form of a rational function in the parameters using a parametric model checker (Hahn, Hermanns, and Zhang 2011). The main technique used by the parametric model checker is as follows: in order to succinctly express the probability of whether a target state is reachable along a path in terms of a closed-form rational function, it successively considers state sequences in the path of the form $s_1 \rightarrow s \rightarrow s_2$ and eliminates $s$ from the path after suitably accounting for the probability of self-transition of $s$ in a direct path from $s_1$ to $s_2$. In the path $s_1 \rightarrow s \rightarrow s_2$, let the probabilities of $s_1 \rightarrow s, s \rightarrow s, s \rightarrow s_2$ be $\Pr(s_1, s), \Pr(s, s), \Pr(s, s_2)$ respectively. We want to eliminate $s$ and replace the path by a direct connection between $s_1$ and $s_2$. Let $\Pr'(s_1, s_2)$ be the probability of transitioning from $s_1$ to $s_2$ along any other path. If we remove $s$ and consider a direct edge from $s_1$ to $s_2$, the transition probability of this new edge can be equivalently expressed as $\Pr'(s_1, s_2) + \Pr(s_1, s).\Pr(s, s_2)/(1 - \Pr(s, s))$, which is a rational function in the parameters of the model. Using successive state elimination, the parametric model checker is able to get an overall non-linear rational function constraint corresponding to the reachability property in Equation 2 (Hahn, Hermanns, and Zhang 2011) — as a result, `ModelRepair` in Equations 1-3 can be expressed as the non-linear optimization problem in Equations 4-6.

**Markov Decision Process:** Let us consider a parametric Markov Decision Process $M = (S, s_0, Act, P_p, V)$, where $S, s_0$ are same as a DTMC, $V$ is the vector of correction parameters as outlined in Section 4, and $Act$ is a finite set of actions of the Markov Decision Process (Puterman 1994) — in a parametric MDP the transition function $P$ is repaired using the parameters $V$. Given a parametric MDP model, we can reduce it to an equivalent parametric DTMC where each state is created by considering a corresponding (state, action) pair in the MDP. There exists a decomposition of the parameter space of the parametric MDP into hyper-rectangles such that the problem of probabilistic reachability in the equivalent parametric DTMC for each hyper-rectangle corresponds to a rational function in the relevant parameters (Hahn, Han, and Zhang 2011). So for a MDP, we finally get a disjunction of rational functions equivalent to a probabilistic reachability property specified in PCTL. For the disjunct of each rational function constraint (corresponding to each parameter hyper-rectangle), we can separately solve the corresponding `ModelRepair` non-linear optimization problem — a suitable combination of the solutions of the different non-linear optimization problems (e.g., $\min$ for $\arg\min$ problems) will give us the final result. So for an MDP, the `ModelRepair` problem can be equivalently expressed as a set of disjoint non-linear optimizations problems. □

[1]Where $G$ is defined as follows: $G\phi = \neg(F\neg\phi)$.

As outlined in Proposition 2, if $M$ is a DTMC or MDP, parametric model checking can convert Equations 1-3 to the following constrained optimization problem:

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

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

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

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 $\phi$ in $M$, to the non-linear equation $f(v)$ in Equation 5 can be obtained using a parametric model checker, e.g., PRISM (Kwiatkowska, Norman, and Parker 2011). 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 (Fourer, Gay, and Kernighan 1989). If the nonlinear optimization problem has a feasible solution, it gives us the optimal values of $Z$ that makes the resulting model $M^*$ satisfy $\phi$.

## Data Repair

In some cases, we try to modify the dataset $D$ to $D'$ so that the model trained on $D'$ satisfies $\phi$. For this we need to solve the Data Repair problem (Definition 2) and is a variant of *machine teaching* (Zhu 2015).

Based on the machine teaching formulation (Mei and Zhu 2015), the Data Repair problem can be formalized as:

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

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

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

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

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 $\Omega$. $E_T$ is the teaching "effort" function of modifying the dataset $D$ to $D'$, $M_{\Theta^*}$ indicates a model that is parameterized by $\Theta^*$, while $g$ and $h$ are other domain constraints.

Let us consider that the dataset $D$ is transformed to $D'$ using a data perturbation vector $p$. In this paper, we consider that a subset of data points need to be dropped from $D$ for the resulting trained model to satisfy $\phi$ (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:

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

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

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

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

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 $\phi$. If $M$ is a parametric Markov Chain (MC) or parametric Markov Decision Process (MDP) and $\phi$ is expressed in Probabilistic Computational Tree Logic (PCTL), then the* `DataRepair` *problem in Definition 2, characterized by Equations 11-14, can be reduced to a set of non-linear optimization problems with non-linear rational constraints.*

*Proof sketch:* We give a brief proof sketch of Proposition 3.

**Markov Chain:** Let us consider a parametric DTMC $M$. Let us also consider the data dropping model of `DataRepair`, where each data point $d_i$ is multiplied by the corresponding parameter $p_i$. The maximum likelihood estimate of the transition probability between any pair of states $s_1$ to $s_2$ in $M$ would be the ratio of the counts of data points having that model transition, normalized by the total count of data points transitioning out of $s_1$. So, each transition probability in the DTMC model can be expressed as the ratio of polynomials in $p$, i.e., a rational function in $p$. As a result, Equations 11-14 can be equivalently expressed as Equations 15-16, which can be then transformed to a non-linear optimization problem with non-linear constraints.

**Markov Decision Process:** For parametric Markov Decision Process (MDP) models, we assume that the states, actions and transitions structure are given — we learn the transition probabilities. Note that we can either assume that the reward function is provided for all state-action pairs, or we can learn it using inverse reinforcement learning since it does not depend on the transition probabilities. For MDP models too, we can show that the maximum likelihood transition probabilities are rational functions of the data

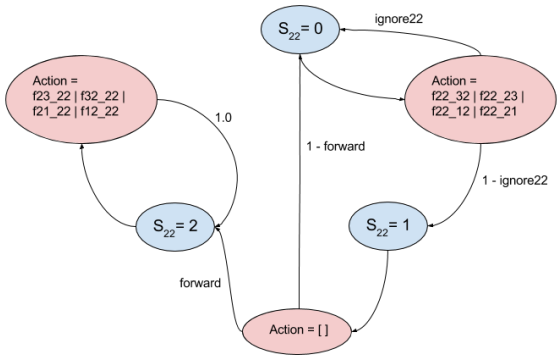


Figure 2: Subset of WSN MDP corresponding to node $n_{22}$ for Model Repair.

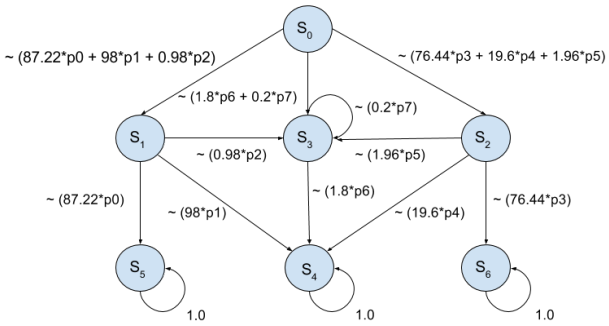


Figure 3: Transition weights in DTMC under Data Repair.

repair parameters, so that Equations 11-14 reduces to a set of non-linear optimization problems of the form Equations 15-16. □

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:

$$\arg \quad \max_{p} ||p||^2 \tag{15}$$

$$\texttt{s.t.} \quad M_p \models \phi. \tag{16}$$

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 for Data Repair when we consider data points being added or replaced, or data features or labels being modified.

## 5 Case Studies

### A) Car Controller: DTMC

We first outline Model Repair and Data Repair examples for the car controller Markov Chain outlined in Section 3.

**Model Repair in Car Controller** We will use the running example of the car-controller described in Section 3. We assume that the initial values of the DTMC model in Figure 1 are learned using maximum likelihood estimation from simulated car traces. We work through 4 different cases:

**Model satisfies property:** Consider the property `Prop1` $= \Pr_{>0.99}\ [F(changedLane\ or\ reduceSpeed)]$, as described in Section 1. We consider a model $M$ in PRISM corresponding to Figure 1, where we consider two controllable correction variables: $p$ and $q$. Initially we consider $p = q = 0$, i.e., we do not consider any Model Repair — in this case, $M$ satisfies `Prop1` without the need for any repair.

**Model Repair gives feasible solution:** Consider the property `Prop2` $= \Pr_{>0.8}[F(reduceSpeed)]$. When we run PRISM on $M$ with $p = q = 0$ for `Prop2`, PRISM reports that the property is not satisfied from the initial state $S_0$. We run parametric model checking on $M$, which converts `Prop2` to a non-linear parametric equation: $0.8 < 0.0001(-20000qp + 16700p + 9800q + 1817)$. Using this as a constraint in AMPL, we minimize the objective: $p^2 + q^2$ and get the Model Repair solution $p = 0.3609, q = 0.0601$.

**Model Repair initially gives infeasible solution:** Consider the property `Prop3` $= \Pr_{<0.1}[F(reduceSpeed)]$. Parametric model checking and non-linear optimization states this to be a "infeasible problem", which indicates that Model Repair cannot perturb $M$ in order to satisfy `Prop3`. So, Model Repair fails to find a solution in the $\epsilon$-neighborhood of the original model $M$ to satisfy `Prop3`, considering the correction parameters $p$ and $q$. Subsequently, we consider relaxing the correction parameters beyond $p$ and $q$.

**Model Repair gives feasible solution with more parameters:** We increase the number of correction parameters in the model in Figure 1, such that each edge in the DTMC has an associated perturbation variable. As expected, with the extra degrees of freedom Model Repair is able to make make $M$ satisfy `Prop3`.

**Data Repair in Car Controller** We assume that the structure of the Markov Chain in Example 3 is given, and the training data is used to learn the transition probabilities. So, we consider that the data is available in the form of traces of states We consider a sample dataset with 8 points for the case study, where each data point is a tuple of the form $(p_i, d_i, w_i)$ — a data trace $d_i$ has an associated data dropping probability $p_i$ of dropping the trace from the training set, and a weight $w_i$ that indicates the weight given to that data trace during training the model: { (p0, (0,1,5), 87.22), (p1, (0,1,4), 98), (p2, (0,1,3), 0.98), (p3, (0,2,6), 76.44), (p4, (0,2,4), 19.6), (p5, (0,2,3), 1.96), (p6, (0,3,4), 1.8), (p7, (0,3,3), 0.2) }. Figure 3 shows the transition weights of the DTMC, estimated from the training data using maximum likelihood — in the model, the transition weights are normalized so that the output probabilities out of each state sum to 1.0.

**Data Repair gives feasible solution:** We consider the property `Prop3` $= \Pr_{<0.1}[F(reduceSpeed)]$. We run parametric model checking with PRISM on this model $M$ to get the following non-linear equation corresponding to the property: $(4900p1 + 49p2 + 980p4 + 98p5 + 90p6 + 10p7)/(4361p0 + 4900p1 + 49p2 + 3822p3 + 980p4 + 98p5 + 90p6 + 10p7)$. On running the AMPL non-linear solver, we get the following solution set for Data Repair: $p0 = 1, p1 = 0.18, p2 = 0.00001, p3 = 1, p4 = 0.00001, p5 = 0.00001, p6 = 0.00001, p7 = 1$. $p0$, $p3$ and $p7$ are kept unchanged by Data Repair, implying that the probabilities of data traces corresponding to right lane change, left lane change and staying in the same lane without reducing speed are kept the same, while the probabilities of the other data traces are reduced. The probability of reducing the speed is brought down, thereby satisfying the property.

**Data Repair becomes infeasible with less parameters:** Note that if we restrict the number of allowable data perturbations such that $< 20\%$ of the data points can be dropped, then Data Repair fails to give a feasible solution. So, we need $\geq 20\%$ of the data to possibly change, in order to satisfy the property using Data Repair.

## B) Wireless Sensor Network: MDP

We next outline Model Repair and Data Repair case studies on a Markov Decision Process (MDP) model for query routing in wireless sensor networks (WSN).

**Query Routing in Wireless Sensor Network** 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 (Ghosh and Lincoln 2012). We model the network as a Markov Decision Process (MDP). Figure 2 shows the sub-component of the overall MDP centered around the states and action-related transitions corresponding to one node in the network (node $n_{22}$ in the Figure). 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.

**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, and each node has a state/action transition diagram similar to Figure 2. 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.

**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. We verified that Data Repair indeed worked by plugging in these values into the model and checking that the property is satisfied by Data Repair.

## 6 Related Work

Machine learning (ML) has a rich history of learning under constraints (Dietterich 1985; Miller and MacKay 1994) — different types of learning algorithms have been proposed for handling various kinds of constraints. Propositional constraints on size (Bar-Hillel et al. 2005), monotonicity (Kotlowski and Slowiński 2009), time and ordering (Laxton, Lim, and Kriegman 2007), etc. have been incorporated into learning algorithms using constrained optimization (Bertsekas 1996) or constraint programming (Raedt, Guns, and Nijssen 2010), while first order logic constraints have also been introduced into ML models (Mei, Zhu, and Zhu 2014; Richardson and Domingos 2006). 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 (Neapolitan 2003), LSTMs (Hochreiter and Schmidhuber 1997), and also efforts in learning temporal logic relations (Fern, Givan, and Siskind 2011; Maggi et al. 2013). Sadigh et al. (Sadigh et al. 2014) study the problem of human driver behavior using Convex Markov Chains, and show how we can verify PCTL properties for these models. (Puggelli et al. 2013) 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 (Hermanns, Wachter, and Zhang 2008).

## 7 Conclusions and Future Work

Our approach to Trusted Machine Learning (TML) uses symbolic analysis (specifically parametric model checking) for learning ML models that satisfy properties in temporal logic. We have developed the techniques of Model Repair and Data Repair for DTMC and MDP models satisfying PCTL properties, and discussed a possible application of our approach to two domains (car controller and sensor networks).

In this paper, we focused on DTMCs and MDPs since they are commonly used to model controllers. Other types of models (e.g., continuous-time Markov models, 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 learning algorithm. In the future, we would also like to extend TML to other types of logical properties (e.g., Linear Temporal Logic), other mission-critical domains (e.g., robot control, cyber security), and non-probabilistic models (e.g., SVM, regression models). We are working on more scalable versions of the approaches outlined here, where we restrict Model Repair to only certain parameters, and restrict Data Repair to only certain data points (that we know are potentially corrupted/noisy). We plan to use TML for training a DTMC lane change controller on the Udacity car controller data (Udacity 2016), and a MDP controller for the SRI UR5 robotic arm. In the future, we also plan to investigate the connection, tradeoffs and potential synergies between Model Repair and Data Repair.

## Acknowledgments

The authors would like to thank Dr. Rodrigo de Salvo Braz for his valuable feedback regarding this work, and Dr. Natarajan Shankar for helpful discussions.

## References


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

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

Bertsekas, D. P. 1996. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific.

Dietterich, T. G. 1985. *Constraint Propagation Techniques for Theory-driven Data Interpretation (Artificial Intelligence, Machine Learning)*. Ph.D. Dissertation, Stanford University.

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

Fourer, R.; Gay, D. M.; and Kernighan, B. 1989. Algorithms and model formulations in mathematical programming. chapter AMPL: A Mathematical Programming Language.

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

Hahn, E. M.; Han, T.; and Zhang, L. 2011. Synthesis for PCTL in parametric Markov decision processes. In *NFM*.

Hahn, E. M.; Hermanns, H.; and Zhang, L. 2011. Probabilistic reachability for parametric Markov models. *International Journal on Software Tools for Technology Transfer* 13(1).

Hermanns, H.; Wachter, B.; and Zhang, L. 2008. Probabilistic CEGAR. In *CAV*.

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

Kotlowski, W., and Slowiński, R. 2009. Rule learning with monotonicity constraints. In *ICML*.

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

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

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

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

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

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

Neapolitan, R. E. 2003. *Learning Bayesian Networks*. Prentice-Hall, Inc.

Ogawa, K.; Nakagawa, H.; and Tsuchiya, T. 2015. An experimental evaluation on runtime verification of self-adaptive systems in the presence of uncertain transition probabilities. In *SEFM*.

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

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

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

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

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

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

Udacity. 2016. https://www.udacity.com/self-driving-car.

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