# Stream Reasoning using Temporal Logic and Predictive Probabilistic State Models

Mattias Tiger
*Department of Computer and Information Science*
*Linköping University, Sweden*
*Email: mattias.tiger@liu.se*

Fredrik Heintz
*Department of Computer and Information Science*
*Linköping University, Sweden*
*Email: fredrik.heintz@liu.se*

***Abstract*—Integrating logical and probabilistic reasoning and integrating reasoning over observations and predictions are two important challenges in AI. In this paper we propose P-MTL as an extension to Metric Temporal Logic supporting temporal logical reasoning over probabilistic and predicted states. The contributions are (1) reasoning over uncertain states at single time points, (2) reasoning over uncertain states between time points, (3) reasoning over uncertain predictions of future and past states and (4) a computational environment formalism that ground the uncertainty in observations of the physical world. Concrete robot soccer examples are given.**

## I. INTRODUCTION

Autonomous systems such as robots need to do both logic and probabilistic inference over observations and other uncertain information to reason about itself and the world. The information available is fundamentally incremental in nature. A flow of incrementally available information is called a *stream* of information. To draw logical conclusions and react to new situations with minimal delays incremental reasoning over streams is necessary. This is *stream reasoning*.

Logical inference is mainly used to reason about high-level facts in for example planning and execution monitoring. Logics are useful since they can be used to formalize abstract problem domains, allow questions in human language and common sense reasoning to be easily expressed. A central problem is however the disconnect between lower-level probabilistic inference over sensor data and higher-level logical reasoning, the so called *sense-reasoning gap* [1].

Good state estimation and prediction quality is a necessity for a good world model, anticipatory behavior and safe operations of an autonomous system. However, estimations of the present and predictions about the future (and the past) will be uncertain. Probabilistic inference is needed to estimate relevant states, to make predictions about them and to draw probabilistic conclusions.

Logic-based stream reasoning is about incrementally evaluating temporal logical formulas over streams of information, which differs substantially from theorem proving. The focus of this paper is on such incremental reasoning over incrementally available data (*stream reasoning*). *Metric Temporal Logic* (MTL) [2] is one example of a logic that has been used successfully in the robotics domain for execution monitoring [3]. Statements in MTL are efficiently and incrementally evaluated using *progression* [4].

Logic-based stream reasoning has previously only been done over observations. To also reason over predictions we present *Predictive MTL (P-MTL),* an extension of MTL supporting reasoning over both stochastic states and stochastic predictions of states.

The contributions are (1) reasoning over uncertain states at single time points, (2) reasoning over uncertain states between time points, (3) reasoning over uncertain predictions of future and past states and (4) a computational environment formalism that ground the uncertainty in observations of the physical world. The grounding of probabilistic terms is important for the reasoning to be meaningful. We therefore present a full theoretical chain from sensor observations to answers to logical queries and provide practical examples of real systems from the robot soccer domain.

To achieve (1)-(4) we considers the logic, the probabilistic models and inference as well as the computational environment for grounding the probabilistic and logical reasoning. P-MTL includes both logical and probabilistic terms clearly separating the logical and probabilistic inferences.

A full unification of logic and probability theory is conceptually difficult for several reasons. A complete axiomatization of first order logic with real valued probability values is not even possible [5]. Despite the many difficulties there are several impressive approaches to probabilistic logics which show practical usefulness in areas such as robotics [6][7]. One significant drawback is however that the inference methods rely on general purpose statistical sampling methods, such as MCMC, which are too slow for many practical usages with large models or with requirements on low latency. Many common probabilistic inference problems (and models) used in practice in the field of probabilistic robotics have specialized and highly optimized inference methods. This is something we want to take advantage of.

The work in this paper differs from work on probabilistic logics since P-MTL is not a probabilistic logic since formulas are either true or false, while in probabilistic logics truth values are replaced with probabilities. The semantics of P-MTL is not directly concerned with probabilistic inference, instead it handles probabilistic statements through the grounding in the computational environment. This al-

lows P-MTL to remain a first order logic with efficient inference schemes for incremental reasoning without adding limitations on what probabilistic inference can be done and used as terms in P-MTL. Existing methods for probabilistic inference can as a consequence be used with P-MTL.

The paper is structured as follows. Background to stream reasoning is provided in Section II followed by an introduction to predictive stochastic stream reasoning in Section III. Section IV presents the syntax of P-MTL. In Section V the computational environment for grounding probabilistic terms is described. The P-MTL semantics is presented in Section VI. Sections VII and VIII provide concrete case studies. The paper is concluded with a discussion on related work in Section IX and a conclusion in Section X.

## II. Stream Reasoning With MTL

We consider a stream to be a potentially infinite sequence of time stamped states with a strict ordering on the time stamps. States in the stream arrive incrementally. The time of a state is assumed to be strictly larger than the time of the previous state. It is important to note that the stream reasoning which is considered in this paper determines if statements are true or false over the entire stream.


Figure 1. A stream is a sequence of time stamped states

Stream reasoning with MTL is done in two parts. First order logic is used to reason about single states. The second part is the metric temporal operators $\Box$ (Always), $\Diamond$ (Eventually) and $\bigcirc$ (Next) which allow statements across the state sequence to be made, i.e. formulas that relate multiple states. The temporal operators can refer to the entire stream (unbounded) or be constrained to a time interval (bounded). For example, the query *It is always the case that if the altitude of uav is below 10 meters then it will be above 10 meters within 5 seconds.* can be written as

$$\Box\big((\text{Altitude}[\textbf{uav}] < 10) \rightarrow \Diamond_{[0,5]}(\text{Altitude}[\textbf{uav}] > 10)\big)$$

where **uav** is an object and Altitude is a feature which is a property or relation whose value may change over time. The interpretation of the above formula is as follows. If an arriving state $S_{t_k}$ contains an altitude of uav that is less than 10 meters, then a state $S_{t_n}$ must arrive in which the altitude of uav is higher than 10 meters and where $t_n - t_k \leq 5$ seconds. The formula will never become true unless the stream is finite, but it can become false.

The semantics of these formulas are defined over infinite state sequences. To make metric temporal logic suitable for stream reasoning, formulas are incrementally evaluated using progression over a stream of timed states [3]. The result of progressing a formula through the first state in a stream is a new formula that holds in the remainder of the state stream if and only if the original formula holds in the complete state stream. If progression returns true (false), the entire formula must be true (false), regardless of future states. Even though the size of a progressed formula may grow exponentially in the worst case, it is always possible to use bounded intervals to limit the growth.

## III. Predictive Stochastic Stream Reasoning

In typical robotics applications features are estimated from noisy numeric observations. The noise is often handled by treating the estimated feature as a stochastic variable with an associated probability distribution. In logic-based stream reasoning these stochastic variables must be reduced to numerical values before they can be used in the logical reasoning. It is common to use the mean or first moment, throwing away information including the uncertainty.

Reasoning over uncertain states is however useful since it takes into account the uncertainty of observed values through sensors and allows the reasoning to be constrained by how certain a system currently is about some state of the world (something which might change over time). The formula

$$\Box\big(Pr((\text{Altitude}[\textbf{roof}] - \text{Altitude}[\textbf{uav}]) > 2m) \geq 0.99\big)$$

can guard a UAV (Unmanned Aerial Vehicle) from crossing above a roof top if it is not sufficiently certain (here with 99% probability) that it is higher up than the roof. A comparison of the raw observation values of UAV and roof altitudes provides no such guarantee for uncertain measurements. Reasoning over the difference in uncertainty between time points allows an autonomous system to further monitor the quality of perception of various properties of the world and how it varies over time.

The stochastic feature at a specific time point can be estimated in isolation from past observations, treating them as independent across time. A stochastic time series model (temporal model for short) over features however can relate feature observations over time. It provides the possibility to also consider previous observations to improve the estimate. A temporal model is also necessary to estimate features that are only indirectly observable, such as estimating the velocity from position observations.

A temporal model does not only relate estimates at different observation time points but also to time points for which there are no observations. This allow interpolation and extrapolation of stochastic features across time (referred to as predictions in this paper).

In total there are three variants of the same feature, such as the altitude of a UAV. First there are raw numerical observations, secondly there are stochastic estimates and thirdly there are stochastic predictions based on several observations. The first is non-stochastic while the other two are stochastic. The time of an estimate corresponds to the

time of the observation, while predictions can be made for any time including the future or the never observed past.

To make these variants explicit P-MTL includes stochastic features terms representing estimates and predictions. Regular features represent observed features (which are non-stochastic). This means that the state for each time point is extended with estimations and predictions. The stream can be considered 2-dimensional with one dimension for stream time and one for prediction time, as is illustrated in Figure 2.

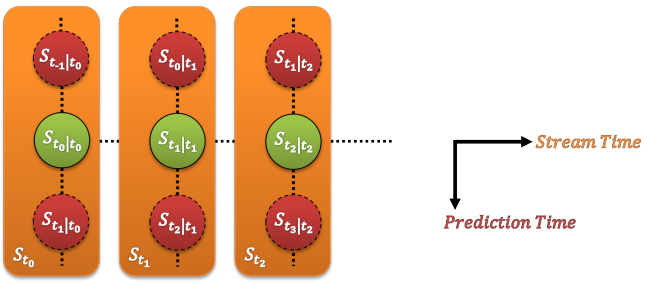


Figure 2. States of the stream state sequence in orange (which include observations), estimated states in green and predicted states in red.

Some interesting queries require the expressiveness to talk about predictions made from other states than the current state. Figure 3 show a rearrangement of Figure 2 where the prediction time now is aligned with the stream time. For example, consider the comparison between predicted state from time 0 about time 2 ($S_{t_2|t_0}$) and the observed state (or state estimate) at time 2 ($S_{t_2|t_2}$). This requires the predictive temporal operator to operate over both stream time and prediction time.

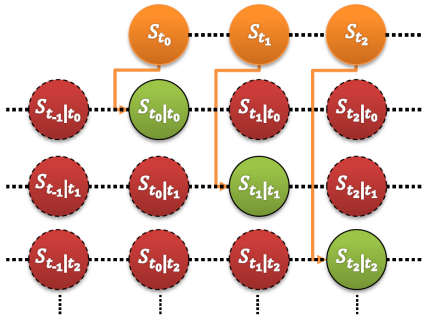

Figure 3. Observed state sequence in orange (first row). Each row below the first is the predicted state sequence of each observed state. Predicted state is in red and estimated states (special case of prediction) are in green.

The predictive operator $\bullet_{t'|t}$ is introduced to refer to observed, estimated and predicted features. Estimated and predicted features are stochastic variables and observations are non-stochastic variables. The operator works as follows:

| | |
|---|---|
| Observed feature value: | $\bullet_t$Altitude[**uav**] |
| Estimated feature value: | $\bullet_{t\|t}$Altitude[**uav**] |
| Predicted feature value: | $\bullet_{t'\|t}$Altitude[**uav**] |

where $t$ is observation time (stream time) and $t'$ is the prediction time. The interpretation of $\bullet_t F$ is that it is the observed value of feature $F$ at time $t$. The interpretation of $\bullet_{t|t} F$ is that it is the estimated value of feature $F$ at time $t$, which is a prediction made at time $t$ about the value of $F$ at time $t$. The interpretation of $\bullet_{t'|t} F$ is that it is a prediction made at time $t$ about the value of feature $F$ at time $t'$. $t'$ may be larger than $t$ (prediction about the future) or smaller than $t$ (prediction about the past). Time is relative.

This allows P-MTL to compare feature values from different time points, which is not possible in MTL. It is important to note that we do not present any mechanism to distinguish if observations from two different time points are different observations with the same value or simply the same observation. This means that it is possible to query if there has been no changes in observation values of a feature in the past 5 seconds, but not if it is due to the lack of new observations (i.e. a constant inertia assumption) or not.

Reasoning over uncertain predictions allows an autonomous system to monitor the prediction quality and consistency of different probabilistic models used to estimate the state of the world, both current, past and future. It is important for autonomous systems operating in unstructured environments to be able to monitor if the model of the world (and the future expectations/predictions of the world) becomes inconsistent with observations. If crucial parts of the world state can no longer be predicted sufficiently well then more sophisticated models might have to be used, learning needs to be prioritized or maybe the autonomous system needs to enter a safety mode such as making an emergency stop.

This is especially important if the risk to hurt humans or to damage expensive property becomes unacceptable. To recognize such important situations it is necessary to be able to reason over uncertain states and predictions with the stochastic models that are used for probabilistic state estimation and prediction in robotics.

For example, consider the UAV predicting its position 3 seconds into the future when it is about to cross over a roof top. It is reasonable to require the UAV's motion to satisfy safety constraints in terms of acceptable risk to damage property and hurt humans according to policy and legislation. An example could be a requirement that it always have to be the case that **a)** the UAV must know its own position to be within a 1m radius sphere with 99% probability, **b)** the certainty of predicting the position of the UAV 3 seconds from now must be so high that it is within a 1m radius sphere with 95% probability and **c)** the quality of the prediction in b must be so high that it matches with the observed position in 3 seconds with at least 50%. If **a-c** are satisfied then the UAV is allowed to perform motions, otherwise it should halt its operations and enter a safe mode.

The requirements **a-c** can be stated as

$$\Box\big(Pr(insideRel(\bullet_{0|0}\text{Pos}[\textbf{uav}], \textbf{Sphere1.0m})) > 0.99 \\ \wedge Pr(insideRel(\bullet_{3|0}\text{Pos}[\textbf{uav}], \textbf{Sphere1.0m})) > 0.95 \\ \wedge \bigcirc_3\big(similarity(\bullet_{0|0}\text{Pos}[\textbf{uav}], \bullet_{0|-3}\text{Pos}[\textbf{uav}]) > 0.5\big)\big)$$

where the third conjunctive term means that *after 3 seconds the estimated position now and the predicted position about now from 3 seconds ago should have a similarity of at least 50%*. Sphere1.0m is the constraint of a sphere with radius 1.0 meters. The similarity between the stochastic variables (estimate and prediction) can for example be calculated using the Hellinger distance between their probability distributions.

## IV. Syntax of P-MTL

The syntax of P-MTL is defined given a tuple $S = \langle \mathcal{P}, \mathcal{FU}, \mathcal{C}, \mathcal{V}, \mathcal{F}, \mathcal{PR} \rangle$, where $\mathcal{P}$ is a set of predicate symbols, $\mathcal{FU}$ a set of function symbols, $\mathcal{C}$ a set of constant symbols, $\mathcal{V}$ a set of variable symbols, $\mathcal{F}$ a set of feature symbols, and $\mathcal{PR}$ a set of probability reasoner symbols. The set of well-formed *probabilistic terms* $\mathcal{PT}_S$ is recursively defined from $\bar{f} \in \mathcal{F}$, $const \in \mathcal{C}$, $t_1, t_2 \in \mathbb{N}$ and probabilistic terms $\tau_p$ as

$$\bar{f}[const] \mid \bullet_{t_1}\, \bar{f}[const] \mid \bullet_{t_1|t_2}\, \bar{f}[const]$$

The set of well-formed *logical terms* $\mathcal{LT}_S$ is recursively defined from $const \in \mathcal{C}$, $f \in \mathcal{FU}$, $g \in \mathcal{PR}$, $c_i \in \mathcal{C}$, probabilistic terms $\tau_p$ and logical terms $\tau_i$ as

$$const \mid f(\tau_1, \ldots, \tau_n) \mid Pr(g(\tau_p, c_1, \ldots, c_n))$$

The set of well-formed formulas $\mathcal{L}_S$ is recursively defined from $P \in \mathcal{P}$, $x \in \mathcal{V}$, $t_1, t_2 \in \mathbb{N}$, logical terms $\tau_i$, probabilistic terms $\tau_p$ and well-formed formulas $\alpha$, $\beta$ as

$$P(\tau_1, \ldots, \tau_n) \mid \neg\alpha \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid \alpha \rightarrow \beta \mid \\ \forall x[\alpha] \mid \exists x[\alpha] \mid \bigcirc_{t_1} \alpha \mid \Box_{[t_1,t_2]}\, \alpha \mid \Box\, \alpha \mid \Diamond_{[t_1,t_2]}\, \alpha \mid \Diamond\, \alpha$$

where $P(\tau_1, \ldots, \tau_n)$ is a predicate over terms $\tau_1, \ldots, \tau_n$, $\neg\alpha$ is the *negation* of $\alpha$, $\alpha \wedge \beta$ is the *conjunction* of $\alpha$ and $\beta$, $\alpha \vee \beta$ is the *disjunction* of $\alpha$ and $\beta$, $\alpha \rightarrow \beta$ is the *implication* of $\alpha$ to $\beta$. $\bigcirc_{t_1}\alpha$ is the *next* operator where $\alpha$ is true after $t_1$ time points from now, $\Box_{[t_1,t_2]}\alpha$ is the *always* operator where $\alpha$ is true at all times in the interval between time point $t_1$ and $t_2$ from now, $\Diamond_{[t_1,t_2]}\alpha$ is the *eventually* operator where $\alpha$ is true for at least one time point in the interval between $t_1$ and $t_2$ time points from now.

## V. Computational Environment

To interpret the probabilistic terms in P-MTL we need to consider the internals of a robotics system operating in an uncertain world observed through inexact sensors. The running example will be a Nao robot playing soccer.

P-MTL supports statements about the state of the world. These can be used both for decision making and monitoring of the execution of the robot. For example, the ball needs to be located and the robot wants to know where it is likely to end up in the near future when it decides what to do. The robot needs to stay localized on the soccer field and it must make sure to not risk walking outside of the soccer field or collide with other robots. We further want the robot to behave anticipatory, weight expected risk and make decisions based on predictions, rather than reactively when it might be too late to act with sufficiently low risk.

The domain consists of a set of object sorts such as robot and ball. There is a set of object instances such as the ball and the robot itself, its team members and those of the opposite team. These objects have various features so there is a set of feature sorts such as position, velocity, acceleration and rotation in 2D. The object instances of a specific object sort has a set of features of the same sorts as the other object instances in the object sort. Robots have for example a rotation feature whereas balls do not.

The properties of objects and the world are assumed to be changing over time and to not be directly observable, except indirectly through noisy sensors. A feature value is a raw non-stochastic sensor observation of the property of a object at a specific time point.

Some feature sorts may further depend on other feature sorts, such as velocity being the first derivative of position, which can be used for estimating the velocity of an object when only its positions is observable (or vice versa). We therefore use multivariate stochastic variables, stochastic features, to represent noisy feature instances (or constellations of features). We have a set of stochastic feature sorts where each element represents a single feature sort or a vector of feature sorts (where the same feature type is allowed to occur more than once). We also have a set of stochastic variable instances where each stochastic instance is of a specific stochastic variable sort, representing a vector of feature instances with matching feature type as the stochastic variable sort described and have a specific time point.

For example, the time series of the position of the ball is represented as an ordered set of stochastic variables (ordered over the unique time points of each stochastic variable in the set) which are all of the same sort (position feature) and associated with the ball object instance. The reason for modeling features with stochastic variables is to represent the uncertainty of the feature values, as each stochastic variable is distributed according to a probability distribution. We have a set of probability distribution sorts and a probability distribution instance has a specific set of parameters which fully describes the probability distribution (e.g. the mean and variance of a Gaussian).

By utilizing assumptions about how we expect features to change over time from for example physics (motion models) we can calculate a better estimate of the feature at the current time point by taking some extent of the history into consideration. An *estimator* takes the non-stochastic observation of the measured ball position of the current

time point together with the previous temporal model of the ball position and produces a new temporal model of the ball position for the current time point. The same assumptions about expected feature-changes over time can be used to make predictions about future or past time points. A *predictor* takes a temporal model of the ball position at any given time point together with another time point and makes a prediction about the ball position at the other time point. The ball prediction is a stochastic variable instance of the other time point. If the other time point is of the same time as the temporal model, then the predicted stochastic variable is the best current estimate of the position of the ball, given the temporal model in question.

Estimated and predicted stochastic variables can further be used to answer probability queries, such as how probable it is that the predicted ball (position wise) after 3 seconds is inside the boundaries of the soccer field. A *probability reasoner* instance takes the predicted stochastic variable representing the ball position 3 seconds into the future and computes a probability between 0 and 1 of the query. A probability query can be expressed as an integral of a restriction (a constraint) in the state space (feature space) of a stochastic variable over this stochastic variable's probability density function. The produced probability is in turn used by the *progressor* as a substitute number for the probabilistic sub-query $Pr\big(inside(\bullet_{3|0}\text{Pos}[\textbf{ball}], \textbf{soccerfield})\big)$.

The computational environment is made up of a number of computation units which refine data received from sensors to probability values as required by the progressor. What probability function expressions are valid depends on which can be grounded in the computational environment and thereby calculated. The different steps (Figure 4) in the computational environment are

1) Observe: $World \rightarrow \text{Pos}[\textbf{ball}]_t^{obs}$
2) Update: $\text{Pos}[\textbf{ball}]_t^{obs} \rightarrow \mathcal{TM}_t$
3) Estimate: $\mathcal{TM}_t \rightarrow \text{Pos}[\textbf{ball}]_{t|t}^{est}$
4) Predict: $\mathcal{TM}_t \rightarrow \text{Pos}[\textbf{ball}]_{t'|t}^{pred}$
5) Reason(probabilistic): $inside(\text{Pos}[\textbf{ball}]_{t'|t}^{pred}, \textbf{areaA}) \rightarrow [0, 1]$
6) Reason(logic): $Pr(inside(\bullet_{t'|t}\text{Pos}[\textbf{ball}], \textbf{areaA})) \geq 0.95 \rightarrow T/F$

### A. Formalization

The computational environment $\mathcal{E}$ is defined as a tuple,

$$\mathcal{E} = \langle T, \mathcal{O}, \mathcal{F}, \bar{\mathcal{F}}, \mathcal{X}, \mathcal{D}, \mathcal{T}, \mathcal{P} \rangle, \tag{1}$$

where $T$ is a finite ordered set of time points, $\mathcal{O}$ is a set of object sorts, $\mathcal{F}$ is a set of feature sorts, $\bar{\mathcal{F}}$ is a set of feature vector sorts, $\mathcal{X}$ is a set of stochastic feature sorts, $\mathcal{D}$ is a set of probability distribution sorts, $\mathcal{T}$ is a set of stochastic time series model sorts, $\mathcal{P}$ is a set of probability reasoner sorts.

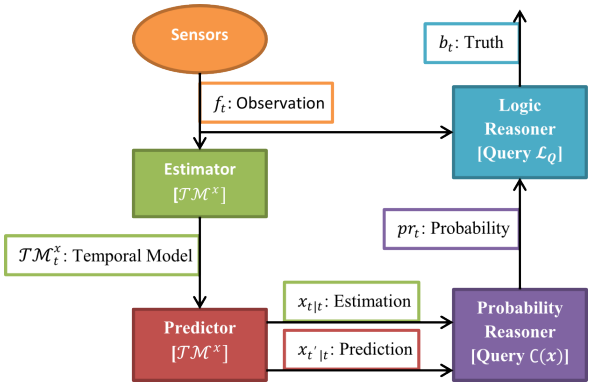


Figure 4. Schematic overview of a chain from observations to truth values.

Each of the sets $\mathcal{O}, \mathcal{F}, \bar{\mathcal{F}}, \mathcal{X}, \mathcal{D}, \mathcal{T}, \mathcal{P}$ is a set of sets, where the elements of a sort set is the instances of that sort. Let $inst(A)$ denote the set of instances of all sorts of respective $A \in \{\mathcal{O}, \mathcal{F}, \bar{\mathcal{F}}, \mathcal{X}, \mathcal{D}, \mathcal{T}, \mathcal{P}\}$. E.g. $inst(\mathcal{O})$ is the set of all object instances of all object sorts

$$inst(\mathcal{O}) = \{instance : \exists sort \in \mathcal{O}\ s.t.\ instance \in sort\}. \tag{2}$$

**Definition** Time: $T$
A time point $t \in T \subset \mathbb{R}$ is a real value, reflecting physical time of the system in seconds from some fixed starting point. The set of time points $T$ is finite and contains all time points used by the system.

**Definition** Object: $\mathcal{O}$
An object can be anything. There are sorts of objects $\mathcal{O}_S \in \mathcal{O}$ which differ by what features they have. All object individuals of a specific sort $o \in \mathcal{O}_S$ has the same features. The set of features an object sort $\mathcal{O}_S \in \mathcal{O}$ have is a feature vector sort $features(\mathcal{O}_S) \in S_{\bar{\mathcal{F}}}$. For example **ball** $\in$ Balls $\in \mathcal{O}$.

**Definition** Feature: $\mathcal{F}$
A feature is a property of an object such as its position. There are different sorts of features $\mathcal{F}_S \in \mathcal{F}$. All feature individuals $f[o]_t \in \mathbb{R}$ belong to a single object $o \in inst(\mathcal{O})$ and a time point $t \in$ T. For example *the velocity of the ball at time point $t$ is 1.0 m/s*.

$$\mathcal{F}_S = \text{Velocity}, \qquad \text{Velocity}[\textbf{ball}]_t = 1.0 \tag{3}$$

where $\text{Velocity}[\textbf{ball}]_t \in \mathcal{F}_S \in \mathcal{F}$ and $t \in$ T.

**Definition** Feature Vector: $\bar{\mathcal{F}}$
A feature vector is an ordered set of features. There are different sorts of feature vectors $\bar{\mathcal{F}}_S \in \bar{\mathcal{F}}$ and they differ by the order and type of the features. A feature vector individual $\bar{f} \in inst(\bar{\mathcal{F}})$ is an ordered set of feature individuals $\bar{f} = \{f_{t_n}^n\}_{n=1}^N$ where $t_n \in$ T. $\bar{f}_t$ is used if $t_n$ is equal for all $n$. Example: *the vector of position and velocity*

*of the ball at time point t*

$$\bar{\mathcal{F}}_S = [\text{Pos}, \text{Velocity}] \tag{4}$$

$$\bar{f}_t = [\text{Pos}[\textbf{ball}]_t, \text{Velocity}[\textbf{ball}]_t] \tag{5}$$

where $\bar{f}_t \in \bar{\mathcal{F}}_S \in \bar{\mathcal{F}}$, **ball** $\in inst(\mathcal{O})$ and Pos,Velocity $\in \mathcal{F}_S$.

**Definition** Probability Distribution: $\mathcal{D}$
A probability distribution assigns a probability to each measurable subset of the possible values of a stochastic variable. In the scope of this article we restrict the examples to continuous probability distributions which are specified by their probability density function (pdf). This can be extended to other probability distribution classes such as discrete probability distributions specified by their probability mass functions. Each probability distribution sort $\mathcal{D}_S \in \mathcal{D}$ has a different probability density function that assigns probability over the value of a specific stochastic feature sort. The pdf of a sort $\mathcal{D}_S^{pdf} = f_{\mathrm{x}}(x, \theta)$ has free parameters $\theta \in \mathbb{R}^N$ with $\mathrm{x} \in \mathcal{X}$ and the range of $x$ is over the domain of x. The individuals of each sort $d \in \mathcal{D}_S$ differ by the realization of $\theta$. For example the Gaussian (Normal) distribution sort $\mathcal{D}_S$ and an individual $d$:

$$\mathcal{D}_S = \mathcal{N} \in \mathcal{D} \tag{6}$$

$$\mathcal{D}_S^{pdf} = f_{\mathrm{x}}(x, \theta) = \mathcal{N}(x; \theta) = \mathcal{N}(x; \mu, \Sigma) \tag{7}$$

$$d \in \mathcal{D}_S,\ d_\theta = [0, 0.2],\ d^{pdf} = f_{\mathrm{x}}(x) = \mathcal{N}(x; 0, 0.2) \tag{8}$$

where $\mathcal{N}(\mu, \Sigma)$ is a Gaussian distribution with mean vector $\mu$ and covariance matrix $\Sigma$, and $\mathcal{N}(x; \mu, \Sigma)$ is its pdf

$$\mathcal{N}(x;\ \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{M}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}[x-\mu]\Sigma^{-1}[x-\mu]^T} \tag{9}$$

where $M$ is the dimension of $x$.

**Definition** Stochastic Feature: $\mathcal{X}$
A stochastic feature is a multivariate stochastic variable of a feature vector sort with a joint probability distribution over all features in the feature vector. Different sorts $\mathcal{X}_S \in \mathcal{X}$ of stochastic features differ by their feature vector sort and probability distribution sort. A stochastic feature individual $\mathrm{x}_{t_1|t_2} \in inst(\mathcal{X})$ consists of the object instances of each feature in the feature vector, a probability distribution instance and two time points $t_1, t_2 \in \mathrm{T}$. $t_1$ is prediction time and $t_2$ is stream time. Example: *the velocity of the ball at time point t is a stochastic feature that is Gaussian distributed with mean velocity 1.0 m/s and a standard deviation of 0.1 m/s.*

$$\mathcal{X}_S = \langle \text{Velocity}, \mathcal{N} \rangle \tag{10}$$

$$\mathrm{x}_{t|t} = \text{Velocity}[\textbf{ball}]_{t|t} \sim \mathcal{N}(1.0, 0.1^2) \tag{11}$$

$$= \langle t, t, \text{Velocity}, \textbf{ball}, \mathcal{N}(1.0, 0.1^2) \rangle \tag{12}$$

where $\mathrm{x}_{t|t} \in \mathcal{X}_S \in \mathcal{X}$, $t \in \mathrm{T}$, Velocity $\in \bar{\mathcal{F}}$, $\mathcal{N}(1.0, 0.1^2) \in \mathcal{N} \in \mathcal{D}$, **ball** $\in inst(\mathcal{O})$. A feature vector $\bar{f}$ can be converted to a stochastic feature x by setting x to be distributed according to a dirac distribution with its parameter $\theta$ set to the values of the feature vector instance, $\mathrm{x} \sim \delta_{\bar{f}}$, where the pdf is $f_{\mathrm{x}}(x) = \delta_{\bar{f}}(x)$ and

$$\int_{\mathrm{C}(x)} f_{\mathrm{x}}(x) dx = \begin{cases} 1.0 & \text{if } \bar{f} \in \mathrm{C}(\mathrm{x}) \\ 0.0 & \text{if } \bar{f} \notin \mathrm{C}(\mathrm{x}) \end{cases} \tag{13}$$

where $\mathrm{C}(x)$ is a constraint on the domain of x.

**Definition** Stochastic Time Series Model: $\mathcal{T}$
A stochastic time series model (temporal model for short) is a model that relates stochastic features over time taking into consideration uncertainty of the observations and of the generating process. There are different sorts of temporal models, $\mathcal{T}_S \in \mathcal{T}$. The sorts differ by the feature vector sort that is used as input (observations) and the stochastic feature sort that is estimated (predictions). A temporal model instance $\mathcal{TM}_t \in inst(\mathcal{T})$ may contain a history of observations and represents the best model instance for the estimated stochastic feature at a specific time point $t \in \mathrm{T}$. Each sort has an estimator function ${\mathcal{T}_S}^{est}$ and a predictor function ${\mathcal{T}_S}^{pred}$. The estimator takes a temporal model instance of a previous time point $t'$ together with a feature vector individual of a later time point $t$ (an observation) and produces a new temporal model instance of time point $t$. The predictor takes a temporal model instance of a specific time point $t$ together with a predicted time point $t'$ and produces a prediction of the stochastic feature at time point $t'$. If $t = t'$ the prediction is the estimate of the stochastic feature at time $t$.

$$\mathcal{TM}_t = {\mathcal{T}_S}^{est}(\mathcal{TM}_{t-\tau}, \bar{f}_t^{obs}),\ \ \tau > 0 \tag{14}$$

$$\mathrm{x}_{t'|t}^{pred} = {\mathcal{T}_S}^{pred}(\mathcal{TM}_t, t') \tag{15}$$

$$\mathrm{x}_{t|t}^{est} = {\mathcal{T}_S}^{pred}(\mathcal{TM}_t, t) \tag{16}$$

where $\mathcal{TM}_{t-\tau}, \mathcal{TM}_t \in \mathcal{T}_S \in \mathcal{T}$.

**Definition** Probability Reasoner: $\mathcal{P}$
A probability reasoner (PR) evaluates a probabilistic query that is of the form of *What is the probability that* x *is constrained according to* $\mathrm{C}(x \in domain(\mathrm{x}))$ *given that* x *have pdf* $f_{\mathrm{x}}$*?*. The different sorts of PR differ by what stochastic feature type taken as input together with what constraint $\mathrm{C}(x)$ is used. Given a continuous stochastic variable $\mathrm{x} \in S_{\mathcal{X}}$ with pdf $f_{\mathrm{x}}(x)$ and constraint $\mathrm{C}(x)$ the probability is calculated by integrating $\mathrm{C}(x)$ over $f_{\mathrm{x}}(x)$

$$P(\mathrm{C}(\mathrm{x})) = \int_{\mathrm{C}(x)} f_{\mathrm{x}}(x) dx \tag{17}$$

## VI. Semantics of P-MTL

The semantics of P-MTL is defined over the model

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{E}, \mathcal{I} \rangle \tag{18}$$

where $\mathcal{S}$ is a set of symbols as defined in section IV, $\mathcal{E}$ is a computational environment as defined in equation 1 and $\mathcal{I}$ is an interpretation. The interpretation maps predicate symbols to relations $\mathcal{O}^N$ where $N$ is the arity of the predicate, function symbols to functions $\mathcal{O}^N \mapsto \mathcal{O}$ where $N$ is the artify of the function, constant symbols to objects in $\mathcal{O}$, feature symbols to features in $\mathcal{F}$, probability reasoner symbols to probability reasoners in $\mathcal{PR}$.

A formula $\alpha$ is true in $\mathcal{M}$ at $t$ iff $\mathcal{M}, t \models \alpha$ defined as:

$$\begin{aligned}
\mathcal{M}, t \models P(\tau_1, \ldots, \tau_n) \text{ iff } &\langle eval(\mathcal{M}, t, \tau_1), \ldots, \\
&eval(\mathcal{M}, t, \tau_n)\rangle \in I(P) \\
\mathcal{M}, t \models \neg\alpha \text{ iff } &\mathcal{M}, t \not\models \alpha \\
\mathcal{M}, t \models \alpha \wedge \beta \text{ iff } &\mathcal{M}, t \models \alpha \text{ and } \mathcal{M}, t \models \beta \\
\mathcal{M}, t \models \alpha \vee \beta \text{ iff } &\mathcal{M}, t \models \alpha \text{ or } \mathcal{M}, t \models \beta \\
\mathcal{M}, t \models \alpha \rightarrow \beta \text{ iff } &\mathcal{M}, t \not\models \alpha \text{ or } \mathcal{M}, t \models \beta \\
\mathcal{M}, t \models \exists x[\alpha] \text{ iff } &\exists o \in \mathcal{O}\ \mathcal{M}, t \models \alpha[x/o] \\
\mathcal{M}, t \models \forall x[\alpha] \text{ iff } &\forall o \in \mathcal{O}\ \mathcal{M}, t \models \alpha[x/o] \\
\mathcal{M}, t \models \bigcirc_{t'}(\alpha) \text{ iff } &\mathcal{M}, t + t' \models \alpha \\
\mathcal{M}, t \models \Box_{[t_1,t_2]}(\alpha) \text{ iff } &\forall t_1 \leq t' \leq t_2\ \mathcal{M}, t + t' \models \alpha \\
\mathcal{M}, t \models \Box(\alpha) \text{ iff } &\mathcal{M}, t \models \Box_{[0,\infty]}\alpha \\
\mathcal{M}, t \models \Diamond_{[t_1,t_2]}(\alpha) \text{ iff } &\exists t_1 \leq t' \leq t_2\ \mathcal{M}, t + t' \models \alpha \\
\mathcal{M}, t \models \Diamond(\alpha) \text{ iff } &\mathcal{M}, t \models \Diamond_{[0,\infty]}\alpha
\end{aligned}$$

The function $eval(\mathcal{M}, t, p)$ recursively evaluates logical and probabilistic terms according to:

$$\begin{aligned}
eval(\mathcal{M}, t, const) &\equiv I(const) \\
eval(\mathcal{M}, t, f(\tau_1, \ldots, \tau_n)) &\equiv I(f)(eval(\mathcal{M}, t, \tau_1), \ldots, \\
&\qquad eval(\mathcal{M}, t, \tau_n)) \\
eval(\mathcal{M}, t, \bar{f}[const]) &\equiv I(\bar{f})[I(const)]_t \\
eval(\mathcal{M}, t, \bullet_{t_1}\bar{f}[const]) &\equiv eval(\mathcal{M}, t + t_1, \bar{f}[const]) \\
eval(\mathcal{M}, t, \bullet_{t_1|t_2}\bar{f}[const]) &\equiv \mathcal{T}_S^{pred}\big(\mathcal{TM}_{t+t_2}^{\bar{f}[const]}, t + t_1\big) \\
eval(\mathcal{M}, t, Pr(g(\tau_p, c_1, .., c_m))) &\equiv \int_{\mathrm{C}(x)} f_{\mathrm{x}}(x)dx
\end{aligned}$$

where $\mathcal{T}_S^{pred}$ is the predictor and $\mathcal{TM}$ is the temporal model associated with the feature $\bar{f}[const]$, $f_{\mathrm{x}}$ is the probability density function associated with the stochastic feature $x = eval(M, t, \tau_p)$ and $C(x) = C(x, c_1, .., c_m)$ is the probabilistic reasoner associated with the symbol $g$. This provides the bridging between the logical reasoning and the probabilistic reasoning.

Since the probabilistic terms can be computed independently for each time-point P-MTL can be used for stream reasoning in the same way as MTL through progression [4].

## VII. Example Scenario 1 – Ball Tracking

For a soccer playing robot it is important to know where the ball is and where it is likely to end up in the future. This is especially true when the ball is moving fast since the robots are substantially slower. Example queries are *"Is the probability of the ball staying inside the soccer field within the next 3 seconds always at least 95%?"*

$$\Box\big(\bigwedge_{t=0}^{3} Pr(inside(\bullet_{t|0}\text{Pos[}\textbf{ball}\text{]}, \textbf{soccerField})) \geq 0.95\big) \tag{19}$$

and *"Is the probability always less than 95% that the ball is getting as close as 1m or less to the robot within the next 3 seconds?"*

$$\Box\big(Pr(insideRel(\bullet_{3|0}\text{ Pos[}\textbf{nao}\text{]}, \bullet_{3|0}\text{Pos[}\textbf{ball}\text{]}, \textbf{circle1.0m})) < 0.95\big) \tag{20}$$

where the latter example can be used as a trigger when evaluated to false for a robot to advance towards a ball only when sufficiently close.

We assume that the position of the ball is passively observed by a Nao robot according to an assumed measurement model with additive Gaussian noise

$$\text{Pos[}\textbf{ball}\text{]}_t^{obs} = \text{Pos[}\textbf{ball}\text{]}_t^{world} + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \Sigma_\epsilon^t) \tag{21}$$

where $\text{Pos[}\textbf{ball}\text{]}_t^{obs}$ is the observed position of the ball at time point $t$, $\text{Pos[}\textbf{ball}\text{]}_t^{world}$ is the true position of the ball and $\Sigma_\epsilon$ is a covariance matrix we assume is known which capture the uncertainty of the observation. The uncertainty of the position observation might vary over time as the ball move closer or further away from the robot's camera. The uncertainty increases the further away the ball is since it will appear smaller on the image sensor of the camera. By assuming that the soccer field is flat and by knowing the transformation from the image-plane to the soccer field plane which the robot feet are standing on we can estimate the distance and the uncertainty.

We use a Kalman Filter [8] since an additive Gaussian noise assumption of our observations is sufficient for our needs. The inference can be expressed in closed form and is very efficiently computed in comparison to if we would have employed a Particle Filter with the same assumptions.

We assume that the initial physical state and physical state uncertainty of the ball is $\hat{x}_{0|0}$ and $P_{0|0}$ respectively, which are given or guessed. Further we use a constant acceleration motion model for modeling the dynamics of the ball through time. The reason for this is that a free moving ball will eventually stop on the soccer field due to friction from the grass which provides a negative acceleration that we assume is constant with some noise. By modeling the acceleration it is possible to predict the ball more accurately until rest.

The state $\mathrm{x}_t$ consists of the stochastic position, velocity and acceleration with 2 dimensions each and we assume the state to be Gaussian distributed with covariance matrix $P_t$

$$\mathrm{x}_t \sim \mathcal{N}(\hat{x}_t, P_t), \qquad \mathbb{E}(\mathrm{x}_t) = \hat{x}_t, \qquad Cov(\mathrm{x}_t) = P_t \tag{22}$$

The linear state space model used by the Kalman filter describes how the state progresses through time (the dynamics) and how the state is observed ($y_t$)

$$x_{t+\tau} = F_\tau x_t + G_\tau v, \qquad Cov(v) = Q, \tag{23}$$
$$y_t = H x_t + \epsilon_t \qquad Cov(\epsilon_t) = \Sigma_\epsilon^t \tag{24}$$

With a constant acceleration motion model and where no actions are performed (passive observation only), the matrices are given by

$$F_\tau = \begin{bmatrix} I_2 & \tau I_2 & \frac{\tau^2}{2} I_2 \\ 0_2 & 0_2 & I_2 \\ 0_2 & 0_2 & 0_2 \end{bmatrix}, \quad G_\tau = \begin{bmatrix} \frac{\tau^3}{6} I_2 \\ \frac{\tau^2}{2} I_2 \\ \tau I_2 \end{bmatrix}, \tag{25}$$
$$H = [1\ 1\ 0\ 0\ 0\ 0]^T,$$
$$Q = diag([0.1^2\ 0.1^2\ 0.1^2\ 0.1^2\ 0.05^2\ 0.05^2]^T),$$

where $I_n$ is the identity matrix of size $n \times n$, $\tau$ is the time difference between the current time point $t$ and some other time point, $F_\tau$ is the state transition matrix, $G_\tau$ is the process noise, $Q$ is the process noise covariance, $H$ is the observation matrix, $x_t$ is the state vector at time $t$, $y_t$ is the observation vector at time $t$ and $\epsilon_t$ is the observation noise at time $t$. We assume that the process noise is equal in both spatial dimensions and that the standard deviation is $0.1m$ for position, $0.1m/s$ for the velocity and $0.05m/s^2$ for the acceleration.

The measurement update of the Kalman filter is calculated

$$\hat{x}_{t|t} = \hat{x}_{t|t'} + K_t\big(y_t - H\hat{x}_{t|t'}\big), \tag{26}$$
$$P_{t|t} = (I - K_t H) P_{t|t'} \tag{27}$$

where $t' < t$ and $K_t = P_{t|t'} H^T \big(H P_{t|t'} H^T + \Sigma_\epsilon^t\big)^{-1}$.

The prediction of the state is calculated as

$$\hat{x}_{t'|t} = F_{t-t'} \hat{x}_{t|t} \tag{28}$$
$$P_{t'|t} = F_{t-t'} P_{t|t} F_{t-t'}^T + G_\tau Q G_\tau^T \tag{29}$$

where $t'$ can be smaller or larger than $t$ depending on if the prediction is into the past or into the future.

The computational environment $\mathcal{E}$ is in this example

$$\mathcal{F} = \{\text{Pos}, \text{Velocity}, \text{Acceleration}\} \tag{30}$$
$$\bar{\mathcal{F}} = \{\bar{\mathcal{F}}_{S_1} : [\text{Pos}], \tag{31}$$
$$\bar{\mathcal{F}}_{S_2} : [\text{Pos}, \text{Velocity}, \text{Acceleration}]^T\}$$
$$\mathcal{O} = \{Ball : \bar{\mathcal{F}}_{S_1}\} \tag{32}$$
$$\mathcal{D} = \{\mathcal{D}_{S_1} : N(x, \theta)\} \tag{33}$$
$$\mathcal{X} = \{\mathcal{X}_{S_1} : \langle \bar{\mathcal{F}}_{S_2}, \mathcal{D}_{S_1} \rangle\} \tag{34}$$
$$\mathcal{T} = \{KalmanConstAccelerationModel : \tag{35}$$
$$\langle observation : \bar{\mathcal{F}}_{S_1},\ \ state : \mathcal{X}_{S_1} \rangle\}$$
$$\mathcal{P} = \{inside : \langle C_1(x), \{\mathcal{X}_{S_1}\}\rangle, \tag{36}$$
$$insideRel : \langle C_2(x), \{\mathcal{X}_{S_1}, \mathcal{X}_{S_1}\}\rangle\}$$

The estimator of $KalmanConstAccelerationModel$ is specified by eq. (26)-(27), where the observed feature vector is $y_t = \text{Pos}[\textbf{ball}]_t^{obs} \in \bar{\mathcal{F}}_{S_1}$, together with eq. (28)-(29). The predictor is specified by eq. (28)-(29) and each temporal model instance is simply $\mathcal{TM}_t = \langle \bar{x}_{t|t}, P_{t|t} \rangle$. Prediction of the ball position at time point $t'$ given measurements up to time point $t$ is given by

$$\text{Pos}[\textbf{ball}]_{t'|t} \sim \mathcal{N}(\hat{x}_{t'|t(1:2)}, P_{t'|t}(1:2, 1:2)) \tag{37}$$

where $\hat{x}_{t'|t(1:2)}$ is the vector of the first two elements of $\hat{x}_{t'|t}$ containing the 2D position mean and $P_{t'|t}(1:2, 1:2)$ is the $2 \times 2$ sub-matrix of $P_{t'|t}$ containing the covariance of the 2D position.

The probability reasoner $inside$ is evaluated as

$$inside(\mathrm{x} \in \mathcal{X}_{S_1},\ \textbf{soccerField}) \tag{38}$$
$$= \int_{X^-}^{X^+} \int_{Y^-}^{Y^+} \mathcal{N}(x;\ \hat{x}_{\mathrm{t}_2|t_1(1:2)}, P_{\mathrm{t}_2|t_1}(1:2, 1:2)) dx_{(1)} dx_{(2)}$$

where the soccer field is aligned with the coordinate system shared with the ball and bounded by $X^-$ and $X^+$ on the x-axis and $Y^-$ and $Y^+$ on the y-axis respectively. $dx_{(1)}$ and $dx_{(2)}$ denote that the integration is over the first and second dimension of $x$ respectively.

The probability reasoner $insideRel$ is evaluated as

$$insideRel(\mathrm{x}_{t_2|t_1} \in \mathcal{X}_{S_1},\ \mathrm{y}_{t_4|t_3} \in \mathcal{X}_{S_1},\ \textbf{circle1.0m})$$
$$= \iint_{circle1.0m} \mathcal{N}(x;\ \hat{z}, P_z) dx_{(1)} dx_{(2)} \tag{39}$$

where $\hat{z} = \mu_{\mathrm{x}_{t_2|t_1}} - \mu_{\mathrm{y}_{t_4|t_3}}$, $P_z = P_{\mathrm{x}_{t_2|t_1}} + P_{\mathrm{y}_{t_4|t_3}}$ and $circle1.0m$ is the constraint of a circle with a radius of 1 meter centered at the origin in Cartesian coordinates over both dimensions of $x$.

## VIII. Example Scenario 2 – Normative Behavior

It is important for autonomous systems that operate in an unstructured environment with requirements on safety to be aware of the short-comings of their world models. New situations are bound to appear and incorrect as well as dangerous actions can be taken when the world model does not sufficiently match the real world.

Consider an industry assembly robot that works along side with humans in a factory. Its models might be based on some expected behavior of the human workers such that they walk and move slowly and consistently when close to the robot. For trained workers this is not an issue, but for a visitor that passes dangerously close to a working robot this can be a problem. To ensure that an unexpected situation results in no danger the robot should stop its work and enter a temporary safe mode. To stop whenever a human is within a certain distance will be too late if the distance is small and the human moves fast. On the other hand, if the distance or area is too large then it is hard for the humans and machine to work tightly together which reduce efficiency. The key here is that the suitable behavior of the machine is for it to stop not only reactively but rather when the expected near

future behavior of the humans have a high risk of being dangerous. Especially important is it for the machine to stop in an unexpected situation, where it cannot confidently be sure that there is no imminent high risk.

How do we detect an unexpected situation then? An obvious way is to compare how well one's predictions of the environment correspond to what is actually happening. If the predictive capability is bad then it is highly uncertain if safety can be assured as a future consequence of one's actions. A suitable example query is "*Is the precision of the predicted human position in 2 seconds always at least within a 0.5m radius circle and is the prediction always within 50% of the actual position of the human in 2 seconds?*":

$$\Box\big(Pr(insideRel(\bullet_{2|0}\text{Pos}[\textbf{human}], \textbf{circle0.5m})) > 0.95 \\ \wedge \bigcirc_2\big(similarity(\bullet_{0|-2}\text{Pos}[\textbf{human}], \\ \bullet_{0|0}\,\text{Pos}[\textbf{human}]) > 0.5\big)\big)$$

In truly unstructured environments unexpected situations will frequently occur and it is not sufficient to always stand still doing nothing when there are tasks that need to be done. The solution is then to observe and learn normative patterns of the surrounding environment while standing still, to be able to make better predictions next time and thereby avoid being needlessly paralyzed. The idea is simply: stop when the world diverges too much from the current world model, which is detected by when predictions are starting to become way off. While still, observe and learn about the unexpected behavior. Next time the same behavior happens it can now be predicted and it is no longer necessary to stop. Of course there are additional issues when safety has to be guaranteed as well as the fact that it might not be safe to stop wherever or whenever. It is however an important step towards autonomous systems that can operate in natural and unstructured environments in a safe way.

## IX. Related Work

A lot of progress has been made in the area of stream reasoning, covering both Data Stream Management Systems (DSMS) and Complex Event Processing (CEP) approaches. A comprehensive survey on these approaches is presented in [9]. Recent work on logic-based stream reasoning was done on DyKnow [10] and LARS [11]. In the context of the Semantic Web, which set out to make the web machine-readable, there has been an increasing awareness of the importance of stream reasoning. Continuous RDF query languages for stream processing are for example Event Processing SPARQL (EPSPARQL) [13] and Continuous SPARQL (C-SPARQL) [14].

Two different approaches to semantics of first-order logics of probabilities are considered and combined by [5]. The first approach is probabilistic over the domain ("*The probability that a randomly chosen UAV is above the roof top is 0.9*") and the second is over possible worlds ("*The probability that uav is above the roof top is 0.9*"). The probabilities are over predicates and well formed formulas, with a special probability operator that takes a well formed formula and produces a probability value term. The probability term can in turn be compared with other numeric terms using the ordinary comparator predicates. The probability distributions considered are discrete, either a distribution over all the objects in a domain or over the truth values of a well formed formula in possible worlds.

The work presented in this paper is complementary and concerns probabilities that arise from uncertain term values from observations and predictions. The latter is regarding possible future and past worlds but regarding terms as properties of objects. The uncertainty in terms propagates upwards and implicitly affects the probabilities of formulas being true and false, both at the current time and at other predicted times. Formulas are however not probabilistic themselves.

When using P-MTL for stream reasoning the goal is to incrementally evaluate formulas over streams of information. This is different from model checking as it only considers one particular execution of the system rather than a model of the system. There are approaches to model checking logics combining both temporal and probabilistic aspects [15].

In P-MTL it is possible to explicitly talk about the past using $\bullet_t F$ as long as $t$ is a valid time and history exists for $F$. MTL is an extension of Linear Temporal Logics (LTL) and there exists extensions such as LTL with Past (PLTL) that in addition has corresponding temporal operators about the past as LTL has about the future. It is possible to translate all temporal formulas about the past from PLTL to LTL, it is however expensive. The succinctness of a translation is exponential in the number of propositions [16].

There exists work on robust system verification and execution monitoring from noisy continuous and discrete signals using MTL [17]. MTL is used to write specifications that describe how the value of a signal is allowed to vary over time. Degree of robustness of a signal is defined as the bound of perturbations (noise) that the signal can have before it violates the MTL formulas. The treatment of the signals and the noise is non-stochastic. Our work provides several complementary abilities which could benefit their approach.

Discrete and continuous stochastic variables as well as their probability distributions are treated as distinct objects in distributional clauses in the work on inference for hybrid relational domains [6]. A clause is a first-order formula with a head and a body, where the head is an atomic formula and the body is a conjunction of atomic formulas or their negations. A distributional clause is then a clause $(h \sim D \leftarrow b_1, \ldots b_n)\theta$ which means that the stochastic variable $h$ is distributed according to probability distribution $D$ with parameters $\theta$ whenever $b_1, \ldots, b_n$ holds. Further they introduce a term $\approx (h)$ that represents the value of the stochastic variable $h$, i.e. when a random sample is

drawn from its probability distribution. Inference is based on statistical simulation by using Monte-Carlo sampling techniques. It is a very powerful formalism for describing logical and probabilistic relations, which includes relations over time. They are for example defining a stochastic transition function for the position of an object from one time point to the next as the current position with additive Gaussian noise but only if certain other clauses hold.

The work presented in this paper takes a slightly different route, where the details of probability distributions, temporal models (e.g. transition function) and probabilistic queries are abstracted away. Instead there is a common formal interface that allow for arbitrary stochastic variables, temporal models and probabilistic queries without additional extensions to the logic. Probabilities for probabilistic queries over stochastic variables produced from temporal models can be reasoned over at a higher level. Switching between different inference methods for temporal models is easier in our approach. Further, adding a non-parametric probability distribution such as a Gaussian process requires no extensions on our part (it is simply another temporal model) but it is not a minor extension to their work. Gaussian processes can for example be used to build good models about the history of a signal by collecting observations of the signal as the growing set of parameters of the distribution. This is important in robotics applications such as classification and prediction of motion patterns learned online in unstructured environments [18].

## X. Conclusion

We have presented P-MTL, a metric temporal logic which can reason over probabilistic and predicted states in a stream reasoning context. This addresses two important problems in AI, integrating logical and probabilistic reasoning and integrating reasoning over observations and predictions. This allows a robot to explicitly reason about the uncertainty of the world, the expected development of the world and the quality of its observations and predictions.

The benefit of our approach is that we impose very few limitations on the probabilistic reasoning (such as methods and models), allowing the usage of existing methods commonly used in practice in robotics. The logical stream reasoning inference in P-MTL is as efficient as for MTL since the necessary extension to the progression inference method is regarding the grounding of terms only. The evaluation of the predictive operators and the probability reasoners will however contribute to the overall complexity of evaluating P-MTL formulas.

Important temporal and probabilistic aspects of the probabilistic side is expressible on the logical level together with the results of probabilistic queries as terms to reason over. This allow logical reasoning upon probabilistic reasoning in an explicit and semantically connected manner with many immediate uses in robotics.

## References


[1] F. Heintz, J. Kvarnström, and P. Doherty, "Stream-based reasoning support for autonomous systems," in *Proc. ECAI*, 2010.

[2] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[3] P. Doherty, J. Kvarnström, and F. Heintz, "A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 3, pp. 332–377, 2009.

[4] F. Heintz, "Dyknow : A stream-based knowledge processing middleware framework," Ph.D. dissertation, Linkping UniversityLinkping University, The Institute of Technology, KPLAB - Knowledge Processing Lab, 2009.

[5] J. Y. Halpern, "An analysis of first-order logics of probability," *Artificial Intelligence*, vol. 46, pp. 311–350, 1990.

[6] D. Nitti, T. D. Laet, and L. D. Raedt, "A particle filter for hybrid relational domains," in *Proc. IROS*, 2013.

[7] S. Russell, *Unifying Logic and Probability: A New Dawn for AI?*, 2014.

[8] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960.

[9] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)*, vol. 44, no. 3, p. 15, 2012.

[10] F. Heintz and P. Doherty, "DyKnow: An approach to middleware for knowledge processing," *Journal of Intelligent and Fuzzy Systems*, vol. 15, no. 1, 2004.

[11] H. Beck, M. Dao-Tran, T. Eiter, and M. Fink, "Lars: A logic-based framework for analyzing reasoning over streams," in *Proc. AAAI*, 2015.

[12] ——, "Towards a logic-based framework for analyzing stream reasoning," in *Proc. Int. Workshop on Ordering and Reasoning*, 2014.

[13] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic, "Ep-sparql: a unified language for event processing and stream reasoning," in *Proc. WWW*, 2011.

[14] D. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus, "C-sparql: Sparql for continuous querying," in *Proc. WWW*, 2009.

[15] S. Konur, M. Fisher, and S. Schewe, "Combined model checking for temporal, probabilistic, and real-time logics," *Theor. Comput. Sci.*, no. 503, pp. 61–88, 2013.

[16] F. Laroussinie and N. Markey, "Temporal logic with forgettable past," in *Proc. LICS*, 2002.

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

[18] S. Ferguson, B. Luders, R. C. Grande, and J. P. How, *Algorithmic Foundations of Robotics XI*, 2015, ch. Real-Time Predictive Modeling and Robust Avoidance of Pedestrians with Uncertain, Changing Intentions.