International Journal of Approximate Reasoning 119 (2020) 325–352

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Incremental reasoning in probabilistic Signal Temporal Logic

Mattias Tiger *, Fredrik Heintz

Linköpings Universitet, 581 83 Linköping, Sweden

ARTICLE INFO

Article history:
Received 14 May 2019
Received in revised form 7 January 2020
Accepted 20 January 2020
Available online 24 January 2020

Keywords:
Knowledge representation
Stream reasoning
Incremental reasoning
Probabilistic logic
Temporal logic
Runtime verification

ABSTRACT

Robot safety is of growing concern given recent developments in intelligent autonomous systems. For complex agents operating in uncertain, complex and rapidly-changing environments it is difficult to guarantee safety without imposing unrealistic assumptions and restrictions. It is therefore necessary to complement traditional formal verification with monitoring of the running system after deployment. Runtime verification can be used to monitor that an agent behaves according to a formal specification. The specification can contain safety-related requirements and assumptions about the environment, environment-agent interactions and agent-agent interactions. A key problem is the uncertain and changing nature of the environment. This necessitates requirements on how probable a certain outcome is and on predictions of future states. We propose Probabilistic Signal Temporal Logic (ProbSTL) by extending Signal Temporal Logic with a sub-language to allow statements over probabilities, observations and predictions. We further introduce and prove the correctness of the incremental stream reasoning technique progression over well-formed formulas in ProbSTL. Experimental evaluations demonstrate the applicability and benefits of ProbSTL for robot safety.

© 2020 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Robot safety is paramount when robotic platforms such as industrial robots, intelligent vehicles and aerial drones are intended to co-exist, side-by-side, with people in human-tailored environments. Consider the following example scenario. An Unmanned Aerial Vehicle (UAV) is tasked with delivering a package from a warehouse to a designated drop-site. The UAV is however not allowed to fly around freely; it has to adhere to various constraints on where to fly, how to behave and how to interact with other agents. Some constraints are for safety reasons such as staying at an altitude above peoples' heads but below other air traffic, while other constrains may include no-fly-zones (Fig. 1) and the requirement to behave predictable close to other UAVs. Adding to the difficulty is the fact that the current and predicted position, altitude and other states of the UAV, and of other nearby agents, are uncertain due to imprecise perception, noisy GPS-signals and the sheer complexity of the real world.

Complex agents operating in uncertain, complex and changing environments make it difficult to guarantee safety before deployment without unrealistic assumptions and restrictions. It is therefore necessary to complement traditional formal verification with monitoring of the running system after deployment. In particular, robotic systems perceive the world

* Corresponding author.
E-mail address: mattias.tiger@liu.se (M. Tiger).
URL: http://www.ida.liu.se~matti23/ (M. Tiger).

https://doi.org/10.1016/j.ijar.2020.01.009
0888-613X/© 2020 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Nomenclature

Signal

$x(t), x(t_n)$	Signal
$x^c(t)$	A continuous-time signal which is sampled to produce a time-discrete signal $x(t_n)$.
$\boldsymbol{x}(t)$	Stochastic signal
S	Tuple of signals
S_n	Tuple of signals' states
$S_{\leq n}$	Tuple of signal prefixes

Sets

$\mathbb{B}$	The Boolean numbers
$\mathbb{N}$	The natural numbers
$\mathbb{R}$	The real numbers
$\mathbb{S}$	Stochastic variables
$\mathbb{E}$	Events
$\mathbb{T}, \mathbb{T}_x$	Time domain (of signal x)
$\mathbb{D}, \mathbb{D}_x$	Value domain (of signal x)
$F_{\mathbb{B}}, F_{\mathbb{R}}, F_{\mathbb{S}}$	Functions

Other Symbols

E	Event
$f_{\mathbb{B}}, f_{\mathbb{R}}, f_{\mathbb{S}}$	Functions
$\mathcal{M}$	Logical model
ϕ, ψ	Well formed formula in a logic
$\boldsymbol{x}, \boldsymbol{x}_t, \boldsymbol{x}_{t'\|t}$	Stochastic variable.
$p(\boldsymbol{x})$	Probability density function over stochastic variable $\boldsymbol{x}$.
t, t_n, t'	Time point (real valued)
X	Tuple of stochastic variables
$\mathcal{I}$	Interval Coverage (a sequence of intervals)
$\mathcal{I}_n$	Interval (left-closed right-open)
I	Interval closed at both ends

Syntax

const	Numerical-constant symbol ($\mathsf{const} \in \mathcal{C}$)
C	Set of numerical constants
E	Event symbol ($\mathsf{E} \in \mathcal{E}$)
$\mathcal{E}$	Set of event symbols
$\mathsf{f}_{\mathbb{B}}, \mathsf{f}_{\mathbb{R}}, \mathsf{f}_{\mathbb{S}}$	Function symbol
$\mathsf{F}_{\mathbb{B}}, \mathsf{F}_{\mathbb{R}}, \mathsf{F}_{\mathbb{S}}$	Set of function symbols
$\mathsf{X}_d, \mathsf{X}_p$	Set of Signal-state symbols
$\mathsf{x}_t, \mathsf{x}_{t'\|t}$	Signal state symbol
p	Proposition

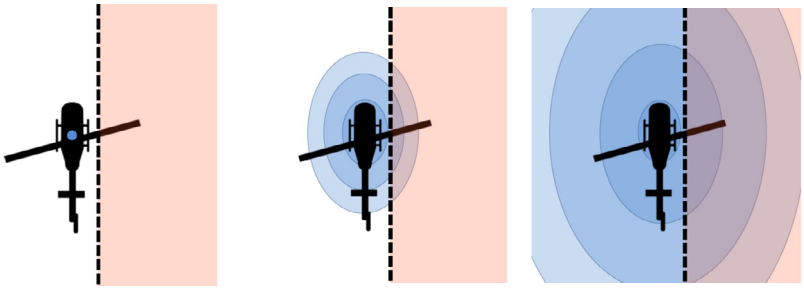

Fig. 1. *Is the UAV inside the no-fly-zone?* The truth of statements about uncertain states such as GPS-positions are unclear unless uncertainty is taken into account explicitly. According to the point estimate (**left**) the UAV is not inside the no-fly-zone. With some position uncertainty it is however 10% probable that it is inside (**middle**) and 40% probable with even higher uncertainty (**right**). It is not uncommon for state uncertainty to vary over time in robotic systems. Robustness to perception uncertainty and its possible variation over time is very important for runtime verification of robot safety.

through sensors and produce state streams about the environment. Properties can be monitored in a timely manner by incremental reasoning over the rapidly-changing information in the stream, known as *stream reasoning*. A recent survey [1] in the stream reasoning field identified the need for robustness to imperfect data such as noisy data. Previously, the logic P-MTL was proposed [2] for supporting progression-based runtime verification over uncertain streams where the semantics was informally grounded in a stream reasoning architecture. P-MTL has since been used to improve UAV motion planning robustness and enable anticipatory behavior by reasoning over uncertain predictions [3].

The main contribution of this paper is the Probabilistic Signal Temporal Logic ProbSTL for predictive stream reasoning to assure the safe behavior of robotic systems in shared environments. ProbSTL extends Signal Temporal Logic and provides a formal semantic grounding for a fragment of P-MTL.

Runtime verification is an active research field [4] concerned with monitoring that properties hold over traces, streams or signals. The focus is mostly on monitoring software program traces, but it also covers monitoring streams or signals from cyber-physical systems and simulators of such systems [5]. Most real-world physical systems are stochastic in nature, both in terms of process noise and observations noise. To deal with uncertainty in the stream/signal values it is common in the runtime verification literature to either have error margins in the formulas or to use fuzzy logic for expressing the formulas [6,7]. Neither approach considers the probability-theoretic modeling of the uncertainty in the causal physical system and the stochastic signals they produce. Yet this is what is done in estimation theory and statistical sensor fusion [8], which

is typically applied to produce useful estimates about physical properties of concern in the field of robotics. For example, incremental probabilistic inference using Bayesian filtering [9] allows for on-line estimation of physical states over time.

To make use of the probabilistic information produced by state estimation it is necessary to formulate the monitoring formulas using probabilistic logic. There are many different probabilistic logics such as probabilistic distributional clauses [10] or temporal logics with uncertainty about time [11] rather than state. There are also full probabilistic logic programming languages [12] which are powerful enough to describe many cases of state estimation directly in the language. The main limitation of these are restrictions on what state estimation is supported, and the reliance on statistical sampling methods as a general purpose inference engine. Such inference methods are typically too slow for real-time state estimation and runtime verification in applications such as robotics where there is a need for timely formula violation detection. These probabilistic logics do not make use of the highly optimized inference methods from the fields of signal processing, automatic control or robotics. Examples of such methods are analytical closed-form solutions, variational approximations or Rao-Blackwellized particle filters [8].

Temporal logics have been used to do runtime verification for execution monitoring in robotics [13] using the stream reasoning technique progression over Metric Temporal Logic (MTL) formulas. MTL [14] extends LTL with time-intervals on the temporal operators. Metric Interval Temporal Logic (MITL) [15] is a decidable fragment of MTL[1] where the temporal operator interval cannot be a single time-point. Signal Temporal Logic (STL) [16] extends MITL with real-time (dense-time) semantics, grounding statements in the logic to continuous-time signals. There are many extensions to STL such as a freeze operator for signal values [17] and work where STL specifications (set of formulas) are inferred from example signals and used to classify new signals [18]. STL allows for specifying requirements over deterministic signals such as real and Boolean signals. To the best of the authors knowledge there exist no extension to stochastic signals where the signal values are probabilistically described as uncertain.

PrSTL [19] is a Probabilistic Signal Temporal Logic where STL is extended with a probabilistic proposition. This is used for specifying probabilistic safety properties for controllers in controller synthesis. The interpretation of the proposition is that a stochastic function is applied to the STL signal for each time point and the probability of the resulting stochastic variable's value being less than 0 is constrained to be larger than a threshold,

$$\text{Interpretation}[\mathrm{p}^{\text{threshold}}_{\boldsymbol{x}_t}] = Pr\big(f_{\boldsymbol{x}_t}(\text{signal}(t)) < 0\big) > \text{threshold},$$

where $\boldsymbol{x}_t$ is a stochastic variable which makes the function $f_{\boldsymbol{x}_t}$ stochastic. The stochastic variable $\boldsymbol{x}_t$ is external to the deterministic STL signal.

In this work we extend STL to stochastic signals, grounded in estimation theory and physical signals. ProbSTL has a much richer syntax for probabilistic events, signal transformations, and expressions for thresholds than PrSTL. ProbSTL further allows for incremental runtime verification where constraints over deterministic observations and uncertain predictions are expressed explicitly in the logic, can be efficiently checked incrementally and constraint violations are detected rapidly.

2. Preliminaries

Sensors allow information-processing systems such as robotic systems to observe properties of the physical environment. Many physical properties vary over time, such as the position of an object in relation to some fixed frame of reference. Some of these continuous-time signals are directly observable from sensors which produce discrete-time signals through sampling. Other properties are only indirectly observed by signal transformations of the observable signals.

Definition 1 *(Signal).* A signal $x(t)$ with time domain $\mathbb{T}$ and value domain $\mathbb{D}$ is a function $x : \mathbb{T} \to \mathbb{D}$. A tuple of signals is denoted by S and S_t denote the tuple of signal values $x(t)$ for all $x \in S$ at time $t \in \mathbb{T}$. When considering multiple signals we let $\mathbb{T}_x$ denote the time domain and $\mathbb{D}_x$ denote the value domain of signal $x(t)$.

Definition 2 *(Continuous-time signal).* A continuous-time signal is a signal with time domain $\mathbb{T} = \mathbb{R}$.

Definition 3 *(Discrete-time signal).* A discrete-time signal (or stream) $x(t_n) = x_0 x_1 x_2 \dots$ is a signal with time domain $\{t_0, t_1, \dots, t_n, \dots\} = \mathbb{T} \subset \mathbb{R}$ where $n \in \mathbb{N}_{\geq 0}$. The notation $x(t_n) = x_n$ is used. The tuple of signal values S_n at time-point t_n is called a state. The first n states in a discrete-time signal (stream) is denoted as a stream prefix $S_{\leq n}$.

Let $x^c(t)$ denote a continuous-time signal of interest, e.g. the *actual* altitude of a UAV, and let $x(t_n)$ denote the observable discrete-time signal which is a (potentially noisy) discretized surrogate of $x^c(t)$. A fundamental problem is to draw conclusions given $x(t_n)$ which are also valid for $x^c(t)$.

[1] It is however decidable to verify that a formula holds over a (possibly infinite) trace/stream in MTL.

2.1. Signal Temporal Logic

Signal Temporal Logic (STL) [16] is an extension of MITL where propositions are grounded in Boolean-valued functions over signals. The continuous-time signals are typically assumed to be of finite length in time ($\mathbb{T} = \{t \in \mathbb{R} \mid 0 \leq t \leq t_N < \infty\}$) and a common value domain of the signals is the real numbers, $\mathbb{D} = \mathbb{R}$. It is further typically assumed in the STL literature that the continuous-time signals are of *finite variability*, allowing the signals to be fully represented by a finite length discrete-time signal.

A continuous-time signal $x^c(t)$ is of finite variability if it has a *finite interval coverage* [16]. A finite interval coverage is a finite set of non-overlapping intervals which spans $\mathbb{T}$ and for which the signal is constant for the duration within each individual interval, see Definition 4.

Definition 4 *(Finite interval coverage).* The sequence of left-closed right-open intervals $\mathcal{I} = \mathcal{I}_1, \mathcal{I}_2, \ldots$ is a *finite interval coverage* of continuous-time signal $x(t)$ with time domain $\mathbb{T}$ if the following properties hold

1. $|\mathcal{I}| < \infty$
2. $\mathbb{T} = \bigcup_{\mathcal{I}_n \in \mathcal{I}} \mathcal{I}_n$
3. For all $(\mathcal{I}_n, \mathcal{I}_m) \in \mathcal{I} \times \mathcal{I}$ where $n \neq m$ it is the case that $\mathcal{I}_n \cap \mathcal{I}_m = \emptyset$.
4. For all $t, t' \in \mathcal{I}_n$ for all $\mathcal{I}_n \in \mathcal{I}$ it is the case that $x(t) = x(t')$.

A continuous-time signal $x^c(t)$ with a finite interval coverage can always be fully represented by a discrete-time signal $x(t_n)$. That is, a discrete-time signal $x(t_n)$ can always be constructed such that, for any $t \in \mathbb{T}_{x^c}$ there exists a $t_n \in \mathbb{T}_x$ such that $x^c(t) = x(t_n)$. This is the case since the $x^c(t)$ has a finite number of signal values coinciding with the finite set of time points which mark the start of each of the intervals in the finite interval coverage. That is, the time domain of the discrete-time signal $x(t_n)$ is the set of time points $t_n \in \mathbb{T}_x = \{t_n \mid [t_n\ t_{n+1}) = \mathcal{I}_n \in \mathcal{I}\}$ and the discrete-time signal $x(t_n)$ is defined as $\forall t_n \in \mathbb{T}_x[x(t_n) = x^c(t_n)]$. For any $t \in \mathbb{T}_{x^c}$ then t_n is uniquely determined by $[t_n\ t_{n+1}) = \mathcal{I}_n \in \mathcal{I}$ for the $\mathcal{I}_n$ where $t \in \mathcal{I}_n$.

Definition 5 *(STL model).* STL is defined over a model $\mathcal{M} = \langle S, F_{\mathbb{B}} \rangle$ where S is a tuple of finite discrete-time signals with the same time domain $\mathbb{T}$ and $F_{\mathbb{B}}$ is a set of functions from real-valued signal values to Boolean-values $\left(f_{\mathbb{B}} \in F_{\mathbb{B}} \subset \{f \mid f : \mathbb{R}^{|S|} \to \{\top, \bot\}\}\right)$. Each of the finite discrete-time signals in S *represents* a continuous-time signal with *finite variability*.

Definition 6 *(STL syntax).* An STL statement ϕ for an STL model $\mathcal{M}$ is a well-formed formula (wff) iff it follows the syntax:

$$\phi := \top \mid \mathsf{p} \mid \neg\phi \mid \phi \vee \psi \mid \phi\, \mathcal{U}_I\, \psi$$

where $I = [a\ b]$ is an interval with $a < b$, $a, b \in \mathbb{R}$ and p is a proposition. Additionally, we allow for the usual logical connectives $\wedge, \rightarrow, \leftrightarrow$ as well as the temporal operators $\Diamond_I\ \phi = \top\, \mathcal{U}_I\ \phi$, $\Box_I\ \phi = \neg\, \Diamond_I\, \neg\, \phi$, $\Diamond\ \phi = \Diamond_{[0,\infty)}\ \phi$, $\Box\ \phi = \Box_{[0,\infty)}\ \phi$ as syntactic sugar.

Definition 7 *(STL semantics).* The semantics of STL are defined for an STL model, discrete time-point $\{n \in \mathbb{N}_{\geq 0} \mid n < |\mathbb{T}|\}$, a tuple of signals S (with S_n denoting a tuple of signal values), intervals $I \in \{[a\ b] \mid a < b \wedge a, b \in \mathbb{R}\}$, functions $f_{\mathsf{p}} \in F_{\mathbb{B}}$ (each proposition p is associated with a function $f_{\mathsf{p}} \in F_{\mathbb{B}}$) and well-formed formulas ϕ, ψ:

$$\begin{aligned}
&\mathcal{M}, n \models \top && \\
&\mathcal{M}, n \models \mathsf{p} && \text{iff } f_{\mathsf{p}}(S_n) \\
&\mathcal{M}, n \models \neg\phi && \text{iff } \mathcal{M}, n \not\models \phi \\
&\mathcal{M}, n \models \phi \vee \psi && \text{iff } \mathcal{M}, n \models \phi \text{ or } \mathcal{M}, n \models \psi \\
&\mathcal{M}, n \models \phi\, \mathcal{U}_I\, \psi && \text{iff } \exists n' \in n + I \\
& && \quad \left(\mathcal{M}, n' \models \psi \text{ and } \forall n'' \in [n, n')\ \mathcal{M}, n'' \models \phi\right)
\end{aligned}$$

where we also define the following operations on intervals $I = [a\ b]$

$$\begin{aligned}
n\ +\ I &:= [t_n + a\ \ t_n + b] \\
n\ \in\ I &:= n \in \{j \mid t_j \in \mathbb{T} \wedge t_j \in I\} \\
I\ <0 &:= b < 0
\end{aligned}$$

The STL monitoring algorithm [16] determines the truth value of a well-formed STL formula by a backwards-going procedure. The procedure requires the complete (finite-length) signal to have been observed, which may be undesirable depending on application.

2.2. Progression

Progression [20] is an incremental formula rewriting procedure for determining the truth value of a well-formed MTL formula over the trace of a discrete-time signal. Satisfaction can be determined in a timely manner before the signal has terminated (the continuous-time signal can in-fact be of infinite length in time, $t_N = \infty$) and the discrete-time signal is allowed to be of countably infinite length.

A discrete-time signal of countably infinite length can fully represent a continuous-time signal of *local* finite variability. The finite variability property can straight-forwardly be generalized to *local* finite variability as follows. Let $\mathcal{I}$ be an interval coverage of a continuous-time signal with infinite length in time. Let a subset of the interval coverage be denoted $\mathcal{I}_{t_a \leq t \leq t_b} = \{[a\ b] \,|\, t_a \ \leq\ a,\ b\ \leq\ t_b,\ [a\ b] \in \mathcal{I}\}$. Any interval subset $\mathcal{I}_{t_a \leq t \leq t_b} \subset \mathcal{I}$, over a finite-time interval $[t_a\ t_b]$, has to be finite. A countably-infinite length discrete-time signal $x(t_n)$ with time domain $\mathbb{T}$ can then similarly as before be constructed from the interval coverage, $\forall t_n \in \mathbb{T}\,[x(t_n) = x^c(t_n)]$.

Repeated application of progression yields a path checking procedure which, given a stream (interpretation) and a formula, checks if this formula is true or false given this interpretation at time-point zero [21]. Progression works by syntactic re-writing and has linear time complexity in the length of the formula. But since progression is an iterative procedure the re-written formula can grow to be exponentially large. For typical applications this has however been shown [22] not to be an issue. We introduce a progression procedure for STL in Algorithm 1 and show its correctness in Theorem 1.

Algorithm 1: Progression for STL.

```
1  function PROGRESS(φ, S_n, Δ):
2  if φ = φ_1 ∨ φ_2 then
3  |  return PROGRESS(φ_1, S_n, Δ) ∨ PROGRESS(φ_2, S_n, Δ)
4  else if φ = ¬φ_1 then
5  |  return ¬PROGRESS(φ_1, S_n, Δ)
6  else if φ = φ_1 U_I φ_2 then
7  |  if I < 0 then
8  |  |  return ⊥
9  |  else if 0 ∈ I then
10 |  |  return PROGRESS(φ_2, S_n, Δ) ∨ (PROGRESS(φ_1, S_n, Δ) ∧ φ_1 U_{I−Δ} φ_2)
11 |  else
12 |  |  return PROGRESS(φ_1, S_n, Δ) ∧ φ_1 U_{I−Δ} φ_2
13 |  end
14 else if φ = p then
15 |  return f_p(S_n)
16 else
17 |  return ⊤
18 end
```

Theorem 1 *(Correctness of STL progression). The* PROGRESS *procedure (Algorithm 1) is correct with respect to the semantics of STL,*

$$\mathcal{M}, n \models \phi \text{ iff } \mathcal{M}, n{+}1 \models \mathit{PROGRESS}(\phi, S_{n+1}, \Delta),$$

for STL-model $\mathcal{M}$, logical time-points $n, n{+}1$, wff ϕ, tuple of signal values S_{n+1} at time point t_{n+1} and $\Delta = t_{n+1} - t_n$.

Proof. [20] prove the correctness of PROGRESS for MTL and consequently for MITL. MITL is a fragment of MTL for which all intervals $I = [a\ b]$ have to fulfill the criteria that $a < b$, i.e. no point-intervals may exist. STL extends MITL with the semantics for $\mathcal{M}, n \models p$ which is affected only when $\phi = p$ in an STL-statement. This affects line 14-17 only, which are time-independent, while the rest of PROGRESS is the same as for MTL/MITL.

If ϕ is a wff then $f_p \in F_{\mathbb{B}}$ according to Definition 6. According to the definition of the model $\mathcal{M}$ it follows that $f_p : \mathbb{R}^{|S|} \to \{\top, \bot\}$. From Definition 3 and the definition of STL it is the case that the state of a signal $S_n \in \mathbb{R}^{|S|}$ for all $n \in \mathbb{N}_{\geq 0}$ such that $n < |\mathbb{T}|$. Since $n \in \mathbb{N}_{\geq 0}$ such that $n < |\mathbb{T}|$, by Definition 7 it is the case that $S_n \in \mathbb{R}^{|S|}$ and therefore $f_p(S_n) \in \{\top, \bot\}$. Consequently, $f_p(S_n) = \top$ on line 15 iff $\mathcal{M}, n \models \mathtt{PROGRESS}(p, S_n, \Delta)$.

Finally, the only other symbol in the syntax of STL apart from the cases on line 2,4,6,14 is $\phi = \top$ for which $\mathtt{PROGRESS}(\top, S_n, \Delta) = \top$ according to line 17. □

2.3. Stochastic variables and events

Consider a scenario where the altitude of a UAV is of interest, and also if the altitude is above a certain threshold. We are however uncertain about the altitude and are consequently also uncertain if it is above the threshold or not. Due to the uncertainty the altitude is one of many possible altitude-values, and only some of these constitutes the event of the UAV being above the threshold. Further, some of the altitude-values may be more probable than other values based on our assumptions, historical data or a sensor model.[2] Representing and reasoning about uncertainty is formalized in probability theory, where the altitude of the UAV can be represented by a stochastic variable.

A stochastic variable $\boldsymbol{x}$ is a measurable function from the sample space Ω to a value domain $\mathcal{D}_{\boldsymbol{x}}$ (all possible values $\boldsymbol{x}$ can take),

$$\boldsymbol{x} : \Omega \to \mathcal{D}_{\boldsymbol{x}}.$$

If $\boldsymbol{x}$ represents the altitude of a UAV then the different values the altitude may take can for example be $\mathcal{D}_{\boldsymbol{x}} = \mathbb{R}_{\geq 0}$.

A set of events $\mathcal{F}_{\boldsymbol{x}}$ with respect to $\boldsymbol{x}$ is an σ-algebra on measurable subsets of $\mathcal{D}_{\boldsymbol{x}}$. In the UAV example, the event $E := \{x \,|\, x > \mathit{threshold},\ x \in \mathbb{D}_{\boldsymbol{x}}\}$ given a $\mathit{threshold} \in \mathbb{D}_{\boldsymbol{x}}$ is an element of $\mathcal{F}_{\boldsymbol{x}}$. Given a realization of $\boldsymbol{x}$, $x \in \mathcal{D}_{\boldsymbol{x}}$, it is the case that E holds iff $x \in E$ and the opposite $\neg E$ iff $x \notin E$.

A probability measure $\mathbb{P}[\cdot]$ is a mapping from events to probabilities, assigning each event a real value between 0 and 1,

$$\mathbb{P}[\cdot] : \mathcal{F} \to [0,\ 1].$$

$\mathbb{P}[E]$ for $E \in \mathcal{F}_{\boldsymbol{x}}$ consequently denotes the probability that E is true with respect to $\mathcal{D}_{\boldsymbol{x}}$. If $\mathbb{D}_{\boldsymbol{x}}$ is a measurable subset of $\mathbb{R}^n$ for some $n \geq 1$ then $\boldsymbol{x}$ is called a continuous stochastic variable. It is common to specify a continuous stochastic variable in terms of its probability density function $p(\boldsymbol{x})$ which is a mapping from the domain to a non-negative real value,

$$p : \mathcal{D}_{\boldsymbol{x}} \to [0,\ \infty)].$$

The probability of an event $E \in \mathcal{F}_{\boldsymbol{x}}$ is then related to the density by

$$\mathbb{P}[E] = \int_{w \in E} p(\boldsymbol{x} = w) dw \leq \int_{w \in \mathcal{D}_{\boldsymbol{x}}} p(\boldsymbol{x} = w) dw = 1.$$

Definition 8 *(Probability-theoretic event).* A probability-theoretic event E with respect to stochastic variable $\boldsymbol{x}$ is a measurable subset of the value domain $\mathbb{D}_{\boldsymbol{x}}$ of $\boldsymbol{x}$, and an element of an σ-algebra $\mathcal{F}_{\boldsymbol{x}}$ with respect to $\boldsymbol{x}$. The probability of E is $\mathbb{P}[E] = \int_{w \in E} p(\boldsymbol{x} = w) dw$ and the probability of $\neg E$ is $\mathbb{P}[\neg E] = 1 - \mathbb{P}[E]$, where $\mathbb{P}[\cdot]$ is a probability measure on $\mathcal{F}_{\boldsymbol{x}}$ and $p(\boldsymbol{x})$ is a probability density function over $\boldsymbol{x}$.

A tuple of stochastic variables[3]

$$\boldsymbol{z} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \dots \rangle$$

is also a stochastic variable

$$\boldsymbol{z} : \Omega \to \mathbb{D}_{\boldsymbol{x}_1} \times \mathbb{D}_{\boldsymbol{x}_2} \times \dots$$

with for example an associated joint probability density function

$$p(\boldsymbol{z}) = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots).$$

Another way to specify a stochastic variable is in terms of its cumulative distribution function $F_{\boldsymbol{x}}$, which is a mapping from the value domain to probabilities,

$$F_{\boldsymbol{x}} : \mathcal{D}_{\boldsymbol{x}} \to [0,\ 1]$$

and defined as the probability that $\boldsymbol{x}$ is less than or equal to the value θ,

$$\mathbb{P}[\boldsymbol{x} \leq \theta] = F_{\boldsymbol{x}}(\theta), \quad \theta \in \mathcal{D}_{\boldsymbol{x}}.$$

The cumulative probability distribution function is related to the probability density function by

[2] A model for how uncertainty is propagated through a sensor.
[3] A random vector.

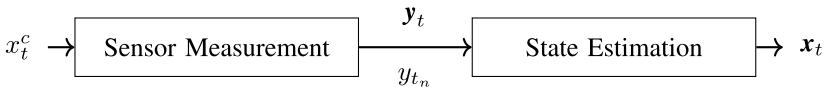

Fig. 2. The continuous-time signal $x^c(t)$ of a physical property is indirectly observed by noisy sensor measurements y_{t_n} sampled at time-points $t_n \in \mathbb{T}$. The sensor measurements y_{t_n} are realizations of the stochastic variables $\boldsymbol{y}_{t_n}$ of the stochastic signal $\boldsymbol{y}(t)$ which represents the sensor noise. The value x_t^c and our uncertainty in it is inferred for arbitrary time-points t using state estimation and it is represented as a stochastic variable $\boldsymbol{x}_t$ of the stochastic signal $\boldsymbol{x}(t)$.

$$F_{\boldsymbol{x}}(\theta) = \int_{w \leq \theta} p(\boldsymbol{x} = w) dw, \quad \theta, w \in \mathcal{D}_{\boldsymbol{x}}$$

The event $\boldsymbol{x} \leq \theta$ can be viewed as a parameterized event, which is made concrete $E \in \mathcal{F}_{\boldsymbol{x}}$ for any concrete value θ and concrete stochastic variable $\boldsymbol{x}$. The cumulative distribution function then defines the probability for any concrete event in the event class $E(\boldsymbol{x}, \theta) := \boldsymbol{x} \leq \theta$, $\theta \in \mathbb{D}_{\boldsymbol{x}}$.

Definition 9 *(Parameterized probability-theoretic event).* A parameterized event $E(\boldsymbol{x}, \theta)$ is a concise representation of a set of events parameterized by a tuple $\boldsymbol{x}$ of stochastic variables and on a tuple of parameters θ. For any concrete tuple $\boldsymbol{x}$ and θ then $E(\boldsymbol{x}, \theta)$ corresponds to a concrete probability-theoretic event $E \in \mathcal{F}_{\boldsymbol{x}}$ with respect to stochastic variable $\boldsymbol{x}$.

In this work we are primarily interested in cases $n \geq 1$ where $\mathbb{D}_{\boldsymbol{x}} \subseteq \mathbb{R}^n$ with $\mathcal{F}_{\boldsymbol{x}}$ being a subset of all Lebesgue measurable subsets of $\mathbb{D}_{\boldsymbol{x}}$ and $\mathbb{P}[\cdot]$ the n-dimensional Lebesgue measure. It is straightforward to extend the work to include discrete stochastic variables (finite $\mathbb{D}_{\boldsymbol{x}}$).

2.4. Stochastic signals and state estimation

Physical sensors are imperfect and the signals they produce contain noise. There also exist other sources of uncertainty in the environment such as the unobservable mental state of other agents or chaotic processes such as wind. Representing and managing uncertainty is consequently important, both for safety and to reach high task efficiency. Further, some properties of the environment are directly observable from sensors and some only indirectly observed by signal transformations. The uncertainty about the value of a signal resulting from a signal transformation is dependent on the uncertainty about the value of the transformed signals. A stochastic signal (Definition 10) is a signal in which the uncertainty about the signal value is made explicit.

Definition 10 *(Stochastic signal).* A stochastic signal $\boldsymbol{x}(t)$ with time domain $\mathbb{T}$ is a stochastic function $\boldsymbol{x} : \mathbb{T} \to \mathbb{S}_{\boldsymbol{x}}$ mapping from time-points to stochastic variables. For all $t \in \mathbb{T}$ the stochastic variable $\boldsymbol{x}(t) \in \mathbb{S}_{\boldsymbol{x}}$ has value domain $\mathbb{D}_{\boldsymbol{x}}$. Any finite tuple of stochastic variables $\langle \boldsymbol{x}(t_k), \ldots, \boldsymbol{x}(t_{k+n}) \rangle$, for $t_k, \ldots, t_{k+n} \in \mathbb{T}$, have a joint probability density function $p(\boldsymbol{x}(t_k), \ldots, \boldsymbol{x}(t_{k+n}))$.

State estimation machinery is commonly used to handle noise and recover a good estimate of the continuous-time signal of the physical property of interest. Fig. 2 shows a simplified overview of the signals relevant to state estimation, where noisy sensor measurements y_{t_n} are observed instead of direct measurements of the physical property x_t^c or its discretization x_{t_n}. The uncertainty in the signal value of $x^c(t)$ is represented by a stochastic signal $\boldsymbol{x}(t)$ and the value at time-point t is represented as a stochastic variable $\boldsymbol{x}_t$.

The uncertainty in the value of $x^c(t)$ at the time-point of the latest sensor measurement t_n, or any other time point t', can be estimated (predicted) given the sensor measurements $y_{t_0}, \ldots, y_{t_n}$ and we use the following short-hand notation through-out the paper:

$$p(\boldsymbol{x}_{t_n|t_n}) := p(\boldsymbol{x}_{t_n} | y_{t_0}, \ldots, y_{t_n}),$$
$$p(\boldsymbol{x}_{t'|t_n}) := p(\boldsymbol{x}_{t'} | y_{t_0}, \ldots, y_{t_n}),$$

where $t_n \in \mathbb{T}$ and $t' \in \mathbb{R}$.

The sensor observations y_{t_n} are assumed to be sampled from the sensor model (which relates the uncertainty in the observation value y_{t_n} to the uncertainty in the value of $x_{t_n}^c$):

$$y_{t_n} \sim p(\boldsymbol{y}_{t_n} | \boldsymbol{x}_{t_n} = x_{t_n}^c).$$

An example is shown in Fig. 3 where $\boldsymbol{x}_{t_n}$ is modeled as a Gaussian distributed stochastic variable $\left(\boldsymbol{x}_{t_n} \sim \mathcal{N}(\mu, \sigma^2)\right)$ and the sensor model used is

$$p(\boldsymbol{y}_{t_n} | \boldsymbol{x}_{t_n} = x_{t_n}^c) = \mathcal{N}(x_{t_n}^c, 0.5^2)$$

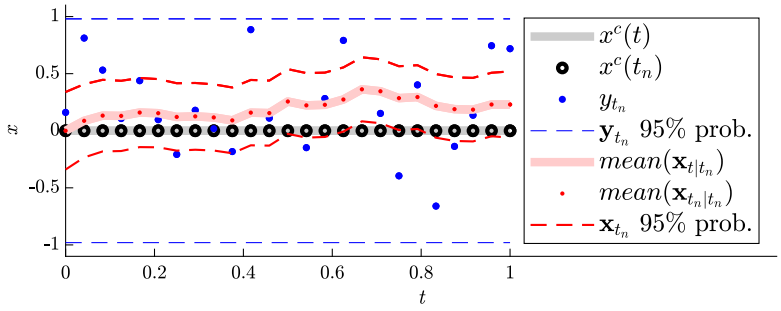

Fig. 3. Illustration of signals at various stages in the signal processing pipeline. From underlying continuous signal $x^c(t)$ (e.g. actual UAV altitude) to noisy observation y_{t_n} and estimated signal $\boldsymbol{x}_{t|t_n}$ (e.g. recovered altitude point estimate). The 95% probability intervals (approximately two standard deviations) are shown with dashed lines.
The underlying signal can be seen to be outside of the 95% probability interval within the time interval $0.6 < t < 0.8$. If we neither underestimate nor overestimate our uncertainty in $x^c(t)$ then this is not unexpected, given the Markov assumption common to Kalman filters. It is rather unlikely that $x^c(t)$ will not be outside the 95% probability interval: $P(x^c(t)$ inside $\text{PI}_{95\%}$ for 25 observations$) = 1 - 0.95^{25} = 0.72$. It is more likely that $x^c(t)$ is inside the 99% probability interval and so on. Another way to make it likely that we capture $x^c(t)$ within a certain probability interval is to overestimate our uncertainty in $x^c(t)$, that is to dilute the stochastic variables with additional uncertainty. This would make the estimation statistically more conservative than previously. It is a common method when the true probabilistic model is not known (or too computationally expensive) and an approximate model is used instead.

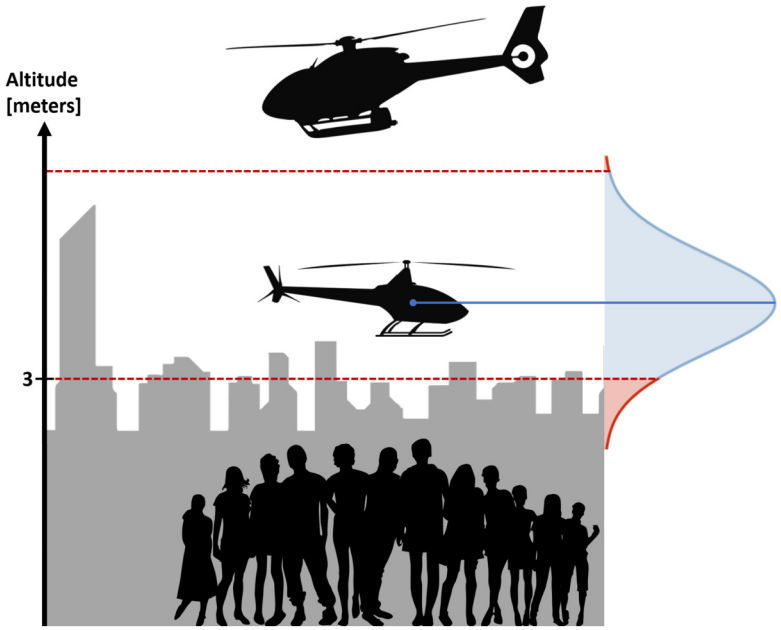

Fig. 4. A UAV has to adhere to various constraints on where to fly, how to behave and how to interact with other agents. Some constraints are for safety reasons such as staying at an altitude above peoples' heads but below other air traffic, while other constrains may include no-fly-zones and the requirement to behave predictable close to other UAVs. The figure illustrates a setting where $[\![altitude > 3]\!] = \top$ but $[\![Pr(\boldsymbol{altitude} > 3) \geq 0.99]\!] = \bot$ because $[\![Pr(\boldsymbol{altitude} > 3)]\!] = 0.90$. The operation $[\![\cdot]\!]$ is defined as the interpretation of a well-formed formula in a logic.

where the observation noise is assumed to be statistically independent across time. $\boldsymbol{x}_{t_n}$ is estimated using a Kalman filter with a constant-position motion model with process noise variance 1.

Stochastic signals can be approximated with deterministic signals and as such be used in STL. For example, by taking the mean (expectation) of the estimated property $\boldsymbol{x}_{t_n|t_n}$ as $x_{t_n} = \text{mean}(\boldsymbol{x}_{t_n|t_n}) = \int_{\mathbb{D}_x} x\, p(\boldsymbol{x}_{t_n|t_n} = x)dx$, the largest mode $x_{t_n} = \arg\max_x p(\boldsymbol{x}_{t_n|t_n} = x)$, using the observations directly $x_{t_n} = y_{t_n}$ or some other point estimator. These choices are however not explicit in the STL syntax, and the grounding of the signals in STL in the state estimation is not part of the STL semantics.

3. Probabilistic extension for STL

Consider a low-flying UAV drone, for example for delivery or surveillance, in public spaces with people walking around (Fig. 4). A reasonable safety constraint is that the UAV is not supposed to fly at an altitude lower than or equal to 3 meters,

$$\Box\,(altitude > 3),$$

or else it might hurt or cause discomfort to people. However, if the altitude is uncertain then this constraint makes little sense in terms of safety. If the uncertainty is large then it is not unlikely that the UAV is at a dangerously low altitude. It is common that the uncertainty changes over time, as a result of variation in perception quality. If using a safety margin it must account for the worst-case uncertainty, say maybe 5 meters extra, and consequently might be too large and cause the task efficiency to drop below useful levels. These issues can be remedied by specifying how likely it has to be that the UAV is fulfilling the altitude requirement,

$$\square\,(Pr(\boldsymbol{altitude} > 3) \geq 0.99),$$

where the specification is that the probability that the altitude is greater than 3 meters is at least 99%.

It is even more useful to detect a possible (and probable) altitude violation before it happens. It might be too late to react to a dangerous situation if it has already occurred, but it could be possible to prevent it if reacting before it happens. For example, by monitoring the predicted altitude 2 seconds into the future,

$$\square\,(Pr(\boldsymbol{altitude}_{2|0} > 3) \geq 0.99),$$

where 0 indicates that the prediction is made from the current time-point. The formula now relies on the underlying state estimation machinery to be able to make conservative predictions (in a probabilistic sense) such that the probability of violating the constraint is never underestimated. It also relies on the control and decision making processes to be planning ahead since the condition is about the altitude of itself and not of another agent. Consequently, it can be important to also monitor if the predictions fluctuate over time and also if the predictions are in fact conservative enough. Monitoring the predictability is useful for checking that the prediction machinery works according to specification. It is even more important in multi-agent settings where the safety and task-efficiency of agents will depend on how precise and correctly they can predict the behaviors and intentions of each other.

We introduce ProbSTL, a probabilistic extension to STL where the signals are considered stochastic and predictions about unobserved signals' values are accessible (produced from for example underlying state estimation processes). We begin by defining the ProbSTL stream consisting of the signals over which the logic is defined. Then we continue by defining the sub-language $\mathcal{L}_{\text{prob}}$ which allows for expressing the exemplified statements of the previous paragraphs. Finally we present ProbSTL as a formal extension of STL. Computational and precision properties are analyzed both with the finite variability assumption on the underlying continuous signal and when relaxing it.

Before defining the ProbSTL Stream we first define a prediction stream. A prediction stream $\boldsymbol{X}$ is a tuple of length $|\mathbb{T}_{\boldsymbol{X}}|$, where each element of the stream is a tuple of infinite length representing the prediction from time point $t_n \in \mathbb{T}_{\boldsymbol{X}}$ about all possible real time-points $t' \in \mathbb{R}$. The inner tuple is ordered over the value of $t' \in \mathbb{R}$. The outer tuple (the prediction stream) is ordered over the value of $t_n \in \mathbb{T}_{\boldsymbol{X}}$.

Definition 11 *(Prediction stream).* A prediction stream $\boldsymbol{X}(t_n) = \langle \langle \boldsymbol{x}_{t'|t_n} \mid t' \in \mathbb{R} \rangle \mid t_n \in \mathbb{T}_{\boldsymbol{X}} \rangle$ with time domain $\mathbb{T}_{\boldsymbol{X}}$ is a discrete-time signal representing all temporal predictions about the value of $x^c(t)$ given the observations available at every time point t_n. More precisely, every stream state $\langle \boldsymbol{x}_{t'|t_n} \mid t' \in \mathbb{R} \rangle$ is a tuple of stochastic variables consisting of predictions $\boldsymbol{x}_{t'|t_n} \sim p(\boldsymbol{x}_{t'|t_n})$ about all possible time points t' from the specific stream time t_n.

Definition 12 *(ProbSTL stream).* A ProbSTL stream $S = \langle x^{(1)}(t_n), \ldots, x^{(J)}(t_n), \boldsymbol{X}^{(1)}(t_n), \ldots, \boldsymbol{X}^{(K)}(t_n) \rangle$ is a tuple consisting of deterministic discrete-time signals $x^{(j)}$ and prediction streams $\boldsymbol{X}^{(k)}$ where all signals share the same timed domain $\mathbb{T}$. If $x^{(j)}(t_n)$ and $\boldsymbol{X}^{(k)}(t_n)$ share the same name x, then $x^{(j)}(t_n)$ are sensor measurements (observations) of $x^c(t)$ and $\boldsymbol{X}^{(k)}(t_n)$ are estimates (predictions) of $x^c(t)$.

Let $S_n = \langle x_{t_n} \in x^{(j)} \mid j = 1, \ldots, J \rangle \cdot \langle \boldsymbol{x}_{t'|t_n} \in \boldsymbol{X}^{(k)} \mid k = 1, \ldots, K,\ t' \in \mathbb{R} \rangle$ denote a tuple of signal states at time point $t_n \in \mathbb{T}$. The operator $\cdot$ denote tuple concatenation.[4]

The predictions $\boldsymbol{x}_{t'|t_n}$ are usually represented by a generative model in the state estimation machinery, and only calculated explicitly for a few time points t' of interest. Since formulas, in which predictions are referred to, are finite in the number of symbols they are made up of, there will ever only be a finite number of predictions necessary to calculate for any t_n.

Definition 13 *(ProbSTL model).* Let A be a set and A^n the n-ary Cartesian power, where $A^0 = \{\varnothing\}$ and consequently $A^0 \times A^n = A^n$. We denote $A^* = A^n$ for $n \geq 0$ and $A^+ = A^n$ for $n \geq 1$. Let $\mathbb{S}$ be a set of stochastic variables.

ProbSTL is defined over a model $\mathcal{M} = \langle S, F_{\mathbb{B}}, F_{\mathbb{R}}, F_{\mathbb{S}}, \mathbb{E} \rangle$ where S is a ProbSTL stream, $F_{\mathbb{B}}$ is a set of functions from real-values to Boolean-values $\left(f_{\mathbb{B}} \in F_{\mathbb{B}} \subset \{f \mid f : \mathbb{R} \to \{\top, \bot\}\}\right)$, $F_{\mathbb{R}}$ is a set of functions from stochastic variables to real-values $\left(f_{\mathbb{R}} \in F_{\mathbb{R}} \subset \{f \mid f : \mathbb{S}^* \times \mathbb{R}^* \to \mathbb{R}\}\right)$, $F_{\mathbb{S}}$ is a set of functions mapping from stochastic variables and real-values to stochastic variables $\left(f_{\mathbb{S}} \in F_{\mathbb{S}} \subset \{f \mid f : \mathbb{S}^* \times \mathbb{R}^* \to \mathbb{S}\}\right)$ and $\mathbb{E}$ is a set of probability-theoretic events (Definition 8).

[4] $\langle a_1, \ldots, a_N \rangle \cdot \langle b_1, \ldots, b_M \rangle = \langle a_1, \ldots, a_N, b_1 \ldots, b_M \rangle$.

3.1. Probability statements

Definition 14 *(Probabilistic language $\mathcal{L}_{prob}$).* A statement ℓ in the expandable probabilistic language $\mathcal{L}_{prob}$ over the vocabulary $\langle \mathcal{E}, \mathsf{F}_{\mathbb{R}}, \mathsf{F}_{\mathbb{S}}, \mathsf{X}_d, \mathsf{X}_p, \mathsf{C} \rangle$ is a well formed statement (wfs) if it adheres to the following syntax, where time points are relative,

$$\ell := Pr(\mathsf{E}(\tau_p, \dots, \tau_d, \dots)) \mid \mathsf{f}_{\mathbb{R}}(\tau, \dots)$$
$$\tau := \tau_d \mid \tau_p$$
$$\tau_d := \ell \mid \mathsf{x}_t \mid \mathsf{const}$$
$$\tau_p := \mathsf{x}_{t'|t} \mid \mathsf{f}_{\mathbb{S}}(\tau, \dots)$$

where E is an element in the set of event symbols $\mathcal{E}$, $\mathsf{f}_{\mathbb{R}}$ is an element in the set of real-valued function-symbols $\mathsf{F}_{\mathbb{R}}$, const is an element in the set of numerical constants C, $\mathsf{f}_{\mathbb{S}}$ is an element in the set of stochastic-function-symbols $\mathsf{F}_{\mathbb{S}}$, x_t is an element in the set of deterministic-signal symbols X_d and $\mathsf{x}_{t'|t}$ is an element in the set of stochastic-signal symbols X_p with time point symbols $t \in \mathbb{R}_{\leq 0}$ and $t' \in \mathbb{R}$.

Definition 15 *(Real interpretation of $\mathcal{L}_{prob}$).* We define a real-valued evaluation function $eval(S_{\leq n}, t_n, \ell) := eval(S_{\leq n}, t_n, \ell')$ given a set of stream prefixes $S_{\leq n}$, a time-point $t_n \in \mathbb{T}$ and wfs $\ell \in \mathcal{L}_{prob}$ recursively over valid symbols $\ell' \in \ell$.

$$eval(S_{\leq n}, t_n, \mathsf{const}) = const$$
$$eval(S_{\leq n}, t_n, \mathsf{f}_{\mathbb{R}}(\tau, \dots)) = f_{\mathbb{R}}(eval(S_{\leq n}, t_n, \tau), \dots)$$
$$eval(S_{\leq n}, t_n, \mathsf{f}_{\mathbb{S}}(\tau, \dots)) = f_{\mathbb{S}}(eval(S_{\leq n}, t_n, \tau), \dots)$$
$$eval(S_{\leq n}, t_n, \mathsf{x}_{t'|t}) = \boldsymbol{x}_{t_n+t'|t_{n-k}}, \qquad k = f_{\mathbb{T}}(n, t)$$
$$eval(S_{\leq n}, t_n, \mathsf{x}_t) = x_{t_{n-k}}, \qquad k = f_{\mathbb{T}}(n, t)$$
$$eval(S_{\leq n}, t_n, Pr(\mathsf{E}(\tau_p, \dots, \tau_d, \dots)))$$
$$= \int_{w \in E(X,\theta)} p(X = w)\, dw,$$
$$X = \langle eval(S_{\leq n}, t_n, \tau_p^i) \rangle_i,\ \theta = \langle eval(S_{\leq n}, t_n, \tau_d^j) \rangle_j$$

where $E(X, \theta)$ is a parameterized event made concrete with the given X and θ (Definition 9), k (for $x_{t_{n-k}}$ and $\boldsymbol{x}_{t_n+t'|t_{n-k}}$) is calculated by

$$f_{\mathbb{T}}(n, t) := \underset{k \in \langle 0,1,\dots,n \rangle}{\arg\min} \left| t_{n-k} - (t_n + t) \right|, \tag{1}$$

(where $t \leq 0$ from Definition 14) and, for $m = n - k$,

$$f_{\mathbb{R}} \in F_{\mathbb{R}},\ f_{\mathbb{S}} \in F_{\mathbb{S}},\ E \in \mathbb{E},\ x_{t_m} \in S_m,\ \boldsymbol{x}_{t_n+t'|t_m} \in S_m,\ const \in \mathbb{R}^+$$

If there are multiple k which minimizes (1) then the smallest such k is chosen.

The terms x_t and $\mathsf{x}_{t'|t}$ refer to earlier states in the stream prefix $S_{\leq n}$ than the current state S_n if $t < 0$. How much of the prefix is necessary to ground the terms is determined by the past-reach over $\mathcal{L}_{\text{prob}}$ (Theorem 2).

Theorem 2 *(Past-reach for $\mathcal{L}_{prob}$). Let $\ell \in \mathcal{L}_{prob}$ and $S_{\leq n}$ be the current stream prefixes (signal prefixes). The past-reach $pr(\tau)$ of a symbol τ in ℓ is the minimal required prefix-length of $S_{\leq n}$ for which the interpretation of τ exists. Given a symbol τ of the form $\tau = \mathsf{x}_t$ or $\tau = \mathsf{x}_{t'|t}$ then $pr(\tau) = k$, where k is calculated from t according to (1). The past-reach $pr(\ell)$ for $\ell \in \mathcal{L}_{prob}$ is the largest k of all τ in ℓ. The earliest state required in the prefix $S_{\leq n}$ for $\ell \in \mathcal{L}_{prob}$ is $S_{(n-pr(x))}$.*

Proof. There is no temporal operators in $\mathcal{L}_{\text{prob}}$ and consequently no nested temporal operators. By Definition 15 at time-point t_n and with signal prefixes $S_{\leq n}$ the interpretation of x_t is $x_{t_{n-k}}$, using the definition of k from (1). Further, $x_{t_{n-k}} \in S_{(n-k)}$ since $0 \leq n - k \leq n$ according to (1). The proof is analog for $x_{t'|t}$. □

3.2. $\mathcal{L}_{prob}$ core

The semantics of events $E \in \mathbb{E}$ will vary depending on the choice of $\mathbb{E}$ including the kinds of stochastic variables $\boldsymbol{x} \in \mathbb{S}$ (in terms of their probability density functions $p(\boldsymbol{x})$) under consideration. $\mathcal{L}_{\text{prob}}$ is consequently separated in a core part and optional plugins to provide flexible extendability.

The core probabilistic language has the events,

$$\mathbb{E} = \{\text{comparator}\},$$

parameterized by a stochastic variable $\boldsymbol{x} \in \mathbb{S}$ and a constant $c \in \mathbb{D}_{\boldsymbol{x}}$, with probabilistic interpretation

$$\mathbb{P}[\text{comparator}(\boldsymbol{x}, \div, c)] = \int_{x \div c} p(\boldsymbol{x} = x)dx,$$

where $\div \in \{<, \leq, =, \geq, >\}$ element-wise. The core language is sufficient for expressing the previously shown example statements of the form $Pr(\mathrm{x}_{t'|t} > \mathrm{c})$.

3.3. $\mathcal{L}_{prob}$ spatial plugin

To provide an example of a plugin we introduce a small spatial plugin. Many other are possible to define for whatever need an application might have, such as collisions between spherical objects with non-isometric Gaussian-distributed position uncertainty[23].

In [24] they do maritime monitoring by complex event processing based on first constructing primitive events. These events are based on vessel positions as points being for example inside a crisp region. The positions of the vessels are based on GPS-measurements. Sometimes GPS-measurements are imprecise and the truth of such a statement is then unclear. Following their example we introduce probabilistic spatial events where this uncertainty is considered explicitly and for which the truth of statements about them is clear.

The real-valued functions and stochastic functions are

$$\{\text{mean}\} \subseteq F_{\mathbb{R}}, \qquad \{\text{distance}, \text{centered}\} \subseteq F_{\mathbb{S}},$$

and are defined as

$$\text{mean}(\boldsymbol{x}) = \int x\, p(\boldsymbol{x} = x)dx$$

$$\text{distance}(\boldsymbol{x}, \boldsymbol{y}) \sim p(\boldsymbol{x} - \boldsymbol{y})$$

$$\text{distance}(\boldsymbol{x}, y) \sim p(\boldsymbol{x} - y)$$

$$\text{centered}(\boldsymbol{x}) \sim p(\boldsymbol{x} - \text{mean}(\boldsymbol{x}))$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{S}$ are stochastic variables (e.g. $\boldsymbol{x}_{t'|t}$) and $y \in \mathbb{D}_{\boldsymbol{x}}$.

The probabilistic events of the plugin are

$$\{\text{greaterThan}, \text{lessThan}, \text{insideInterval}, \text{insideRectangle}\} \subset \mathbb{E}$$

and are defined for $\mathbb{D}_{\boldsymbol{x}} = \mathbb{R}$, $c, r_1, r_2 \in \mathbb{R}$, $r_1 < r_2$, as

$$\mathbb{P}[\text{greaterThan}(\boldsymbol{x}, r_1)] = \int_{x > r_1} p(\boldsymbol{x} = x)dx$$

$$\mathbb{P}[\text{lessThan}(\boldsymbol{x}, r_2)] = \int_{x < r_2} p(\boldsymbol{x} = x)dx$$

$$\mathbb{P}[\text{insideInterval}(\boldsymbol{x}, r_1, r_2)] = \int_{r_1 < x < r_2} p(\boldsymbol{x} = x)dx$$

and defined for $\mathbb{D}_{\boldsymbol{x}} = \mathbb{R}^2$, $c, r \in \mathbb{R}^2$ as

$$\mathbb{P}[\text{insideRectangle}(\boldsymbol{x}, c, r)] = \int_{|x - c| \lessdot r} p(\boldsymbol{x} = x)dx$$

where $\lessdot$ is the element-wise less-than operator.

3.4. Integration with STL

In STL the signals and the Boolean-valued functions applied to these signals are hidden behind proposition symbols in the syntax. The proposition symbols are associated with specific Boolean-valued functions, providing their interpretation in the semantics. In ProbSTL we want the individual signals to be addressable directly in the logical language. Similarly for functions over signals, and also which signals the functions are applied to. The syntax is consequently constructed with symbols of Boolean-valued functions present in the syntax, which have a direct interpretation as Boolean-valued functions in the semantics. These functions are applied to statements in $\mathcal{L}_{prob}$ and therefore map from a real value to a Boolean, as opposed to from a signal tuple to a Boolean value as is the case in STL. Functions over multiple signals are instead expressible in $\mathcal{L}_{prob}$ allowing ProbSTL to express everything that STL can express.

Definition 16 *(ProbSTL syntax).* A ProbSTL statement is a well-formed formula iff it adheres to the following syntax:

$$\phi := \top \mid \mathsf{f}_{\mathbb{B}}(\ell) \mid \neg\phi \mid \phi \vee \psi \mid \phi \,\mathcal{U}_I\, \psi$$

where $\mathsf{f}_{\mathbb{B}}$ is a element in the set of Boolean-valued function-symbols $\mathsf{F}_{\mathbb{B}}$ and ℓ is a $\mathcal{L}_{prob}$ statement. Additionally, we allow for the usual logical connectives $\wedge, \rightarrow, \leftrightarrow$ as well as the temporal operators $\Diamond_I \phi = \top \,\mathcal{U}_I \phi$ and $\Box_I \phi = \neg\Diamond_I \neg\phi$ as syntactic sugar.

Definition 17 *(ProbSTL semantics).* The semantics of ProbSTL extend the semantics of STL with

$$\mathcal{M}, n \models \mathsf{f}_{\mathbb{B}}(\ell) \quad \text{iff} \quad f_{\mathbb{B}}(eval(S_{\leq n}, t_n, \ell)),$$

where $f_{\mathbb{B}} \in F_{\mathbb{B}}$ from Definition 13.

4. Implementation

In the implementation the assumption is made that all stochastic variables $\boldsymbol{x} \in \mathbb{S}$ are Gaussian distributed and consequently have the probability density function

$$\begin{aligned} p(\boldsymbol{x} = x) &= \mathcal{N}(x;\, \mu_{\boldsymbol{x}},\, \Sigma_{\boldsymbol{x}}) \\ &= |2\pi\, \Sigma_{\boldsymbol{x}}|^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}(x - \mu_{\boldsymbol{x}})^T\, \Sigma_{\boldsymbol{x}}^{-1}(x - \mu_{\boldsymbol{x}})\Big), \end{aligned}$$

where $\mu_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{x}}$ are the mean and covariance function respectively of the stochastic variable $\boldsymbol{x}$. We further assume for simplicity that the Gaussian distributions of the stochastic variables are isometric which means that the covariance matrix $\Sigma_{\boldsymbol{x}}$ is diagonal. The real-valued functions and stochastic functions of the spatial plugin are realized as

$$\begin{aligned} \text{mean}(\boldsymbol{x} \sim \mathcal{N}(\mu_{\boldsymbol{x}}, \Sigma_{\boldsymbol{x}})) &= \mu_{\boldsymbol{x}} \\ \text{distance}(\boldsymbol{x} \sim \mathcal{N}(\mu_{\boldsymbol{x}}, \Sigma_{\boldsymbol{x}}),\, \boldsymbol{y} \sim \mathcal{N}(\mu_{\boldsymbol{y}}, \Sigma_{\boldsymbol{y}})) &\sim \mathcal{N}(\mu_{\boldsymbol{x}} - \mu_{\boldsymbol{y}}, \Sigma_{\boldsymbol{x}} + \Sigma_{\boldsymbol{y}}) \\ \text{distance}(\boldsymbol{x} \sim \mathcal{N}(\mu_{\boldsymbol{x}}, \Sigma_{\boldsymbol{x}}),\, y) &\sim \mathcal{N}(\mu_{\boldsymbol{x}} - y, \Sigma_{\boldsymbol{x}}) \\ \text{centered}(\boldsymbol{x} \sim \mathcal{N}(\mu_{\boldsymbol{x}}, \Sigma_{\boldsymbol{x}})) &\sim \mathcal{N}(0, \Sigma_{\boldsymbol{x}}) \end{aligned}$$

where $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{S}$ are stochastic variables (e.g. $x_{t'|t}$). The events in $\mathbb{E}$ are realized as

$$\begin{aligned} &\mathbb{P}[\text{greaterThan}(\boldsymbol{x}, r_1)] \\ &\quad = \int_{x > r_1} p(\boldsymbol{x} = x)dx = \int_{r_1}^{\infty} p(\boldsymbol{x} = x)dx = 1 - \Phi(r_1) \\ &\mathbb{P}[\text{lessThan}(\boldsymbol{x}, r_2)] \\ &\quad = \int_{x < r_2} p(\boldsymbol{x} = x)dx = \int_{-\infty}^{r_2} p(\boldsymbol{x} = x)dx = \Phi(r_2) \\ &\mathbb{P}[\text{insideInterval}(\boldsymbol{x}, r_1, r_2)] \\ &\quad = \int_{r_1 < x < r_2} p(\boldsymbol{x} = x)dx = \int_{r_1}^{r_2} p(\boldsymbol{x} = x)dx = \Phi(r_2) - \Phi(r_1) \end{aligned}$$

$$\mathbb{P}[\text{insideRectangle}(\boldsymbol{x}, c, r)]$$
$$= \int_{|x-c|\ll r} p(\boldsymbol{x}=x)dx = \int_{|x_1-c_1|<r_1} \int_{|x_2-c_2|<r_2} p(\boldsymbol{x}=x)dx_1dx_2$$
$$= (\Phi(c_1+r_1) - \Phi(c_1-r_1))(\Phi(c_2+r_2) - \Phi(c_2-r_2))$$

where $\Phi(x) = \frac{1}{2}\left(1 + \text{erf}(\frac{x-\mu}{\sigma\sqrt{2}})\right)$ is the Gaussian cumulative probability distribution function $\left(\int_{-\infty}^{x} \mathcal{N}(x|\mu, \sigma^2)dx\right)$ and erf(x) is the Gauss error function.

We prove the correctness of Progression for ProbSTL (Algorithm 3) in Theorem 3, and it makes use of the Eval algorithm (Algorithm 2) which realizes Definition 15.

Algorithm 2: Eval for $\mathcal{L}_{\text{prob}}$.

```
1  function EVAL(S_{≤n}, t_n, ℓ):
2  if ℓ = const then
3  |  return const
4  else if ℓ = f_ℝ(ℓ_1, ℓ_2, ...) then
5  |  return f_ℝ(EVAL(S_{≤n}, t_n, ℓ_1), EVAL(S, t_n, ℓ_2), ...)
6  else if ℓ = f_𝕊(ℓ_1, ℓ_2, ...) then
7  |  return f_𝕊(EVAL(S_{≤n}, t_n, ℓ_1), EVAL(S_{≤n}, t_n, ℓ_2), ...)
8  else if ℓ = x_t then
9  |  return x_{t_{n-k}},   k ← f_𝕋(t, n)   (Equation 1)
10 else if ℓ = x_{t'|t} then
11 |  return x_{t_n+t'|t_{n-k}},   k ← f_𝕋(t, n)   (Equation 1)
12 else if ℓ = Pr(E(ℓ_{p_1}, ..., ℓ_{p_N}, ℓ_{d_1}, ..., ℓ_{d_M})) then
13 |  X = ⟨eval(S_{≤n}, t_n, ℓ_{p_j})⟩_{j=1}^{N}
14 |  θ = ⟨eval(S_{≤n}, t_n, ℓ_{d_j})⟩_{j=1}^{M}
15 |  return ∫_{w∈E(X,θ)} p(X = w) dw
16 end
```

Algorithm 3: Progression for ProbSTL.

```
1  function PROGRESS(φ, S_{≤n}, Δ, t_n):
2  if φ = φ_1 ∨ φ_2 then
3  |  return PROGRESS(φ_1, S_{≤n}, Δ, t_n) ∨ PROGRESS(φ_2, S_{≤n}, Δ, t_n)
4  else if φ = ¬φ_1 then
5  |  return ¬PROGRESS(φ_1, S_{≤n}, Δ, t_n)
6  else if φ = φ_1 U_I φ_2 then
7  |  if I < 0 then
8  |  |  return ⊥
9  |  else if 0 ∈ I then
10 |  |  return PROGRESS(φ_2, S_{≤n}, Δ, t_n) ∨ (PROGRESS(φ_1, S_{≤n}, Δ, t_n) ∧ φ_1 U_{I-Δ} φ_2)
11 |  else
12 |  |  return PROGRESS(φ_1, S_{≤n}, Δ, t_n) ∧ φ_1 U_{I-Δ} φ_2
13 |  end
14 else if φ = p_ℓ then
15 |  return f_p(EVAL(S_{≤n}, t_n, ℓ))
16 else
17 |  return ⊤
18 end
```

Theorem 3 *(Correctness of ProbSTL progression). The* PROGRESS *procedure (Algorithm 3) is correct with regards to the semantics of ProbSTL,*

$$\mathcal{M}, n \models \phi \text{ iff } \mathcal{M}, n{+}1 \models \mathit{PROGRESS}(\phi, S_{\leq n+1}, \Delta, t_{n+1}),$$

for ProbSTL-model $\mathcal{M}$, time-points $n, n{+}1$, real-valued time-point t_n, wff ϕ, tuple of signal prefixes $S_{\leq n}$ and $\Delta = t_{n+1} - t_n$.

Proof. ProbSTL extends STL with a sub-language $\mathcal{L}_{\text{prob}}$ which always evaluates to real-valued terms in place of real-valued signals in STL as long as the statement is a wfs. The sub-language can consequently clearly be plugged into STL. The ProbSTL extension of STL only affects line 15 in Algorithm 3 where S_n is replaced by $\texttt{Eval}(S_{\leq n}, t_n, \ell)$ with respect to line 15 in Algorithm 1. It is sufficient to show that $\texttt{Eval}(S_{\leq n}, t_n, \ell) \in \mathbb{R}$ defined by Algorithm 2 for a wfs $\ell \in \mathcal{L}_{\text{prob}}$ and the rest follows from Theorem 1. For ℓ to be a wfs (Definition 14) it can only appear in EVAL as an argument on lines 4 and

12. The interpretation of the respective (wfs) ℓ is a real valued number (Definition 15). Consequently, the associated return statement of each of these three cases all return a value in $\mathbb{R}$. □

The computational complexity of incremental evaluation of wffs in ProbSTL depends on the plugins used. The integrals of all the listed events, e.g. *InsideInterval*, has constant time complexity ($\mathcal{O}(1)$), but other integrals do not have cheap closed form solutions and the computational complexity of numerical integration depends on the integrand [25]. Integrals are also commonly approximated using techniques such as variation methods, Monte Carlo based sampling methods [26] or efficient hybrid methods [8]. The use of plugins allows for a *pay-as-you-go* principle, where the computational complexity and other desirable theoretical properties depends on the choice of plugin. The computational complexity of progression over ProbSTL-formulas is the same as for STL when including the additional signal transformations which would be hidden in STL but are made explicit in ProbSTL.

5. Properties of ProbSTL for physical signals

Physics and computational hardware put limits on how fast the sampling in a sensor (the sample rate) can be performed. If the underlying continuous-time signal $x^c(t)$, which the sensor observes, varies faster than the sample rate then the produced discrete-time signal $x(t_n)$ will miss out on information present in the underlying signal. It is also common that a value change in $x^c(t)$ will show up at a later time-point in $x(t)$, since the sampling is not being triggered by a value change in most sensor types. The produced signal will in both cases consequently not fully represent the underlying continuous-time signal. Both cases violate the finite variability assumption.

In [16] they use a simulator in which a discrete-time signal can be constructed and which by design represents every signal-value change produced by the simulator. This does ground the semantic validity in the simulator abstraction but not necessarily in the physical process which is simulated. Finite variability is a very strong and, in general, unrealistic assumption for many real physical signals. Further, a sampling procedure which captures every signal-value change, even of a signal with finite variability, is also in general unrealistic for most physical systems. A finite interval coverage $\mathcal{I}$ of $x^c(t)$ is for example typically neither known nor available for use by the sampling procedure. It is therefore necessary to consider realistic relaxations on the finite variability assumption and on the use of a finite interval coverage sampling process[5] (*FIC sample process* for short). It is important to analyze how such relaxations affect the computational complexity, validity and latency of drawn conclusions in STL and ProbSTL.

Given continuous-time signal $x^c(t)$ and a FIC sampling process (which implies that $x^c(t)$ is of finite variability), it is the case that a formula ϕ which holds for the produced discrete-time signal $x(t_n)$ also holds for the continuous-time signal $x^c(t)$:

$$x \models \phi \quad \Longrightarrow \quad x^c \models \phi$$

However, under what circumstances is it the case given realistic relaxations to finite variability and to a FIC sampling process? The relaxations we consider here are

- The sample rate is bounded between a minimal and a maximal rate.
- The sample rate is independent of the value changes in the underlying signal.
- The continuous-time signal $x^c(t)$ does not have to be of (local) finite variability.

We focus on these three relaxations because they correspond to the most common realistic settings for cyber-physical systems in practice, such as in robotics or in the signal processing field. In signal theory this setting is for example accompanied with the common additional assumption that the (minimal) sample rate is at least twice the maximum frequency of the underlying noise-free continuous signal (Nyquist theorem [27]). This is among other things necessary in general in order to do a full reconstruction of the signal using the sampled signal.

To analyze the consequences of these relaxations we therefore introduce a time difference bound $\Delta_{limits} = \{\Delta_{min}, \Delta_{max}\}$ as a minimal and maximal time difference between consequent samples as a result of a sampling procedure. This directly corresponds to a maximum and minimum sample rate. To be precise, Δ_{limits} puts constraints on the discrete-time signal $x(t_n) = x^c(t_n)$ as it is produced by a sampling procedure of the (noisy) continuous-time signal $x^c(t)$ describing observations of a physical state. The constraint is defined as $\forall n\,(\Delta_{min} \leq \Delta_n \leq \Delta_{max})$ where $\Delta_n = t_{n+1} - t_n$ with $t_n, t_{n+1} \in \mathbb{T}$ and $\Delta_{min}, \Delta_{max} \in \mathbb{R}_{\geq 0}$.

Firstly, we investigate how a FIC sampling process over a finite variability signal relates to the Δ_{limits} bound. Since the discrete-time signal by definition of FIC sampling process with finite variability has to contain all signal value variations in the continuous-time signal it is the case that the bound cannot be tighter than the rate of change allowed. Finite variability puts constraints on the number of value changes, but not how fast they are allowed to change as long as consecutive

[5] A sampling procedure such that a discrete-time signal is produced, which fully captures the observed continuous-time signal with (local) finite variability, by sampling at every signal-value change.

time points are distinguishable (i.e. not the same time-point). Nor is there a requirement that the signal has to change. In the general case of local finite variability the first value change could appear in the limit. Consequently, the bound is fully determined by the smallest and largest time-difference between value changes in the continuous-time signal. If these properties are not known (the common case) then the bounds exists in the limit ($\Delta_{min} \to 0$ and $\Delta_{max} \to \infty$) and $0 < \Delta_n < \infty$ is consequently all we can say. This is a direct consequence of the finite variability assumption and why it is a physically infeasible assumption. Theorem 4 summarizes these conclusions.

Theorem 4 *(Δ_{limits} for a FIC sampling process). The bounds $\Delta_{limits} = \{\Delta_{min}, \Delta_{max}\}$ for a FIC sampling process exists in the limit*

$$\Delta_{min} \to 0 \text{ and } \Delta_{max} \to \infty$$

Proof. Let $x^c(t)$ be a continuous-time signal which fulfills the (local) finite variability assumption, and let $x(t_n)$ be a discrete-time signal produced by a FIC sampling procedure from $x^c(t)$ which means that $x(t_n)$ fully represent $x^c(t)$. This assumes a sampling procedure in which a new sample $x(t_n)$ is produced for every time-point t_n where $x^c(t)$ has changed to a new value. Consequently, Δ_n can potentially be as short as the fastest value change, $0 < \Delta_n$, and potentially as long as the entire signal, $\Delta_n < \infty$. Since Δ_{limits} consequently is directly dependent on the characteristics of the continuous-time signal, and we cannot directly observe this signal, the general bounds exists in the limit $\Delta_{min} \to 0$ and $\Delta_{max} \to \infty$. □

A frequently used assumption in signal processing is that uniform sampling (fixed sample rate) is used as a sampling procedure. This can leverage techniques such as the Fast Fourier Transforms in order to significantly reduce the computational complexity of certain signal processing tasks. In the uniform sampling case $\Delta_{min} = \Delta_{max}$.

Proposition 1 *(Δ_{limits} for uniform sampling). For uniform sampling (fixed sample rate) it is the case that $\Delta_{min} = \Delta_{max}$.*

The cases in Theorem 4 and Proposition 1 span a range of the bound Δ_{limits}. From event rate based sampling (Theorem 4) to fixed rate sampling (Proposition 1). We will now investigate aspects of validity, latency and computational complexity within this span.

Since the discrete-time signal $x(t_n)$ produced from a sampling process is not guarantied to fully represent the continuous-time signal $x^c(t)$ (as opposed to with FIC sampling) there is a possibility that signal value changes will be missed. The consequence is that the violation of a formula given $x^c(t)$ will not result in the violation of the formula given $x(t_n)$. Only signal value changes that happens slow enough will be reliably captured by $x(t_n)$ and consequently any conclusions drawn from such value-changes as well.

Theorem 5 *(Reliable formula violation detection). Only formula violations due to a signal violating the formula for a time duration (violation interval) $\geq \Delta_{max}$ will be reliably detected in STL and ProbSTL for deterministic signals.*

Proof. The time between consecutive stream states is at most Δ_{max} from the definition of Δ_{limits}. If the underlying signal values $x^c(t)$, for $t \in [a\ b]$, falsifies a formula but the interval duration $b - a$ is shorter than Δ_{max} and the interval is between two sample time-points $t_n < a < b < t_{n+1}$, for $t_n, t_{n+1} \in \mathbb{T}_x$, then none of the underlying signal values within the interval will be present in the signal values of $x(t_n)$. If the underlying signal values do not falsify the formula immediately before a and immediately after b then this violation interval will be missing from the sampled signal.

If a violation interval is longer than Δ_{max} then at least one falsifying underlying signal value will be sampled and present in $x(t_n)$ and consequently provide a detectable violation. □

Example 1 *(Reliable formula violation detection).* Consider the illustrative example in Fig. 5. Let $\mathbb{T}_{altitude} = \langle t_0 = 0, t_1 = 1, t_2 = 2, t_3 = 3, t_4 = 4 \rangle$, shown as the 5 black bars in Fig. 5, and $\phi := \mathsf{altitude}_0 > 3$. The interpretation of ϕ at t_n is $[\![\mathsf{altitude}_0 > 3]\!] = altitude_{t_n} > 3$. For signal $altitude^c(t)$ it is the case that

$$altitude^c(t) > 3 \quad = \quad \begin{cases} \top, & I_1 := 0.0 \leq t < 0.3 \\ \bot, & I_2 := 0.3 \leq t < 0.7 \\ \top, & I_3 := 0.7 \leq t < 2.3 \\ \bot, & I_4 := 2.3 \leq t < 3.8 \\ \top, & I_5 := 3.8 \leq t \end{cases}.$$

The interpretation of ϕ is at $n = 0, 1, 2, 3, 4$

$$t_0: \quad [\![\mathsf{altitude}_0 > 3]\!] = altitude_{t_0} > 3 = \top,$$
$$t_1: \quad [\![\mathsf{altitude}_0 > 3]\!] = altitude_{t_1} > 3 = \top,$$

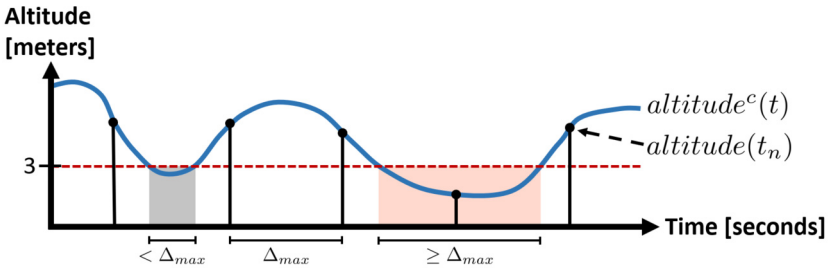

Fig. 5. The ProbSTL formula $\text{altitude}_0 > 3$ is monitored over signal $altitude(t_n)$ (**black** dots) which is a discretization of signal $altitude^c(t)$ (**blue** curve). Filled red area is a detected violation over $altitude^c(t)$ ($altitude(t_n) > 3 = \bot$ for some t_n in the violation interval). Filled gray area is a violation over $altitude^c(t)$ which is not detected (there is no t_n in the violation interval such that $altitude(t_n) > 3 = \bot$). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$t_2: \quad [\![\text{altitude}_0 > 3]\!] = altitude_{t_2} > 3 = \top,$$
$$t_3: \quad [\![\text{altitude}_0 > 3]\!] = altitude_{t_3} > 3 = \bot,$$
$$t_4: \quad [\![\text{altitude}_0 > 3]\!] = altitude_{t_4} > 3 = \top.$$

The violation of $altitude^c(t) > 3$ at I_2 has the length $|I_2| = 0.3999... < \Delta_{max} = 1$. The formula violation at I_2 is not detected. The violation of $altitude^c(t) > 3$ at I_4 however has the length $|I_4| = 1.4999... \geq \Delta_{max}$ and results in a detected violation of ϕ. No matter how the sampling of $altitude^c(t)$ is shifted in time, there will always be a sample from within interval I_4 present in $altitude(t_n)$ which consequently results in a detected violation of ϕ.

In robot safety it is important to react quickly. Detecting violations in a timely manner for which the robot should react to is consequently very important. The time difference bound Δ_{limits} puts a bound on the detection delay of a formula violation. That is, how long it can take for a signal value change in $x^c(t)$ to take place before it is possible to conclude that a formula is false given the value-change in $x(t_n)$.

Theorem 6 *(Formula violation detection delay). Let C be the computationally-induced time-delay in `eval`, `Progression` and other external code. Then for reliable formula violation detection (Theorem 5) the detection delay d will be bounded by*

$$C < d \leq \Delta_{max} + C$$

and with a FIC sampling process the formula violation delay is always

$$d = C.$$

Proof. For the case $C < d$: If a reliable formula violation interval starts immediate before time point $t_n \in \mathbb{T}_x$ then the formula violation signal value will be present in the produced signal $x(t_n)$ at time-point t_n. The delay is therefor strictly larger than C.

For the case $d < \Delta_{max} + C$: If a reliable formula violation interval starts immediate after time point $t_n \in \mathbb{T}_x$ then the formula violating signal value will not be present in the produced signal $x(t_n)$ at time-point t_n. The formula violation interval is by Theorem 5 long enough for the violating signal value to be present at the next time-point t_{n+1}. By the definition of Δ_{limits} the maximum difference between two time-points, $t_{t+1} - t_n$ of the produced signal $x(t_n)$ is Δ_{max}. The delay will consequently be at most Δ_{max} in addition to C.

Finally, with a FIC sampling process there is no delay due to Δ_{limits} since (by Theorem 4) any changes in the continuous-time signal at $t = t_n$ have an immediate corresponding state at t_n in the discrete-time signal produced. The delay is consequently only due to C. □

Example 2 *(Formula violation detection delay).* Consider the illustrative example in Fig. 6 where the formula violation detection delay d is shown. The formula violation (filled red area) start at time-point t' in the figure. We will now consider a few cases with t' shifted in time (the formula violation will start at different time-points) and calculate what the delay d is in each case.

Let $\mathbb{T}_{altitude} = \langle t_0 = 0, t_1 = 1, t_2 = 2 \rangle$ and $\phi := \text{altitude}_0 > 3$. For $\mathbb{T}_{altitude}$ it is the case that $\Delta_{max} = 1$. Let $altitude^c(t)$ be a signal for which

$$altitude^c(t) > 3 \quad = \quad \begin{cases} \top, & 0.0 \leq t < t' \\ \bot, & t' \leq t < 1.1 \end{cases}.$$

For $t' \in \{0.9,\ 1.0,\ 1.1\}$ it is the case that

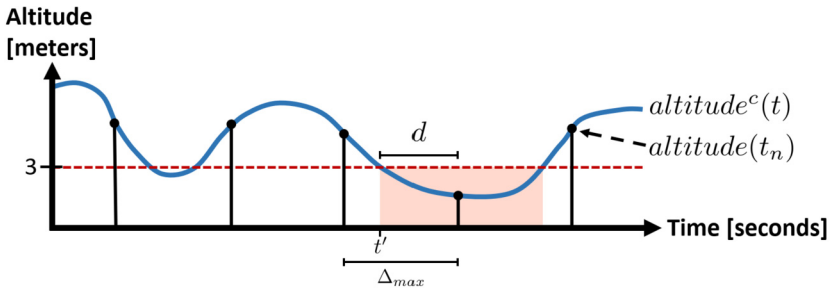

Fig. 6. The ProbSTL formula $\text{altitude}_0 > 3$ is monitored over signal $altitude(t_n)$ (**black** dots) which is a discretization of signal $altitude^c(t)$ (**blue** curve). Filled red area is a formula violation which is detected. The detection is d seconds after the start (t') of the formula violation.

$$
\begin{aligned}
t' = 0.9: \quad & \langle [\![\text{altitude}_0 > 3]\!] \text{ at } t_n \rangle_{t_n = t_0, t_1, t_2} = \langle \top, \bot, \bot \rangle, \\
& d = t_1 - t' + C = \underline{0.1 + C} \\
t' = 1.0: \quad & \langle [\![\text{altitude}_0 > 3]\!] \text{ at } t_n \rangle_{t_n = t_0, t_1, t_2} = \langle \top, \top, \bot \rangle, \\
& d = t_2 - t' + C = \underline{1.0 + C} \\
t' = 1.1: \quad & \langle [\![\text{altitude}_0 > 3]\!] \text{ at } t_n \rangle_{t_n = t_0, t_1, t_2} = \langle \top, \top, \bot \rangle, \\
& d = t_2 - t' + C = \underline{0.9 + C}
\end{aligned}
$$

In the first case ($t' = 0.9$) the first violation of ϕ occur at t_1 which is 0.1 seconds after the first violation of $altitude^c(t) > 3$ at t'. The violation detection occurs at $t_1 + C$, which includes the time C of evaluating ϕ at time-point t_1.

In the second case ($t' = 1.0$) the first violation of $altitude^c(t) > 3$ occur at time-point t' which is strictly after t_1. The first violation of ϕ is therefore not until t_2, which is 1 second after t'. This is the longest possible detection delay ($1.0 + C$) and increasing t' further ($t_1 < t' \leq t_2$) decreases the detection delay (the third case).

The bound $C < d \leq \Delta_{max} + C$ from Theorem 6 holds for all three cases.

The computational complexity of a progression iteration (one call to `Progression`) depends on the computational complexity of `Eval` which depends on the computational complexity of the inference algorithms in the $\mathcal{L}_{\text{prob}}$-plugins used.

Theorem 7 *(Computational complexity of single-step ProbSTL progression). The computational complexity of a progression iteration over a ProbSTL statement is*

$$\mathcal{O}\Big(|\phi| f(F_{\mathbb{R}}, F_{\mathbb{S}}, \mathbb{E})\Big)$$

where $|\phi|$ is the length of the ProbSTL formula and $f(F_{\mathbb{R}}, F_{\mathbb{S}}, \mathbb{E})$ is a function describing the computational complexity of computing functions and probabilities of events in `Eval` given the sets of real-valued functions $F_{\mathbb{R}}$, stochastic-valued function $F_{\mathbb{S}}$ and probability-theoretic events $\mathbb{E}$.

Proof. The length of the ProbSTL formula is the number of nodes in the full abstract syntax tree of ϕ which includes the sub trees of $\mathcal{L}_{\text{prob}}$ statements. Eval (Algorithm 2) visits each node in each $\mathcal{L}_{\text{prob}}$ abstract syntax sub tree once, and Progression (Algorithm 3) visits each non-$\mathcal{L}_{\text{prob}}$ node once. Together each node in the full abstract syntax tree is visited once.

Every visit to a node in Progression is a constant time operation except the nodes which are root nodes for $\mathcal{L}_{\text{prob}}$ sub trees (Line 15 in Algorithm 3). The complexity for evaluating a $\mathcal{L}_{\text{prob}}$ statement is a function of the real-valued functions $F_{\mathbb{R}}$, stochastic-valued function $F_{\mathbb{S}}$ and probability-theoretic events $\mathbb{E}$ given by the chosen ProbSTL plugins. □

ProbSTL allows terms on the form x_t and $\mathsf{x}_{t'|t}$ where $t \in \mathbb{R}_{\leq 0}$. Specifying $t = 0$ refer to the current time-point $t_n \in \mathbb{T}_x$ of the stream. For other t, that is for $t \in \mathbb{R}_{<0}$, then t will refer to some $t_m \in \mathbb{T}_x$ calculated using Equation (1). If $t_n + t \notin \mathbb{T}_x$ then there will be a temporal precision error between the time-point specified by the term $(t + t_n)$ and the time-point of the interpretation of the term (t_m).

The constraint $t_n + t \in \mathbb{T}_x$ can be enforced when writing formulas if the time domain $\mathbb{T}_x$ of signal $x(t_n)$ is known. The time domain is however in general unknown when constraint formulas are written. A special case when $\mathbb{T}_x$ is in fact known is when uniform sampling (Proposition 1) is performed and Δ_n is known.

For a FIC sampling process there is no problem with temporal precision since $t_n + t$ will fall into the suitable interval $\mathcal{I}_m$ corresponding to $t_m \in \mathbb{T}_x$ which exactly describes the signal values at time $t_n + t$.

Without (local) finite variability, and especially with variable sample rates, it is important to take temporal preciseness into consideration when writing constraint formulas and designing safe systems. The temporal preciseness of $t_n + t$ in

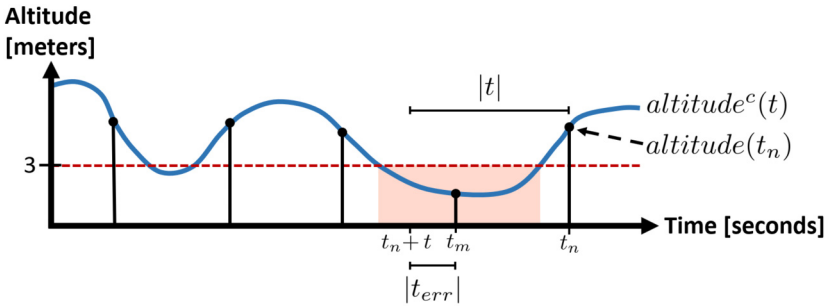

Fig. 7. The ProbSTL formula $\mathsf{altitude}_t > 3$, where t is a non-positive real number, is monitored over signal $altitude(t_n)$ which is a discretization of signal $altitude^c(t)$. $t_{err} = t_m - (t_n + t)$.

relation to $\mathbb{T}_x$ determines the temporal resolution of the terms with the temporal precision error being bounded by $\frac{\Delta_{max}}{2}$ (Theorem 8).

Theorem 8 *(Preciseness of time in terms). Let $\Delta_{max} < \infty$ be the maximal time difference between any two subsequent states $\mathsf{x}_{t_{j+1}}, \mathsf{x}_{t_j}$ in signal $x(t_n)$: $t_{j+1} - t_j \leq \Delta_{max}, \forall t_j \in \mathbb{T}$. The error between the specified stream time $t_n + t$ for a term of the form x_t or $\mathsf{x}_{t'|t}$ and the actual time-point t_m of a stream state which will represent this symbol is bounded by $|t_m - (t_n + t)| \leq \frac{\Delta_{max}}{2}$ for $t \geq t_0$.*

Proof. Let $S_{\leq n}$ be the current stream prefixes (signal prefixes). $m = n - k$ as calculated by (1) is the discrete-time-point closest to $\tilde{t} := t_n + t$ (if many, then it is the earliest) and is in the range $0 \leq m \leq n$. The difference in time between $\tilde{t}$ and t_m is $|t_m - \tilde{t}|$. Since t_m is the closest time-point to $\tilde{t}$ it is the case that $|\tilde{t} - t_{m-1}| \geq |t_m - \tilde{t}|$ and $|\tilde{t} - t_{m+1}| \geq |t_m - \tilde{t}|$. Since $|t_m - t_{m-1}| \leq \Delta_{max}$ and $|t_m - t_{m+1}| \leq \Delta_{max}$ then $|t_m - \tilde{t}| = |t_m - (t_n + t)| \leq \frac{\Delta_{max}}{2}$. □

Example 3 *(Preciseness of time in terms).* Consider the illustrative example in Fig. 7. We will consider a few possible values for t in the formula $\mathsf{altitude}_t > 3$ and calculate the temporal preciseness of respective case. We only consider three time-points in $\mathbb{T}_{altitude}$ for simplicity.

Let $\mathbb{T}_{altitude} = \langle t_0 = 1, t_1 = 2, t_2 = 3 \rangle$ and $\ell^1 := \mathsf{altitude}_{-0.4}$, $\ell^2 := \mathsf{altitude}_{-0.5}$, $\ell^3 := \mathsf{altitude}_{-0.9}$, $\ell^4 := \mathsf{altitude}_{-1.2}$. For $\mathbb{T}_{altitude}$ it is the case that $\Delta_{max} = 1$. The interpretation of ℓ^i is at $t_n = t_2$ (using Equation (1))

$$
\begin{aligned}
[\![\ell^1]\!] &= altitude_{t_k^1} \quad \text{where} \quad t_k^1 = f_{\mathbb{T}}(2, -0.4) = t_2, \\
[\![\ell^2]\!] &= altitude_{t_k^2} \quad \text{where} \quad t_k^2 = f_{\mathbb{T}}(2, -0.5) = t_1, \\
[\![\ell^3]\!] &= altitude_{t_k^3} \quad \text{where} \quad t_k^3 = f_{\mathbb{T}}(2, -0.9) = t_1, \\
[\![\ell^4]\!] &= altitude_{t_k^4} \quad \text{where} \quad t_k^4 = f_{\mathbb{T}}(2, -1.2) = t_1.
\end{aligned}
$$

From Theorem 8 it is the case that $|t_m - (t_n + t)| \leq \Delta_{max}/2$. It holds for each ℓ^i:

$$
\begin{aligned}
\ell^1: &\quad |t_2 - (3 - 0.4)| = 0.4 \leq 0.5, \\
\ell^2: &\quad |t_1 - (3 - 0.5)| = 0.5 \leq 0.5, \\
\ell^3: &\quad |t_1 - (3 - 0.9)| = 0.1 \leq 0.5, \\
\ell^4: &\quad |t_1 - (3 - 1.2)| = 0.2 \leq 0.5.
\end{aligned}
$$

Reliable formula violation detection (Theorem 9) is limited by the minimal sample rate (Δ_{max}). Using a prediction stream (produced by a time series model) it is possible to up-sample the signal with predictions to achieve reliable formula violation detection at higher time-resolution than Δ_{max}. The prediction stream represents the uncertainty of the signal value on the real-valued time-line (not just at the observations) and the formulas consequently have to be probability-constrained to deal with this uncertainty.

Theorem 9 *(Improved reliability of formula violation detection by interpolation). Let ℓ be a probability-constrained formula $Pr(\mathsf{E}(\mathsf{x}_{0|0}, \ldots)) > \mathsf{probability}$. Let $\Delta_n = t_n - t_{n-1}$. The violation-interval of reliable formula violation detection of probability-constrained formulas can be reduced from that of Theorem 5 down to $\frac{\Delta_n}{K}$ by using prediction terms to interpolate (up-sample) the stream,*

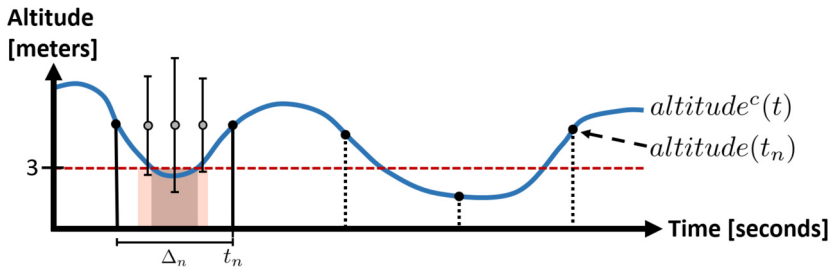

Fig. 8. The ProbSTL formula $Pr(\text{altitude}_{0|0} > 3) > 0.95$ is monitored over signal $altitude(t_n)$ which is a discretization of signal $altitude^c(t)$. The gray dots are predictions of *altitude* and the error-bars show the 90% probability interval of the prediction. The previously undetected violation (filled gray area) is now detected via the detectable uncertainty-constrained violation (filled red area).

$$\bigwedge_{k=1}^{K} Pr(\mathsf{E}(\mathsf{x}_{\frac{-(k-1)\Delta_n}{K}|0}, \dots)) > \text{probability},$$

if the probability density function over property $\boldsymbol{x}$ *is conservative with respect to* E.

Proof. The probability density function over property $\boldsymbol{x}$ is conservative with respect to E if the probability of $E(\boldsymbol{x})$ is not overestimated. Following this it is the case that $Pr(E(\boldsymbol{x}_t^c, \dots)) \geq Pr(E(\boldsymbol{x}_{t|t}, \dots)) > c$. Consequently, a violation will not be missed (but false violations w.r.t. $x^c(t)$ can occur). By definition $\Delta_n \leq \Delta_{max}$ which means that $\frac{\Delta_n}{K} \leq \frac{\Delta_{max}}{K}$. □

Example 4 *(Improved reliability of formula violation detection by interpolation).* Consider the illustrative example in Fig. 8.

The formula $\ell := Pr(\text{altitude}_{0|0} > 3) > 0.95$ is replaced with

$$\begin{aligned}
\ell' := & \bigwedge_{k=1}^{4} Pr(\text{altitude}_{-\frac{(k-1)\Delta_n}{4}|0} > 3)) > 0.95 \\
= \quad & Pr(\text{altitude}_{0|0} > 3)) > 0.95 \\
& \wedge Pr(\text{altitude}_{-\frac{1\Delta_n}{4}|0} > 3)) > 0.95 \\
& \wedge Pr(\text{altitude}_{-\frac{2\Delta_n}{4}|0} > 3)) > 0.95 \\
& \wedge Pr(\text{altitude}_{-\frac{3\Delta_n}{4}|0} > 3)) > 0.95
\end{aligned}$$

in order to improve the reliability of formula violation detection. The temporal resolution is increased for formula violation detection, but with the draw-back that the predictions are more uncertain further away from the observed data points.

For simplicity a Kalman smoother with constant-position motion model is used in this example. The three predictions $\text{altitude}_{-\frac{1\Delta_n}{4}|0}$, $\text{altitude}_{-\frac{2\Delta_n}{4}|0}$ and $\text{altitude}_{-\frac{3\Delta_n}{4}|0}$ are shown in Fig. 8 as gray dots with 90% probability intervals. There is consequently 5% probability density above and below the error-bar. The uncertainty is smaller closer to observations (t_n and t_{n-1}) and at its highest at the midpoint.

The filled red area in Fig. 8 shows the interval where $[\![\ell']\!] = \bot$. That is, where at least 5% probability density is below the 3 meter threshold (the probability of the altitude being higher than 3 meters is 95% or lower). The filled red area is wider or equal to the filled gray area as long as the predictions are conservative with respect to the event $>$. That is, as long as the probability of the event is not overestimated.

The reliability is improved with ℓ' over ℓ by a reduction in false positives (missed true formula violations). The reduction is payed for by an increase in false negatives, that is formula violations with respect to $altitude(t_n)$ which actually are not violations with respect to $altitude^c(t)$.

The prediction stream further allows formula violation detection over the predicted future. This is useful for detecting likely formula violations before they happen, which among other things circumvents the formula violation detection delay from Theorem 6. Similarly to Theorem 9 it is possible to perform reliable formula violation detection over the predicted future at a higher temporal resolution than the sampled signal (Theorem 10).

Theorem 10 *(Improved reliability of formula violation detection by extrapolation). Let* ℓ *be a probability-constrained formula* $Pr(\mathsf{E}(\mathsf{x}_{0|0}, \dots)) > \text{probability}$. *The violation-interval of reliable formula violation detection of probability-constrained formulas can be reduced from that of Theorem 5 down to* $\frac{\Delta_{max}}{K}$ *by using prediction terms,*

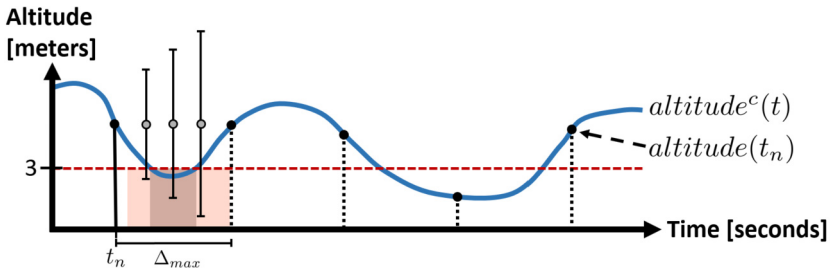

Fig. 9. The ProbSTL formula $Pr(\text{altitude}_{0|0} > 3) > 0.95$ is monitored over signal $altitude(t_n)$ which is a discretization of signal $altitude^c(t)$. The gray dots are predictions of *altitude* and the error-bars show the 90% probability interval of the prediction. The previously undetected violation (filled gray area) is now detected via the detectable uncertainty-constrained violation (filled red area).

$$\bigwedge_{k=1}^{K} Pr(\mathsf{E}(\mathsf{x}_{\frac{(k-1)\Delta_{max}}{K}|0}, \dots)) > \text{probability},$$

if the probability density function over property $\boldsymbol{x}$ *is conservative with respect to* E.

Proof. Identical to the Proof of Theorem 9 but where Δ_n is substituted for $\Delta_{max} \geq t_{n+1} - t_n$. □

Example 5 *(Improved reliability of formula violation detection by extrapolation).* Consider the illustrative example in Fig. 9 which is similar to Example 4 but with predictions into the future (extrapolations) as opposed to predictions of intermediary signal values (interpolation).

The formula $\ell := Pr(\text{altitude}_{0|0} > 3) > 0.95$ is replaced with

$$\begin{aligned}
\ell' &:= \bigwedge_{k=1}^{4} Pr(\text{altitude}_{\frac{(k-1)\Delta_{max}}{4}|0} > 3)) > 0.95 \\
&= \quad Pr(\text{altitude}_{0|0} > 3)) > 0.95 \\
&\quad \wedge Pr(\text{altitude}_{\frac{1\Delta_{max}}{4}|0} > 3)) > 0.95 \\
&\quad \wedge Pr(\text{altitude}_{\frac{2\Delta_{max}}{4}|0} > 3)) > 0.95 \\
&\quad \wedge Pr(\text{altitude}_{\frac{3\Delta_{max}}{4}|0} > 3)) > 0.95
\end{aligned}$$

in order to improve the reliability of formula violation detection. The difference to Example 4 is that the uncertainty grows over time after t_n (using the same Kalman smoother). The consequence of this is that the trade-off between false positives and false negatives becomes more severe, which is seen in Fig. 9 by the wider red area compared to Fig. 8

Corollary 1 *(Increased complexity). The improved formula violation detection reliability in Theorem 9 and Theorem 10 increases the computational complexity by a factor* K*, respectively, on the computational complexity of* `Eval`.

We have in this chapter considered several aspects which are of important if STL or ProbSTL is to be used for monitoring physical signals, for example in robot safety applications. These are important to keep in mind and highly relevant if working with cyber-physical systems or robots in practice.

6. Empirical evaluation

To provide more insights into the details of ProbSTL and its possible use we here consider two use-cases. Both use-cases consider constraints on spatio-temporal movement of a UAV. In the first use-case a simulated altitude signal is used and in the second use-case a high-fidelity simulator of the physical dynamics of a DJI M100 drone is used to generate realistic data for monitoring UAV behaviors.

6.1. UAV altitude constraint

Consider the setting in Fig. 4 where a UAV has to stay above a certain altitude for safety reasons. Assume that some UAV task requires the UAV to move up and down repeatedly. The reason for the motion could be to avoid obstacles that gets in the way of the UAV such as street lights, power-cables, buildings or other UAVs. Further assume that the uncertainty in the UAV altitude changes over time, for example due to GPS interference/shadowing or due to varying precision in the visual odometry. The altitude observations are generated from the noisy process with a 10 hz sample rate,

$$x_t^c = 0.5\sin(2\pi 0.2t) + 3.2$$
$$y_{t_n} \sim \mathcal{N}(x_{t_n}^c, \sigma_\epsilon^2(t_n))$$

where y_{t_n} are noisy observations of the altitude produced from sensor measurements as a stream over time points $t_n \in \mathbb{T}$. The uncertainty (variance) varies over time between $\sigma_{min}^2 = 0.01^2$ and $\sigma_{max}^2 = 0.4^2$ through $\sigma_\epsilon^2(t) = \sigma_{min}^2 + (\sigma_{max}^2 - \sigma_{min}^2)(0.5 + 0.5\sin(2\pi f t))$. The observation uncertainty $\sigma_\epsilon^2(t_n)$ is assumed to be known, and it has the uncertainty varying frequency $f = \frac{1}{3}$ hz.

The altitude x_t^c and our uncertainty about it is represented as a gaussian distributed stochastic variable $\boldsymbol{x}_t$ and estimated via Bayesian filtering using a Kalman filter with a constant-velocity motion model. The constant-velocity motion model is a crude approximation of the true dynamics, but it is compensated for by making sure that the uncertainty is conservative (the uncertainty is overestimated). The state $\boldsymbol{z}_t$ consists of the stochastic altitude $\boldsymbol{x}_t$ and its velocity $\boldsymbol{v}_t$. We represent our uncertainty of the state with a multivariate Gaussian distribution

$$\boldsymbol{z}_t \sim \mathcal{N}(\mu_{\boldsymbol{z}_t}, \Sigma_{\boldsymbol{z}_t}), \quad \mu_{\boldsymbol{z}_t} = \begin{bmatrix} \mu_{\boldsymbol{x}_t} \\ \mu_{\boldsymbol{v}_t} \end{bmatrix}, \ \Sigma_{\boldsymbol{z}_t} = \begin{bmatrix} \sigma_{\boldsymbol{x}_t}^2 & \sigma_{\boldsymbol{x}_t \boldsymbol{v}_t} \\ \sigma_{\boldsymbol{x}_t \boldsymbol{v}_t} & \sigma_{\boldsymbol{v}_t}^2 \end{bmatrix}$$

and the altitude when marginalizing out the velocity is

$$\boldsymbol{x}_t \sim \mathcal{N}(\mu_{\boldsymbol{x}_t}, \sigma_{\boldsymbol{x}_t}^2).$$

The state space model of the Kalman filter is

$$\begin{aligned} z_{t+\Delta} &= F_\Delta z_t + w_\Delta, & w_\Delta &\sim \mathcal{N}(0, Q_\Delta), \\ y_t &= H z_t + \epsilon_t, & \epsilon_t &\sim \mathcal{N}(0, R_t), \end{aligned} \tag{2}$$

where the matrices F_Δ, Q_Δ, H and R_t are defined as

$$F_\Delta = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix},$$
$$Q_\Delta = \begin{bmatrix} 1.5^2 \frac{\Delta^2}{2} & 0 \\ 0 & 1.5^2 \Delta \end{bmatrix},$$
$$H = [1\ 0],$$
$$R_t = \sigma_\epsilon^2(t),$$

and where $\Delta = t' - t$ is the time difference between time point t' and the current time point t, F_Δ is the state transition matrix, Q_Δ is the process noise covariance, H is the observation matrix, z_t is the state vector at time t, y_t is the observation vector at time t and R_t is the observation noise covariance at time t. We represent our uncertainty in the model with the process noise as 1.5 m standard deviations position-wise and 1.5 m/s velocity-wise.

The measurement update of the Kalman filter, incorporating a new observation y_{t_n} given a previous state for time-point t_{n-1}, is given by

$$\mu_{z_{t_n|t_n}} = \mu_{z_{t_n|t_{n-1}}} + K_{t_n|t_{n-1}}\left(y_{t_n} - H\mu_{z_{t_n|t_{n-1}}}\right), \tag{3}$$
$$\Sigma_{z_{t_n|t_n}} = (I - K_{t_n|t_{n-1}} H)\Sigma_{z_{t_n|t_{n-1}}} \tag{4}$$

where the Kalman gain is $K_{t_n|t_{n-1}} = \Sigma_{z_{t_n|t_{n-1}}} H^T \left(H \Sigma_{z_{t_n|t_{n-1}}} H^T + R_t\right)^{-1}$.

The Kalman filter prediction is given by

$$\mu_{z_{t'|t_n}} = F_{t'-t_n} \mu_{z_{t_n|t_n}}$$
$$\Sigma_{z_{t'|t_n}} = F_{t'-t_n} \Sigma_{z_{t_n|t_n}} F_{t'-t_n}^T + Q_{t'-t_n}$$

where t' can be smaller or larger than t_n depending on if the prediction is into the past or into the future. The prediction for $t' = t_{n+1}$ is used in (3) - (4).

The ProbSTL stream is defined as $S = \langle altitude(t_n), \boldsymbol{Altitude}(t_n) \rangle$ where $altitude(t_n)$ is a deterministic discrete-time signal and $\boldsymbol{Altitude}(t_n)$ is a prediction stream. The two signals are defined for $\forall t_n \in \mathbb{T}$ as

$$altitude_{t_n} \in altitude(t_n)$$
$$\boldsymbol{altitude}_{t_n|t_n} \in \boldsymbol{Altitude}(t_n)$$

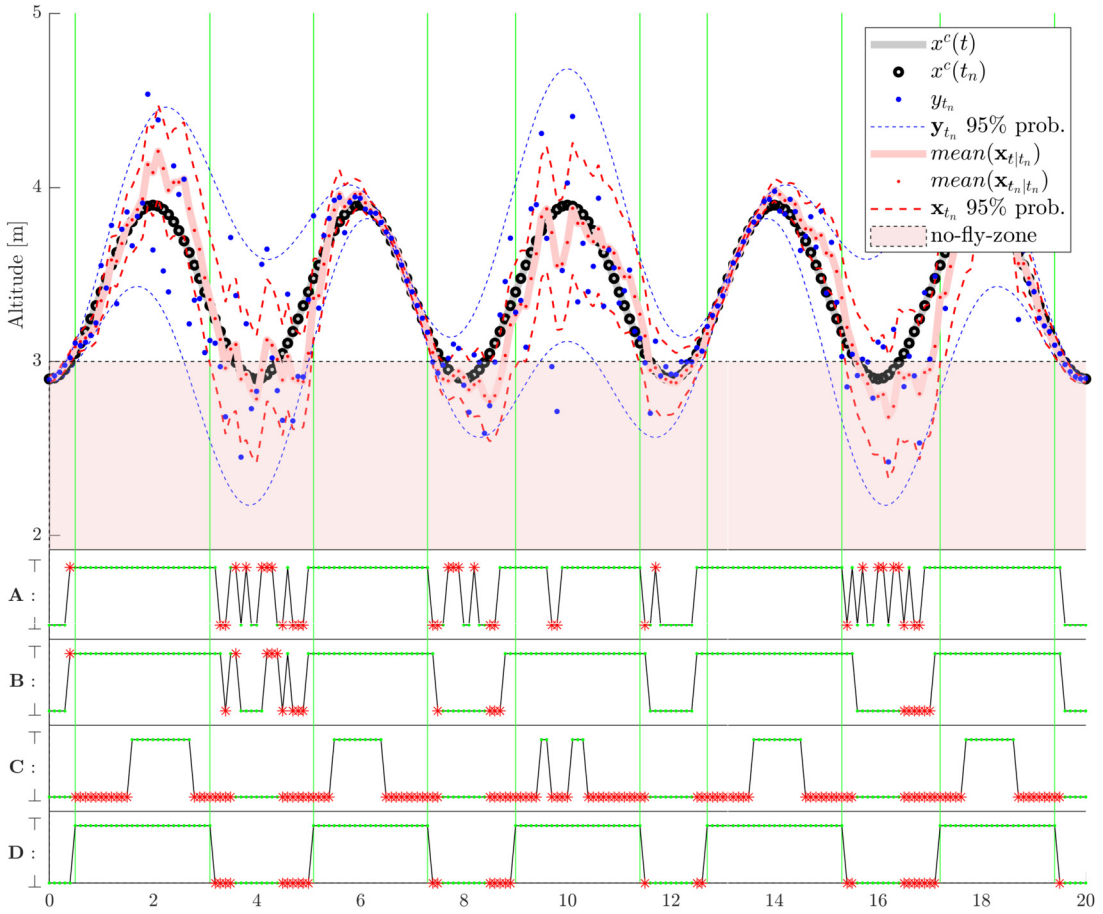

Fig. 10. Illustration of the altitude experiment for the four constraint formulas **A**, **B**, **C**, **D**. The truth value of an evaluated formula is plotted with **green dots** whenever it has the same truth value ($\top, \bot$) as the ground truth ($x^c(t)$) and as a **red star** when not. Incorrect $\bot$ is a false alarm and incorrect $\top$ is a missed constraint violation. The 95% probability intervals (approximately two standard deviations) are shown with dashed lines.

where the stochastic variables are defined as $\boldsymbol{altitude}_{t_n|t_n} \sim \mathcal{N}(\mu_{x_{t_n|t_n}}, \sigma^2_{x_{t_n|t_n}})$ from (3) - (4) and $altitude_{t_n} = y_{t_n}$. We consider the monitoring of four different constraints on the UAV altitude,

$$\begin{aligned}
&\mathbf{A}: && \text{altitude}_{t_n} > 3 \\
&\mathbf{B}: && \text{mean}(\text{altitude}_{t_n|t_n}) > 3 \\
&\mathbf{C}: && \text{mean}(\text{altitude}_{t_n|t_n}) > 3 + 0.7840 \\
&\mathbf{D}: && Pr(\text{altitude}_{t_n|t_n} > 3) > 0.95
\end{aligned}$$

where the purpose is to detect violations to the constraint that the altitude must be larger than 3 at all times. Constraint **C** contains a safety margin of 0.78403 meters. The safety margin is calculated such that it should hold with 0.95 probability given observations drawn from $\mathcal{N}(\mu_{\boldsymbol{x}_{t|t}}, \sigma^2_{max})$. The safety margin is calculated using the function **Q**,

$$\mathbf{Q}(p, \mu, \sigma) = \mu + \sigma\ \Phi^{-1}\Big(\frac{p+1}{2}\Big) = \mu + \sigma\sqrt{2}\,\text{erf}^{-1}(p),$$

where $p \in [0\ 1]$ is a probability and erf^{-1} is a standard function in most mathematical software libraries. The interpretation of **Q** is that if $c = \mathbf{Q}(p, \mu, \sigma)$ then $\int_{-\infty}^{c} \mathcal{N}(x; \mu, \sigma^2)dx = p$. The safety margin is given by $\mathbf{Q}(0.95, 0, \sigma_{max}) = 0.7840$.

Fig. 10 shows the signals and the constraint satisfaction for 20 seconds. The full experiment lasts 300 seconds. The result of the experiment is summarized in Table 1.

Formula **A** is the base case where the raw and noisy sensor measurements are used directly. The true positive rate and true negative rate is about 0.926 and 0.683 respectively (Table 1). These rates are improved for Formula **B** by using the estimated mean altitude which suppresses much of the noise present in the sensor measurements.

Table 1
Results of the altitude experiment. Constraint formulas are presented in the order **A**, **B**, **C**, **D**. Formula **A** and **B** both have a high False Negative rate which makes them unsuitable as safety constraints. Formula **D** is almost as safe as formula **C** (low False Negative rate) but has far fewer false alarms (False Positive rate).

Constraint formula	True Positives	False Positives	True Negatives	False Negatives
$\text{altitude}_{t_n} > 3$	0.92559	0.074409	0.68296	0.31704
$\text{mean}(\text{altitude}_{t_n \mid t_n}) > 3$	0.93505	0.064946	0.86667	0.13333
$\text{mean}(\text{altitude}_{t_n \mid t_n}) > 3 + 0.7840$	0.34409	0.65591	1.00000	0.00000
$Pr(\text{altitude}_{t_n \mid t_n} > 3) > 0.95$	0.81333	0.18667	0.99852	0.00149

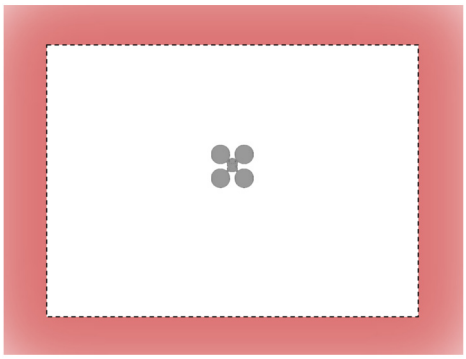

Fig. 11. UAV inside an allowed space specified as a rectangle. The restricted space is in red.

The true negative rate corresponds to the number of true constraint violations for which the monitoring of a formula also reported a violation. From a safety perspective it is important to never miss any true violations, or as few as possible given the inherent uncertainty in physical sensing. The true negative rate is increased substantially for Formula **C** by adding a static safety margin to Formula **B**. The cost of this safety margin is however that the true positive rate drops significantly (0.344). The false positive rate corresponds to the rate of false alarms in detecting constraint violations. A low false alarm rate (a high true positive rate) is crucial in order to keep a high task efficiency. Formula **D** improves the true positive rate drastically (0.813) while maintaining a high true negative rate. This is achieved by a dynamic safety margin implicitly coupled to the uncertainty in the altitude directly. If the uncertainty is large then the safety margin is as large as for **C**, but if it is small then the safety margin also becomes small and thus allows for higher task efficiency.

All four formulas can be expressed explicitly in ProbSTL, but only the first one (formula **A**) can be explicitly expressed in STL. Formula **D** is the best performing formula in this experiment and it is clearly specified in ProbSTL with a clear interpretation.

6.2. UAV 2D use-case

Assume that there is an area where a UAV is allowed to operate. Outside of it there might be (untrained) people or reduced support infrastructure for the UAV positioning. Sometimes the UAV might venture outside the area for short amounts of time. It might be on purpose, or accidentally due to for example strong wind pulling it away. Whenever the UAV ends up outside of the area it must get back inside right away.

The high-fidelity simulator used in this experiment operates in 3D space. The experiment is however conducted in a spatial 2D plane at a fixed altitude. This means that the trajectory followed by the MPC lies on this plane, while the simulated UAV state will stay as close to this altitude as the controller can manage given the UAV dynamics. The experiment can trivially be extended to full 3D trajectories, but this would make the presentation unnecessary complicated (visualizations in 3D instead of 2D). The simplification of 2D instead of 3D does not affect the results nor the conclusions.

Let the allowed area to be defined by a rectangle (Fig. 11) where the rectangle is defined by a center $rect_c$, a width $rect_w$ and height $rect_h$. Let $rect = \langle c, \bar{r} \rangle$ where

$$
\begin{aligned}
c &= rect_c, \\
\bar{r} &= [r_1\ r_2]^T, \\
r_1 &= \frac{rect_w}{2}, \\
r_2 &= \frac{rect_h}{2}
\end{aligned}
$$

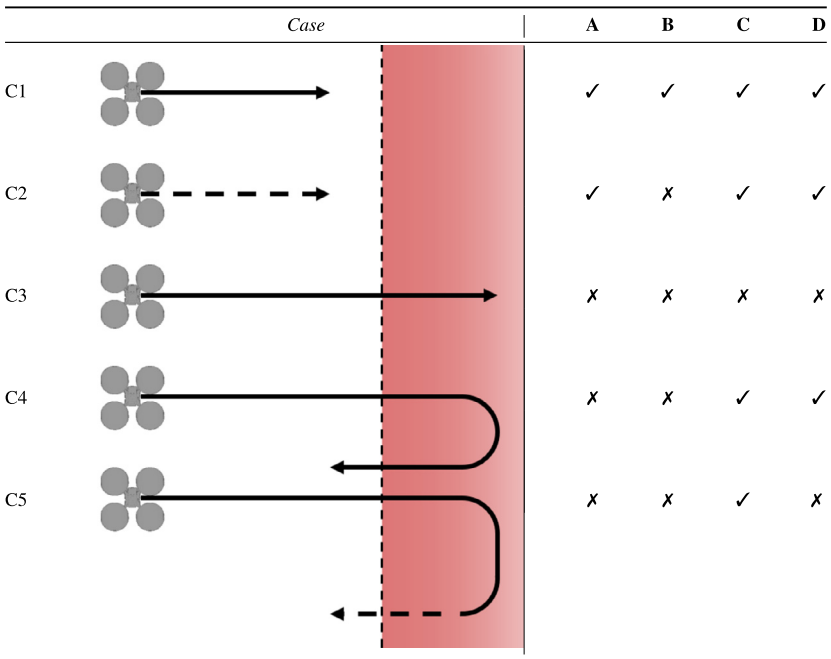

Fig. 12. Five example cases C1-C5 of UAV motions are illustrated together with if the formulas **A**-**D** hold (✓) or not (✗) for each case. Case **C1**: Motion to a point close to the restricted space. Case **C2**: High-speed motion to a point close to the restricted space. Case **C3**: Motion to a point far into the restricted space. Case **C4**: Motion into the restricted space with an immediate return to the allowed space. Case **C5**: Motion into the restricted space with a dragged-out U-turn and a sudden high-speed return to the allowed space.
Dashed arrow indicates higher speeds.

We consider the monitoring of four different ProbSTL formulas **A**, **B**, **C**, **D** constraining the behavior of the UAV in relation to the rectangular area in different ways,

$$\mathbf{A}: \quad Pr(\text{insideRectangle}(\text{Position}_{0|0}, rect)) > 0.95$$

$$\mathbf{B}: \quad Pr(\text{insideRectangle}(\text{Position}_{1|0}, rect)) > 0.95$$

$$\mathbf{C}: \quad \neg(Pr(\text{insideRectangle}(\text{Position}_{0|0}, rect)) > 0.95) \rightarrow \Diamond_{[0\ 4]}(Pr(\text{insideRectangle}(\text{Position}_{0|0}, rect)) > 0.95)$$

$$\mathbf{D}: \quad \neg(Pr(\text{insideRectangle}(\text{Position}_{0|0}, rect)) > 0.95) \rightarrow (\quad \Diamond_{[0\ 4]}(Pr(\text{insideRectangle}(\text{Position}_{0|0}, rect)) > 0.95) \wedge \Box_{[2\ 4]}(Pr(\text{insideRectangle}(\text{Position}_{1|0}, rect)) > 0.95))$$

Several interesting cases of UAV behavior are considered in Fig. 12, where it is also shown if the monitoring formulas will hold true or detect a violation for the different behavior cases.

Formula **A** is a reactive constraint, considering if the UAV is outside the area at the current time, as opposed to Formula **B** which is predictive since it considers if the UAV is expected (predicted) to be outside of the area 1 second from now. A violation to Formula **B** gives the UAV a chance to make some action in order to do something about leaving the area. When Formula **A** is violated it is too late to do something about it. This gets even more important when considering collisions.

Formula **C** constrains the UAV to be back inside the area again within 5 seconds from when it left the area. Formula **D** adds to Formula **C** that the UAV also have to be predicted to get inside the area in the near future when it has been outside the area for 2 seconds already.

In a multi-agent cooperative setting without perfect information it is important that the agents behave predictable as a way to show their intent. This is important both for task efficiency and for safety. Formula **B** can be seen as a constraint which assures that the behavior of the UAV does not reflect the intent to leave the area. Similarly, Formula **D** reflects the intent of entering the area again (case C4) as opposed to behaving less predictable (case C5).

For this experiment we consider the non-linear UAV dynamics described in [28] and use a high-fidelity simulator of the dynamics. The virtual UAV is controlled using a Model Predictive Controller (MPC) based on the work in [29]. A MPC solves an on-line optimization problem to find a discrete sequence of control signals which produce the optimal trajectory

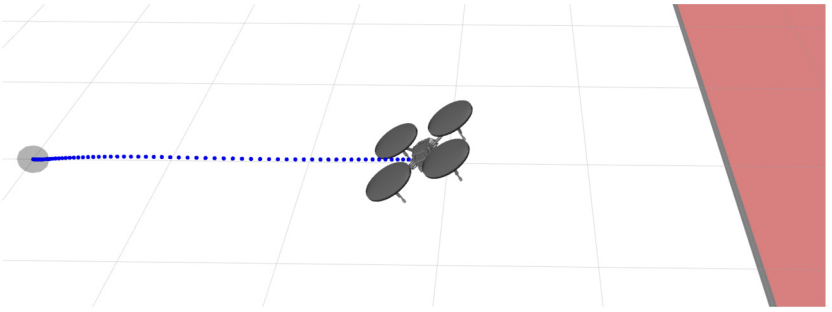

Fig. 13. The UAV is slowing down at the end of case **C2**. The grid cells have 1 meter sides and the no-fly-zone begins 5 meters to the right (in the x-direction) of the starting location (gray sphere). The end location is at 3.5 meters to the right of the staring location.

according to some goal while adhering to specified constraints. Typical constraints are limitations in angle, thrust or velocity. The scenario with all five cases described in Fig. 12 are realized in ROS (Robot Operating System) using the dynamics simulator and MPC. A screen shot from Rviz when the UAV is running case **C2** is shown in Figure Fig. 13. The controller and sampling is running at 50 hz.

We consider a Kalman filter with a 2D constant-velocity motion model as a simple and very approximate state estimation method. The Kalman filter has a state-space model (2), where the state consist of 2D position and 2D velocity, with the matrices F_Δ, Q_Δ, H and R_t defined as

$$F_\Delta = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$Q_\Delta = \begin{bmatrix} 1.5^2\frac{\Delta^2}{2} & 0 & 0 & 0 \\ 0 & 1.5^2\frac{\Delta^2}{2} & 0 & 0 \\ 0 & 0 & 1.5^2\Delta & 0 \\ 0 & 0 & 0 & 1.5^2\Delta \end{bmatrix},$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$R_t = [0.0001^2\ 0.0001^2]^T.$$

The Kalman filter is applied on the observed trajectory of 2D positions. The results are shown in Fig. 14. It can be seen that the predictions made for the high-speed situations in **C2** are very wrong, causing formula **B** to fail, and that the actual future positions are far away from the 95% probability ellipse. The process noise of the Kalman filter has been tuned to be statistically conservative for lower velocities, but the predictions are not conservative given the higher velocities. The 95% probability ellipse would have to be much larger for the predictions to be conservative.

The assumption that predictions are statistically conservative is important, because otherwise the prediction cannot be relied upon. This can happen if the state estimation is incorrectly implemented (e.g. wrongly tuned) or if the robot operates outside of assumed specifications (e.g. in a new situation). One possible statistically conservative prediction constraint is to require that 95% probability density of the current best estimate should be within the 95% probability interval of the prediction. This constraint to be monitored can be formulated as

$$\mathbf{E}: \quad Pr(\text{insidePI}(\text{Position}_{0|0}, \text{Position}_{0|\text{-}1}, 0.95) \geq 0.95$$

where, for $\boldsymbol{x}, \boldsymbol{y}$ being N-dimensional multivariate isometric Gaussian distributions,

$$\boldsymbol{x} \sim \mathcal{N}(\mu_{\boldsymbol{x}}, \Sigma_{\boldsymbol{x}}), \qquad \mu_{\boldsymbol{x}} = \begin{bmatrix} \mu_{\boldsymbol{x}_1} \\ \mu_{\boldsymbol{x}_2} \\ \vdots \\ \mu_{\boldsymbol{x}_N} \end{bmatrix}, \qquad \Sigma_{\boldsymbol{x}} = \begin{bmatrix} \sigma^2_{\boldsymbol{x}_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma^2_{\boldsymbol{x}_N}, \end{bmatrix}$$

$Pr(\text{insidePI}(\boldsymbol{x}, \boldsymbol{y}, p))$ is defined as

$$\mathbb{P}[\text{insidePI}(\boldsymbol{x}, \boldsymbol{y}, p)] = \prod_{k=1}^{N} \mathbb{P}[\text{insidePI}(\boldsymbol{x}_k, \boldsymbol{y}_k, p)]$$

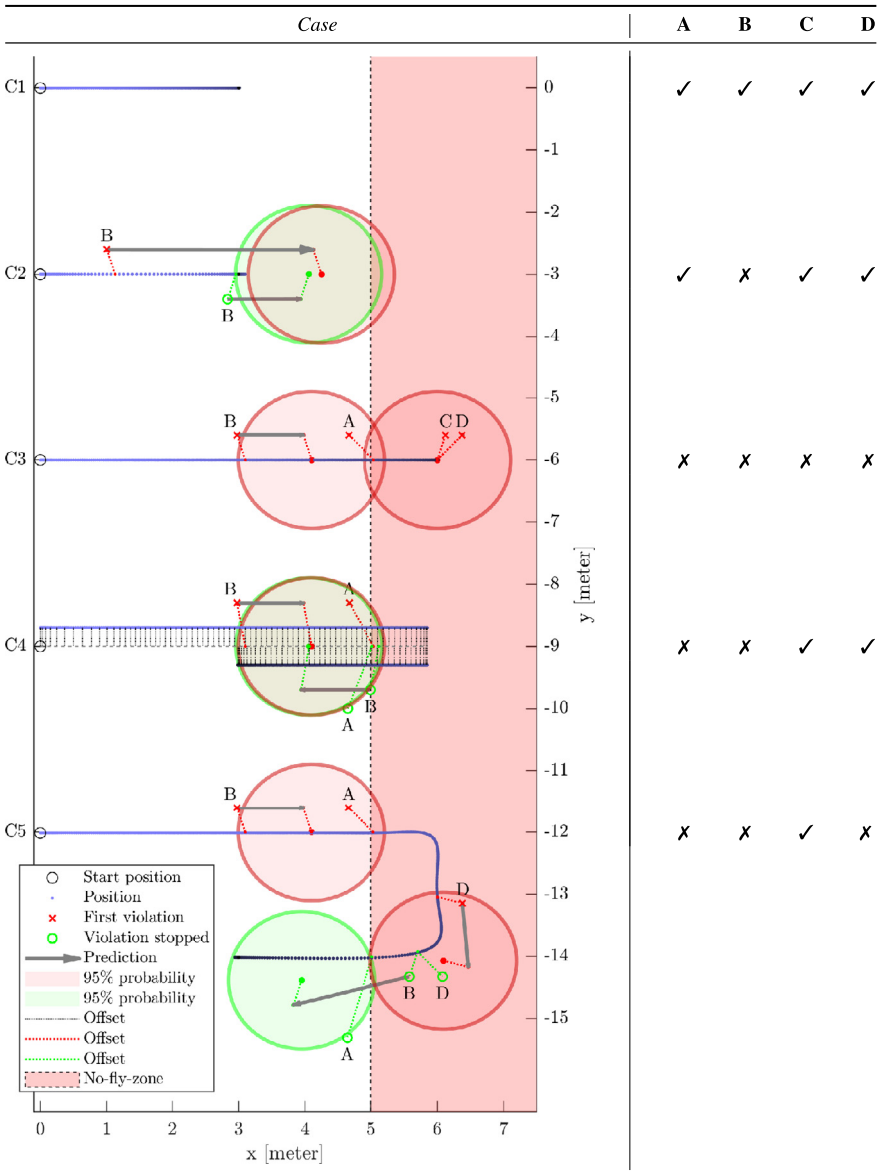

Fig. 14. Simulated UAV experiment. Offsets (**black**/**red**/**green** dashed lines) are used to separate overlapping points/lines/arrows for clarity. Predictions are shown with their mean (thick **red**/**green** dot) and 95% probability interval (ellipse), centered at the tip of an arrow.
C4: The UAV moves back again without turning when it reaches 1 meter inside the no-fly-zone. The dashed gray line shows the actual y-positions with the trajectory to and from the no-fly-zone are shown above and below this line. **C5**: Both B and D stop being violated at the same time since the prediction (shown for only B here) has 95% probability density outside the no-fly-zone.

with

$$\mathbb{P}[\text{insidePI}(\boldsymbol{x}_k, \boldsymbol{y}_k, p)]$$
$$= \mathbb{P}[\text{insideInterval}(\boldsymbol{x}_k, \mu_{\boldsymbol{y}_k} - \mathbf{Q}(p, 0, \sigma_{\boldsymbol{y}_k}), \mu_{\boldsymbol{y}_k} + \mathbf{Q}(p, 0, \sigma_{\boldsymbol{y}_k}))]$$

Monitoring constraint formula **E** for case **C1-C5** (Fig. 15) shows that the UAV actually violates this conservative-prediction constraint for **C2** and **C5**. The violations occur due to faster velocity and acceleration than what we have modeled for in terms of process noise. The violations can mean three different things. (1) The Kalman filter is wrong and we need to increase the process noise to compensate for the allowed agility of the controller. This will however cause the uncertainty to increase which typically decreases task efficiency. (2) The controller is wrong and the constraints on velocity and acceleration must be adjusted to restrict the UAV motion. (3) Assumptions about the environment are incorrect and there are external forces (e.g. wind) which affect the UAV more than what the controller can compensate for.

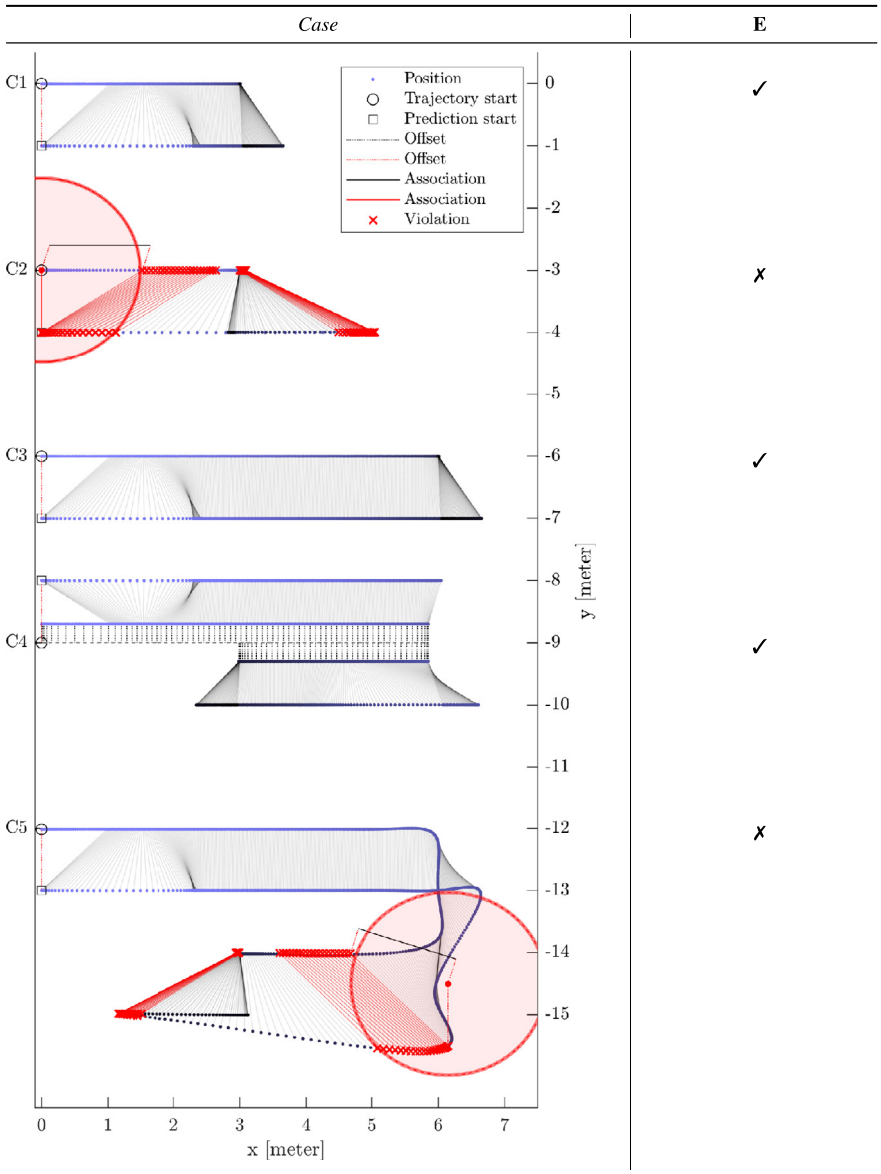

Fig. 15. Simulated UAV experiment. Estimated position and predicted position is shown as blue dots. The predictions have a 1 meter offset vertically to the estimated positions. Predictions are 1 second into the future and the estimated position at the time the predictions are about are shown with a black association line. The 95% probability interval (ellipse) is shown for the first prediction which causes a violation of formula **E**.

Robot safety is becoming increasingly important when robotic platforms are intended to operate close to humans in human-tailored environments. It is then crucial to monitor for perception/model errors (1), controller errors (2) and environmental errors (3). It might be impossible to conclude which one of perception, control or environment that is the reason for a constraint violation. It is regardless possible to monitor the necessary condition on robot safety that they should all be consistent (no violation).

We have shown that it is possible to formulate and monitor relevant safety constraints using ProbSTL. It is important to know what a safety specification means, and to know that its monitoring is correct. ProbSTL provides a clearer and more expressive formalism than STL. Progression allows for timely incremental evaluation of formulas and we have proved its correctness for STL and ProbSTL. The plugin modularization further allows ProbSTL to be easily extended to suite specific needs and future state-of-the-art probabilistic inference techniques. The plugin separation makes it easier to prove correctness and computational complexity for ProbSTL extended with specific plugins since it is made clear how correctness and complexity in the plugins affect the properties of ProbSTL. It is consequently easier to add plugins and derive theoretical guarantees of monitoring with available plugins than what is the typical case otherwise.

7. Conclusions and future work

Robot safety can be strengthened by runtime verification via incremental evaluation of logically specified constraints on the environment, including the behavior of the own platform. We propose ProbSTL, a probabilistic extension to STL where the expressiveness is increased by adding new symbols that are well defined and useful for robot safety applications. This specialization with respect to STL allow for powerful constraints to be formulated and monitored, with symbols grounded in probability and estimation theory. ProbSTL is easily extendable and we provide an example spatial plugin. Properties of ProbSTL are analyzed given realistic assumptions on the underlying signals and we prove the correctness of the incremental reasoning technique progression over ProbSTL formulas. We demonstrate the usefulness and importance of ProbSTL for robot safety in two different UAV experiments.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is partially supported by grants from the National Graduate School in Computer Science, Sweden (CUGS), the Swedish Research Council (VR) Linnaeus Center CADICS, ELLIIT Excellence Center at Linköping-Lund for Information Technology, and partially supported by Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] D. Dell'Aglio, E. Della Valle, F. van Harmelen, A. Bernstein, Stream reasoning: a survey and outlook, Data Sci. 1 (2017) 59–83.
[2] M. Tiger, F. Heintz, Stream reasoning using temporal logic and predictive probabilistic state models, in: Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning (TIME), IEEE, 2016, pp. 196–205.
[3] S. Alqahtani, S. Taylor, I. Riley, R. Gamble, R. Mailler, Predictive path planning algorithm using kalman filters and mtl robustness, in: 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), IEEE, 2018, pp. 1–7.
[4] E. Bartocci, Y. Falcone, A. Francalanza, G. Reger, Introduction to Runtime Verification, Springer International Publishing, Cham, 2018, pp. 1–33.
[5] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, S. Sankaranarayanan, Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications, in: Lectures on Runtime Verification, Springer, 2018, pp. 135–175.
[6] K.B. Lamine, F. Kabanza, Using fuzzy temporal logic for monitoring behavior-based mobile robots, in: Proc. of Int. Conf. on Robotics and Applications (IASTED), 2000, pp. 116–121.
[7] J. Pérez, J. Jiménez, A. Rabanal, A. Astarloa, J. Lázaro, Ftl-cfree: a fuzzy real-time language for runtime verification, IEEE Trans. Ind. Inform. 10 (3) (2014) 1670–1683.
[8] T. Schön, F. Gustafsson, P.-J. Nordlund, Marginalized particle filters for mixed linear/nonlinear state-space models, IEEE Trans. Signal Process. 53 (7) (2005) 2279–2289.
[9] D. Fox, J. Hightower, L. Liao, D. Schulz, G. Borriello, Bayesian filtering for location estimation, IEEE Pervasive Comput. 3 (2003) 24–33.
[10] D. Nitti, T.D. Laet, L.D. Raedt, A particle filter for hybrid relational domains, in: Proc. International Conference on Intelligent Robots and Systems (IROS), 2013.
[11] A. Kovtunova, R. Penaloza, Cutting diamonds: a temporal logic with probabilistic distributions, in: Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR), 2018.
[12] L. De Raedt, A. Kimmig, Probabilistic (logic) programming concepts, Mach. Learn. 100 (1) (2015) 5–47.
[13] P. Doherty, J. Kvarnström, F. Heintz, A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems, Auton. Agents Multi-Agent Syst. 19 (3) (2009) 332–377.
[14] R. Koymans, Specifying real-time properties with metric temporal logic, Real-Time Syst. 2 (4) (1990) 255–299.
[15] R. Alur, T. Feder, T.A. Henzinger, The benefits of relaxing punctuality, J. ACM 43 (1) (1996) 116–146.
[16] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, in: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Springer, 2004, pp. 152–166.
[17] L. Brim, P. Dluhoš, D. Šafránek, T. Vejpustek, Stl∗: extending signal temporal logic with signal-value freezing operator, Inf. Comput. 236 (2014) 52–67.
[18] C. Yoo, C. Belta, Rich time series classification using temporal logic, in: Robotics: Science and Systems (RSS), 2017.
[19] D. Sadigh, A. Kapoor, Safe control under uncertainty with probabilistic signal temporal logic, in: Proceedings of Robotics: Science and Systems, AnnArbor, Michigan, 2016.
[20] F. Bacchus, F. Kabanza, Planning for temporally extended goals, Ann. Math. Artif. Intell. 22 (1–2) (1998) 5–27.
[21] N. Markey, P. Schnoebelen, Model checking a path, in: International Conference on Concurrency Theory, Springer, 2003, pp. 251–265.
[22] D. Basin, B.N. Bhatt, D. Traytel, Almost event-rate independent monitoring of metric temporal logic, in: Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), 2017, pp. 94–112.
[23] R. Serra, D. Arzelier, M. Joldes, J.-B. Lasserre, A. Rondepierre, B. Salvy, Fast and accurate computation of orbital collision probability for short-term encounters, J. Guid. Control Dyn. (2016) 1009–1021.
[24] G.M. Santipantakis, A. Vlachou, C. Doulkeridis, A. Artikis, I. Kontopoulos, G.A. Vouros, A stream reasoning system for maritime monitoring, in: 25th International Symposium on Temporal Representation and Reasoning (TIME), 2018.
[25] E. Novak, Some results on the complexity of numerical integration, in: Springer Proceedings in Mathematics and Statistics, vol. 163, 163rd edition, 2016, pp. 161–183.
[26] S. Thrun, Particle filters in robotics, in: Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence (UAI), Morgan Kaufmann Publishers Inc., 2002, pp. 511–518.
[27] H. Nyquist, Certain topics in telegraph transmission theory, Trans. Am. Inst. Electr. Eng. 47 (2) (1928) 617–644.
[28] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, F. Heintz, Receding-horizon lattice-based motion planning with dynamic obstacle avoidance, in: 2018 IEEE Conference on Decision and Control (CDC), 2018, pp. 4467–4474.
[29] M. Kamel, T. Stastny, K. Alexis, R. Siegwart, Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system, in: Robot Operating System (ROS), Springer, 2017, pp. 3–39.