View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarlyCommons@Penn

University of Pennsylvania
ScholarlyCommons

Departmental Papers (CIS) Department of Computer & Information Science

11-2020

Compositional Probabilistic Analysis of Temporal Properties over Stochastic Detectors

Ivan Ruchkin
University of Pennsylvania, iruchkin@seas.upenn.edu

Oleg Sokolsky
University of Pennsylvania, sokolsky@cis.upenn.edu

James Weimer
University of Pennsylvania, weimerj@cis.upenn.edu

Tushar Hedaoo
University of Pennsylvania, tghedaoo@seas.upenn.edu

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Follow this and additional works at: https://repository.upenn.edu/cis_papers

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation

Ivan Ruchkin, Oleg Sokolsky, James Weimer, Tushar Hedaoo, and Insup Lee, "Compositional Probabilistic Analysis of Temporal Properties over Stochastic Detectors", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 39(11), 3288-3299. November 2020.

This article was presented at the International Conference on Embedded Software 2020; original version appears as part of the ESWEEK-TCAD special issue.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_papers/863
For more information, please contact repository@pobox.upenn.edu.

Compositional Probabilistic Analysis of Temporal Properties over Stochastic Detectors

Abstract

Run-time monitoring is a vital part of safety-critical systems. However, early-stage assurance of monitoring quality is currently limited: it relies either on complex models that might be inaccurate in unknown ways, or on data that would only be available once the system has been built. To address this issue, we propose a compositional framework for modeling and analysis of noisy monitoring systems. Our novel 3-value detector model uses probability spaces to represent atomic (non-composite) detectors, and it composes them into a temporal logic-based monitor. The error rates of these monitors are estimated by our analysis engine, which combines symbolic probability algebra, independence inference, and estimation from labeled detection data. Our evaluation on an autonomous underwater vehicle found that our framework produces accurate estimates of error rates while using only detector traces, without any monitor traces. Furthermore, when data is scarce, our approach shows higher accuracy than non-compositional data-driven estimates from monitor traces. Thus, this work enables accurate evaluation of logical monitors in early design stages before deploying them.

Disciplines

Computer Engineering | Computer Sciences

Comments

This article was presented at the International Conference on Embedded Software 2020; original version appears as part of the ESWEEK-TCAD special issue.

This journal article is available at ScholarlyCommons: https://repository.upenn.edu/cis_papers/863

Compositional Probabilistic Analysis of Temporal Properties over Stochastic Detectors

Ivan Ruchkin, Oleg Sokolsky, James Weimer,
Tushar Hedaoo, and Insup Lee
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, USA
{iruchkin,sokolsky,weimerj,tghedaoo,lee}@seas.upenn.edu[*]

Abstract

Run-time monitoring is a vital part of safety-critical systems. However, early-stage assurance of monitoring quality is currently limited: it relies either on complex models that might be inaccurate in unknown ways, or on data that would only be available once the system has been built. To address this issue, we propose a compositional framework for modeling and analysis of noisy monitoring systems. Our novel 3-value detector model uses probability spaces to represent atomic (non-composite) detectors, and it composes them into a temporal logic-based monitor. The error rates of these monitors are estimated by our analysis engine, which combines symbolic probability algebra, independence inference, and estimation from labeled detection data. Our evaluation on an autonomous underwater vehicle found that our framework produces accurate estimates of error rates while using only detector traces, without any monitor traces. Furthermore, when data is scarce, our approach shows higher accuracy than non-compositional data-driven estimates from monitor traces. Thus, this work enables accurate evaluation of logical monitors in early design stages before deploying them.

[*]This article was presented at the International Conference on Embedded Software 2020; original version appears as part of the ESWEEK-TCAD special issue.

1 Introduction

Many cyber-physical systems (CPS) operate in challenging safety-critical contexts, such as roadways, urban airspaces, and coastal waters. Not only is the control of the CPS expected to be safe, but also safety monitoring is necessary to trigger fail-safes and notify human operators. This monitoring typically detects whether the CPS has experienced a fault/failure or finds itself in an environment where it may be unsafe. Such environments include plants with hardware malfunctions, severe weather conditions, and unmapped physical locations.

In the last decade, the runtime verification research has developed formal techniques and tools to increase confidence in the system's safety [5,15]. Logically and temporally related events can be monitored based on specifications in Linear Temporal Logic (LTL) [20] and its descendants. When observations violate these specifications, the conditions are possibly unsafe and the system should execute a fail-safe fallback maneuver (e.g., stopping or landing).

Runtime safety monitors rely on inputs from noisy and potentially unreliable perception mechanisms. For example, the algorithms processing visual sensor data may randomly fail to detect features, or instead report non-existent artifacts [24]. Further, some perception systems are based on statistical tests, which at times may not have sufficient data for a conclusive answer. Such detectors feed stochastic errors into monitors, which become difficult to analyze only using non-deterministic logical constructs: they do not distinguish the more and less likely scenarios. Thus, we require *probabilistic guarantees for logical monitors.* These guarantees often take the form of false positive and false negative rates.

Unfortunately, the existing work falls short of providing such guarantees given perception uncertainties, especially in early design stages when data is scarce. One way to address this problem is model-driven: probability constructs are embedded into a logical property (e.g., in PCTL [14]), and the property is model-checked on a probabilistic dynamics model. This method requires a detailed closed-loop model of the whole system — not only of the perception mechanism, but also of the controller and the environment. Such models are rarely available in practice, and even when available they tend to be inaccurate, hard to validate, and lack scalability.

Another path to probabilistic monitoring guarantees is directly through the testing/execution data: safety monitors are treated as black-box binary random variables, thus turning the problem into observation of Bernoulli trials and estimating their error probabilities using standard statistical techniques [11]. In this case, the binary variable representation unnecessarily abstracts away our knowledge about the perception uncertainties and the logical structure of the monitor. This method requires substantial and specific data that may be difficult to collect; e.g., the system needs to be repeatedly put in an unsafe state to calculate the false negative rate of a safety monitor. Usually such data is unavailable in early design stages.

In this paper, we present an approach to *estimating the error rates of logic-based runtime monitors*. We combine the strengths of model-based and data-driven approaches by taking advantage of logical specifications and using only perception (non-monitor) data — without any closed-loop system models. Thus, we can evaluate monitors before they are built.

Shown in Figure 1, our approach consists of two parts: *modeling* and *analysis*. Inspired by statistical hypothesis tests [8], the modeling part focuses on binary ground-truth conditions (e.g., whether a vehicle faces an obstacle), which are true or false at each time. At the heart of our model are *stochastic detectors* of those ground truths with three outcomes: true, false, and unknown. Logical monitors are modeled by composing, these detectors using LTL operators; the composed detection outcomes are determined by the 3-value[1] LTL [19], whereas the ground truths are composed by classic binary LTL.

The analysis part, performed offline by a heuristic computational assistant, treats a monitor's error rate as a probability in an outcome space composed from the detector spaces, in accordance with the monitor's logical definition. This probability is represented by a symbolic formula that can be decomposed into smaller probabilities of events associated with individual detectors. These probabilities (shown at the bottom on Figure 1) are easier to estimate because they require only labeled data from the detectors, but not monitors.

We evaluated our approach on a simulated underwater vehicle that detects and follows an underwater pipeline. One safety property we monitor is that, after losing sight of the pipeline, the vehicle rediscovers it within a given period of time; otherwise, the vehicle declares that it is lost, surfaces, and needs to be evacuated by a ship crew. We implemented a monitor for this property and applied our approach to estimate its false negative rate (along with the false positive rate of a different monitor, which checks

[1]We do *not* mean the 3-value inconclusiveness due to incomplete traces [6].

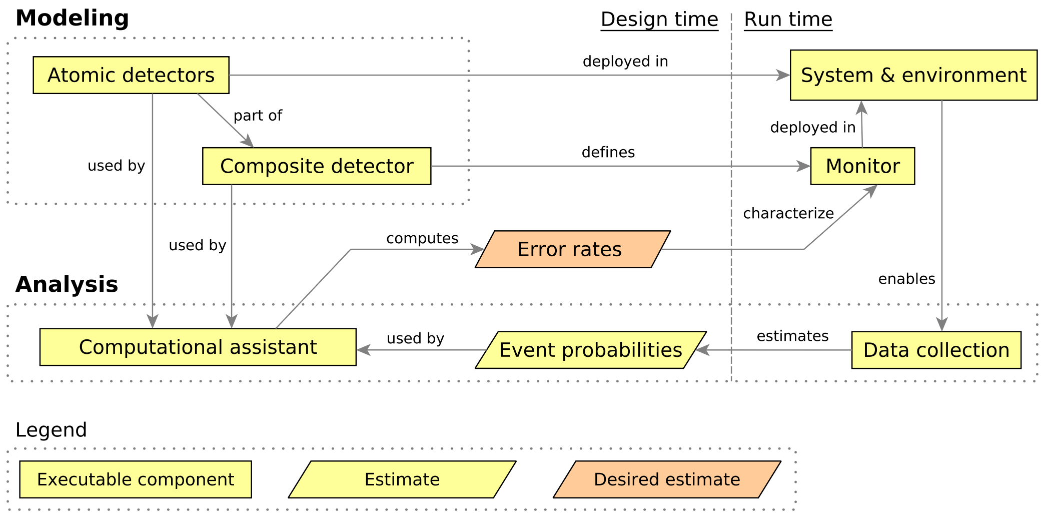

Figure 1: Our modeling and analysis framework.

whether the vehicle reliably follows the pipeline). This evaluation showed that our approach provides estimates on par with purely data-driven estimation, exceeds them when little data is available, reuses labeled detector data due to its compositional nature, and executes within seconds. However, the approach is dependent on accurate independence assumptions between detectors.

This paper makes three contributions:

- A probabilistic model of 3-outcome detectors (Section 4) based on a logic for these detectors (Section 3).
- An analysis of error rates based on probabilistic calculations and using labeled detection data (Section 5).
- An application of the above model and analysis to two monitors in a simulated underwater vehicle (Section 6).

This paper is accompanied by a supplement [21] with additional derivations and illustrations. The source code and data can be found at `https://github.com/bisc/prob-comp-asst`.

The next section introduces a running example.

2 Motivating Example

The DARPA Assured Autonomy program considers an *unmanned underwater vehicle* (UUV) that follows a continuous underwater pipeline and scans it for defects. To discover and track the pipeline, the vehicle uses two side-looking sonars pointing to the sides and downwards from the vehicle. Thus, going above the pipeline in parallel to it provides the necessary data. The quality of the pipeline is investigated manually offline, after the UUV mission is complete, by processing the collected scan data via synthetic aperture imaging. The UUV is also equipped with a forward-facing sonar for obstacle avoidance.

The UUV software is comprised of a perception subsystem, a controller, and a safety monitor. The perception subsystem continuously processes the sensor data from both side-scan sonars. Since the data is noisy, the pipeline detector (represented as a predicate $\boldsymbol{Pl}$ here) produces a potentially inaccurate 3-valued output at each moment:

- Confident that the pipeline is visible (value T).
- Confident that the pipeline is not visible (value F).
- Not confident in either outcome (value U).

This output of $\boldsymbol{Pl}$ is fed into the controller and the safety monitor. The former actuates the propeller and fins to follow the pipeline as well as possible. The goal of the latter is to determine when the UUV has definitively lost track of the pipeline and needs to be evacuated. The monitor is required to raise an alarm when the pipeline is not visible (i.e., F) for $d \in [5, 30]$ seconds. To define this monitor in classic LTL, we map T to the boolean truth value, and both F and U to the falsehood value. Thus, we write the following LTL formula, from which a monitor[2] can be automatically generated [5]:

$$\neg \boldsymbol{Pl} \to \Diamond_{[1,d]} \boldsymbol{Pl}. \tag{1}$$

Our goal is to determine two probabilistic properties of the above monitor at design time: the *false positive rate* (FPR: "detected but not present") and the *false negative rate* (FNR: "not detected but present"). High FPR is undesirable because the it takes the effort of an entire vessel to evacuate the UUV. High FNR is a safety issue because if the UUV is lost for longer than

[2]A monitor raises an alarm when the formula it monitors evaluates to F.

d seconds, it may find itself in an area where surfacing may be unsafe due to potential collisions (e.g., near a busy port). The next section describes our logical model for monitors.

3 Three-Value Temporal Logic for Detectors

Our model of monitors has two aspects: logical and stochastic. The logical aspect (this section) specifies monitors as logical formulas over detector outputs. The stochastic aspect (next section) represents these monitors as probabilistic compositions of detectors. Both aspects focus on the outputs of detectors, abstracting away heterogeneous details of the exact functionality of the detectors.

A detector $\boldsymbol{D}$ is — abstractly for now — a random process that at each moment tries to determine the presence or absence of some hypothesis H_1. We model the output of $\boldsymbol{D}$ at a certain moment as a pair of random variables (DO, GT):

- *Detection outcome* $DO \in \{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ is the 3-value output of the detector: T (confident in H_1), F (confident in the absence of H_1, i.e., the presence of null hypothesis H_0), and U (not confident in either H_1 or H_0).
- *Ground truth* $GT \in \{\mathsf{T}, \mathsf{F}\}$ is the actual presence of H_1, which can either be true (T) or false (F).

Sampled at consecutive times, a detector produces a pair of finite equal-length sequences (s_{do}, s_{gt}), where s_{do} is a sequence (or, trace) of DO values and s_{gt} is a sequence of GT values. For example, $\boldsymbol{Pl}$ is a detector for H_1: "pipeline present". For it, $s_{gt}(t)$ is whether the pipeline is indeed present at time t, and $s_{do}(t)$ is the 3-value detection output.

Monitors are specified in LTL$_{3d}$, a $\underline{3}$-value linear temporal logic for $\underline{\text{d}}$etectors with bounded temporal modalities. This logic is our modification of the 3-value LTL with unbounded modalities [19], which in turn builds on Kleene's strong logic of indeterminacy K$_3$ [17]. LTL$_{3d}$ adds bounds on modalities and two extra negations to handle uncertain values. LTL$_{3d}$ considers fully known traces: the value U is orthogonal to the inconclusiveness from not knowing a complete trace [6].

Formulas in LTL$_{3d}$ are constructed from atomic detectors $\boldsymbol{D}^1 \ldots \boldsymbol{D}^k$, the truth constant T, three operators of negation ($\neg_s$, $\neg_w$, $\neg_{se}$), conjunction $\boldsymbol{\wedge}$, and bounded until $\boldsymbol{U}_{[m,n]}$ (where $m, n \in \mathbb{N}_0$ and $m < n$ for the rest of this section).

$$\varphi ::= \varphi\, \boldsymbol{U}_{[m,n]}\, \varphi \mid \varphi \wedge \varphi \mid \neg_s \varphi \mid \neg_w \varphi \mid \neg_{se} \varphi \mid \boldsymbol{D} \mid \mathsf{T}$$

where $\boldsymbol{D} \in \{\boldsymbol{D}^1 \dots \boldsymbol{D}^k\}$. These operators conventionally define other syntax: disjunction $\vee$, three implications $\rightarrow_s$, $\rightarrow_w$, $\rightarrow_{se}$, and modalities eventually $\Diamond$ and always $\Box$; their definitions are available in the supplement [21].

Any LTL$_{3d}$ formula φ can be evaluated in two ways. The first evaluation is over the values of DO, calculating the 3-value detection results. The second evaluation is over the values of GT, characterizing the binary ground-truth "oracle" (and coinciding with the standard LTL semantics). Both evaluations happen on a finite state trace $s = s_1 s_2 \dots s_L$, where $L \in \mathbb{N}$ is the trace length. We assume we always have a trace of sufficient length to evaluate a given formula: $L \geq n$. A tail of trace s by starting from position $t < L$ is denoted as $s(t..) = s_t s_{t+1} s_{t+2} \dots s_L$.

We formulate the 3-value evaluation first, denoted $[\![\varphi]\!]_s$. Here the state trace s is a collection of the DO traces $s^1_{do} \dots s^k_{do}$ of the respective detectors.

$$[\![\mathsf{T}]\!]_s = \mathsf{T}$$

$$[\![\boldsymbol{D}^j]\!]_s = s^j_{do}(1), \text{ where } j \in [1,k]$$

$$[\![\neg_s \varphi]\!]_s = \begin{cases} \mathsf{T} & \text{if } [\![\varphi]\!]_s = \mathsf{F} \\ \mathsf{F} & \text{if } [\![\varphi]\!]_s = \mathsf{T} \\ \mathsf{U} & \text{if } [\![\varphi]\!]_s = \mathsf{U} \end{cases}$$

$$[\![\neg_w \varphi]\!]_s = \begin{cases} \mathsf{T} & \text{if } [\![\varphi]\!]_s = \mathsf{F} \text{ or } [\![\varphi]\!]_s = \mathsf{U} \\ \mathsf{F} & \text{if } [\![\varphi]\!]_s = \mathsf{T} \\ \mathsf{U} & \text{never} \end{cases}$$

$$[\![\neg_{se} \varphi]\!]_s = \begin{cases} \mathsf{T} & \text{if } [\![\varphi]\!]_s = \mathsf{F} \\ \mathsf{F} & \text{if } [\![\varphi]\!]_s = \mathsf{T} \text{ or } [\![\varphi]\!]_s = \mathsf{U} \\ \mathsf{U} & \text{never} \end{cases}$$

$$[\![\varphi_1 \wedge \varphi_2]\!]_s = \begin{cases} \mathsf{T} & \text{if } [\![\varphi_1]\!]_s = \mathsf{T} \text{ and } [\![\varphi_2]\!]_s = \mathsf{T} \\ \mathsf{F} & \text{if } [\![\varphi_1]\!]_s = \mathsf{F} \text{ or } [\![\varphi_2]\!]_s = \mathsf{F} \\ \mathsf{U} & \text{otherwise} \end{cases}$$

$$[\![\boldsymbol{\varphi}_1\ \boldsymbol{U}_{[m,n]}\ \boldsymbol{\varphi}_2]\!]_s = \begin{cases} \mathsf{T} & \text{if } \exists i \in [m,n] \cdot [\![\boldsymbol{\varphi}_2]\!]_{s(i..)} = \mathsf{T} \\ & \text{and } \forall j \in [m,i) \cdot [\![\boldsymbol{\varphi}_1]\!]_{s(j..)} = \mathsf{T} \\ \mathsf{F} & \text{if either } \forall i \in [m,n] \cdot [\![\boldsymbol{\varphi}_2]\!]_{s(i..)} = \mathsf{F} \\ & \text{or } \exists i \in [m,n] \cdot [\![\boldsymbol{\varphi}_1]\!]_{s(i..)} = \mathsf{F} \text{ and} \\ & \forall j \in [m,i) \cdot [\![\boldsymbol{\varphi}_2]\!]_{s(j..)} = \mathsf{F} \\ \mathsf{U} & \text{otherwise} \end{cases}$$

Definition 3.1 (3-value Satisfaction)**.** Formula φ is *3-value satisfied* on state trace s, denoted $s \models_3 \boldsymbol{\varphi}$, if $[\![\boldsymbol{\varphi}]\!]_s = \mathsf{T}$.

A few remarks are in order. Similarly to K$_3$, we interpret U as a value of insufficient information, leading to the semantics that take advantage of the known information, e.g., $\mathsf{U} \boldsymbol{\wedge} \mathsf{F} \equiv \mathsf{F}$. We adopt this principle because CPS often need to make decisions with limited information rather than waiting for the complete picture to be known. Similarly to the 3-value LTL [19], our Until operator handles uncertainty in the values of its operands — not in whether the available trace values are sufficient to evaluate the Until.

Unlike the existing 3-value logics, our LTL$_{3d}$ has several negations. Operator $\boldsymbol{\neg}_s$ is equivalent to the negation in K$_3$: it flips the certain outcomes (T, F) while preserving the uncertain value. The two other negations, *weak* $\boldsymbol{\neg}_w$ and *strong exclusive* $\boldsymbol{\neg}_{se}$, are never uncertain, with $\boldsymbol{\neg}_w$ erring on the side of H_1 and $\boldsymbol{\neg}_{se}$ erring on the side of H_0.

The 2-value evaluation ($[\![\boldsymbol{\varphi}]\!]_s$) is based on the s_{gt} traces $s_{gt}^1 \ldots s_{gt}^k$ of the respective detectors. The semantics are the same as for $[\![\boldsymbol{\varphi}]\!]_s$ except not using the U values: atomic detectors can only evaluate to T or F because here the state trace s is based on the ground-truth traces:

$$[\![\boldsymbol{D}^j]\!]_s = s_{gt}^j(1), \text{ where } j \in [1,k]$$

Definition 3.2 (2-value Satisfaction)**.** Formula $\boldsymbol{\varphi}$ is *2-value satisfied* on state trace s, denoted $s \models_2 \varphi$, if $[\![\boldsymbol{\varphi}]\!]_s = \mathsf{T}$.

Remark 1. We use bold operators (e.g., $\boldsymbol{\wedge}$, $\boldsymbol{\neg}_s$, $\boldsymbol{U}$) where *both* 3-value and 2-value interpretations are possible; non-bold operators (e.g., $\wedge$, $\neg$) have the usual 2-value interpretation.

Relations $\models_3$ and $\models_2$ are used to define a monitor's output and ground truth, respectively.

Definition 3.3 (Monitor)**.** A *monitor* M, defined by formula $\boldsymbol{\varphi}$ in LTL$_{3d}$ over detectors $\boldsymbol{D}^1 \dots \boldsymbol{D}^k$, is a function that maps the detector traces $(s^1_{do}, s^1_{gt}) \dots (s^k_{do}, s^k_{gt})$ to a *pair of monitor traces* (m_{do}, m_{gt}):

$$\begin{aligned} m_{do}(t) = \; & s^1_{gt}(t..) \dots s^k_{gt}(t..) \models_3 \boldsymbol{\varphi} \\ m_{gt}(t) = \; & s^1_{gt}(t..) \dots s^k_{gt}(t..) \models_2 \boldsymbol{\varphi} \end{aligned}$$

Notice that detectors produce a 3-value s_{do} and a 2-value s_{gt}, whereas monitors produce two 2-value traces: m_{do} and m_{gt}. In all cases, we interpret the value T an "alarm", and F and U as "no alarm".

To sum up, the ground truth values are substituted into logical formulas in the 2-value case, and the detector output values are substituted in the 3-value case.

With the logical aspect defined, we rewrite the motivating property (Equation (1)) in LTL$_{3d}$. We use $\neg_w$ to be conservative: the unknown presence of pipeline (U) is interpreted as its loss:

$$\boldsymbol{\varphi}_{pr} := \neg_w \boldsymbol{Pl} \rightarrow_w \Diamond_{[1,d]} \boldsymbol{Pl} \tag{2}$$

We define a pipe recovery monitor M_{pr} to alarm only when *certain* about *violations* of $\boldsymbol{\varphi}_{pr}$. Thus, we use $\neg_s$ to indicate this requirement of certainty:

$$\boldsymbol{M}_{pr} = \neg_s \boldsymbol{\varphi}_{pr} = \neg_s(\neg_w \boldsymbol{Pl} \rightarrow_w \Diamond_{[1,d]} \boldsymbol{Pl}) \tag{3}$$

We implemented $\boldsymbol{M}_{pr}$ over the s_{do} trace of detector $\boldsymbol{Pl}$, and it raises an alarm when T occurs in its m_{do}. The GT trace m_{gt} is calculated from s_{gt} of $\boldsymbol{Pl}$. Monitor code generation relies on the standard approaches [5] and is out of this paper's scope.

A *monitoring error* occurs if, for some t, $m_{do}(t) \neq m_{gt}(t)$. We are interested in two types of errors: if $m_{do}(t) = \mathsf{T}$ and $m_{gt}(t) = \mathsf{F}$, it is a *false positive*; if $m_{do}(t) = \mathsf{F}$ and $m_{gt}(t) = \mathsf{T}$, it is a *false negative*. Since detectors and monitors are random variables, we talk about probabilities, or *rates*, of either type of errors.

Having monitors with high error rates can be a safety-critical concern, as illustrated on $\boldsymbol{M}_{pr}$ in Section 2. Typically, these rates are evaluated by collecting, labeling, and analyzing their output data. The data is collected over multiple executions, each providing one m_{do}, labeling which yields one m_{gt}. Then the error rate estimate is the count of errors (e.g., false positives) divided by the count of cases where this type of error (e.g., a false positive) was possible (e.g., all true negatives).

We believe that this estimation can be done in *early design stages* and using *only* the data from the atomic detectors (without executing the monitors). The central problem of this paper is, hence, to estimate the error rates of a monitor of a LTL$_{3d}$ formula. This estimation relies on labeled traces of atomic detectors (paired s_{do} and s_{gt}).

4 Probabilistic Composition of Detectors

This section models detectors as probability spaces behind the random variables (DO, GT), described in the previous section. Then, we show how compositions of these spaces correspond to formulas in LTL$_{3d}$, in order to analyze these formulas probabilistically in Section 5.

4.1 Stochastic Detectors

We start by defining a probability space of one detector.

Definition 4.1 (Atomic stochastic detector)**.** An *atomic stochastic detector* ($\boldsymbol{D}$) is a pair of random variables $(DO,\ GT)$ over a probability space $(\Omega, \mathcal{F}, \Pr)$:

- Ω is an elementary outcome space, defined as the set of all six possible pairs of values for (DO, GT): $(\mathsf{T},\mathsf{T}), (\mathsf{T},\mathsf{F}), (\mathsf{F},\mathsf{T}), (\mathsf{F},\mathsf{F}), (\mathsf{U},\mathsf{T}), (\mathsf{U},\mathsf{F})$.
- $\mathcal{F} = 2^{\Omega}$ is a sigma-algebra of events over signature Ω, defined as the powerset of Ω (including the empty set). We use the following *marginal events* as shortcuts for the individual values of either random variable: [3]

$$\begin{aligned}
\mathrm{gtt}(\boldsymbol{D}) &= (*,\mathsf{T}) = \{(\mathsf{T},\mathsf{T}), (\mathsf{F},\mathsf{T}), (\mathsf{U},\mathsf{T})\},\\
\mathrm{dot}(\boldsymbol{D}) &= (\mathsf{T},*) = \{(\mathsf{T},\mathsf{T}), (\mathsf{T},\mathsf{F})\},\\
\mathrm{gtf}(\boldsymbol{D}) &= (*,\mathsf{F}) = \{(\mathsf{T},\mathsf{F}), (\mathsf{F},\mathsf{F}), (\mathsf{U},\mathsf{F})\},\\
\mathrm{dof}(\boldsymbol{D}) &= (\mathsf{F},*) = \{(\mathsf{F},\mathsf{T}), (\mathsf{F},\mathsf{F})\},\\
\mathrm{dou}(\boldsymbol{D}) &= (\mathsf{U},*) = \{(\mathsf{U},\mathsf{T}), (\mathsf{U},\mathsf{F})\}.
\end{aligned}$$

[3]By definition, events dof, dot, and dou partition Ω. The same applies to gtt and gtf. Any such partitioning events constitute a well-formed detector.

- Pr is a single-detector discrete probability measure over $\mathcal{F}$, subject to the usual Kolmogorov axioms.

Our concept of a stochastic detector encompasses a wide range of algorithms with ternary outcomes over time series, including anomaly detectors, machine learning classifiers, and sensor fusion algorithms. We have conveniently abstracted away the inputs of these algorithms, making this model broadly applicable. As mentioned in Section 3, our convention is that a detector raises an alarm iff event dot occurs, thus introducing the asymmetry inherent in statistical tests (which reject H_0 only with sufficient evidence).

Now we handle time and multiple detectors. When detector $\boldsymbol{D}$ produces readings over time points $t_1 \dots t_k$ (e.g., a UUV gets consecutive readings of the pipeline), we model it with a sample of k "instances" of the original detector: $\boldsymbol{D}^1 \dots \boldsymbol{D}^k$, i.e., a stochastic process. These copies have their own respective variables $DO^1 \dots DO^k$ and $GT^1 \dots GT^k$. These instances are not necessarily i.i.d.; in general, their individual probability spaces are coupled to form a larger, *multi-detector probability space*, which we use to describe compositions.

Definition 4.2 (Multi-detector probability space)**.** For detectors $\boldsymbol{D}^1 \dots \boldsymbol{D}^k$, a *multi-detector probability space* $(\boldsymbol{\Omega}, \boldsymbol{\mathcal{F}}, \mathsf{Pr})$ is defined as follows:

- $\boldsymbol{\Omega} = \Omega^1 \times \cdots \times \Omega^k$ is a k-dimensional space of elementary outcomes; each dimension produces an elementary outcome pair for the respective detector.

- $\boldsymbol{\mathcal{F}} = 2^{\boldsymbol{\Omega}}$ is an event sigma-algebra, which is a powerset of the outcome space $\boldsymbol{\Omega}$ (including the empty set). Thus, an event is a set of k-sized vectors.

- Pr is a global probability measure over $\boldsymbol{\mathcal{F}}$, extended from $\Pr^1 \dots \Pr^k$ in a manner consistent with them (this requirement is formalized in the supplement [21]).

For a given set of atomic detectors, there are infinite multi-detector spaces because individual probabilities $\Pr^1 \dots \Pr^k$ can be coupled in many ways to form the joint Pr. Analysis of possible couplings is one of the tasks we handle in Section 5.

Now consider k different[4] detectors for n time points, leading to a $k \times n$ multi-detector space. Although we can handle the general joint distribution from Definition 4.2, for tractability it is common to impose domain-specific constraints on Pr. A common type of them is independence constraints, denoted $A \perp\!\!\!\perp B$ for random variables A and B; a set of independence constraints is denoted $\mathcal{I}$. Detectors of a random walk would have independent sequential actions: $A^i \perp\!\!\!\perp A^{i+1}$. This means that *any combination* of marginal events produced by these variables is independent (as expanded in the supplement [21]).

One of the common measurement models is that the error is *stateless* between measurements, meaning that previous measurements do not provide any extra information[5] given the current ground truths. In our terms, this implies the independence of any sequence of outcomes given their respective ground truths — an assumption we will use in Section 5:

$$\begin{aligned} &\forall i, j : \mathbb{N}, 1 \leq i < j \leq k \cdot \\ &DO^i \perp\!\!\!\perp DO^{i+1} \perp\!\!\!\perp \ldots \perp\!\!\!\perp DO^j \mid GT^i \ldots GT^j \end{aligned} \tag{4}$$

Remark 2. For convenience, we will perform symbolic calculations over marginal events introduced in Definition 4.1. These events can be connected with operators from classic binary logic: $\neg$, $\wedge$, and $\vee$. We use non-bold fonts for them interpret as usual (e.g., $\wedge$ means that both events occurred). These operators correspond to set operations over events in $\boldsymbol{\mathcal{F}}$: complement, intersection, and union respectively.

The above probabilistic model expresses any error rate, denoted er, straightforwardly as a conditional probability over marginal events. This paper focuses on two specific error rates:

- *False positive rate*: $\mathrm{fpr}(\boldsymbol{D}) = \Pr(\mathrm{dot}(\boldsymbol{D}) \mid \mathrm{gtf}(\boldsymbol{D}))$.
- *False negative rate*: $\mathrm{fnr}(\boldsymbol{D}) = \Pr(\neg\, \mathrm{dot}(\boldsymbol{D}) \mid \mathrm{gtt}(\boldsymbol{D}))$.

Our model allows error rates to change between detectors $\boldsymbol{D}^1 \ldots \boldsymbol{D}^n$. However, in this paper we assume the error rates to be constant across time

[4] The detectors may be of the same phenomena or different ones. We address the general case where all ground truths are different.

[5] A slightly stronger assumption, common in time series analysis [7], is written $DO^t = GT^t + \epsilon_t$, where ϵ_t is a white noise series.

moments because here they represent inherent qualities of detectors. Mathematically, this means that our model averages out temporal variation in an error rate.

4.2 Composition of Stochastic Detectors

Now we will show how to interpret formulas in LTL$_{3d}$ as compositions of detector spaces. Intuitively, a composite detector defined by a formula is a result of applying this formula's evaluation to the atomic detectors. The composite outputs are distributed over a multi-detector probability space.

Definition 4.3 (Composite detector)**.** A *composite detector* $\boldsymbol{D}'$ for formula $\boldsymbol{\varphi}$ over detectors $\boldsymbol{D}^1 \ldots \boldsymbol{D}^k$ with state trace s is a pair of random variables (DO', GT') over a multi-detector probability space for $\boldsymbol{D}^1 \ldots \boldsymbol{D}^k$, with these values at time t:

$$DO'(t) = [\![\boldsymbol{\varphi}(\boldsymbol{D}^1 \ldots \boldsymbol{D}^k)]\!]_{s(t..)}$$
$$GT'(t) = [\![\boldsymbol{\varphi}(\boldsymbol{D}^1 \ldots \boldsymbol{D}^k)]\!]_{s(t..)}$$

Our insight here is that logical formulas correspond to composite detectors in multi-detector spaces. For example, $\boldsymbol{\varphi}_{pr}$ and $\boldsymbol{M}_{pr}$ are composite detectors over $\boldsymbol{Pl}, \boldsymbol{Pl}^1 \ldots \boldsymbol{Pl}^d$ defined in a $(d+1)$-dimensional event space, according to Equations (2) and (3). These spaces can be analyzed probabilistically, yielding estimates of error rates.

Given a fully defined Pr in some multi-detector space, a logical formula determines the probabilities of events of the corresponding composite detector. In our experience, however, a fully defined probability measure on a multi-detector space is rarely available or needed: it would require a lot of data and/or prior knowledge to determine. Instead, this space can be manipulated symbolically (i.e., without assigning numeric values) to express the desired probabilities through the known or easy-to-measure ones (e.g., probabilities of the atomic events), without committing to a fully defined Pr. These symbolic manipulations are convenient to carry out in the classic binary propositional logic, so we calculate using probability formulas over event predicates.

Let us provide some examples to illustrate event predicates in compositions, say conjunction. Its marginal events are derived from the logical semantics to be as follows: [6]

$$\begin{aligned}
\mathrm{gtt}(\boldsymbol{D}^a \boldsymbol{\wedge} \boldsymbol{D}^b) &= \mathrm{gtt}(\boldsymbol{D}^a) \wedge \mathrm{gtt}(\boldsymbol{D}^b) \\
\mathrm{gtf}(\boldsymbol{D}^a \boldsymbol{\wedge} \boldsymbol{D}^b) &= \mathrm{gtf}(\boldsymbol{D}^a) \vee \mathrm{gtf}(\boldsymbol{D}^b) \\
\mathrm{dot}(\boldsymbol{D}^a \boldsymbol{\wedge} \boldsymbol{D}^b) &= \mathrm{dot}(\boldsymbol{D}^a) \wedge \mathrm{dot}(\boldsymbol{D}^b) \\
\mathrm{dof}(\boldsymbol{D}^a \boldsymbol{\wedge} \boldsymbol{D}^b) &= \mathrm{dof}(\boldsymbol{D}^a) \vee \mathrm{dof}(\boldsymbol{D}^b) \\
\mathrm{dou}(\boldsymbol{D}^a \boldsymbol{\wedge} \boldsymbol{D}^b) &= \mathrm{dou}(\boldsymbol{D}^a) \wedge \mathrm{dou}(\boldsymbol{D}^b) \vee \\
&\quad \mathrm{dou}(\boldsymbol{D}^a) \wedge \mathrm{dot}(\boldsymbol{D}^b) \vee \\
&\quad \mathrm{dou}(\boldsymbol{D}^b) \wedge \mathrm{dot}(\boldsymbol{D}^a)
\end{aligned}$$

Such rewritings for other compositions can be found in the supplement [21].

Another rewriting to note: in our finite-trace semantics, the temporal modalities "always" and "eventually" for integers m and $n > m$ can be expressed with $\boldsymbol{\wedge}$ and $\boldsymbol{\vee}$ (which in turn can be analyzed based on their marginal events):

$$\begin{aligned}
\Box_{[m,n]}\boldsymbol{D} &::= \boldsymbol{D}^m \boldsymbol{\wedge} \ldots \boldsymbol{\wedge} \boldsymbol{D}^n \\
\Diamond_{[m,n]}\boldsymbol{D} &::= \boldsymbol{D}^m \boldsymbol{\vee} \ldots \boldsymbol{\vee} \boldsymbol{D}^n
\end{aligned}$$

In summary, this section showed that an error rate of a LTL$_{3d}$ formula can be represented as a probability of events in a multi-detector space. We take advantage of this fact to calculate estimates of error rates in the next section.

5 Analysis of Error Rates

The goal of our analysis is to compute a numeric estimate of the false positive or false negative rate of a monitor, such as $\mathrm{fnr}(\boldsymbol{M}_{pr})$, which is treated as a probability in an event space $\boldsymbol{\mathcal{F}}$ over some detectors $\boldsymbol{D}^1 \ldots \boldsymbol{D}^k$. This is challenging because, even though it is precisely defined, a computable formula for that probability is *not known a priori* — and there is generally an infinite number of them. Finding a computable formula is complicated by knowing

[6] The form of $\mathrm{dou}(\boldsymbol{D}^a \boldsymbol{\wedge} \boldsymbol{D}^b)$ follows from events $\mathrm{dot}(\boldsymbol{D}_a \boldsymbol{\wedge} \boldsymbol{D}_b)$ and $\mathrm{dof}(\boldsymbol{D}^a \boldsymbol{\wedge} \boldsymbol{D}^b)$, being a complement of their union. Specifying the marginal events fully defines a detector, as shown in the supplement [21].

probability values for some set of events and having traces for another set of events.

We factor this problem into three analyses; each takes different inputs and produces a numeric estimate $\hat{\text{er}}$ of a given error rate er:

- *Exact Compositional Calculation (ECC):* to compute $\hat{\text{er}}$, this analysis requires a monitor formula $\boldsymbol{M}$, user-specified probability values Pr of selected events in $\boldsymbol{\mathcal{F}}$, independence assumptions $\mathcal{I}$, and transformation rules $\mathcal{R}$:

$$\hat{\text{er}}(\boldsymbol{M}) = ECC_{\text{er}}(\boldsymbol{M}, \mathsf{Pr}, \mathcal{I}, \mathcal{R})$$

- *Noisy Compositional Calculation (NCC):* this analysis requires a monitor formula $\boldsymbol{M}$, labeled traces for atomic detectors (for simplicity represented by one trace pair: s_{gt} and s_{do}), independence assumptions $\mathcal{I}$, a set of symbolic "preferred" probabilities $\hat{\mathbb{P}}$ of user-specified events in $\boldsymbol{\mathcal{F}}$, and transformation rules $\mathcal{R}$ (which we elaborate in the next subsection). The preferred probabilities is a set of probability symbols (Pr) over the events of $\boldsymbol{\mathcal{F}}$. The user chooses $\hat{\mathbb{P}}$ as a heuristic[7] for probabilities to express the error rate through. For example, we use $\hat{\mathbb{P}} = \{\text{fpr}(\boldsymbol{Pl}), \text{fnr}(\boldsymbol{Pl})\}$ because the composite error rates can often be compactly expressed using the atomic error rates. The goal is to find a rate formula with a minimal number of non-preferred probabilities and estimate these probabilities from the labeled detector traces.

$$\hat{\text{er}}(\boldsymbol{M}) = NCC_{\text{er}}(\boldsymbol{M}, s_{gt}, s_{do}, \mathcal{I}, \hat{\mathbb{P}}, \mathcal{R})$$

- *Black-Box Calculation (BBC):* this analysis uses only labeled monitor traces, represented with a pair (m_{gt}, m_{do}), produced by monitor $\boldsymbol{M}$ whose rate is being estimated. This analysis does not need the $\boldsymbol{M}$ formula: it estimates the error rate directly from the traces.

$$\hat{\text{er}}(\boldsymbol{M}) = BBC_{\text{er}}(m_{gt}, m_{do})$$

To perform BBC, we use the standard approach using count-based frequency estimation, exemplified in the second last paragraph of Section 3,

[7]The reasons for choosing a specific set $\hat{\mathbb{P}}$ may vary. Some probabilities have strong priors or more data for accurate estimation; others can limit the formula search or help find a small formula.

which is is an optimal estimator for event probabilities [8]. Generally, to estimate an error rate $\rho = \Pr(A \mid B)$, BBC is computed as follows:

$$BBC_{\Pr(A|B)}(m_{gt}, m_{do}) = \frac{|\{(m_{gt}(t), m_{do}(t)) \text{ s.t. } A(t) \land B(t)\}|}{|\{(m_{gt}(t), m_{do}(t)) \text{ s.t. } B(t)\}|} \tag{5}$$

This paper proposes a novel semi-automated heuristic analysis to compute ECC and NCC. To support the analysis, we implemented a semi-automated computational assistant based on the Mathematica's symbol manipulation (see its source code at `https://github.com/bisc/prob-comp-asst`)

5.1 Transformation Rules

Our computational assistant symbolically transforms formulas and computes numeric estimates. The assistant starts with the formula of the desired error rate (er) of a monitor and transforms it towards a computable formula. Here, "computable" means that each probability in this formula is either known or can be estimated from (s_{gt}, s_{do}).

This assistant manipulates formulas using transformation rules. Each transformation rule has this form:

$$[G]A \to C, \text{ where}$$

- A is the antecedent to replace. E.g., it can be $\mathsf{Pr}(X, Y)$ where X and Y are placeholders for any events.
- C is the consequent to replace the antecedent. E.g., it can be $\mathsf{Pr}(X)\mathsf{Pr}(Y)$, where X and Y are from$A$.
- G is the guard condition, such as $X \perp\!\!\!\perp Y$.

All transformation rules can be partitioned into four lists, the combination of which is denoted $\mathcal{R}$. The four lists are below (their rules are fully specified in the supplement [21]):

- $\mathcal{R}_{\text{log}}$ are logical tautologies of LTL$_{3d}$, e.g., $\boldsymbol{\neg}_s\boldsymbol{\neg}_s\boldsymbol{D} = \boldsymbol{D}$.
- $\mathcal{R}_{\text{ev}}$ are rules substituting the logical operators of LTL$_{3d}$ for their definitions in terms of detector events.

- $\mathcal{R}_{\text{prob}}$ are rules to algebraically manipulate probability expressions. This includes the rules to turn probabilities of event conjunctions into products of event probabilities, if these events are independent.

- $\mathcal{R}_{\text{indep}}$ are rules to infer conditional independence. We use the standard semi-graphoid axioms: contraction, decomposition, weak union, and intersection [18].

Each of the four lists can be *applied* to a formula. The application of each list proceeds one rule at a time in the order of the list. A rule is applied to every matching subformula and until no subformulas match the antecedent and satisfy the guard. If/once a rule can no longer be applied, the next rule is read. If no rule is applicable, the list application terminates.

All rule lists use only sound transformations: assuming G holds, we know that $A = C$. We ensure that by using only the standard rules of probability, Boolean algebra,conditional independence, as well as the easy-to-show tautologies of LTL_{3d}. Therefore, applying $\mathcal{R}$ never produces a formula not equivalent to the original one, $\text{er}(\boldsymbol{M})$, under the assumptions $\mathcal{I}$.

The outcome of rule application significantly depends on the rule lists chosen by the user. While we provide large sets of rules, it is up to the user to pick a subset and order that would be effective for a given scenario. To do so, the user needs to understand the rules and their effects on probability expressions.

The presence and order of rules in $\mathcal{R}$, and particularly $\mathcal{R}_{\text{prob}}$, determine the completeness, termination, and algorithmic efficiency of ECC and NCC. Some rule lists may lack the necessary transformations (e.g., adding and removing negations to events) to produce a computable formula. Application of other rule lists may not terminate; e.g., replacing $\Pr(X)$ with $\Pr(X \wedge Y)$ requires a rule that introduces a variable Y, which could be applied indefinitely. Such rules can lead to a set of formulas that grows exponentially (to consider all combinations of events) and is possibly unbounded (if normal forms for event expressions are not enforced).

We expect a large number of formula-finding problems to be solvable with rule lists. Unfortunately, we do not have a precise characterization of this class, leaving it for future work. For the monitors and error rates in this paper, we found sufficient and efficient rule lists, given in full detail in the supplement [21].

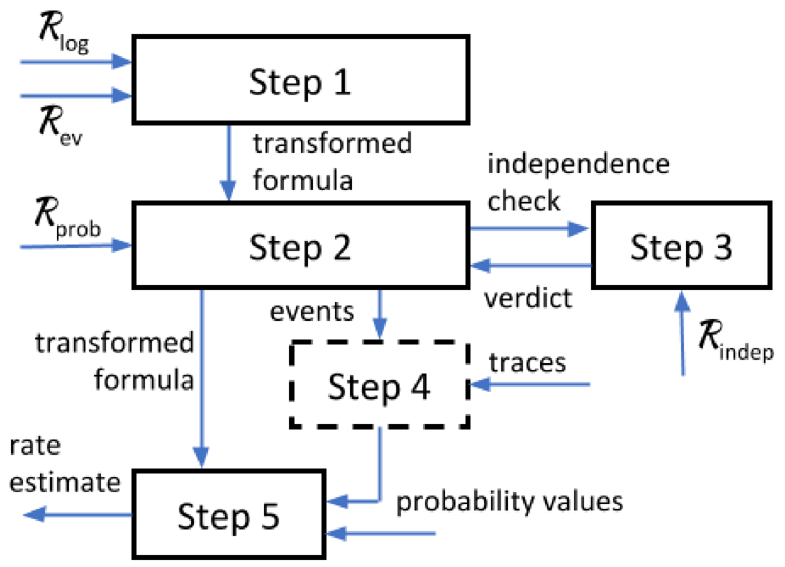

Figure 2: The ECC/NCC analysis steps.

5.2 Steps of ECC and NCC

Tne ECC and NCC analyses proceed in five steps, illustrated in Figure 2. Their abstract description below is exemplified in Subsection 5.3.

Step 1: rewrite the formula in terms of marginal events dot, dof, dou, gtt, gtf—without any operators of LTL_{3d}.

Inputs:

- $\text{er}(\boldsymbol{M})$ — the goal error rate er of a monitor $\boldsymbol{M}$.
- $\mathcal{R}_{\text{log}}$ — transformation rules for the logic operators.
- $\mathcal{R}_{\text{ev}}$ — transformation rules for events of detectors.

Output: $f(\mathsf{Pr}(\boldsymbol{\mathcal{F}}))$ — a formula f over probabilities of Boolean combinations of marginal events of atomic detectors.

Sub-steps:

1.1 Apply rules $\mathcal{R}_{\text{log}}$ to $\text{er}(\boldsymbol{M})$ to simplify the composite detector formula. While not required, this sub-step can shorten the subsequent steps, e.g., by removing negation operators when valid (examples in the supplement [21]).

1.2 Apply rules $\mathcal{R}_{\text{ev}}$ to the result of the previous step. This sub-step replaces the 3-value logical constructs with an equivalent encoding using replacing them with binary events dot, dof, dou, gtt, and gtf. The expression becomes more verbose but also manipulable with the classical Boolean logic.

Step 2: algebraically reduce the formula to the preferred or known probabilities.

Inputs:

- $f(\mathsf{Pr}(\mathcal{F}))$ — a formula f over probabilities of Boolean combinations of marginal events of atomic detectors.
- $\mathcal{R}_{\text{prob}}$ — transformation rules for probabilities.
- (ECC only) $\mathsf{Pr}(\boldsymbol{\mathcal{F}})$ — known probabilities of some given events in $\boldsymbol{\mathcal{F}}$. The rest are unknown.
- (NCC only): $\hat{\mathbb{P}}(\boldsymbol{\mathcal{F}})$ — a set of preferred probabilities.

Output:$f(\mathsf{Pr}(\boldsymbol{\mathcal{F}}))$ — a formula f over Pr of events in $\boldsymbol{\mathcal{F}}$.

Sub-steps:

2.1 Apply rules from $\mathcal{R}_{\text{prob}}$, with two caveats:

- Do not apply rules to known/preferred probabilities.
- If a rule has an independence guard, send this independence statement to Step 3. If Step 3 returns T, apply the rule; otherwise, skip to the next rule.

2.2 After all $\mathcal{R}_{\text{prob}}$ have been applied, proceed to this Step:

- For ECC, go to Step 5.
- For NCC, send the events under probabilities in the current formula f to Step 4 (then go to Step 5).

Step 3: check of whether an independence statement hold under the independence assumptions.

Inputs:

- An independence statement of form $X \perp\!\!\!\perp Y$ or $X \perp\!\!\!\perp Y \mid Z$, where X, Y,and Z are event expressions.
- $\mathcal{I}$ — a set of independence assumptions.
- $\mathcal{R}_{\text{indep}}$ — a list of transformation rules for independence.

Output: T (if the statement can be assumed to be true) or F (if not enough information to conclude independence).
Sub-steps:

3.1 Apply rules $\mathcal{R}_{\text{indep}}$ to $\mathcal{I}$. This produces a larger set of independence statements assumed to be true.

3.2 Return T if the given statement is in the produced set. Otherwise, return F.

Step 4 (NCC only): produce numeric probability estimates for the set of events requested in Substep 2.2..
Inputs:

- Events $\mathbb{E}$ in $\boldsymbol{\mathcal{F}}$. requested in Substep 2.2
- A labeled detector trace (s_{gt}, s_{do}).

Output: probability estimates $\hat{\mathsf{Pr}}$ for the requested events.
Sub-steps:

4.1 For each event in $\mathbb{E}$, calculate a count-based estimate analogously to Equation (5) but over trace (s_{gt}, s_{do}).

4.2 Send all probability estimates $\hat{\mathsf{Pr}}(\mathbb{E})$ to Step 5.

Step 5: calculate a numeric estimate $\hat{\text{er}}$ of error rate er.
Inputs:

- Formula $f(\mathsf{Pr}(\boldsymbol{\mathcal{F}}))$ from Step 2.2.
- (ECC only) Known probabilities $\mathsf{Pr}(\boldsymbol{\mathcal{F}})$.
- (NCC only) Estimated probabilities $\hat{\mathsf{Pr}}(\mathcal{F})$.

Output: a numeric estimate of error rate $\hat{\text{er}}(\boldsymbol{M})$.
Sub-steps:

5.1 Replace each probability in $f(\mathsf{Pr}(\boldsymbol{\mathcal{F}}))$ with its value:

- ECC uses known probability values $\mathsf{Pr}(\boldsymbol{\mathcal{F}})$.
- NCC uses the estimates $\hat{\mathsf{Pr}}(\boldsymbol{\mathcal{F}})$ from Step 4.

5.2 Calculate the numeric expression and return the result.

5.3 Example of ECC/NCC Application

As an illustration, we walk through[8] the NCC calculation of $\text{fnr}(\boldsymbol{M}_{pr})$. As inputs we have the definition of $\boldsymbol{M}_{pr}$ in Equation (2) with some known value of d, a labeled trace (s_{gt}, s_{do}) for $\boldsymbol{Pl}$, and $\hat{\mathbb{P}} = \{\text{fpr}(\boldsymbol{Pl}), \text{fnr}(\boldsymbol{Pl})\}$, rules $\mathcal{R}$, and two independence assumptions: Equation (4) and that only the current ground truth affects the detector output:

$$\begin{aligned} &\forall i : \mathbb{N}, 1 \leq i \leq k \cdot \\ &DO^i \perp\!\!\!\perp GT^1 \dots GT^{i-1}, GT^{i+1} \dots GT^k \mid GT^i \end{aligned} \tag{6}$$

The initial formula $\text{er}(\boldsymbol{M})$ is below:

$$\text{fnr}\left(\boldsymbol{\neg}_s(\boldsymbol{\neg}_w \boldsymbol{Pl} \boldsymbol{\rightarrow}_w \boldsymbol{\Diamond}_{[1,d]} \boldsymbol{Pl})\right) \tag{7}$$

Step 1.1 replaces $\boldsymbol{\rightarrow}_w$ with $\boldsymbol{\vee}$ and $\boldsymbol{\neg}_w$, rewrites $\boldsymbol{\Diamond}_{[1,d]}$ using $\boldsymbol{\vee}$, and distributes $\boldsymbol{\neg}_s$ over the $\boldsymbol{\vee}$ operators in the parentheses. The outcome is a different detector under fnr:

$$\text{fnr}\left(\boldsymbol{\neg}_w \boldsymbol{Pl} \boldsymbol{\wedge} \boldsymbol{\neg}_s \boldsymbol{Pl}^1 \boldsymbol{\wedge} \dots \boldsymbol{\wedge} \boldsymbol{\neg}_s \boldsymbol{Pl}^d\right) \tag{8}$$

Step 1.2 rewrites Equation (8) as a conditional probability over the marginal events of the detector under fnr. Specifically, the rewriting uses events dot and gtf according to the definitions of operators $\boldsymbol{\vee}, \boldsymbol{\neg}_w, \boldsymbol{\neg}_s$:

$$\begin{aligned} &\left(\text{dof}(\boldsymbol{Pl}^0) \vee \text{dou}(\boldsymbol{Pl}^0)\right) \wedge \text{dof}(\boldsymbol{Pl}^1) \wedge \dots \wedge \text{dof}(\boldsymbol{Pl}^d), \\ &\text{gtf}(\boldsymbol{Pl}^0) \wedge \text{gtf}(\boldsymbol{Pl}^1) \wedge \dots \wedge \text{gtf}(\boldsymbol{Pl}^d) \end{aligned}$$

[8]The full derivation for this error rate formula is in the supplement [21].

Hence, Step 1.2 produces this probability formula:

$$\mathsf{Pr}\Big(\neg\big((\mathrm{dof}(\boldsymbol{Pl}^0) \vee \mathrm{dou}(\boldsymbol{Pl}^0)) \wedge \mathrm{dof}(\boldsymbol{Pl}^1) \ \wedge \ldots \\ \wedge \mathrm{dof}(\boldsymbol{Pl}^d)\big) \ | \ \mathrm{gtf}(\boldsymbol{Pl}^0) \wedge \ldots \wedge \mathrm{gtf}(\boldsymbol{Pl}^d)\Big) \tag{9}$$

Then Step 2.1 switches the negation to $1 - \mathsf{Pr}$:

$$1 - \mathsf{Pr}\Big(\big(\mathrm{dof}(\boldsymbol{Pl}^0) \vee \mathrm{dou}(\boldsymbol{Pl}^0)\big) \wedge \mathrm{dof}(\boldsymbol{Pl}^1) \ \wedge \ldots \\ \wedge \mathrm{dof}(\boldsymbol{Pl}^d) \ | \ \mathrm{gtf}(\boldsymbol{Pl}^0) \wedge \ldots \wedge \mathrm{gtf}(\boldsymbol{Pl}^d)\Big) \tag{10}$$

Now Step 2.1 matches a rule with an independence guard and sends the events under Pr to Step 3. Step 3.2 finds that the assumedEquation (4) matches the provided events and returns T. Step 2.1 now replaces the above probability with a product of probabilities of each detector's events. Further, Step 2.1 reduces the conditioning of dof^i to gtf^i using a rule that checks Equation (6). A few algebraic transformations in Step 2.1 yield the *final formula* for $\mathrm{fnr}(\boldsymbol{M}_{pr})$:

$$1 - \big(1 - \mathrm{fpr}(\boldsymbol{Pl})\big)\Big(1 - \mathrm{fpr}(\boldsymbol{Pl}) - \mathsf{Pr}\big(\,\mathrm{dou}(\boldsymbol{Pl}) \ | \ \mathrm{gtf}(\boldsymbol{Pl})\big)\Big)^d \tag{11}$$

Step 2.2 leads to Step 4, which estimates the probabilities $\mathrm{fpr}(\boldsymbol{Pl})$ and $\mathsf{Pr}(\mathrm{dou}(\boldsymbol{Pl}) \ | \ \mathrm{gtf}(\boldsymbol{Pl}))$ from the trace (s_{gt}, s_{do}) of $\boldsymbol{Pl}$. Step 4.1 counts the events of interest (e.g., false positives for $\mathrm{fpr}(\boldsymbol{Pl})$) and divides it by the counts of possibilities for false positives (i.e., ground-truth negatives for $\mathrm{fpr}(\boldsymbol{Pl})$).

Finally, Step 5.1 substitutes the estimates of $\mathrm{fpr}(\boldsymbol{Pl})$ and $\mathsf{Pr}(\mathrm{dou}(\boldsymbol{Pl}) \mid \mathrm{gtf}(\boldsymbol{Pl}))$ into Equation (11). Step 5.2 yields a numeric estimate of $\mathrm{fnr}(\boldsymbol{M}_{pr})$. We investigate the accuracy of such estimates in the next section.

To summarize, this section introduced three analyses of error rates. ECC consists of two stages: symbolic manipulation (Steps 1–3) and numeric computation (Step 5). NCC consists of three stages: symbolic manipulation (Steps 1–3), estimation from data (Step 4), and numeric computation (Step 5). BBC consists only of one step — estimation from data.

ECC/NCC can take advantage of their modularity: (i) prior knowledge about detectors can be used for accurate of (ii) that the symbolic manipulation can be performed once per monitor and reused should the data or numerical inputs change, (iii) labeled monitor traces are not required.

6 Evaluation: UUV Case Study

Our validation goals are four-fold; we evaluate (a) the *accuracy* of predictions of monitor error rates by our approach (NCC) compared to the purely data-driven approach (BBC), (b) the *dependency* of this accuracy on the amount of provided data, (c) the *sensitivity* of our approach to the independence assumptions, and (d) the *computational costs* of our approach. Note that in (a), NCC is *not* intended to beat the accuracy of BBC because it is used on different inputs in different circumstances; instead, NCC aims to achieve a comparable accuracy without any traces (m_{gt}, m_{do}) from monitors.

At a high level, our evaluation goes as follows. First, we specify and implement two monitors: $\boldsymbol{M}_{pr}$ in Equation (3) and $\boldsymbol{M}_{rf}$ defined below in Equation (13). Second, we collect multiple labeled traces (outputs and ground truth) of each monitor and detector. Third, we produce three estimates of an error rate of each monitor: ECC uses the error rates of the atomic detector known-by-construction, NCC uses the labeled traces of the atomic detector, and BBC uses the labeled traces of monitors.

To evaluate (a), we compare the residuals of NCC and BBC relative to ECC, which in this case plays the role of the true monitor error rate. To evaluate (b), we compare these residuals at different amounts of data provided to NCC and BBC. To evaluate (c), we compare NCC and BBC on traces with sensor noise that invalidate our independence assumption in Equation (4). Finally, to evaluate (d), we measure the time it takes to perform the five steps of NCC.

6.1 Data Collection

We based our evaluation on the DARPA Assured Autonomy case study described in Section 2. Data was collected a ROS Gazebo UUV simulator (`https://github.com/uuvsimulator/uuv_simulator`) customized with underwater vehicle dynamics and pipeline generation. We executed 73 pipeline-following missions, totalling 7.7 hours of simulation time (the data can be found at `https://github.com/bisc/prob-comp-asst`). Each mission started with the pipeline in view and terminated 30 seconds after the pipeline was no longer visible (either due to a successful completion or getting lost). We kept all the system design parameters (including the controller) fixed across all missions, with the only variables being the random seed that determines the pipeline shape[9] and the initial position relative to the first pipeline segment.

During each mission, the pipeline detector $\boldsymbol{Pl}$ provided a trace of binary values, each indicating the presence of a pipeline in the UUV's view at that moment, at the rate of 1 Hz. Pr(dou) for $\boldsymbol{Pl}$ is set to 0 for simplicity. We used two (mutually exclusive) types of noise in $\boldsymbol{Pl}$: *stateless* and *stateful*. The former has fixed fpr and fnr equal to 0.1 by construction, without any dependency on the past noise (only on the current GT). The latter is statefully dependent on the past 5 detection results, varying with the noise weight $w \in [0, 1]$ from 0.1 at best to 0.5 at worst (a coin toss equivalent), simulating the propensity of noise to persist over time: [10]

$$\begin{aligned} \mathrm{fpr}(t) &= 0.1 + 0.4w \cdot \frac{\sum_{i:1..5} \mathbb{1}(DO^{t-i} = \mathsf{T})}{5} \\ \mathrm{fnr}(t) &= 0.1 + 0.4w \cdot \frac{\sum_{i:1..5} \mathbb{1}(DO^{t-i} = \mathsf{F})}{5} \end{aligned} \tag{12}$$

We investigated two monitors, chosen for their different syntactic structure:

1. The pipeline recovery monitor $\boldsymbol{M}_{pr}$, which has been described in Section 2 and formalized in Equation (3):

$$\boldsymbol{M}_{pr} := \neg_s(\neg_w \boldsymbol{Pl} \rightarrow_w \Diamond_{[1,d]} \boldsymbol{Pl}).$$

2. A monitor $\boldsymbol{M}_{rf}$ for the "reliable following" detector $\boldsymbol{\varphi}_{rf} := \Box_{[0,d]} \boldsymbol{Pl}$, requiring that the pipeline is confidently perceived for d samples:

$$\boldsymbol{M}_{rf} := \neg_s \Box_{[0,d]} \boldsymbol{Pl}. \tag{13}$$

By feeding the ground-truth and noisy detector samples into each monitor, we obtained the m_{gt} and m_{do} traces respectively. Thus, for each of the two composite detectors, each mission provided four binary-valued traces of equal length: a ground truth/noisy pair of traces for the pipeline detection, and a ground truth/noisy pair of traces for the monitor.

All experiments were run on a Lenovo X1 laptop with Ubuntu 18.04 LTS, Intel Core i7-6600U CPU at 2.60GHz, 16 GB DDR3 RAM, and an internal SATA SSD hard drive.

[9]The pipeline is always a non-circular and non-self-intersecting sequence of linear segments, and no segment connection angle is less than 90 degrees.

[10]$\mathbb{1}(a)$ is an indicator function; it returns 1 if $a = \mathsf{T}$ and 0 otherwise.

6.2 Variables and Sampling

We estimate one rate for each monitor: FNR for $\boldsymbol{M}_{pr}$ using Equation (11), and FPR for $\boldsymbol{M}_{rf}$ using a formula derived in the supplement [21]. We chose these specific rates because they have a larger range of values for different d, from 0.2 to over 0.95. The other two rates — $\text{fpr}(\boldsymbol{M}_{pr})$ and $\text{fnr}(\boldsymbol{M}_{rf})$ — have low values, barely exceeding 0.01 at their highest.

Thus, our results have four *independent variables:* a chosen monitor, a mission number, a deadline[11], and a noise weight w. Our primary *dependent variables* are ECC, NCC, and BBC. Given a monitor, ECC depends only on d, whereas NCC and BBC depend on d and a subset of missions from which the data is used. In all comparisons below, we always provided the same subset of missions to both NCC and BBC.

Our secondary dependent variables are the errors between the estimates and the amounts of information available in the traces. We use the *root-mean-squared error* (RMSE) as a standard measure between real-valued point estimates. For estimating FNR, the amount of information is determined by the number of gtt events of the monitor: BBC considers the FNR a probability parameter of a Binomial random variable, and each gtt is a sample of this variable. Likewise, for estimating FPR, the information is measured by the count of gtf events.

The evaluation (b) samples across all non-trivial subsets of our trace dataset (sizes from 1 to 72) with replacement. We pick a fixed number (40) of randomly (uniformly) chosen subsets for each size. In the analyses related to RMSE, the averaging is done over fixed-sized bins for the amounts of information (i.e., the counts of monitor gtt or gtf events). The only exception to this sampling strategy is the first accuracy analysis, where we use our full dataset for an estimate.

6.3 Results

Evaluation (a): Accuracy

We evaluated the accuracy of calculations for different values of deadline d using the full dataset with stateless noise ($w = 0$). For both monitors the differences between NCC, BBC, and ECC were small, with all three closely tracing each other (see the supplement [21] for the figures). For FNR of $\boldsymbol{M}_{pr}$,

[11]The "deadline" is the value of time bound d in both monitor definitions.

the error of BBC increases slightly for $d > 20$. The RMSE of NCC (relative to ECC) across all values of $d \in [2, 30]$ is 0.015, and 0.013 for BBC (also relative to ECC). While NCC has a larger error, it stays consistent across all d values, whereas BBC accuracy varies because BBC needs data for each value of d. For larger $d > 20$, fewer datapoints trigger the monitors ground-truth alarm, making it harder for BBC to produce accurate estimates. For the FPR of $\boldsymbol{M}_{rf}$, the differences were negligible: the average absolute error is on the order of 0.001 for both NCC and BBC.

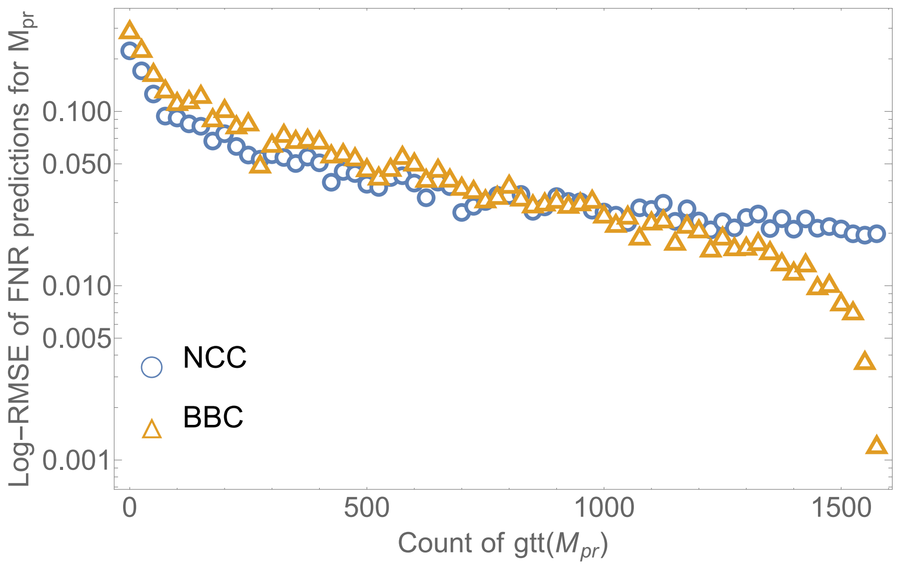

Figure 3: The log-RMSE of FNR predictions ($d = 10$) for M_{pr}, relative to the amount of information.

Evaluation (b): Data Dependency

Here we analyze how estimate error changes relative to the amount of information (keeping $w = 0, d = 10$ for illustration). Figure 3 shows the RMSE of NCC and BBC for $\boldsymbol{M}_{pr}$ produced by sampling different data subsets. Each point in the plot is produced by averaging the squared error for a 25-wide bin[12], and the minimum number of samples per bin is 37. This figure indicates that NCC is more accurate than BBC for small amounts of data, while BBC gets increasingly more accurate for larger amounts. Notice a

[12]Binned over the x-axis; each bin's result is plotted at its lower bound.

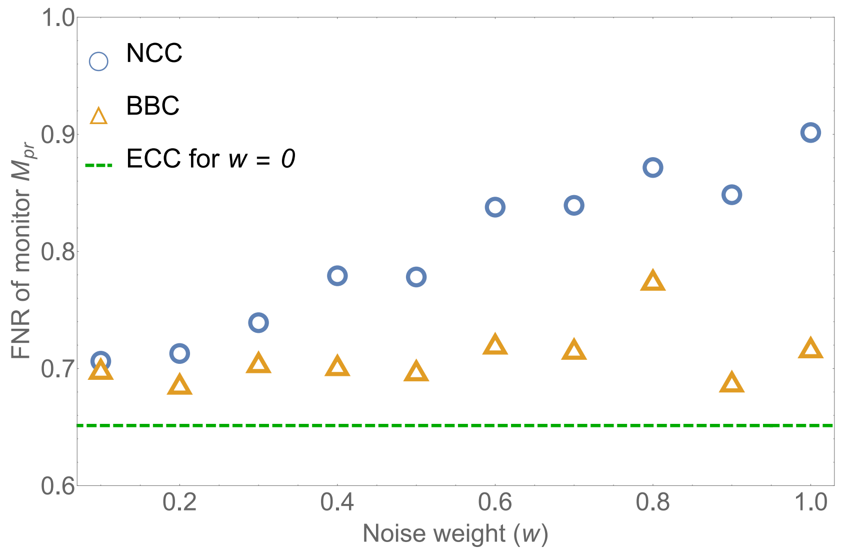

Figure 4: NCC and BBC for FNR of $\boldsymbol{M}_{pr}$ $(d = 10)$ relative to weight w. Baseline ECC is shown for $w = 0$, for context.

logarithmic scale: the small-data errors are orders of magnitude larger than the large-data errors. The sudden drop of BBC is a sampling artifact: BBC converges to the full-dataset estimate of FNR, which in this case happens to be close to ECC. No such phenomenon was observed for $\boldsymbol{M}_{rf}$: NCC is more accurate for all data amounts, as shown in the supplement [21], confirming that the BBC error is higher for larger deadlines.

Evaluation (c): Sensitivity to Independence Assumptions

In the previous two evaluations, the noise profile ($w = 0$) matched our independence assumption (Equation (4)). Here though, we intentionally set $w \in (0, 1]$ to disrupt this assumption and observe the effect on the NCC estimates. Figure 4 shows that for full-data estimates NCC and BBC increase significantly for larger w. We do not have the ground-truth values for monitor error rates for $w > 0$: a true Markov model for their calculation needs the true detector event probabilities, which are simulated and not known exactly. Therefore, ECC is only shown to illustrate the increase in the FNR. NCC diverges from BBC by over 0.1 for $w \geq 0.9$, over-estimating the FNR. The supplement [21] shows an analogous picture for $\boldsymbol{M}_{rf}$.

Evaluation (d): Computational Costs

To measure the computation times of our approach, we consider its two parts: *formula manipulation* (Steps 1, 2, 3, and 5)[13] and *estimation from data* (Step 4). The former is performed in ECC and NCC and has indistinguishable performance in both cases. The latter is performed in NCC (for detector probabilities $\hat{\mathsf{Pr}}$) and BBC (for monitor rate estimates $\hat{\text{er}}$).

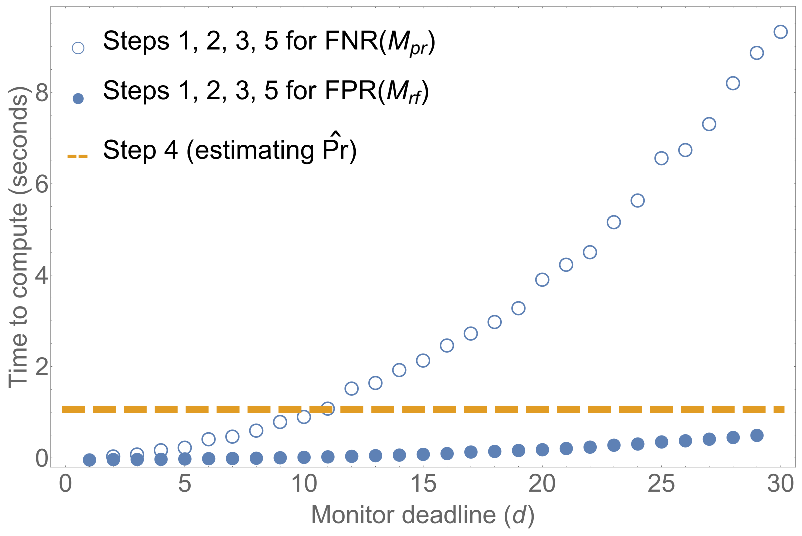

Figure 5: The dependency of computation times of two formula manipulations and data estimation on the formula size.

The computation times for formula manipulation, shown in Figure 5, increase with monitor formula size, linearly determined by parameter d. While small formulas take fractions of a second, the magnitude of growth varies with the formula and the rule list. The time to analyze $\text{fpr}(\boldsymbol{M}_{rf})$ never takes longer than a second, exhibiting a slow and linear growth. On the other hand, the analysis of $\text{fnr}(\boldsymbol{M}_{pr})$ exhibits a polynomial growth in complexity and takes close to 10 seconds for $d = 30$. We witnessed an exponential complexity growth for some rule lists that introduce many probability terms (e.g., by transforming disjunctions into sums of probabilities). The estimation of rates for monitors and detectors alike does not depend on the formula

[13]Step 5 is virtually instantaneous and was timed together with the symbolic manipulation steps (1, 2, and 3), which are computationally intensive.

size and, for our dataset, takes approximately 1 second on average for one detector/monitor.

6.4 Interpretation of Results

NCC and BBC estimates benefit differently from getting more data. How useful extra data is depends on the probability of the GT events used in the respective calculation. For estimating fnr($\boldsymbol{M}_{pr}$), NCC uses events gtf($\boldsymbol{Pl}$) (to estimate fpr($\boldsymbol{Pl}$)), which are 1.58 times more likely than the gtt($\boldsymbol{M}_{pr}$) events (for $d = 10$) used by BBC. Whereas for estimating fpr($\boldsymbol{M}_{rf}$), NCC uses events gtt($\boldsymbol{Pl}$) (to estimate fnr($\boldsymbol{Pl}$)), which are 2.02 times more likely than the gtf($\boldsymbol{M}_{rf}$) events (for $d = 10$) used by BBC. Thus, NCC for $\boldsymbol{M}_{rf}$ gets a larger information advantage over BBC per trace, than in case of $\boldsymbol{M}_{pr}$.

Absence of relevant ground-truth events puts BBC at a significant disadvantage. We notice that larger d for $\boldsymbol{M}_{pr}$, but not $\boldsymbol{M}_{rf}$. This is explained by the reduced chance of pipe-loss events (gtt($\boldsymbol{M}_{pr}$)) as the definition of a pipe loss is relaxed by increasing d. On the other hand, satisfaction events for reliable-following (gtf($\boldsymbol{M}_{pr}$)) do not drop as drastically, since for most of each trace, the UUV follows the pipeline successfully. As a result, NCC performs better than BBC for $d \geq 15$. We conclude that composite detectors with infrequent violations would benefit from our approach for FNR estimation, whereas those with infrequent satisfactions should use our approach for FPR estimation.

The evaluation also highlights two benefits of compositions. First, the events for atomic detectors are typically more prevalent than those for monitors, making it easier to estimate and compose the error rates of detectors — as opposed to observing rare monitored events. Second, NCC can reuse the results of its steps. If the monitor formula changes, it can re-do the formula manipulation and reuse the estimates $\hat{\mathsf{Pr}}$. If the detectors or available data change, NCC can re-do Steps 4 and 5 and reuse the outcomes of symbolic manipulation for any value of d. In contrast, whenever the formula (e.g., the value of d) changes, BBC needs new monitor traces and re-does the whole analysis. These benefits have the potential to reduce the data collection burden and speed up design space exploration: poor-quality monitors can be eliminated from the design or used to justify hardware/software upgrades.

6.5 Limitations

Our approach is indeed sensitive to the independence assumptions: the more they are violated, the more we deviate from the data-driven estimates. However, for $\boldsymbol{M}_{pr}$ and $\boldsymbol{M}_{rf}$, assuming independence of events that are in fact dependent leads to upper-bounding the true error rate. This happens because independent events lead to smaller probabilities in Equation (10) compared to a stateful dependency. Hence, in this case, our estimates are conservative upper bounds.

Therefore, one guideline is to carefully pick independence assumptions by consulting the domain and system experts (by asking them about stateful/stateless events, causal relations, etc.). For example, we could handle $w > 0$ in the noise model in Equation (12) by making Markovian assumptions on detector $\boldsymbol{Pl}$ such as $DO^t \perp\!\!\!\perp DO^{t-j} \mid GT^t, DO^{t-1} \ldots DO^{t-5}$ for $j > 5$, instead of Equation (4). This would change the FNR calculation, requiring estimation of e.g. $\mathsf{Pr}\big(\mathrm{dof}(\boldsymbol{Pl}^t) \mid \mathrm{gtf}(\boldsymbol{Pl}^t), \mathrm{dof}(\boldsymbol{Pl}^{t-1}) \ldots \mathrm{dof}(\boldsymbol{Pl}^{t-5})\big)$. Our analysis procedure would remain the same.

Another way to handle independence assumptions is to validate them on data, e.g., by informally comparing probabilities or doing chi-squared tests. Any way of validating these assumptions would fit into our framework.

The produced error rate estimates are not invariant to some changes in the system. Examples include changing the sampling rate and re-training the perception. To update the estimates for new perception, data collection and Step 4 of our analysis analysis need to be repeated. However, the $\boldsymbol{Pl}$ rate estimates we produced would be invariant to deploying a higher-performance controller, which may increase $\Pr(\mathrm{gtt}(\boldsymbol{Pl}))$ but not change $\mathrm{fpr}(\boldsymbol{Pl})$. We plan to investigate reusing part of the data and calculations for limited changes in the system.

The relative errors of NCC and BBC will vary for other monitors and systems. Simple monitors of low-variance systems may not benefit from our approach. For complex systems though, our approach could avoid prohibitively difficult testing of rare monitored conditions. Either way, we expect our qualitative observations to still hold.

7 Related Work

The existing work related to this paper can be grouped in two large categories: (i) detection and monitoring of uncertain events and (ii) probabilistic and logical reasoning.

7.1 Detection, Monitoring, and Error

Combination of detection algorithms is a well-explored technique in many areas, giving rise to a variety of relations between the objects of composition. In perception, sensor fusion reconciles multiple uncertainties about an observed phenomenon. Popular approaches, such as Kalman filtering and Bayesian methods, link the inputs via process and measurement models [4], whereas we do not rely on such models and therefore only require specific knowledge about certain probabilities. In machine learning, ensemble methods such as bagging, boosting, and stacking [22] aim to improve the learning performance given multiple algorithms (e.g., for anomaly detection [3]). They aggregate the outputs of learners in dynamic ways, such as voting on them or averaging them. Our paper focuses on pre-specified logical/temporal relations between the outputs of black-box detection algorithms.

Logically specified properties are often used for safety monitoring and runtime verification in CPS. For STL, robustness [1, 10] takes a worst-case view on error, measuring the maximum deviation to violate the property. We focus on the average-case (probabilistic) error measures, such as false positive and false negative rates [11]. Typically estimated directly from samples, such measures do not necessarily transfer to the deployment environment. We provide a more rigorous way to estimate error rates based on probabilistic computations.

Monitoring of probabilistic logical properties addresses a different problem, but uses relevant techniques: it relies on the information from the observed data to test whether a probability constraint holds. Approaches like ProMo [12], PTPSC [25], and BaProMon [26] perform online checks of probability constraints via as statistical hypothesis tests. Similarly, statistical approaches can estimate monitor error rates, although at design time: each rate is modeled as a Bernoulli random variable, with its parameter estimated from monitor traces. As we show in Section 6, the downside of this approach is the need to observe and label potentially rare events for each monitor under analysis, necessitating extensive data collection. In contrast, our approach uses atomic, more observable inputs.

7.2 Probability, Logic, and Inference

We distinguish two groups of combinations of formal logics and probability theory [13]: those with probabilistic entailment (i.e., logics of probability [23]), and those with probability operators (i.e., probabilistic logics [9]). The former type of logics (e.g., Adams's logic [2]) use the classic logical syntax and perform *uncertainty propagation*, from the uncertainties of premises to the uncertainty of the conclusion. These logics address a problem different from ours: they manipulate uncertain knowledge about certain statements (e.g., "the system is safe" has a 95% chance of being true), whereas our paper manipulates certain statements about uncertain phenomena (e.g., "the chance of false positive is 5%" is true).

The latter type of logics use probability operators over events; for example, the Probabilistic Computation Tree Logic (PCTL) [14] can describe probabilistic transition systems. A typical use of such logics is to model-check a desired property on a known model, or develop a set of premises that would prove or disprove the property deductively. Similarly to these logics, we reason about probabilities of related events, but do not use explicit probability operators in our syntax. Further, our approach also does not require models of systems dynamics or trustworthy premises that logically entail the desired statement.

A notable recent probabilistic logic is the Chance Constrained Temporal Logic (C2TL) [16], which turns constraints on probabilities of deterministic predicates into optimization problems for controller synthesis. Similar to our approach, C2TL probabilistically handles perception uncertainty of logically related events. The key difference is that we solve a probability estimation — not control — problem and thus rely on error rates of atomic detectors instead of chance constraints and a plant model.

Logic-free probabilistic inference often uses graphical models, such as Bayesian nets and Markov chains [18]. Usually approximate, this inference computes a probability query given a network fitted to the data. Instead of fitting a model, we discharge part of the problem by performing exact symbolic transformations, which allows us to handle arbitrary logical relationships. In the future, we hope to use graphical models to perform (in)dependence calculations for our approach.

8 Discussion

The reader may notice that the values of FPR and FNR described in the evaluation are unacceptably high for a safety-critical monitor: most of the time, the M_{pr} would not detect that the UUV has lost the pipeline. This finding came unexpectedly to the authors, and it illustrates the benefit of applying our approach in early design stages. Two responses would reduce the monitor's error rates: significant improvements to the pipe detector (i.e., reducing its FPR and FNR below 0.1) or re-defining the properties (e.g., the loss of pipeline occurs if it was not visible for 2 out of 3 seconds). For instance, if the pipe detector is improved such that $\text{fpr}(\boldsymbol{Pl}) = 0.01$, then $\text{fnr}(\boldsymbol{M}_{pr})$ is lowered from 0.651 to 0.096 for $d = 10$.Similarly, setting $\text{fnr}(\boldsymbol{Pl}) = 0.01$ reduces $\text{fpr}(\boldsymbol{M}_{pr})$ from 0.686 to 0.105 for $d = 10$. This reduction indicates that our estimation can inform impactful design choices.

We note a dual relationship between the modalities in detector formulas and the FNR/FPR computations. The detectors with the box modality lead to straightforward computations of FNR (given standard independence assumptions) and relatively large values for it, whereas the computations for FPR rely on complex formulas and produce small values. For diamond modalities, the situation is reversed: simple computations for large FPRs, and complex computations for small FNRs. Formally, this is due to the conditioning in the error rate definition — either on a conjunction (simple) or a disjunction (complex).

8.1 Future Work

Several important research directions are opened by this paper. One such direction is enhancing the framework outputs. First, one could improve accuracy for limited data by moving from point estimates to interval estimates with uncertainty. Also, the framework could explore the design space by optimizing error rates over parameters in probability formulas.

Another direction is transitioning the analysis to run time. One could continuously maintain the estimates of the analysis inputs and update the error rate estimates as needed. Monitoring independence assumptions between detectors would be particularly useful to compensate for design-time simplifications. Furthermore, our analysis could predict the probability of future events (e.g., crashes) that are logically connected to the observed events.

Finally, one can envision a CPS design environment that iterates through sets of independence assumptions that would be sufficient to calculate a desired rate given the known inputs. This environment would suggests alternative independence setups to the engineers along with the additional required inputs (and their uncertainties), making the design choices more transparent and automated.

8.2 Conclusion

This paper proposed a modeling and analysis framework for logical compositions of detectors. The modeling part represents atomic detectors as compact probability spaces and composes them with 3-value temporal-logic formulas. The analysis part estimates the error rates of composite detectors by manipulating symbolic probability expressions, via independence inference and estimation from data. Our evaluation on a pipeline detection case study showed that our framework's estimates are accurate within fractions of a percent and comparable to purely data-based (non-compositional) estimates on monitor traces.

Acknowledgments

We thank Taylor Carpenter and Hyonyoung Choi for their help with the UUV simulator. We also thank the anonymous reviewers for their insightful comments. This work is supported in part by the Air Force Research Laboratory and the Defense Advanced Research Projects Agency as part of the Assured Autonomy program under Contract No. FA8750-18-C-0090, and by the Office of Naval Research under Contract No. N68335-19-C-0200.

References

[1] H. Abbas, Y. V. Pant, and R. Mangharam. Temporal Logic Robustness for General Signal Classes. *International Conference on Hybrid Systems: Computation and Control*, Apr. 2019.

[2] E. Adams. *A Primer of Probability Logic.* CSLI Publications, 1998.

[3] C. C. Aggarwal. *Outlier Analysis.* Springer, New York, Jan. 2013.

[4] Y. Bar-Shalom, P. K. Willett, and X. Tian. *Tracking and Data Fusion: A Handbook of Algorithms.* Storrs, Connecticut, Apr. 2011.

[5] E. Bartocci, J. Deshmukh, A. Donze, G. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. *Lecture Notes in Computer Science*, pages 135–175, Jan. 2018.

[6] A. Bauer, M. Leucker, and C. Schallhart. Monitoring of Real-Time Properties. In *FSTTCS*, Berlin, Heidelberg, 2006. Springer.

[7] P. J. Brockwell. *Time Series: Theory and Methods.* Springer, New York, NY, 2009.

[8] H. Cramer. *Mathematical Methods of Statistics.* Princeton UP, 1999.

[9] G. De Bona, F. G. Cozman, and M. Finger. Towards Classifying Propositional Probabilistic Logics. *J. of Applied Logic*, 12(3):349–368, 2014.

[10] J. V. Deshmukh, A. Donze, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia. Robust Online Monitoring of Signal Temporal Logic. *Form. Methods Syst. Des.*, 51(1):5–30, Aug. 2017.

[11] J. L. Fleiss, B. Levin, and M. C. Paik. *Statistical Methods for Rates & Proportions.* Wiley-Interscience, Hoboken, N.J, 3rd edition, 2003.

[12] L. Grunske and P. Zhang. Monitoring Probabilistic Properties. In *ESEC/FSE*, pages 183–192, New York, NY, USA, 2009. ACM.

[13] R. Haenni, J.-W. Romeijn, G. Wheeler, and J. Williamson. *Probabilistic Logics and Probabilistic Networks.* Springer Netherlands, 2011.

[14] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.

[15] K. Havelund and G. Reger. Runtime Verification Logics A Language Design Perspective. In *Models, Algorithms, Logics and Tools: Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, pages 310–338. Springer, Cham, Switzerland, 2017.

[16] S. Jha, V. Raman, D. Sadigh, and S. A. Seshia. Safe Autonomy Under Perception Uncertainty Using Chance-Constrained Temporal Logic. *Journal of Automated Reasoning*, 60(1):43–62, Jan. 2018.

[17] S. C. Kleene. *Introduction to metamathematics*. North-Holland Publishing Co, Amsterdam, The Netherlands, 1952.

[18] D. Koller, N. Friedman, and F. Bach. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, MA, July 2009.

[19] B. Konikowska. A Three-Valued Linear Temporal Logic for Reasoning about Concurrency. Technical report, ICS PAC, Warsaw, Poland, Nov. 1998.

[20] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, 1977*, pages 46–57, Oct. 1977.

[21] I. Ruchkin, O. Sokolsky, J. Weimer, T. Hedaoo, and I. Lee. Supplementary Materials for Compositional Probabilistic Analysis of Temporal Properties over Stochastic Detectors. Technical report, July 2020.

[22] O. Sagi and L. Rokach. Ensemble learning: A survey. *WIREs Data Mining Knowl Discov*, 8(4):e1249, 2018.

[23] A. Sernadas, J. Rasga, and C. Sernadas. On probability and logic. *Portugaliae Mathematica*, 74(4):267–313, Feb. 2018.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[25] P. Zhang, W. Li, D. Wan, and L. Grunske. Monitoring of Probabilistic Timed Property Sequence Charts. *Software: Practice and Experience*, 41(7):841–866, 2011.

[26] Y. Zhu, M. Xu, P. Zhang, W. Li, and H. Leung. Bayesian Probabilistic Monitor: A New and Efficient Probabilistic Monitoring Approach Based on Bayesian Statistics. In *2013 13th International Conference on Quality Software*, pages 45–54, July 2013.