Chapter 2

EVALUATING PERCEPTION FOR SYSTEM-LEVEL TASK REQUIREMENTS

In safety-critical systems, the goal of perception is to aid downstream decision-making modules so that the overall system can meet its safety-critical requirements. Yet, the metrics we often use to evaluate perception performance do not account for system-level requirements or interactions between sub-systems. Usually, not all perception errors are equally safety-critical with respect to system-level requirements. this chapter argues for the importance of system-level reasoning in identifying metrics to evaluate perception. First, we show how existing evaluation metrics for object detection tasks, e.g., confusion matrices, can be leveraged to compute a probabilistic satisfaction of system-level specifications. However, confusion matrices, as traditionally defined, account for all detections equally. The second contribution of this chapter is in identifying that atomic propositions relevant to downstream planning logic and the system-level specification can be used to define new metrics for detection which result in less conservative system-level evaluations. Finally, we illustrate these ideas on a car-pedestrian example in simulation for confusion matrices constructed from the nuScenes dataset. We validate the probabilistic system-level guarantees in simulation.

This chapter is adapted from:

A. Badithela, T. Wongpiromsarn, R. M. Murray. (2021). "Leveraging Classification Metrics for Quantitative System-Level Analysis with Temporal Logic Specifications." In: *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 564–571. DOI: 10.1109/CDC45484.2021.9683611.

A. Badithela, T. Wongpiromsarn, R. M. Murray. (2023). "Evaluation Metrics of Object Detection for Quantitative System-Level Analysis of Safety-Critical Autonomous Systems." In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8651–86581.
DOI: 10.1109/IROS55552.2023.10342465.

A. Badithela, R. Srivastav, T. Wongpiromsarn, R. M. Murray, "Task-relevant evaluation metrics for object detection." *In Preparation for submission to the International Journal of Robotics Research (IJRR).*

Section 2.7 has been adapted from:

I. Incer, A. Badithela, J. Graebener, P. Mallozzi, A. Pandey, S.-J. Yu, A. Benveniste, B. Caillaud, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. (2024). "Evaluation Metrics of Object Detection for Quantitative System-Level Analysis of Safety-Critical Autonomous Systems." Conditionally accepted to: *The ACM Transactions on Cyber-Physical Systems (T-CPS)*.
arXiv preprint: https://arxiv.org/pdf/2303.17751.

2.1 Introduction

The presence of deep neural network architectures in the software stack of safety-critical applications (e.g., self-driving vehicles) necessitates a comprehensive system-level evaluation of these systems. Figure 1.1 is an illustration of the software stack of the system in which the perception component involves a deep learning-based architecture, which perceives the environment and passes its observations as inputs to the downstream planning and control modules. Using this information, the control module computes a trajectory for the vehicle to follow and the corresponding actuation commands to keep the vehicle on the trajectory.

The perception and control modules are typically designed under different principles. For example, the perception module often relies on object classification that is based on deep learning such as the use of convolutional neural networks to distinguish objects of different classes. These learning-based algorithms are often evaluated based on the performance measures such as accuracy, precision, and recall [50, 51].

On the other hand, formal methods have been employed to construct a provably correct controller given a system model and temporal logic specifications [14–18]. The correctness guarantee, typically specified using a temporal logic formula, relies heavily on the assumption that the input (i.e., the perceived world reported by perception module) is perfect. For example, if the perception component only reports the most likely class of each object, the control component assumes that the reported class is correct. Unfortunately, this assumption may not hold in most real-world systems.

To reason about system-level safety, one might consider the paradigm of specifying formal requirements on the entire system and reasoning about it. However, specifying formal requirements on the object detection task of perception is not trivial. Even in the standard classification task of classifying handwritten digits, it

is difficult to formally specify how the digits must be classified. Instead of taking this approach, we leverage metrics that are already used to evaluate learned models for their performance on object detection and classification tasks — confusion matrices. Confusion matrices are a statistical model of sensor error, constructed by evaluating a learned model against a large evaluation set.

The first contribution of this chapter is in identifying confusion matrices as a candidate model of sensor error. Leveraging confusion matrices, we can rigorously define transition probabilities representing the system's state evolution in the presence of detection error. On this model of the overall system, we can quantify system-level satisfaction of specifications via off-the-shelf probabilistic model-checking tools. An important insight gained from this analysis is that even in simple examples, intuitive design methodologies for detection models, such as maximizing recall with respect to pedestrians, might not result in safer systems overall.

However, traditionally defined confusion matrices do not account for the system-level task or the downstream controller. The second contribution of this chapter is proposing two new logic-based evaluation metrics that to account for the downstream planning logic and the system-level task. We replace the object class labels of a confusion matrix with logical formulas that are informed from the downstream controller and system-level guarantee.

Related Work

Evaluating and monitoring perception for safety-critical errors is an emerging research topic [26, 52, 53]. Perception is a complex subsystem responsible for tasks such as detection, localization, segmentation. These recent works have focused on evaluating object detection in the context of system-level safety. We follow this early work and focus on object detection task of perception, which refers to both detecting an object and classifying it correctly. As an initial stage of this study, we assume a static environment and perfect object localization. These assumptions can potentially be relaxed based on an analysis that takes into account partial observability of the environment [54], as discussed in Section 2.8.

The use of Markov chains for probabilistic reasoning about the correctness of high-level robot behaviors in the presence of perception errors was studied in [49]. However, the algorithms in [49] assumed knowledge of the probabilistic sensor model. Rigorously constructing these sensor models from confusion matrices was presented in [55]. In [56], this approach was further extended by providing confi-

dence intervals on the probabilistic sensor models and was applied to a case study on guiding aircraft on taxiways introduced by Boeing [57].

For runtime monitoring of perception systems, Timed Quality Temporal Logic (TQTL) is used to specify spatio-temporal requirements on perception [28, 29] . However, to specify these requirements, the user has to label each scenario with critical objects that need to be detected. This approach is useful in evaluating perception in isolation with respect to the requirements defined on a specific scenario. In [52], temporal diagnostic graphs are proposed to identify failures in object detection during runtime.

In [26], Hamilton-Jacobi reachability was used to account for closed-loop interactions with agents in the environment to identify safety-critical perception zones in which correct detection is crucial. Our work can be viewed as a complementary approach to [26] by allowing crucial misclassifications, according to system-level analysis, to be identified. Task-relevant perception design has been studied in [58] and [59]. In [58], the codesign of control and perception modules has been explored for tasks such as state estimation [58] and behavior prediction [59].

2.2 Preliminaries

In this section, we give an overview of linear temporal logic (LTL), a formalism for specifying system-level requirements. We also describe the performance metrics used to evaluate object detection and classification models in the computer vision community. Finally, we setup a simple discrete-state car-pedestrian system as a running example to illustrate the role of these different concepts.

System-level Task Specifications

System Specification. We use the term system to refer to refer to the autonomous agent and its environment. The agent is defined by variables V_A, and the environment is defined by variables V_E. The valuation of V_A is the set of states of the agent S_A, and the valuation of V_E is the set of states of the environment S_E. Thus, the states of the overall system is the set $S := S_A \times S_E$. Let AP be a finite set of atomic propositions over the variables V_A and V_E. An atomic proposition $a \in AP$ is a statement that can be evaluated to *true* or *false* over states in S.

We specify formal requirements on the system in LTL (see [60] for more details).

Definition 2.1 (Linear Temporal Logic [60])**.** *Linear temporal logic* (LTL) is a temporal logic specification language that allows reasoning over linear-time trace prop-

erties. An LTL formula is defined by (a) a set of atomic propositions, (b) logical operators such as: negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), and implication ($\implies$), and (c) temporal operators such as: next ($\bigcirc$), eventually ($\diamond$), always ($\Box$), and until ($\mathcal{U}$). The syntax of LTL is given as:

$$\varphi ::= \textit{True} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2,$$

with $a \in AP$, where AP is the set of atomic propositions, $\wedge$ (conjunction) and $\neg$ (negation) are the Boolean connectors from which other Boolean connectives such as $\rightarrow$ can be defined, and $\bigcirc$ (next) and $\mathcal{U}$ (until) are temporal operators. Let φ be an LTL formula over AP. We can define the operators $\diamondsuit$ (eventually) and $\Box$ (always) as $\diamondsuit \varphi = \textit{True}\, \mathcal{U} \varphi$ and $\Box\varphi = \neg \diamondsuit \neg\varphi$. The syntax of LTL is read as follows: (a) An atomic proposition p is an LTL formula, and (b) if φ and ψ are LTL formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$, $\varphi \mathcal{U} \psi$ are also LTL formulae. For an execution $\sigma = s_0 s_1 \ldots$ and an LTL formula φ, $s_i \vDash \varphi$ iff φ holds at $i \geq 0$ of σ. More formally, the semantics of LTL formula φ are inductively defined over an execution $\sigma = s_0 s_1 \ldots$ as follows,

- for $a \in AP$, $s_i \vDash a$ iff a evaluates to *True* at s_i,
- $s_i \vDash \varphi_1 \wedge \varphi_2$ iff $s_i \vDash \varphi_1$ and $s_i \vDash \varphi_2$,
- $s_i \vDash \neg\varphi$ iff $\neg(s_i \vDash \varphi)$,
- $s_i \vDash \bigcirc\varphi$ iff $s_{i+1} \vDash \varphi$, and
- $s_i \vDash \varphi_1 \mathcal{U} \varphi_2$ iff $\exists k \geq i, s_k \vDash \varphi_2$ and $s_j \vDash \varphi_1$, for all $i \leq j < k$.

An execution/trace $\sigma = s_0 s_1 \ldots$ satisfies formula φ, denoted by $\sigma \models \varphi$, iff $s_0 \models \varphi$. A *strategy* π is correct (satisfies formula φ), if the trace σ_π resulting from the strategy satisfies φ.

For an infinite trace $\sigma = s_0 s_1 \ldots$, where $s_i \in 2^{AP}$, and an LTL formula φ defined over AP, we use $\sigma \models \varphi$ to denote that σ satisfies φ. For example, the formula $\varphi = \Box p$ represents that the atomic proposition $p \in AP$ is satisfied at every state in the trace, i.e., $\sigma \models \varphi$ if and only if $p \in s_t, \forall t$. In this chapter, these traces σ are executions of the system, which we model using a Markov chain.

Definition 2.2 (Labeled Markov Chain [60])**.** A discrete-time *labeled Markov chain* is a tuple $\mathcal{M} = (S, Pr, \iota_{init}, AP, L)$, where S is a non-empty, countable set of

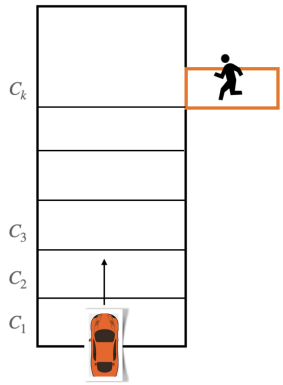


Figure 2.1: Running example of a car and pedestrian. If there is a pedestrian at crosswalk cell C_k, that is, $x_e \models \texttt{ped}$, then the car must stop at cell C_{k-1}. Otherwise, it must not stop.

states, $Pr : S \times S \to [0,1]$ is the *transition probability function* such that for all states $s \in S$, $\Sigma_{s' \in S} Pr(s,s') = 1$, $\iota_{init} : S \to [0,1]$ is the initial distribution such that $\Sigma_{s \in S} \iota_{init}(s) = 1$, AP is a set of atomic propositions, and $L : S \to 2^{AP}$ is a labeling function. The labeling function returns the set of atomic propositions that evaluate to true at a given state. Given an LTL formula φ (defined over AP) that specifies requirements of a system modeled by the Markov Chain $\mathcal{M}$, the probability that a trace of the system starting from $s_0 \in S$ will satisfy φ is denoted by $\mathbb{P}_{\mathcal{M}}(s_0 \models \varphi)$. The definition of this probability function is detailed in [60].

Example

Consider a car-pedestrian example, modeled using discrete transition system as illustrated in Figure 2.1. The true state of the environment is denoted by x_e. The state of the car is characterized by its position and speed, $s_a := (x_c, v_c) \in S_A$. The safety requirement on the car is that it "shall stop at the crosswalk if there is a waiting pedestrian, and not come to a stop, otherwise". The overall system specifications are formally expressed as safety specifications in equations (2.1)-(2.3).

1. If the true state of the environment is not a pedestrian, i.e. $x_e \neq \texttt{ped}$, then the car must not stop at C_{k-1}.

$$\varphi_1 = \Box((x_e \neq \texttt{ped}) \to \neg(x_c = C_{k-1} \wedge v_c = 0)) \,. \tag{2.1}$$

2. If $x_e = \texttt{ped}$, the car must stop on C_{k-1}.

$$\varphi_2 = \square\Big(x_e = \texttt{ped} \rightarrow ((x_c = C_{k-1} \wedge v_c = 0) \vee \neg(x_c = C_{k-1}))\Big). \quad (2.2)$$

3. The agent should not stop at any cell C_i, for all $i \in \{1, \ldots, k-2\}$,

$$\varphi_3 = \square\neg(\bigvee_{i=1}^{k-2}(x_c = C_i \wedge v_c = 0)). \quad (2.3)$$

The overall safety specification for the car is $\varphi := \varphi_1 \wedge \varphi_2 \wedge \varphi_3$. Since the car controller has been designed assuming perfect perception, the specification for the pedestrian and non-pedestrian environment simplifies to,

$$\begin{aligned}\varphi_{\texttt{ped}} =& \square\neg(\bigvee_{i=1}^{k-2}(x_c = C_i \wedge v_c = 0)) \bigwedge \square(\neg(x_c = C_{k-1}) \\ & \vee (x_c = C_{k-1} \wedge v_c = 0)), \\ \varphi_{\texttt{class}} =& \square\neg(\bigvee_{i=1}^{k-1}(x_c = C_i \wedge v_c = 0)), \text{ if } \texttt{class} \in \{\texttt{obs}, \texttt{empty}\}.\end{aligned}$$

As mentioned previously, we assume a static environment. We also assume that the car knows the location of the crosswalk, e.g., from HD map information, and that it can coarsely localize whether the detected object is on the crosswalk. The evaluation framework presented in this chapter is valid for any discrete-state control strategy, both deterministic and probabilistic. To concretize the setup, we consider a car controller that acts corresponding to the detection model's prediction of the environment at the crosswalk. If the car at time step t detects a pedestrian, then it chooses its speed according to a control strategy for $\varphi_{\texttt{ped}}$ to come to a stop before the crosswalk at cell C_{k-1}. If the state of the car is such that it is impossible to find a controller that will bring it to a stop at cell C_{k-1}, then it decelerates as fast as possible. Similarly, if an obstacle or empty sidewalk is detected, then the car chooses its speed according to a control strategy designed correct-by-construction for φ_k.

2.3 Problem Statement

Here, we introduce and define the probability of satisfaction of an LTL formula starting from an initial state, given the true state of the environment.

Definition 2.3 (Model of Sensor Error)**.** Let S_E denote the set of possible environment states. Then, a model of sensor error in identifying the state of the environment $M : S_E \times S_E \rightarrow [0, 1]$ is defined as follows, $M(y, x) = p$, where p is the probability with which the sensor predicts the environment state to be $y \in S_E$ when its true state is $x \in S_E$.

Definition 2.4 (Transition Probability)**.** Let $s_1 = (s_{1,a}, x_e)$, $s_2 = (s_{2,a}, x_e) \in S$ be two states of the overall system, x_e be the true class label of the environment, and let M be a model of sensor error. Let $O(s_1, s_2)$ denote the set of environment observations $y_e \in V_E$ that result in the agent controller transitioning from $s_{1,a}$ to $s_{2,a}$. The *transition probability* $Pr : S \times S \rightarrow [0, 1]$ is defined as,

$$Pr(s_1, s_2) := \sum_{y_e \in O(s_1, s_2)} M(y_e, x_e)\,. \tag{2.4}$$

Since the controller is entirely informed by the outputs of the perception module, and for each output of the perception module, there is a corresponding control action, it is trivial to check that $\sum_{s_2 \in S} Pr(s_1, s_2) = 1$. Therefore, the transition probability between any two states is always in the range $[0, 1]$.

Definition 2.5 (Paths)**.** Choose a state $s_0 = (s_{a,0}, x_e) \in S$ for a fixed true environment state x_e. A finite path starting from s_0 is a finite sequence of states $\sigma(s_0) = s_0, s_1, \ldots, s_n$ for some $n \geq 0$ such that the probability of transition between consecutive states, $Pr(s_i, s_{i+1}) > 0$ for all $0 \leq i < n$ such that $s_i = (s_{a,i}, x_e) \in S$. Similarly, an infinite path $\sigma = s_0, s_1, \ldots$ is an infinite sequence of states such that $Pr(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We denote the set of all paths starting from $s_0 \in S$ by $Paths(s_0)$, and the set of all finite paths starting from $s_0 \in S$ by $Paths_{fin}(s_0)$. For an LTL formula φ on AP, $Paths_{\varphi}(s_0) \subset Paths(s_0)$ is the set of paths $\sigma = s_0, s_1, \ldots$ such that $\sigma_S \models \varphi$.

Semantics

Now, we define probability of satisfaction of a temporal logic formula with respect to a formal specification based on the following definitions derived from [60]. Let $\Omega = Paths(s_0)$ represents the set of all possible outcomes, that is, the set of all paths of the agent, starting from state s_0. Let 2^{Ω} denote the powerset of Ω. Then, $(\Omega, 2^{\Omega})$ forms a σ-algebra. For a path $\hat{\pi} = s_0, s_1, \ldots, s_n \in Paths_{fin}(s_0)$, we define a cylinder set as follows,

$$Cyl(\hat{\pi}) = \{\pi \in Paths(s_0) | \hat{\pi} \in pref(\pi)\}, \tag{2.5}$$

where $pref(\pi) = \{\pi_{...j} = s_0, \ldots, s_j | j \geq 0\}$ is the set of all finite prefix path fragments for $\pi = s_0, s_1, \ldots$, an infinite path. Let $\mathcal{C}_{s_0} = \{Cyl(\hat{\pi}) | \hat{\pi} \in Paths_{fin}(s_0)\}$. The following result can be found in [60], and can be derived from the fundamental definition of a σ-algebra.

Lemma 2.1. The pair $(Paths(s_0),\ 2^{\mathcal{C}_{s_0}})$ forms a σ-algebra, and is the smallest σ-algebra containing $\mathcal{C}_{s_0}$.

The σ-algebra associated with s_0 is $(Paths(s_0), 2^{\mathcal{C}_{s_0}})$. Then, there exists a unique probability measure $\mathbb{P}_{s_0}$ such that

$$\mathbb{P}_{s_0}(Cyl(s_0, \ldots, s_n)) = \prod_{0 \leq i \leq n} Pr(s_i, s_{i+1}). \tag{2.6}$$

Definition 2.6. Consider an LTL formula φ over AP with the overall system starting at state $s_0 = (s_{a,0}, x_e)$. Then, the probability that the system will satisfy the specification φ from the initial state s_0 given the true state of the environment is,

$$\mathbb{P}(s_0 \models \varphi) := \sum_{\sigma(s_0) \in \mathcal{S}(\varphi)} \mathbb{P}_{s_0}(Cyl(\sigma(s_0))), \tag{2.7}$$

where $\mathcal{S}(\varphi) := Paths_{fin}(s_0) \cap Paths_{\varphi}(s_0)$. Note that $\mathcal{S}(\varphi)$ need not be a finite set, but has to be countable.

Definition 2.7 (Controller)**.** For an initial condition $s_0 \in S$ of the system and environment, and the system specification φ, the system controller $K : S^{\omega}S \to S_A$ chooses the next system state based on the trace history of system states and environment observations.

Problem Formulation

Problem 2.1. Given a model of sensor error M for multi-class classification, a controller K, a temporal logic formula φ, the initial state of the agent $s_{a,0}$, and the true state of the static environment x_e, compute the probability $\mathbb{P}(s_0 \models \varphi)$ that φ will be satisfied for a system trace σ starting from initial condition $s_0 = (s_{a,0}, x_e)$?

2.4 Role of Detection Metrics in Quantitative System-level Evaluations

In this section, we will introduce x While the confusion matrix provides useful metrics for comparing and evaluating detection models, we would like to use these metrics in evaluating the overall system with respect to formal constraints in temporal logic. Not all detection errors are equally safety-critical [25, 26].

Confusion Matrix

We consider object detection to include both the detection and the classification tasks. In this section, we provide background on metrics used to evaluate performance with respect to these perception tasks. Let the evaluation dataset $\mathcal{D} = \{(f_i, b_i, d_i, x_i)\}_{i=1}^{N}$ consist of N objects across m image frames $F = \{F_1, \ldots, F_m\}$. For each object, $f_i \in F$ represents the image frame token, b_i specifies the bounding box coordinates, d_i denotes the distance of the object to ego, and x_i denotes the true class of the object. When a specific object detection algorithm is evaluated on $\mathcal{D}$, each object has a predicted bounding box, $\tilde{b}_i$, and predicted object class $\tilde{x}_i$. We store these predictions in the set $\mathcal{E} = \{(\tilde{b}_i, \tilde{x}_i)\}_{i=1}^{N}$.

Definition 2.8 (Confusion Matrix)**.** Let $\mathcal{D}$ be an evaluation set of objects and $\mathcal{E}$ be the corresponding predictions by an object detection algorithm. Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a set of object classes in $\mathcal{D}$, and let n denote the cardinality of $\mathcal{C}$. The confusion matrix corresponding to the classes $\mathcal{C}$ and dataset $\mathcal{D}$, and predictions $\mathcal{E}$ is an $n \times n$ matrix $\texttt{CM}(\mathcal{C}, \mathcal{E}, \mathcal{D})$ with the following properties:

- $\texttt{CM}(\mathcal{C}, \mathcal{E}, \mathcal{D})[i, j]$ is the element in row i and column j of $\texttt{CM}(\mathcal{C}, \mathcal{E}, \mathcal{D})$, and represents the number of objects that are predicted to have class label $c_i \in \mathcal{C}$, but have the true class label $c_j \in \mathcal{C}$, and
- the sum of the j^{th}-column of $\texttt{CM}(\mathcal{C}, \mathcal{E}, \mathcal{D})$ is the total number of objects in $\mathcal{D}$ belonging to the class $c_j \in \mathcal{C}$.

Several performance metrics for object detection and classification such as true positive rate, false positive rate, precision, accuracy, and recall can be derived from the confusion matrix[50, 51, 61].

Definition 2.9 (Precision [50])**.** Given the confusion matrix $\texttt{CM}$ for a multi-class classification, the *precision* corresponding to class c_i is:

$$P(i) = \frac{\texttt{CM}(i,i)}{\texttt{CM}(i,i) + \frac{\sum_{j \neq i} \texttt{CM}(i,j)|\mathcal{D}_j|}{\sum_{j \neq i}|\mathcal{D}_j|}}, \tag{2.8}$$

where $\frac{\sum_{j \neq i} \texttt{CM}(i,j)|\mathcal{D}_j|}{\sum_{j \neq i}|\mathcal{D}_j|}$ is the false positive rate for class c_i, and $\texttt{CM}(i,i)$ is the true positive rate for class c_i.

Definition 2.10 (Recall [50])**.** Given the confusion matrix $\texttt{CM}$ for a multi-class classification, the *recall* corresponding to class label c_i is:

$$R(i) = \frac{\texttt{CM}(i,i)}{\texttt{CM}(i,i) + \sum_{j \neq i} \texttt{CM}(j,i)}, \tag{2.9}$$

where $\sum_{j \neq i} \texttt{CM}(j,i)$ is the false negative rate for class c_i.

Maximizing precision typically corresponds to minimizing false positives while maximizing recall corresponds to minimizing false negatives. However, there is an inherent trade-off in minimizing both false positives and false negatives for classification tasks [50], and often, a good operating point is found in an *ad-hoc* manner. Typically, safety-critical systems are designed for optimizing recall, but as we will show in Section 2.6, this is not always the best strategy to satisfy formal requirements.

Remark 2.1. In this chapter, we use c_n = (referring to the background class) as an auxiliary class label in the construction of confusion matrices. If an object has the true class label c_i but is not detected by the object detection algorithm, then this gets counted in $\texttt{CM}(\mathcal{C}, \mathcal{E}, \mathcal{D})(n,i)$ as a false negative with respect to class c_i. If the object was not labeled originally, but is detected and classified to have class label c_i, then it gets counted in $\texttt{CM}(\mathcal{C}, \mathcal{E}, \mathcal{D})(i,n)$ as a false negative of the $\texttt{empty}$class. We expect that in a properly annotated dataset, false negatives $\texttt{CM}(\mathcal{C}, \mathcal{E}, \mathcal{D})(i,n)$ to be small. We ignore these extra detections in constructing the confusion matrix because by not being annotated, they are not relevant to the evaluation of object detection models.

Definition 2.11 (Transition Probability for Confusion Matrices)**.** Let $s_1 = (s_{1,a}, x_e)$, $s_2 = (s_{2,a}, x_e) \in S$ be two states of the overall system, x_e be the true class label of the environment, and $\texttt{CM}$ be the known confusion matrix associated with the agent's perception model. Let $O(s_1, s_2)$ denote the set of environment observations $y_e \in V_E$ that result in the agent controller transitioning from $s_{1,a}$ to $s_{2,a}$. The transition probability $Pr : S \times S \to [0,1]$ is defined as,

$$Pr(s_1, s_2) := \sum_{y_e \in O(s_1, s_2)} \texttt{CM}(y_e, x_e)\,. \tag{2.10}$$

From the definition, and consequently structure, of the confusion matrix in Definition 2.8, it is trivial to check that $\sum_{s_2 \in S} Pr(s_1, s_2) = 1$. Therefore, the transition probability between any two states is always in the range $[0,1]$.

Class-labeled, distance-parametrized Confusion Matrix

This performance metric builds on the class-labeled confusion matrix defined in Definition 2.8. As denoted previously, let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be the set of different classes of objects in dataset $\mathcal{D}$. For every object in $\mathcal{D}_k$, the predicted class of the object will be one of the class labels $c_1, \ldots, c_n$. For each distance interval z_k, we define the class-labeled confusion matrix as $\texttt{CM}_{\text{class},k} := \texttt{CM}(\mathcal{C}, \mathcal{E}_k, \mathcal{D}_k)$. Algorithm 2 shows the construction of the class-labeled, distance-parametrized confusion matrix. Therefore, the outcomes of the object detection algorithm will be defined by the set $Outc = \{c_1, \ldots, c_n\}^m$, where m is the total number of objects in the true environment in the distance interval z_k. The tuple $(Outc, 2^{Outc})$ forms a σ-algebra for defining a probability function over the class-labeled confusion matrix $\texttt{CM}_{\text{class},k}$. Similar to the definition of a probability function, for every class label c_j, the probability function $\mu_{\text{class},k}(\cdot, c_j) : Outc \to [0, 1]$ is defined as follows,

$$\mu_{\text{class},k}(c_i, c_j) := \frac{\texttt{CM}_{\text{class},k}(c_i, c_j)}{\sum_{l=1}^{n} \texttt{CM}_{\text{class},k}(c_l, c_j)}. \tag{2.11}$$

Algorithm 1: Class-labeled Confusion Matrix

procedure ClassCM(Dataset $\mathcal{D} = \{(f_i, b_i, d_i, x_i)\}_{i=1}^{N}$, Classes $\mathcal{C}$, Distance Parameters $\{D_k\}_{k=0}^{k_{\max}}$)
  From $\{D_k\}_{k=0}^{k_{\max}}$, define distance intervals $\{z_k\}_{k=1}^{k_{\max}}$
  Run object detection algorithm to get predictions $\mathcal{E}$,
  Initialize $\mathcal{D}_1, \ldots, \mathcal{D}_{k_{\max}}$ as empty sets
  Initialize $\mathcal{E}_1, \ldots, \mathcal{E}_{k_{\max}}$ as empty sets
  for $(f_i, b_i, d_i, x_i) \in \mathcal{D}$ **do**
    if $d_i \in z_k$ **then**
      $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{(f_i, b_i, d_i, x_i)\}$
      $\mathcal{E}_k \leftarrow \mathcal{E}_k \cup \{(\tilde{b}_i, \tilde{x}_i)\}$
  for $k \in \{0, \ldots, k_{\max}\}$ **do**
    Denote $\texttt{CM}_{\text{class}}(\mathcal{C}, \mathcal{E}_k, \mathcal{D}_k)$ as $\texttt{CM}_{\text{class},k}$
    $\texttt{CM}_{\text{class},k} \leftarrow$ zero matrix
    for $f_i \in \{f_1, \ldots, f_m\}$ **do** ▷ Loop over images
      for object in $\mathcal{D}_k$ **do**
        $c_i \leftarrow$ Predicted class label of object
        $c_j \leftarrow$ True class label of object in $\mathcal{E}_k$
        $\texttt{CM}_{\text{class},k}(c_i, c_j) \leftarrow \texttt{CM}_{\text{class},k}(c_i, c_j) + 1$
  $\texttt{CM}_{\text{class}}(\mathcal{C}, \mathcal{E}, \mathcal{D}) = \{\texttt{CM}_{\text{class}}(\mathcal{C}, \mathcal{E}_k, \mathcal{D}_k)\}_{k=0}^{k_{\max}}$
  return $\texttt{CM}_{\text{class}}(\mathcal{C}, \mathcal{E}, \mathcal{D})$

Definition 2.12 (Transition probability function for class-labeled confusion matrix)**.** Let the true environment be represented as a tuple x_e corresponding to class labels

in the region z_k (class labels can be repeated in a tuple x_e when multiple objects of the same class are in region z_k). Let $s_{a,1}, s_{a,2} \in S$ be states of the car, and let $O(s_1, s_2)$ denote the set of all predictions of the environment that prompt the system to transition from $s_1 = (s_{a,1}, x_e)$ to $s_2 = (s_{a,2}, x_e)$. Likewise, the tuple y_e represents the object detection model's predictions of the environment. Then, the transition probability function from state s_1 to s_2 is defined as follows,

$$Pr(s_1, s_2) := \sum_{y_e \in O(s_1, s_2)} \prod_{i=1}^{|y_e|} \mu_{\text{class},k}(y_e(i), x_e(i)). \tag{2.12}$$

For both transition probability functions (2.12) and (2.14), we can check (by construction) that $\forall s_1 \in S, \sum_{s_2} Pr(s_1, s_2) = 1$. In the running example, if the crosswalk were to have another pedestrian and a non-pedestrian obstacle, then the probability of detecting each object is considered independently of the others. This results in the product of probabilities $\mu_{\text{class},k}(\cdot, x_e(i))$ in equation (2.12).

Proposition-labeled Confusion Matrix

In several instances, the high-level planner does not necessarily require correct detection of every single object in a frame to make a correct decision. For instance, for the planner to decide to stop for a cluster of pedestrians 20m away, knowledge that there are pedestrians, and not necessarily the exact number of pedestrians is sufficient for the planner to decide to slow down. Accounting for this in quantitative system-level evaluations would make the analysis less conservative. Therefore, we introduce the notion of using atomic propositions as class labels in the confusion matrix instead of the object classes themselves.

Let p_i be the atomic proposition: "*there exists an object of class* $c_i \in \mathcal{C}$," and let $\mathcal{P} = \{p_1, \ldots, p_n\}$ denote the set of all atomic propositions. Let $D_0 < D_1 < \ldots < D_k < \ldots < D_{k_{\max}}$ denote progressively increasing distances from the autonomous vehicle. Let $\mathcal{D}_k \subset \mathcal{D}$ be the subset of the dataset that includes objects that are in the distance interval $z_k = (D_{k-1}, D_k)$ from the autonomous system. Let $\mathcal{E}_k$ denote the predictions of the object detection algorithm corresponding to dataset $\mathcal{D}_k$. For each parameter k, we define the proposition-labeled confusion matrix $\texttt{CM}_{\text{prop},k} = \texttt{CM}_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}_k, \mathcal{D}_k)$ where the classes are characterized by the powerset of atomic propositions $2^{\mathcal{P}}$. Algorithm 1 shows the construction of the proposition-labeled confusion matrix.

The true environment is associated with a set of atomic propositions that are a subset of $\mathcal{P}$ that evaluates to true. Suppose, there is a pedestrian and a trash can in the distance interval z_k from the ego, then the true class label is $\{p_{\texttt{ped}}, p_{\texttt{obs}}\}$ in the distance-parametrized confusion matrix $\texttt{CM}_{\text{prop},k}$. Note that for every possible environment, there is only one corresponding class in the proposition-labeled confusion matrix. Thus, for a given true environment, the predicted class of the environment at distance interval z_k could be any element of the set $2^{\mathcal{P}}$. Therefore, at each time step, the set of detection outcomes is $Outc = 2^{\mathcal{P}}$.

Algorithm 2: Proposition-labeled Confusion Matrix

procedure PropCM(Dataset $\mathcal{D} = \{(f_i, b_i, d_i, x_i)\}_{i=1}^{N}$, Classes $\mathcal{C}$, Distance Parameters $\{D_k\}_{k=0}^{k_{\text{max}}}$)
  From $\{D_k\}_{k=0}^{k_{\text{max}}}$, define distance intervals $\{z_k\}_{k=1}^{k_{\text{max}}}$
  Run object detection algorithm to get predictions $\mathcal{E}$,
  Initialize $\mathcal{D}_1, \ldots, \mathcal{D}_{k_{\text{max}}}$ as empty sets
  Initialize $\mathcal{E}_1, \ldots, \mathcal{E}_{k_{\text{max}}}$ as empty sets
  for $(f_i, b_i, d_i, x_i) \in \mathcal{D}$ **do**
    if $d_i \in z_k$ **then**
      $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{(f_i, b_i, d_i, x_i)\}$
      $\mathcal{E}_k \leftarrow \mathcal{E}_k \cup \{(\tilde{b}_i, \tilde{x}_i)\}$
  for $c_j \in \mathcal{C}$ **do**
    $p_j \equiv$ "there exists an object of class c_j"
  $\mathcal{P} \leftarrow \bigcup_j \{p_j\}$ ▷ Set of atomic propositions
  for $k \in \{1, \ldots, k_{\text{max}}\}$ **do**
    Denote $\texttt{CM}_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}_k, \mathcal{D}_k)$ as $\texttt{CM}_{\text{prop},k}$
    $\texttt{CM}_{\text{prop},k} \leftarrow$ zero matrix
    for $f \in F$ **do** ▷ Loop over image frames
      Group objects in $\mathcal{D}_k$ with image token f.
      Group predictions in $\mathcal{E}_k$ with image token f.
      $P_i \leftarrow$ Predicted set of propositions
      $P_j \leftarrow$ True set of propositions
      $\texttt{CM}_{\text{prop},k}(P_i, P_j) \leftarrow \texttt{CM}_{\text{prop},k}(P_i, P_j) + 1$
  $\texttt{CM}_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}, \mathcal{D}) = \{\texttt{CM}_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}_k, \mathcal{D}_k)\}_{k=0}^{k_{\text{max}}}$
  return $\texttt{CM}_{\text{prop}}(2^{\mathcal{P}}, \mathcal{E}, \mathcal{D})$

The tuple $(Outc, 2^{Outc})$ forms a σ-algebra for defining a probability function over the proposition-labeled confusion matrix. Since the set $Outc$ is countable, we can define a probability function $\mu : Outc \to [0, 1]$ such that $\sum_{e \in Outc} \mu(e) = 1$. For a distance-parametrized confusion matrix $\texttt{CM}_{\text{prop},k}$ with class labels in the set $Outc$, and for every true environment class label P_j, define a probability function

$\mu_{\text{prop},k}(\cdot, P_j) : Outc \to 2^{Outc}$ as follows,

$$\mu_{\text{prop},k}(P_i, P_j) = \frac{\texttt{CM}_{\text{prop},k}[P_i, P_j]}{\sum_{l=1}^{|2^{\mathcal{P}}|} \texttt{CM}_{\text{prop},k}[P_l, P_j]}, \quad \forall P_i \in 2^{\mathcal{P}}, \tag{2.13}$$

where $\texttt{CM}_{\text{prop},k}[P_i, P_j]$ is the element of the confusion matrix $\texttt{CM}_{\text{prop},k}$ with predicted class label P_i and true class label P_j.

That is, for every confusion matrix $\texttt{CM}_{\text{prop},k}$ where $k \in \{1, \ldots, k_{\max}\}$, we define a total of $2^{|\mathcal{P}|}$ different probability functions, one for each possible true environmentP_j. Thus, the probability function $\mu_{\text{prop},k}$ that characterizes the probability of detecting an environment satisfying propositions P_i, given that the true environment at z_k satisfies propositions P_j. This helps to formally define the state transition probability of the overall system as follows.

Definition 2.13 (Transition probability function for proposition-labeled confusion matrices)**.** Let x_e be the true environment state corresponding to propositions P_j evaluating to true, and let $s_{a,1}, s_{a,2} \in S$ be states of the car. Let $O(s_1, s_2)$ denote the set of all predictions of the environment that prompt the system to transition from $s_1 = (s_{a,1}, x_e)$ to $s_2 = (s_{a,2}, x_e)$. At state s_1, let z_k be the distance interval of objects in the environment causing the agent to transition from $s_{a,1}$ to $s_{a,2}$. The corresponding confusion matrix is $\texttt{CM}_{\text{prop},k}$. Then, the transition probability from state s_1 to s_2 is defined as follows,

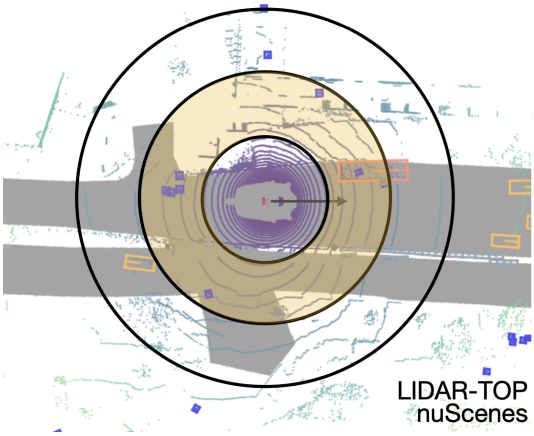
$$Pr(s_1, s_2) := \sum_{P_i \in O(s_1, s_2)} \mu_{\text{prop},k}(P_i, P_j). \tag{2.14}$$

For simplicity, we assume that objects at a specific distance interval influence the agent to transition from $s_{a,1}$ to $s_{a,2}$. However, Definition 2.13 can be extended to cases in which objects at multiple distances can influence transitions.

Choosing Proposition Labels

Generally, the set of atomic propositions $\mathcal{P}$ depends on the logic used by the planner to trigger different operation modes. In the running example, the planner outputs different actions depending on the environment, i.e., pedestrian or other objects. If the planner responds differently to other types of objects, e.g cars, bicycles, cones, those should be included in the set of atomic propositions $\mathcal{P}$. Thus, our approach can generalize to a wider range of scenarios by adapting the set $\mathcal{P}$ accordingly.

In particular, proposition labels of the confusion matrix can be chosen to match the set of environment observations S_E that are acceptable inputs to the controller (Definition 2.7). Proposition labels can be propositional formulas comprising of logical connectives, but not any temporal operators.



Predicted state \ True state of environment	$x_e \vDash ped$	$x_e \vDash obj$	$x_e \vDash \{ped, obj\}$	$x_e \vDash emp$
$y_e \vDash ped$	10	1	1	3
$y_e \vDash obj$	1	25	20	1
$y_e \vDash \{ped, obj\}$	2	2	11	2
$y_e \vDash emp$	3	1	3	10

$\texttt{CM}_{prop}$

Figure 2.2: Proposition-labeled confusion matrices when evaluations are grouped solely by distance. Observe that detecting one pedestrian in the highlighted distance zone will amount to the proposition "there is a pedestrian" evaluating to true. This would not be an appropriate evaluation for the driving task.

Grouping Objects for Evaluation

The choice of evaluation metric for the detection model depends on the observations received by the downstream planner, and how the planner processes these observations to to control the system. Proposition labels are defined over objects in a group, and each group accounts for a single evaluation of the model. For meaningful evaluations of the perception system, the grouping of objects into propositional formulas should be at the right fidelity for the planning module. For example, in a robotic system that has a vision-based perception comprising of only forward-facing cameras and a planner tasked with driving forward, grouping objects by distance to the ego might be sufficient for effectively evaluating the perception with the system-level task. However, in robotic systems equipped with LiDAR sensors and tasked with navigating arbitrarily, the same evaluations might no longer be meaningful. In particular, since LiDAR sensor outputs 360° observations, grouping objects solely by ego-centric distance will be too coarse from a planning standpoint (see Figure 2.2). We denote this proposition-labeled confusion matrix as $\texttt{CM}_{prop,seg}$.

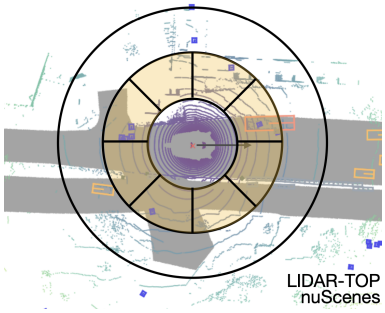


Figure 2.3: Grouping evaluations at the same level of abstraction used by the high-level planner. Evaluating the proposition "there is a pedestrian" in each segment of the distance zone.

The perception system localizes objects in the environment and returns detections in $\mathbb{R}^3$ upto some finite distance around the ego. Let $G = \{C_1, \ldots, C_m\}$ be a finite, discretized ego-centric abstraction of $\mathbb{R}^3$. The labeling functions of the system and environment allow for mapping the states of the system and environment to G. Typically in high-level planning, detections from the perception system are mapped onto the discretized abstraction G, and this information is used by the planner to decide on next actions. Therefore, evaluation of the perception system using proposition labels at the fidelity of G would be as follows. Instead of using distance to group objects into radius bands (lines 17-21 of Algorithm 2), we group objects according to the ego-centric abstraction G, and evaluate proposition labels for each cell (see Figure 2.3) for an illustration).

2.5 Markov Chain Analysis

Our approach to solving Problem 2.1 is based on constructing a Markov chain that represents the state evolution of the overall system, taking into account the control logic as well as detection errors. This Markov chain is constructed for a particular true state of the environment. Given a Markov chain for the state evolution of the system, it is then straightforward to compute the probability of satisfying a temporal logic formula on the Markov chain from an arbitrary initial state [60]. Probabilistic model checking can be used to compute the probability that the Markov chain satisfies the formula using existing tools such as PRISM [62] and Storm [63], which have been demonstrated to successfully analyze systems modeled by Markov chains with billions of states. In addition to the efficient off-the-shelf probabilistic model checkers, our approach is computationally tractable because constructing the Markov chain from the confusion matrix is linear in the number of classes used for perception.

For each confusion matrix, we can synthesize a corresponding Markov chain of the system state evolution as per Algorithm 3. Using off-the-shelf probabilistic model checkers such as Storm [63], we can compute the probability that the trace of a system satisfies its requirement, $\mathbb{P}(s_0 \models \varphi)$, by evaluating the probability of satisfaction of the requirement φ on the Markov chain. Let $O(x_e)$ be the set of all possible predictions of true environment state x_e by the the object detection model. The system controller $K : S \times O(x_e) \rightarrow S$ accepts as inputs the current state of the agent and the environment, $s_0 \in S$, and the environment state predictions $y_e \in O(x_e)$ from object detection. Based on the predictions, it actuates the agent resulting in the end state $s_f \in S$. At each time step, the agent makes a new ob-

servation of the environment (y_e) and chooses a control action corresponding to y_e.

Remark 2.2. Markov chain construction aids in evaluating the overall system. In future work, we plan to address the issue of tracking, in which perception errors are tracked over multiple temporal frames.

Definition 2.14 (Labeled Markov Chain [60])**.** A discrete-time labeled Markov chain is a tuple $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$, where S is a non-empty, countable set of states, $\mathbf{P} : S \times S \to [0, 1]$ is the *transition probability function* such that for all states $s \in S$, $\Sigma_{s' \in S}\mathbf{P}(s, s') = 1$, $\iota_{init} : S \to [0, 1]$ is the initial distribution such that $\Sigma_{s \in S}\iota_{init}(s) = 1$, AP is a set of atomic propositions, and $L : S \to 2^{AP}$ is a labeling function.

The σ-algebra of Markov chain $\mathcal{M}$ is $(Paths(\mathcal{M}, 2^{\mathcal{C}_\mathcal{M}}))$, where $\mathcal{C}_\mathcal{M} = \{Cyl(\hat{\pi}) | \hat{\pi} \in Paths_{fin}(\mathcal{M})\}$ [60]. Let $\mathcal{S}_\mathcal{M}(\varphi)$ denote all paths of the MC $\mathcal{M}$ in $Paths_{fin}(\mathcal{M}) \cap Paths(\mathcal{M})$.

Definition 2.15 (Probability on a Markov Chain)**.** Given an LTL formula φ over AP, a true state of the environment, x_e, an initial system state, $s_0 = (s_{a,0}, x_e)$, and a Markov chain $\mathcal{M}$ describing the dynamics of the overall system, we denote the probability that the system will satisfy φ starting from state s_0 as $\mathbb{P}_\mathcal{M}(s_0 \models \varphi_s)$. This probability can be computed using standard techniques as described in [60].

Algorithm 3: Markov Chain Construction

procedure `Labeled Markov Chain`$(S, x_e, s_0, K, \texttt{CM})$
Input: Product states S, True environment x_e, Initial condition $s_0 \coloneqq (s_{a,0}, x_e) \in S$, Controller K synthesized for s_0 and φ, Confusion matrix `CM`,
Output: Markov Chain $\mathcal{M}$ carrying the probability of detection error
  $Pr(s, s') = 0,\ \forall s, s' \in S$
  $K \leftarrow$ Initialize Controller for initial state s_0
  for $s_i \in S$ **do**
    $\iota_{init}(s_i) = 1$ ▷ Initial Distribution
    for $y_e \in O(x_e)$ **do**
      $s_f \leftarrow K(s_i, y_e)$ ▷ Controller
      Identify z_k according to Definitions 2.12, 2.13
      $\mu_{\text{class},k}, \mu_{\text{prop},k} \leftarrow$ Equations (2.11), (2.13).
      if class-labeled **then**
        $p \leftarrow \prod_{i=1}^{|y_e|} \mu_{\text{class},k}(y_e(i), x_e(i))$
      if proposition-labeled **then**
        $P_j \leftarrow$ Propositions for true x_e
        $P_i \leftarrow$ Propositions for predicted y_e
        $p \leftarrow \mu_{\text{prop},k}(P_i, P_j)$
      $Pr(s_i, s_f) \leftarrow Pr(s_i, s_f) + p$
  return $\mathcal{M} = (S, Pr, \iota_{init}, AP, L)$

Proposition 2.1. Given φ as a temporal logic formula over the agent and the environment states, true state of the environment x_e, agent initial state $s_{a,0}$, and a Markov chain $\mathcal{M}$ constructed via Algorithm 3, then $\mathbb{P}(s_0 \models \varphi)$ is equivalent to computing $\mathbb{P}_{\mathcal{M}}(s_0 \models \varphi)$, where $s_0 = (s_{a,0}, x_e)$.

Proof. We begin by considering the transition probabilities Pr and the transition probabilities on the Markov chain $\mathbf{P}$. Since misclassification errors are the only source of non-determinism in the evolution of the agent state, by construction, we have that $\mathbf{P}(s_i, s_j) = Pr(s_i, s_j)$ for some $s_i, s_j \in S$. Next, we compare the σ-algebra of Markov chain $\mathcal{M}$ with the σ-algebra associated with state s_0. By construction of the Markov chain, observe that any path $p \in Paths(s_0)$ is also a path on the MC $\mathcal{M}$, $p \in Paths(\mathcal{M})$, and as a result $\mathcal{C}_{s_0} \subset \mathcal{C}_{\mathcal{M}}$. Similarly, by construction, there is no finite trace on the Markov chain starting from s_0, $\sigma(s_0) \in \mathcal{S}_{\mathcal{M}}$ that

is not in $\mathcal{S}(\varphi)$.

$$\begin{aligned}
\mathbb{P}(s_0 \models \varphi) &= \sum_{\sigma(s_0)\in\mathcal{S}(\varphi)} \mathbb{P}_{s_0}(Cyl(\sigma(s_0))) \\
&= \sum_{\sigma(s_0)\in\mathcal{S}(\varphi)} \prod_{0\le i<n} Pr(\sigma_i, \sigma_{i+1}) \\
&= \sum_{\sigma(s_0)\in\mathcal{S}(\varphi)} \prod_{0\le i<n} \mathbf{P}(\sigma_i, \sigma_{i+1}) \\
&= \sum_{\sigma(s_0)\in\mathcal{S}_{\mathcal{M}}(\varphi)} \prod_{0\le i<n} \mathbb{P}_{\mathcal{M}}(Cyl(\sigma(s_0))) \\
&= \mathbb{P}_{\mathcal{M}}(s_0 \models \varphi)
\end{aligned}$$

□

2.6 Experiments

In this section, we conduct various experiments the car-pedestrian example in simulation. We present system-level evaluations for the car pedestrian example for various types of confusion matrices.

Fundamental Tradeoffs. Even in the simplest setting of the traditional class-based confusion matrix, we can show that these quantitative evaluations highlight fundamental tradeoffs in detection, and that the right operating point must be informed by system-level specifications as well as the down-stream control logic. Often in autonomous driving applications, maximizing recall is prioritized over precision for safety purposes. In our example, maximizing recall would correspond with increasing tendency to stop at C_{k-1}, even if $x_e \neq \texttt{ped}$. In Figure 2.4, we show how varying precision/recall affects the probability of satisfaction for $V_{max} = 6$. These precision/recall pairs were chosen to reflect the general precision/recall tradeoff trends for classification tasks [50]. For the results presented in this chapter, we construct a confusion matrix as a function of precision (p) and recall (r) as shown in $\texttt{CM}(p, r)$ of Table 2.1, and are in reference to the class label ped. In Table 2.1, TP, FP, TN, FN are the number of true positives, false positives, true negatives, and false negatives, respectively, of the ped class label. These are derived from precision p and recall r as follows,

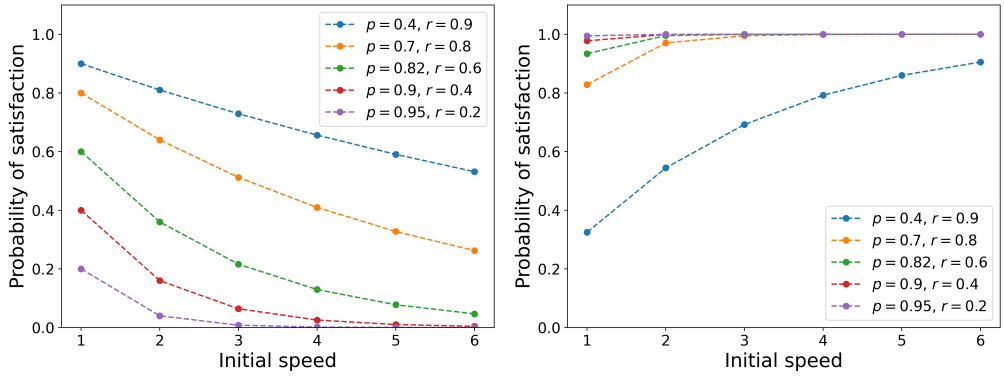
$$\begin{aligned}
&\texttt{TP} = r\,, && \texttt{FP} = \texttt{TP}(\frac{1}{p} - 1)\,, \\
&\texttt{TN} = 2 - \texttt{FP}\,, && \texttt{FN} = 1 - \texttt{TP}\,.
\end{aligned} \tag{2.15}$$

Note that this is one of many possible confusion matrices that could be constructed; we have chosen one of them for illustration, and we use it consistently across all

Table 2.1: Confusion matrices used in simulation for various precision-recall pairs, where TP, TN, FP, FN are given according to equation (2.15).

Predicted	True ($\texttt{CM}(p, r)$)		
	ped	obs	empty
ped	TP	FP/2	FP/2
obs	FN/2	4TN/10	TN/10
empty	FN/2	TN/10	4TN/10

precision/recall pairs.



(a) True environment: *ped* (b) True environment: *obs*

Figure 2.4: For class-labeled confusion matrices with precision-recall values derived according to Table 2.1. (a) Satisfaction probabilities that the car stops at C_{k-1} for $x_e = \texttt{ped}$ under various initial speeds and maximum speeds V_{max} such that $1 \leq V_{max} \leq 6$. (b) Satisfaction probabilities that the car does not stop at C_{k-1} for $x_e = \texttt{obs}$ under various initial speeds and maximum speeds V_{max} such that $1 \leq V_{max} \leq 6$.

nuScenes Dataset: We choose nuScenes [64] to illustrate the metrics introduced in this chapter on a real-world dataset. We choose a state-of-the-art PointPillars detection model for nuScenes that uses the LiDAR modality [65, 66]. The pre-trained model[1] is evaluated on the validation split of the full nuScenes dataset. The resulting dataset has 6019 pointcloud samples, with annotated objects common to urban settings such as pedestrians, cars, trucks, among others. For this detection model, we tabulate the evaluation results according to the various confusion matrices discussed so far. For the car-pedestrian example and its controller described previously, we compute the system-level guarantees, i.e., the probability that the car will satisfy the safety requirements in equations (2.1)-(2.3), given the confu-

[1] Available open source at this Github repository https://github.com/open-mmlab/mmdetection3d/tree/main/configs/pointpillars.

Prediction	True Label								
	$1 \leq d \leq 10$			$11 \leq d \leq 20$			$21 \leq d \leq 30$		
	ped	obs		ped	obs		ped	obs	
ped	1849	11	369	5443	87	963	4290	271	943
obs	56	5697	47	45	12406	354	191	12939	762
	1002	621	6117	2734	1949	12668	2406	3647	8969
	$31 \leq d \leq 40$			$41 \leq d \leq 50$			$51 \leq d \leq 60$		
	ped	obs		ped	obs		ped	obs	
ped	3302	382	213	0	0	0	0	0	0
obs	358	10670	345	0	6981	252	0	0	0
	1824	4358	1285	0	4346	709	0	0	0

Table 2.2: Class labeled confusion matrix, parametrized by distance computed from the full nuScenes dataset for the Pointpillars model

sion matrices from various evaluations of the detection model. Each discrete state corresponds abstracts a 1m distance on the road.

Each scene is 20 seconds long, with 3D object annotations made at 2 Hz for 23 different classes. All objects with nuScenes annotation "human" are clustered under the class `ped`, and all objects annotated as "vehicle", static obstacles, and moving obstacles are annotated as `obs`. We use all 40 pointcloud frames from the `LIDAR-TOP` sensor in each scene to form our dataset $\mathcal{D}$. The LiDAR sweeps accompanying each scene provides distances of annotated objects from the ego vehicle. We use the birds-eye-view to compare predicted bounding boxes to the ground truth, comparing for both l_2-norm in x, y-positions as well as orientation error. These evaluations are used to construct the (distance-parametrized) class-labeled and proposition-labeled confusion matrices from Algorithms 1 and 2 with 10m distance intervals with parameters $D_0 = 0$ and $D_{k_{max}} = 100$m. The class-labeled and proposition-labeled confusion matrices for each distance bin are listed in Tables 2.2 and 2.3, respectively.

For proposition-labeled confusion matrices, ground truth annotations and predictions are grouped according to an occupancy patch that roughly covers the area occupied by the ego. Concretely, the radius band $D_k = (z_k, z_{k+1})$ is split into occupancy patches covering the area every $\theta = \frac{z_k}{2.5}$ radians. The arc length of 2.5 m is a user-specified parameter; here, it is chosen to roughly approximate the width of a car. Table 2.4 is the proposition-labeled confusion matrix, where in addition to distance, evaluations are grouped by the occupancy patch size. There is a considerable difference between the proposition-labeled confusion matrix that is and is not

Prediction	True Label							
	$1 \leq d \leq 10$				$11 \leq d \leq 20$			
	{}	{ped}	{obs}	{ped, obs}	{}	{ped}	{obs}	{ped, obs}
{}	0	141	94	6	0	110	139	24
{ped}	54	373	9	17	34	363	18	81
{obs}	20	3	2122	210	36	1	2301	388
{ped, obs}	0	3	104	415	1	18	233	1400
	$21 \leq d \leq 30$				$31 \leq d \leq 40$			
	{}	{ped}	{obs}	{ped, obs}	{}	{ped}	{obs}	{ped, obs}
{}	0	84	253	27	0	106	331	48
{ped}	34	246	34	74	31	241	45	128
{obs}	25	14	2109	443	17	12	2200	489
{ped, obs}	8	37	343	1565	0	42	245	1240
	$41 \leq d \leq 50$				$51 \leq d \leq 60$			
	{}	{ped}	{obs}	{ped, obs}	{}	{ped}	{obs}	{ped, obs}
{}	0	0	905	0	0	0	0	0
{ped}	0	0	0	0	0	0	0	0
{obs}	42	0	3396	0	0	0	0	0
{ped, obs}	0	0	0	0	0	0	0	0

Table 2.3: Proposition labeled confusion matrix, parametrized by distance computed from the full nuScenes dataset for the pretrained Pointpillars model

grouped according to an occupancy patch that is planner consistent. For example, consider the label $\{ped\}$ in the distance range $1 \leq d \leq 10$ in in the ungrouped (see Table 2.3) and the grouped proposition labeled confusion matrices (see Table 2.4). The true positive rate of matching the label is higher when atomic propositions are not grouped (see Tables 2.3 and 2.4). This is because the proposition must match in every occupancy patch, which is finer, as opposed to every radius band.

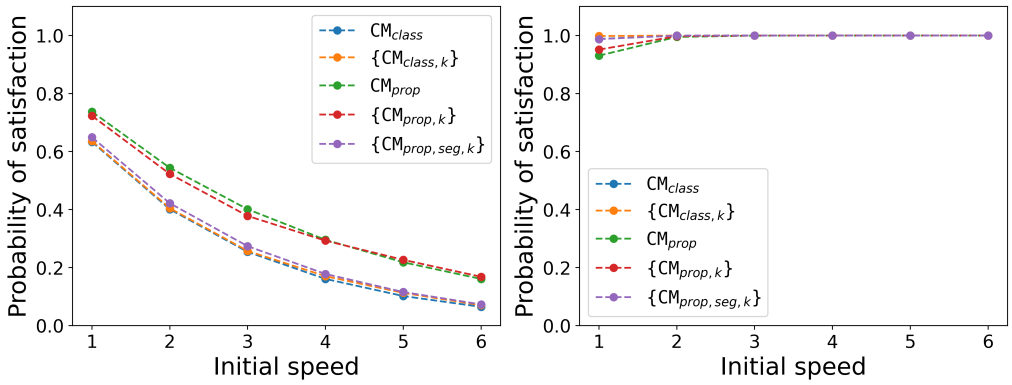
The satisfaction probabilities for the pedestrian case is shown in Figure 2.5a. The system-level satisfaction probability in the case of the true environment not having a pedestrian is given in Figure 2.5b. The full class and proposition labeled confusion matrices are given in Tables 2.5 and 2.6, respectively. The code for this chapter is given in the Python package, TRELPY and is available on GitHub[2]. In both the class labeled and proposition labeled confusion matrices, notice that after a distance of $50m$, there are no more detections output by the model, beyond which the nuScenes LiDAR data is sparse and cannot be reliably inferred from [64]; this is also nuScenes threshold for evaluation and objects beyond $50m$ are in the far-field and not annotated [67].

Figure 2.5 illustrates the importance of choosing perception metrics at the right level of fidelity. The proposition-labeled confusion matrix (green curve) and its dis-

[2] https://github.com/IowaState-AutonomousSystemsLab/TRELPy

Prediction	True Label							
	$1 \leq d \leq 10$				$11 \leq d \leq 20$			
	{}	{ped}	{obs}	{ped, obs}	{}	{ped}	{obs}	{ped, obs}
{}	0	344	280	6	0	1689	1658	10
{ped}	145	649	10	11	582	3183	77	28
{obs}	29	8	4006	133	262	29	11241	94
{ped, obs}	2	3	44	256	20	12	36	175
	$21 \leq d \leq 30$				$31 \leq d \leq 40$			
	{}	{ped}	{obs}	{ped, obs}	{}	{ped}	{obs}	{ped, obs}
{}	0	1615	3150	16	0	1316	3912	13
{ped}	697	2987	260	17	188	2368	353	22
{obs}	658	149	11878	58	317	286	9978	37
{ped, obs} 9	29	64	71	2	30	26	57	
	$41 \leq d \leq 50$				$51 \leq d \leq 60$			
	{}	{ped}	{obs}	{ped, obs}	{}	{ped}	{obs}	{ped, obs}
{}	0	0	4069	0	0	0	0	0
{ped}	0	0	0	0	0	0	0	0
{obs}	245	0	6706	0	0	0	0	0
{ped, obs}	0	0	0	0	0	0	0	0

Table 2.4: Proposition labeled confusion matrix, in which evaluations are groupedboth by distance as well as orientation from the ego. This matrix is derived for the full nuScenes dataset for the pre-trained Pointpillars model.



(a) True environment: *ped* (b) True environment: *obs*

Figure 2.5: System-level probabilistic guarantees for the car-pedestrian example. Figure 2.5a shows the satisfaction probability that the car stops at C_{k-1} for $x_e =$ ped under various initial speeds and maximum speeds V_{max} such that $1 \leq V_{max} \leq 6$. Figure 2.5b shows the satisfaction probability that the car does not stop at C_{k-1} for $x_e =$ obs under various initial speeds and maximum speeds V_{max} such that $1 \leq V_{max} \leq 6$.

Prediction	True Label		
	ped	obs	
ped	14884	751	2488
obs	650	48693	1760
	7966	14921	29748

Table 2.5: Class Labeled Confusion Matrix computed from the full nuScenes dataset for the Pointpillars model

Prediction	True Label			
	{*empty*}	{ped}	{obs}	{ped, obs}
{}	0	441	1722	105
{ped}	153	1223	106	300
{obs}	140	30	12128	1530
{ped, obs}	9	100	925	4620

Table 2.6: Proposition Labeled Confusion Matrix computed from the full nuScenes dataset for the Pointpillars model

tance parametrized counterpart (red curve) result in the highest system-level guarantees for the pedestrian case (see Figure 2.5a). In comparison, the class-labeled and proposition-labeled confusion matrices with grouped evaluations result in lower probabilities of satisfaction. While the class-labeled confusion matrix can result in overly conservative results, the proposition-labeled confusion matrices (without the grouped evaluations) might result in overly relaxed guarantees. For example, suppose there are multiple pedestrians in the radius band D_k, and the model detects just one pedestrian from the LiDAR data. If the pedestrian detected is one that is not going to interact with the car (e.g., it is located laterally distant from or behind the vehicle), then this detection is not safety-critical. However, this still gets counted as a true positive in the proposition-labeled confusion matrix. This coarseness is reduced when evaluations are grouped, especially in a manner consistent with the high-level planner's discrete abstraction. This can be seen in the satisfaction probabilities of the proposition-labeled confusion matrix computed from grouped evaluations (brown curve). This satisfaction probability lies between probability curves for the class-labeled and ungrouped proposition-labeled counterparts, thus illustrating the importance of choosing the right fidelity in grouping abstractions.

Sensitivity Analysis. this chapter is focused on highlighting the importance of system-level reasoning of determining perception metrics that are best suited for system-level analysis. The choice of a stronger object detection model would better highlight the strength of our evaluation framework, as illustrated in Figure 2.6. For each true positive rate for the pedestrian class, 20 random instances of the 4×4 proposition-labeled confusion matrix were generated. Even though the class-labeled confusion matrix is the most conservative, we observe that system-level satisfaction probability is close to 1 when the true positive rate is high ($> 99\%$).

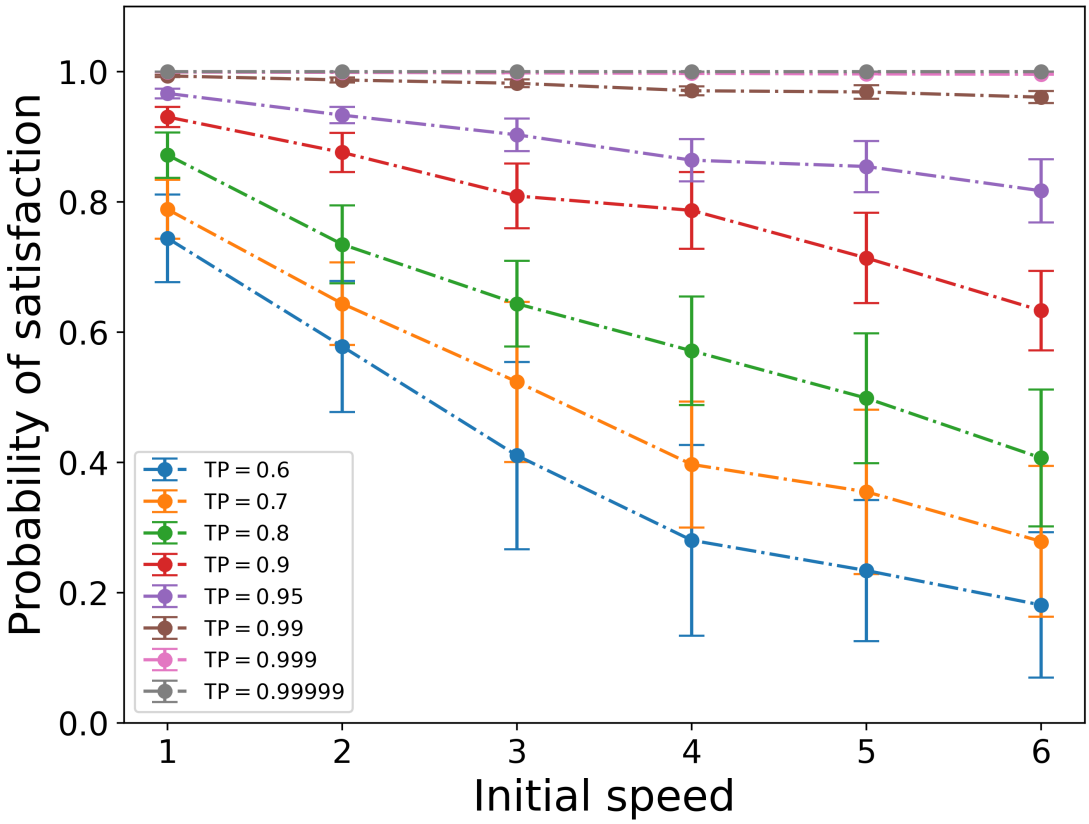


Figure 2.6: Sensitivity plots for satisfaction probability derived from proposition labeled confusion matrix for the specification that the car does not stop at C_{k-1} for $x_e = \texttt{ped}$ under various initial speeds and maximum speeds V_{max} such that $1 \leq V_{max} \leq 6$. The sensitivity is shown for varying true positive rates of detecting pedestrians.

2.7 Lower Bounds for Detection Metrics from System-level Guarantees

In this section, we will cover a case study to illustrate how system-level probabilistic guarantees can inform quantitative evaluation metrics for perception. In particular, we will derive lower bounds on detection metrics from desired system-level guarantees. This was implemented as a case study in the system design and analysis tool, Pacti [68].

Assume-guarantee contracts are a useful formalism to specify assumptions and guarantees of individual sub-systems or scenario viewpoints. Building on fundamentals in category theory, operators for composition, conjunction, refinement, quotient, and others can be rigorously defined over assume-guarantee contracts. This allows for formal reasoning about interactions between component implementations that respect assume-guarantee contracts, allowing for rigorous system

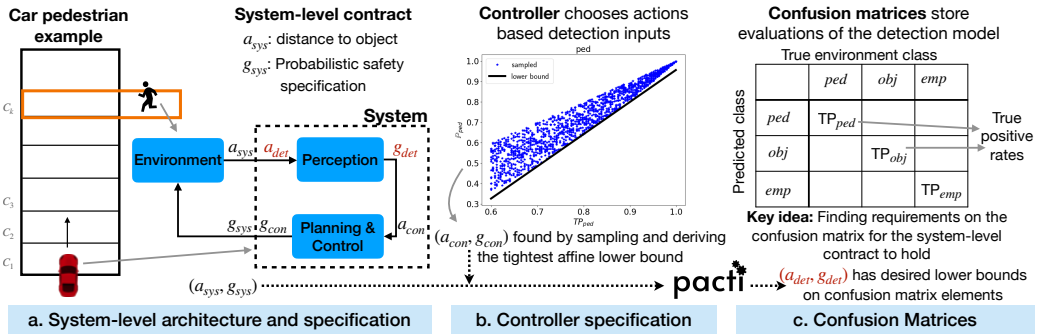


Figure 2.7: Given a system-level specification $\mathcal{C}_{\text{sys}} = (a_{\text{sys}}, g_{\text{sys}})$, and a specification for the controller $\mathcal{C}_{\text{con}} = (a_{\text{con}}, g_{\text{con}})$, derive the object detection specification $\mathcal{C}_{\text{det}} = (a_{\text{det}}, g_{\text{det}})$.

design. In addition to identifying requirements on perception systems from system-level guarantees of safety, we will use this formalism in Chapter 5 for designing compositional test plans.

Definition 2.16 (Assume-Guarantee Contract)**.** Let $\mathcal{B}$ be a universe of behaviors, then a *component* M is a set of behaviors $M \subseteq \mathcal{B}$. A *contract* is the pair $\mathcal{C} = (A, G)$, where A are the assumptions and G are the guarantees. A component E is an *environment* of the contract $\mathcal{C}$ if $E \models A$. A component M is an *implementation* of the contract, $M \models \mathcal{C}$ if $M \subseteq G \cup \neg A$, meaning the component provides the specified guarantees if it operates in an environment that satisfies its assumptions. There exists a partial order of contracts, we say $\mathcal{C}_1$ is a refinement of $\mathcal{C}_2$, denoted $\mathcal{C}_1 \leq \mathcal{C}_2$, if $(A_2 \leq A_1)$ and $(G_1 \cup \neg A_1 \leq G_2 \cup \neg A_2)$. We say a contract $\mathcal{C} = (A, G)$ is in canonical, or saturated, form if $\neg A \subseteq G$.

In this case study, we consider the design of a vehicle that has to satisfy a safety property with a given probability. We understand the vehicle as a system that consists of two subsystems: a perception component (for object detection) and a controller, as shown in Figure 2.7a. From knowledge of a system-level safety contract and of the specification of the control component, the *quotient* operator is used to derive a specification for the perception component.

Consider the car-pedestrian example once again. We encode the notion of safety in linear temporal logic formulas φ_c, where $c \in \{\texttt{ped}, \texttt{obs}, \texttt{empty}\}$. This way, we can specify safe behavior when an element of each class is present on the crosswalk. We synthesized controllers to satisfy these safety properties, assuming perfect per-

ception. The details of the properties and our synthesis approach can be found in [55].

System-level contract. Let $\mathbb{P}_c$ be the probability that the car will satisfy requirement φ_c when the crosswalk object has true class c. We set the system-level contract to

$$\mathcal{C}_{\text{sys}} = (d_l \leq d \leq d_u,\ g_{\texttt{ped}} \wedge g_{\texttt{obs}} \wedge g_{\texttt{empty}}),$$

where $d_l,\ d_u$ are bounds on the distance d to the object in the crosswalk, and g_c characterizes an affine lower bound of $\mathbb{P}_c$. This system-level contract assumes bounded distance to the object of interest, and guarantees affine lower bounds (as a function of d) on probabilistic satisfaction of safety properties. In other words, at the system-level we allow the probability of satisfaction of the safety property to degrade if the vehicle is far away from the crosswalk.

Controller contract. As mentioned, we synthesize three controllers, each making sure that property φ_c would be satisfied under perfect perception. In order to write a contract for each of the controllers, we make use of the fact that the perception component is not perfect. As a result, the controller satisfies its safety specification probabilistically.

To correlate probabilities of property satisfaction to perception errors, we base our approach on [55, 69]. The satisfaction probability $\mathbb{P}_c$ for the safety property φ_c is computed by constructing a Markov chain with transition probabilities derived from the *true positive rates*[3] of the perception component, and then invoking standard statistical model-checking tools.

For this example, $\mathbb{P}_c$ depends mainly on the true positive rate $\texttt{TP}_c$ of the class c. We determine a tight affine lower bound for $\mathbb{P}_c$ as a function of $\texttt{TP}_c$ by sampling and solving a linear program. The data for the linear program is generated by sampling false negatives for each value of $\texttt{TP}_c$ and computing the corresponding $\mathbb{P}_c$ (see Figure 2.7b). This procedure yields the following controller contract corresponding to each object class c:

$$\mathcal{C}_{\text{c}} = (l_c \leq \texttt{TP}_c,\ a_c(\texttt{TP}_c) + b_c \leq \mathbb{P}_c), \tag{2.16}$$

where l_c, a_c, and b_c are reals. The three contracts are composed to find the overall control contract: $\mathcal{C}_{\text{con}} = \mathcal{C}_{\texttt{ped}} \parallel \mathcal{C}_{\texttt{obs}} \parallel \mathcal{C}_{\texttt{empty}}$.

[3] The true positive rate of a perception component for an object class is defined as the probability that the component correctly detects an object to be of that class.

Object detection contract. Now that we have the specifications for the system and for the three controllers, we use contract operations to obtain the specification of the perception component. The detection component contract is found via the quotient $\mathcal{C}_{\text{det}} = \mathcal{C}_{\text{sys}}/\mathcal{C}_{\text{con}}$, where $\mathcal{C}_{\text{sys}}$ is the system-level contract and $\mathcal{C}_{\text{con}}$ is the controller contract. $\mathcal{C}_{\text{det}}$ imposes lower bounds on the true positive rates $\texttt{TP}_c$ of each object class c. We illustrate the results numerically for an instance of the car-pedestrian example. The system contract is set to

$$\begin{aligned}\mathcal{C}_{sys} = (1 \leq d \leq 10,\, 0.99(1-0.1d) \leq \mathbb{P}_{\texttt{ped}} \wedge\, 0.8(1-0.1d) \leq \mathbb{P}_{\texttt{obs}} \\ \wedge\, 0.95(1-0.1d) \leq \mathbb{P}_{\texttt{empty}}),\end{aligned} \tag{2.17}$$

that is, the contract assumes the distance to the crosswalk is bounded between 1 and 10 units, and specifies desired system-level probabilities $\mathbb{P}_c$ as a function of distance d. The controller contracts are computed to be $\mathcal{C}_{\texttt{ped}} = (0.6 \leq \texttt{TP}_{\texttt{ped}},\, 1.58\texttt{TP}_{\texttt{ped}} - 0.622 \leq \mathbb{P}_{\texttt{ped}})$, $\mathcal{C}_{\texttt{obs}} = (0.3 \leq \texttt{TP}_{\texttt{obs}},\, 0.068\texttt{TP}_{\texttt{obs}} + 0.93 \leq \mathbb{P}_{\texttt{obs}})$, and $\mathcal{C}_{\texttt{empty}} = (0.6 \leq \texttt{TP}_{\texttt{empty}},\, 0.2\texttt{TP}_{\texttt{empty}} + 0.799 \leq \mathbb{P}_{\texttt{empty}})$. These contracts impose affine lower bounds on $\mathbb{P}_c$ with respect to the true positive rates $\texttt{TP}_c$. The quotient results in an object detection contract with true positive rates lower bounded by affine functions of the distance d:

$$\begin{aligned}\mathcal{C}_{\text{det}} = (1 \leq d \leq 10,\, (1.02 - 0.063d \leq \texttt{TP}_{\texttt{ped}}) \wedge\, (0.6 \leq \texttt{TP}_{\texttt{ped}}) \wedge \\ (0.3 \leq \texttt{TP}_{\texttt{obs}}) \wedge\, (0.6 \leq \texttt{TP}_{\texttt{empty}})).\end{aligned} \tag{2.18}$$

Now that we have obtained the contract for the perception component, we can give this contract to designers responsible for object detection. The designers can develop the perception component and verify that it satisfies the requirements on true positive bounds as in $\mathcal{C}_{\text{det}}$. If it does, we can infer that the overall system with controller designed according to $\mathcal{C}_{\text{con}}$ will satisfy the system-level requirements.

2.8 Conclusion

The main takeaway of this chapter is that evaluation metrics for perception tasks should be informed by the downstream control logic as well as system-level metrics of safety. We focused on the object detection and classification task of perception, and made the following contributions. First, we proposed the idea of using confusion matrices as probabilistic models of sensor error to inform how system-level guarantees must be computed. Second, we replaced the labels of the confusion matrix with atomic propositions that are used in the system-level specifications and the downstream planner. Third, we finetuned the proposition-labeled confusion matrix

by grouping evaluations to an abstraction that is consistent with the occupancy size of the vehicle. Fourth, we illustrated how assume-guarantee contracts, or system design optimization tools in general, can leverage our framework to inform desired evaluation criteria for the percpetion module from system-level guarantees. Finally, we evaluated a state-of-the art detection model on the nuScenes dataset according to these metrics, and computed the corresponding system-level guarantees for a discrete-state car-pedestrian example.

There are several exciting directions for future work. As illustrated in Figure 2.5, the satisfaction probabilities of safety requirements are still relatively low compared to the high levels of safety guarantees (e.g., $1 - 10^{-6}$ to $1 - 10^{-9}$) that are often expected in these applications. This is for several reasons. First, we evaluated a model trained on one modality (3D object detection from pointcloud); typically the best models are multi-modal and use data from several different sensors. Secondly, we do not consider tracking in our evaluation; once an object is detected, it is tracked across frames and an object misdetected in a single frame need not drastically change the high-level plan. Given the sensitivity analysis, we expect the satisfaction trends to improve with the aforementioned extensions and with better object detection models.

In addition, this paradigm can be extended to evaluate other perception tasks in a task-relevant manner, to handle scenarios with dynamic environments, to synthesize controllers that are optimal for a given perception model, and to validate the framework via experimental demonstrations. For this, we will consider building on the work in [54], which studies quantitative analysis of systems that operate in partially known dynamic environments. It assumes that the environment model belongs to a set $\mathbf{M}^{env}$ of Markov chains. The system does not know the true model of the environment, and instead maintains a belief, which is defined as a probability distribution over all possible environment models in $\mathbf{M}^{env}$. We will extend our work to derive the belief update function based on the perception performance.

Bibliography

[1] Zoox, "Putting Zoox to the test: preparing for the challenges of the road," 2021. https://zoox.com/journal/structured-testing/, Last accessed on 2024-04-11.

[2] Waymo, "A blueprint for av safety: Waymo's toolkit for building a credible safety case," 2020. https://waymo.com/blog/2023/03/a-blueprint-for-av-safety-waymos/#:~:text=A%20safety%20case%20for%20fully,evidence%20to%20support%20that%20determination., Last accessed on 2024-05-05.

[3] F. Favarò, L. Fraade-Blanar, S. Schnelle, T. Victor, M. Peña, J. Engstrom, J. Scanlon, K. Kusano, and D. Smith, "Building a credible case for safety: Waymo's approach for the determination of absence of unreasonable risk," 2023. www.waymo.com/safety.

[4] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[5] N. Webb, D. Smith, C. Ludwick, T. Victor, Q. Hommes, F. Favaro, G. Ivanov, and T. Daniel, "Waymo's safety methodologies and safety readiness determinations," 2020.

[6] I. S. Organization, "Road vehicles: Safety of the intended functionality (ISO Standard No. 21448:2022)," 2022. https://www.iso.org/standard/77490.html, Last accessed on 2024-04-11.

[7] L. Li, W.-L. Huang, Y. Liu, N.-N. Zheng, and F.-Y. Wang, "Intelligence testing for autonomous vehicles: A new approach," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 158–166, 2016.

[8] H. Winner, K. Lemmer, T. Form, and J. Mazzega, "Pegasus—first steps for the safe introduction of automated driving," in *Road Vehicle Automation 5*, pp. 185–195, Springer, 2019.

[9] "DARPA Urban Challenge." https://www.darpa.mil/about-us/timeline/darpa-urban-challenge.

[10] "Technical Evaluation Criteria." https://archive.darpa.mil/grandchallenge/rules.html.

[11] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[12] J. Eskenazi and W. Jarett, "Explore: See the 55 reports — so far — of robot cars interfering with SF fire dept.," 2023. https://missionlocal.org/2023/08/cruise-waymo-autonomous-vehicle-robot-taxi-driverless-car-reports-san-francisco/, Last accessed on 2024-04-11.

[13] H. Zhao, S. K. Sastry Hari, T. Tsai, M. B. Sullivan, S. W. Keckler, and J. Zhao, "Suraksha: A framework to analyze the safety implications of perception design choices in avs," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 434–445, 2021.

[14] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[15] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[16] M. Lahijanian, S. B. Andersson, and C. Belta, "A probabilistic approach for control of a stochastic system from LTL specifications," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 2236–2241, IEEE, 2009.

[17] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, pp. 81–87, IEEE, 2014.

[18] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[19] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*, pp. 97–117, Springer, 2017.

[20] M. Fazlyab, M. Morari, and G. J. Pappas, "Probabilistic verification and reachability analysis of neural networks via semidefinite programming," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 2726–2731, IEEE, 2019.

[21] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *IEEE Transactions on Automatic Control*, 2020.

[22] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The neural network verification tool for

deep neural networks and learning-enabled cyber-physical systems," in *International Conference on Computer Aided Verification*, pp. 3–17, Springer, 2020.

[23] T. Dreossi, S. Jha, and S. A. Seshia, "Semantic adversarial deep learning," in *International Conference on Computer Aided Verification*, pp. 3–26, Springer, 2018.

[24] S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte, and X. Yue, "Formal specification for deep neural networks," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 20–34, Springer, 2018.

[25] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," *Journal of Automated Reasoning*, vol. 63, no. 4, pp. 1031–1053, 2019.

[26] S. Topan, K. Leung, Y. Chen, P. Tupekar, E. Schmerling, J. Nilsson, M. Cox, and M. Pavone, "Interaction-dynamics-aware perception zones for obstacle detection safety evaluation," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1201–1210, IEEE, 2022.

[27] K. Chakraborty and S. Bansal, "Discovering closed-loop failures of vision-based controllers via reachability analysis," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2692–2699, 2023.

[28] A. Dokhanchi, H. B. Amor, J. V. Deshmukh, and G. Fainekos, "Evaluating perception systems for autonomous vehicles using quality temporal logic," in *International Conference on Runtime Verification*, pp. 409–416, Springer, 2018.

[29] A. Balakrishnan, A. G. Puranic, X. Qin, A. Dokhanchi, J. V. Deshmukh, H. B. Amor, and G. Fainekos, "Specifying and evaluating quality metrics for vision-based perception systems," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1433–1438, IEEE, 2019.

[30] B. Bauchwitz and M. Cummings, "Evaluating the reliability of Tesla model 3 driver assist functions," 2020.

[31] H. Kress-Gazit, D. C. Conner, H. Choset, A. A. Rizzi, and G. J. Pappas, "Courteous cars," *IEEE Robotics & Automation Magazine*, vol. 15, no. 1, pp. 30–38, 2008.

[32] H. Kress-Gazit and G. J. Pappas, "Automatically synthesizing a planning and control subsystem for the DARPA Urban Challenge," in *2008 IEEE International Conference on Automation Science and Engineering*, pp. 766–771, IEEE, 2008.

[33] T. Wongpiromsarn, S. Karaman, and E. Frazzoli, "Synthesis of provably correct controllers for autonomous vehicles in urban environments," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1168–1173, IEEE, 2011.

[34] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on Robot Learning*, pp. 1–16, PMLR, 2017.

[35] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and scene generation," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 63–78, 2019.

[36] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 254–257, Springer, 2011.

[37] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[38] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel, "Verification of automotive control applications using s-taliro," in *2012 American Control Conference (ACC)*, pp. 3567–3572, IEEE, 2012.

[39] S. Sankaranarayanan and G. Fainekos, "Falsification of temporal properties of hybrid systems using the cross-entropy method," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pp. 125–134, 2012.

[40] S. Bak, S. Bogomolov, A. Hekal, N. Kochdumper, E. Lew, A. Mata, and A. Rahmati, "Falsification using reachability of surrogate koopman models," in *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[41] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*, pp. 167–170, Springer, 2010.

[42] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1555–1562, IEEE, 2018.

[43] C. Menghi, P. Arcaini, W. Baptista, G. Ernst, G. Fainekos, F. Formica, S. Gon, T. Khandait, A. Kundu, G. Pedrielli, *et al.*, "Arch-comp 2023 category report: Falsification," in *10th International Workshop on Applied Verification of Continuous and Hybrid Systems. ARCH23*, vol. 96, pp. 151–169, 2023.

[44] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *International Conference on Computer Aided Verification*, pp. 432–442, Springer, 2019.

[45] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validatio," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 163–168, IEEE, 2019.

[46] S. Feng, H. Sun, X. Yan, H. Zhu, Z. Zou, S. Shen, and H. X. Liu, "Dense reinforcement learning for safety validation of autonomous vehicles," *Nature*, vol. 615, no. 7953, pp. 620–627, 2023.

[47] X. Qin, N. Arechiga, J. Deshmukh, and A. Best, "Robust testing for cyber-physical systems using reinforcement learning," in *Proceedings of the 21st ACM-IEEE International Conference on Formal Methods and Models for System Design*, MEMOCODE '23, (New York, NY, USA), p. 36–46, Association for Computing Machinery, 2023.

[48] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Toward verified artificial intelligence," *Commun. ACM*, vol. 65, p. 46–55, jun 2022.

[49] B. Johnson and H. Kress-Gazit, "Probabilistic analysis of correctness of high-level robot behavior with sensor error," 2011.

[50] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[51] X. Wang, R. Li, B. Yan, and O. Koyejo, "Consistent classification with generalized metrics," 2019.

[52] P. Antonante, H. Nilsen, and L. Carlone, "Monitoring of perception systems: Deterministic, probabilistic, and learning-based fault detection and identification," *arXiv preprint arXiv:2205.10906*, 2022.

[53] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, pp. 1–11, 2019.

[54] T. Wongpiromsarn and E. Frazzoli, "Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 7644–7651, 2012.

[55] A. Badithela, T. Wongpiromsarn, and R. M. Murray, "Leveraging classification metrics for quantitative system-level analysis with temporal logic specifications," in *2021 60th IEEE Conference on Decision and Control (CDC)*, (Austin, TX, USA (virtual)), pp. 564–571, IEEE, 2021.

[56] C. S. Pasareanu, R. Mangal, D. Gopinath, S. G. Yaman, C. Imrie, R. Calinescu, and H. Yu, "Closed-loop analysis of vision-based autonomous systems: A case study," *arXiv preprint arXiv:2302.04634*, 2023.

[57] S. Beland, I. Chang, A. Chen, M. Moser, J. Paunicka, D. Stuart, J. Vian, C. Westover, and H. Yu, "Towards assurance evaluation of autonomous systems," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–6, 2020.

[58] Y. V. Pant, H. Abbas, K. Mohta, R. A. Quaye, T. X. Nghiem, J. Devietti, and R. Mangharam, "Anytime computation and control for autonomous systems," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 768–779, 2021.

[59] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "Diffstack: A differentiable and modular control stack for autonomous vehicles," in *Proceedings of The 6th Conference on Robot Learning* (K. Liu, D. Kulic, and J. Ichnowski, eds.), vol. 205 of *Proceedings of Machine Learning Research*, pp. 2170–2180, PMLR, 14–18 Dec 2023.

[60] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[61] O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon, "Consistent multilabel classification.," in *NeurIPS*, vol. 29, (Palais des Congrès de Montréal, Montréal CANADA), pp. 3321–3329, Advances in Neural Information Processing Systems, 2015.

[62] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *International conference on computer aided verification*, pp. 585–591, Springer, 2011.

[63] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A Storm is coming: A modern probabilistic model checker," in *International Conference on Computer Aided Verification*, pp. 592–600, Springer, 2017.

[64] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11621–11631, 2020.

[65] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 12689–12697, IEEE Computer Society, jun 2019.

[66] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection." https://github.com/open-mmlab/mmdetection3d, 2020.

[67] S. Gupta, J. Kanjani, M. Li, F. Ferroni, J. Hays, D. Ramanan, and S. Kong, "Far3det: Towards far-field 3d detection," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, (Los Alamitos, CA, USA), pp. 692–701, IEEE Computer Society, jan 2023.

[68] I. Incer, A. Badithela, J. Graebener, P. Mallozzi, A. Pandey, S.-J. Yu, A. Benveniste, B. Caillaud, R. M. Murray, A. Sangiovanni-Vincentelli, *et al.*, "Pacti: Scaling assume-guarantee reasoning for system analysis and design," *arXiv preprint arXiv:2303.17751*, 2023.

[69] A. Badithela, T. Wongpiromsarn, and R. M. Murray, "Evaluation metrics of object detection for quantitative system-level analysis of safety-critical autonomous systems," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Detroit, MI, USA), p. To Appear., IEEE, 2023.

[70] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 92–106, Springer, 2010.

[71] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Falsification of ltl safety properties in hybrid systems," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 4, pp. 305–320, 2013.

[72] G. Chou, Y. E. Sahin, L. Yang, K. J. Rutledge, P. Nilsson, and N. Ozay, "Using control synthesis to generate corner cases: A case study on autonomous driving," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2906–2917, 2018.

[73] T. Wongpiromsarn, M. Ghasemi, M. Cubuktepe, G. Bakirtzis, S. Carr, M. O. Karabag, C. Neary, P. Gohari, and U. Topcu, "Formal methods for autonomous systems," *arXiv preprint arXiv:2311.01258*, 2023.

[74] G. Fainekos, H. Kress-Gazit, and G. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 4885–4890, 2005.

[75] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "Paracosm: A language and tool for testing autonomous driving systems," *arXiv preprint arXiv:1902.01084*, 2019.

[76] L. Tan, O. Sokolsky, and I. Lee, "Specification-based testing with linear temporal logic," in *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, 2004. IRI 2004.*, pp. 493–498, IEEE, 2004.

[77] G. Fraser and F. Wotawa, "Using LTL rewriting to improve the performance of model-checker based test-case generation," in *Proceedings of the 3rd International Workshop on Advances in Model-Based Testing*, pp. 64–74, 2007.

[78] G. Fraser and P. Ammann, "Reachability and propagation for LTL requirements testing," in *2008 The Eighth International Conference on Quality Software*, pp. 189–198, IEEE, 2008.

[79] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[80] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[81] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2208–2224, 2019.

[82] R. Bloem, G. Fey, F. Greif, R. Könighofer, I. Pill, H. Riener, and F. Röck, "Synthesizing adaptive test strategies from temporal logic specifications," *Formal methods in system design*, vol. 55, no. 2, pp. 103–135, 2019.

[83] J. Tretmans, "Conformance testing with labelled transition systems: Implementation relations and test generation," *Computer Networks and ISDN Systems*, vol. 29, no. 1, pp. 49–79, 1996.

[84] B. K. Aichernig, H. Brandl, E. Jöbstl, W. Krenn, R. Schlick, and S. Tiran, "Killing strategies for model-based mutation testing," *Software Testing, Verification and Reliability*, vol. 25, no. 8, pp. 716–748, 2015.

[85] R. Hierons, "Applying adaptive test cases to nondeterministic implementations," *Information Processing Letters*, vol. 98, no. 2, pp. 56–60, 2006.

[86] A. Petrenko and N. Yevtushenko, "Adaptive testing of nondeterministic systems with FSM," in *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering*, pp. 224–228, IEEE, 2014.

[87] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 179–190, 1989.

[88] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[89] M. Yannakakis, "Testing, optimization, and games," in *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.*, pp. 78–88, IEEE, 2004.

[90] L. Nachmanson, M. Veanes, W. Schulte, N. Tillmann, and W. Grieskamp, "Optimal strategies for testing nondeterministic systems," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 4, pp. 55–64, 2004.

[91] A. David, K. G. Larsen, S. Li, and B. Nielsen, "Cooperative testing of timed systems," *Electronic Notes in Theoretical Computer Science*, vol. 220, no. 1, pp. 79–92, 2008.

[92] E. Bartocci, R. Bloem, B. Maderbacher, N. Manjunath, and D. Ničković, "Adaptive testing for specification coverage in CPS models," *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 229–234, 2021.

[93] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.

[94] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science Robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[95] H. Zhang, M. Fontaine, A. Hoover, J. Togelius, B. Dilkina, and S. Nikolaidis, "Video game level repair via mixed integer linear programming," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, pp. 151–158, 2020.

[96] M. Fontaine, Y.-C. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, "On the Importance of Environments in Human-Robot Coordination," in *Proceedings of Robotics: Science and Systems*, (Virtual), July 2021.

[97] J. R. Büchi, *On a Decision Method in Restricted Second Order Arithmetic*, pp. 425–435. New York, NY: Springer New York, 1990.

[98] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, É. Renault, and L. Xu, "Spot 2.0 — a framework for ltl and omega-automata manipulation," in *Automated Technology for Verification and Analysis* (C. Artho, A. Legay, and D. Peled, eds.), (Cham), pp. 122–129, Springer International Publishing, 2016.

[99] F. Fuggitti, "Ltlf2dfa," June 2020.

[100] S. Bansal, Y. Li, L. Tabajara, and M. Vardi, "Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 9766–9774, Apr. 2020.

[101] N. Klarlund and A. Møller, *MONA Version 1.4 User Manual*. BRICS, Department of Computer Science, University of Aarhus, January 2001. Notes Series NS-01-1. Available from `http://www.brics.dk/mona/`.

[102] D. Goktas and A. Greenwald, "Convex-concave min-max Stackelberg games," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[103] I. Tsaknakis, M. Hong, and S. Zhang, "Minimax problems with coupled linear constraints: computational complexity, duality and solution methods," *arXiv preprint arXiv:2110.11210*, 2021.

[104] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, vol. 67. Springer Science & Business Media, third ed., 2021.

[105] V. V. Vazirani, *Approximation algorithms*, vol. 1. Springer, 2001.

[106] M. Fischetti and M. Monaci, "A branch-and-cut algorithm for mixed-integer bilinear programming," *European Journal of Operational Research*, vol. 282, no. 2, pp. 506–514, 2020.

[107] J. B. Graebener, A. S. Badithela, D. Goktas, W. Ubellacker, E. V. Mazumdar, A. D. Ames, and R. M. Murray, "Flow-based synthesis of reactive tests for discrete decision-making systems with temporal logic specifications," *arXiv preprint arXiv:2404.09888*, 2024.

[108] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "Tulip: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pp. 313–314, 2011.

[109] I. Filippidis, S. Dathathri, S. C. Livingston, N. Ozay, and R. M. Murray, "Control design for hybrid systems with tulip: The temporal logic planning toolbox," in *2016 IEEE Conference on Control Applications (CCA)*, pp. 1030–1041, IEEE, 2016.

[110] S. Maoz and J. O. Ringert, "Gr (1) synthesis for ltl specification patterns," in *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pp. 96–106, 2015.

[111] S. A. Cook, "The complexity of theorem-proving procedures," in *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, pp. 143–152, 2023.

[112] C. H. Papadimitriou, *Computational complexity*, p. 260–265. GBR: John Wiley and Sons Ltd., 2003.

[113] W. Ubellacker and A. D. Ames, "Robust locomotion on legged robots through planning on motion primitive graphs," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12142–12148, 2023.

[114] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023.

[115] E. W. Dijkstra, "Guarded commands, nondeterminacy and formal derivation of programs," *Communications of the ACM*, vol. 18, no. 8, pp. 453–457, 1975.

[116] L. Lamport, "win and sin: Predicate transformers for concurrency," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 12, no. 3, pp. 396–428, 1990.

[117] B. Meyer, "Applying 'design by contract'," *Computer*, vol. 25, no. 10, pp. 40–51, 1992.

[118] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis, "Multiple viewpoint contract-based specification and design," in *Formal Methods for Components and Objects: 6th International Symposium, FMCO 2007, Amsterdam, The Netherlands, October 24-26, 2007, Revised Lectures* (F. S. de Boer, M. M. Bonsangue, S. Graf, and W.-P. de Roever, eds.), (Berlin, Heidelberg), pp. 200–225, Springer Berlin Heidelberg, 2008.

[119] A. L. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-based design for cyber-physical systems," *Eur. J. Control*, vol. 18, no. 3, pp. 217–238, 2012.

[120] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2104–2132, 2015.

[121] I. Incer, *The Algebra of Contracts*. PhD thesis, EECS Department, University of California, Berkeley, May 2022.

[122] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. L. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, K. G. Larsen, *et al.*, "Contracts for system design," *Foundations and Trends in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[123] I. Incer, A. L. Sangiovanni-Vincentelli, C.-W. Lin, and E. Kang, "Quotient for assume-guarantee contracts," in *16th ACM-IEEE International Conference on Formal Methods and Models for System Design*, MEMOCODE'18, pp. 67–77, October 2018.

[124] R. Passerone, Í. Íncer Romeo, and A. L. Sangiovanni-Vincentelli, "Coherent extension, composition, and merging operators in contract models for system design," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–23, 2019.

[125] R. Negulescu, "Process spaces," in *CONCUR 2000 — Concurrency Theory* (C. Palamidessi, ed.), (Berlin, Heidelberg), pp. 199–213, Springer Berlin Heidelberg, 2000.

[126] J. B. Graebener^*, A. Badithela^*, and R. M. Murray, "Towards better test coverage: Merging unit tests for autonomous systems," in *NASA Formal Methods* (J. V. Deshmukh, K. Havelund, and I. Perez, eds.), (Cham), pp. 133–155, Springer International Publishing, 2022. A. Badithela and J.B. Graebener contributed equally to this work.

[127] R. Bloem, B. Könighofer, R. Könighofer, and C. Wang, "Shield synthesis," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 533–548, Springer, 2015.

[128] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, pp. 282–293, Springer, 2006.

[129] I. Incer, L. Mangeruca, T. Villa, and A. Sangiovanni-Vincentelli, "The quotient in preorder theories," *arXiv:2009.10886*, 2020.

[130] O. Hussien, A. Ames, and P. Tabuada, "Abstracting partially feedback linearizable systems compositionally," *IEEE Control Systems Letters*, vol. 1, no. 2, pp. 227–232, 2017.

[131] P. Tabuada, G. J. Pappas, and P. Lima, "Composing abstractions of hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 436–450, Springer, 2002.

[132] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, (New York, NY, USA), p. 58–67, Association for Computing Machinery, 2015.

[133] J. Liu and N. Ozay, "Abstraction, discretization, and robustness in temporal logic control of dynamical systems," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pp. 293–302, 2014.