Probabilistic Temporal Logic for Motion Planning with Resource Threshold Constraints

Chanyeol Yoo, Robert Fitch and Salah Sukkarieh
Australian Centre for Field Robotics (ACFR)
The University of Sydney
NSW, 2006, Australia
{c.yoo, rfitch, salah}@acfr.usyd.edu.au

***Abstract*—Temporal logic and model-checking are useful theoretical tools for specifying complex goals at the task level and formally verifying the performance of control policies. We are interested in tasks that involve constraints on real-valued energy resources. In particular, autonomous gliding aircraft gain energy in the form of altitude by exploiting wind currents and must maintain altitude within some range during motion planning. We propose an extension to probabilistic computation tree logic that expresses such real-valued resource threshold constraints, and present model-checking algorithms that evaluate a piecewise control policy with respect to a formal specification and hard or soft performance guarantees. We validate this approach through simulated examples of motion planning among obstacles for an autonomous thermal glider. Our results demonstrate probabilistic performance guarantees on the ability of the glider to complete its task, following a given piecewise control policy, without knowing the exact path of the glider in advance.**

I. Introduction

Task-level programming is a long-standing, decades-old problem in robotics. The goal is to command complex tasks using simple, natural language. Progress towards this goal has been achieved recently using temporal logic to specify *rich tasks* [3, 15] or *complex goals* [4]. In addition to providing powerful means for representing tasks, temporal logic specification offers the additional benefit of proving the correctness of a plan or policy that completes a task. By verifying a plan or policy against its logical specification while considering uncertainty, we can provide probabilistic performance guarantees. We are interested in developing provably correct control policies with expressive task-level specifications.

Formal verification is important for safety-critical systems and currently is widely employed in embedded systems such as pipelined CPUs [7] and medical monitoring systems [8]. In robotics, the significance of a probabilistic performance guarantee is that, given a stochastic transition model, it is possible to provide a probability of success *independent of the path eventually executed*. This principled understanding of level of confidence is necessary for robots operating in complex environments, especially outdoor environments, where failure in the worst case leads to catastrophic loss of the platform and compromises the safety of nearby humans.

This work is supported in part by the Australian Centre for Field Robotics and the New South Wales State Government. We thank Christopher J. Langmead and Nicholas Lawrance for helpful discussions.

One issue in applying formal methods is choosing the appropriate form of temporal logic [3]. Since we are interested in systems with stochasticity, probabilistic computation tree logic (PCTL) is the natural choice [17]. But PCTL as defined cannot fully express the task-level specifications required in robotics. In particular, we are interested in tasks that involve capacity constraints on energy resources. These energy resources are continuous quantities subject to upper and lower bounds. We would like to specify these bounds in continuous form, but PCTL specifies constraints as boolean propositions over discrete states. It is common in applications outside of robotics to extend temporal logic to increase expressive power; in this paper we propose such an extension to PCTL to model resource capacity as a real-valued reward threshold and demonstrate its use in the context of an autonomous gliding aircraft.

An intuitive application of a resource bound would be to constrain fuel or battery level above some minimum. However, in the case of an autonomous thermal glider the 'resource' of interest is altitude. The glider gains altitude, and hence energy, by exploiting favourable wind currents. Gliders must operate within a fixed altitude range for several reasons, including maintaining the safety of the platform and to comply with government regulations for autonomous flight. It may seem reasonable to model altitude discretely, but as with all discrete approximation it is then necessary to choose an appropriate resolution. Even with a very fine resolution, discrete approximation can lead to inaccurate evaluation of the safety criteria. For example, there may exist conditions where the length scale of wind features is less than the discretisation of altitude. A policy could then lead the glider into an unsafe wind gust yet is evaluated as safe. From the perspective of formal methods, there is no strong guarantee in the evaluation since such approximation may find the *presence* of certain behaviour, but not the *absence* of such behaviour.

Our approach is to extend PCTL to admit resource threshold constraints in continuous form. We present *Resource Threshold-PCTL (RT-PCTL)*, in which the logic is not only able to formally represent high-level symbolic specifications, but also a constraint on an accumulated real-valued resource. RT-PCTL includes a set of operators to formally represent *hard guarantees* and *soft guarantees* by considering the probability of mission success in all possible immediate transitions from a

state. For a hard guarantee, all possible immediate transitions from a state lead to a successor state that satisfies a given probability requirement for mission success. For a soft guarantee, there exists at least one immediate transition that leads to a successor state that satisfies the requirement.

Because control actions in our approach depend on temporal conditions (the value of accumulated resource at the time a state is entered), we define a *piecewise control policy*, where the value of a control action varies with accumulated resource. We also define a *piecewise probability function (PPF)* that represents the probability of mission success in a given state with respect to the value of accumulated resource. A set of PPFs, one for each state, represents the formal performance guarantee for a given control policy over all possible paths. Finally, we present algorithms for model-checking the control policy and associated PPF set against a given safety-critical requirement (hard or soft guarantee).

We validate our approach by developing an example with an autonomous thermal glider [19] in a hexagonal grid environment. The glider flies non-holonomically with *a priori* knowledge of the thermal airflow distribution. The resource threshold constraint requires the glider to maintain its altitude between given upper and lower bounds. We investigate two scenarios specified in RT-PCTL: 1) reaching the goal while avoiding danger states, and 2) reaching the goal while traversing either only safe states, or semi-safe states where all successor states are not danger states. The tasks are model-checked against the hard and soft constraints defined. We also compare the PPF evaluation with discrete evaluation at several fixed resolutions to demonstrate the need for representing resource values in continuous form.

II. Related Work

Temporal logic has been used extensively in embedded systems to specify required system properties over all possible paths that are not possible using traditional propositional logic. System properties include *functional correctness*, *liveness*, *safety*, *fairness*, *reachability* and *real-time property* [5, 7, 8, 20]. With the formal specification, *model-checking* is then used to systematically determine if the specification holds [2]. The critical difference between model-checking and testing/simulation is that model-checking is capable of detecting the *absence of error* whereas testing/simulation is only able to detect the *presence of error*. Therefore the formalism plays an important role in safety-critical systems.

Various forms of temporal logic have been proposed, including linear temporal logic (LTL) [21] and computation tree logic (CTL) [11]. Neither is defined to include stochastic transition models in their basic forms. For temporal logic to represent real-world environments with sensor noise and actuation error, probabilistic computation tree logic (PCTL) [17] was introduced to replace the non-determinism of CTL with probabilities.

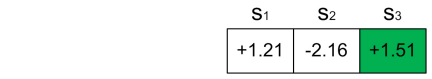
The application of temporal logic has recently become an important topic in robotics where a control policy generated is formally guaranteed to ensure safety [3, 13, 14]. The focus has been on systems where temporal logic is used for high-level discrete mission planning complemented by low-level planners or controllers operating in continuous space [4], and with approximation methods [1]. LTL is often used since stochasticity is not directly considered in the discrete abstraction layer [6, 9, 16], and there have been attempts to use LTL for motion-planning in uncertain environments with probabilistic guarantees [10]. Our work is distinct in that we focus on probabilistic transitions in the high-level discrete layer, with real-valued resource threshold constraints.

Other related work using MDPs and PCTL for complex mission specifications includes extending the MDP formation to maintain the probability of entering undesired states below a given constant [22], extending PCTL for more complex missions [18], and approximating PCTL [12]. Kwiatkowska et al. [17] have shown that PCTL can specify the reward structure such that a mission satisfies reward-related requirements such as 'the expected amount of energy that an agent acquires during the mission'. However, the reward structure in PCTL only considers the expected value at the end of the time horizon and thus is not suitable for a mission where success depends not only on the mission specification but also on the accumulated reward within the path. Our approach addresses this case directly.

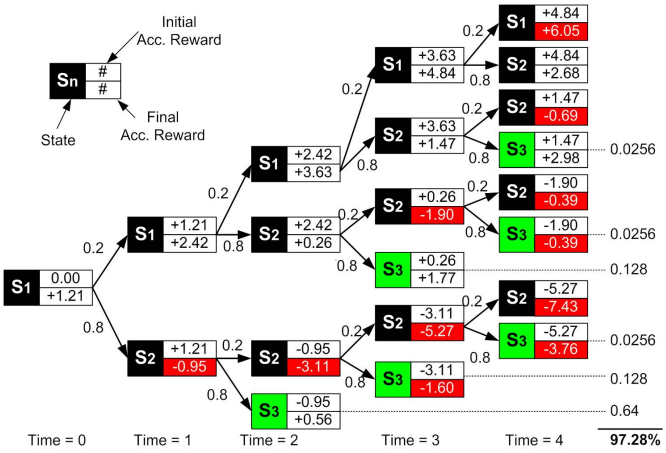
III. Problem Formulation

A labelled discrete-time Markov chain (DTMC) $\mathcal{M}$ has a tuple $< S, s_0, r_s, r_{ss'}, h_s, P, AP, L >$ with a finite set of states S, an initial state $s_0 \in S$, and a set of transition probabilities $P : S \times S \rightarrow [0, 1]$ where $\sum_{s' \in S} P_{ss'} = 1, \forall s \in S$. Scalar $r_s \in \mathbb{R}$ is an instant resource gained when entering the state s and $r_{ss'} \in \mathbb{R}^2$ is a transition resource gained while transiting from state s to state s'. Scalar set $h_s = [h_s^l, h_s^u]$ represents the lower and upper resource bounds respectively at state s. Function $L : S \rightarrow 2^{AP}$ is a labelling function that assigns atomic propositions AP for each state $s \in S$.

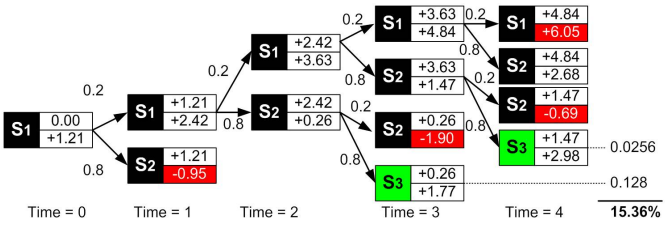
To illustrate the need for resource threshold constraints, a simple environment using the DTMC $\mathcal{M}$ is given in Fig. 1(a) with three states where state s_3 is the goal state. The agent starting from s_1 must maintain its accumulated resource between 0 and 5 until it reaches the goal state. The formulation using PCTL [17] allows us to compute the satisfaction of a property over an indefinite number of paths as shown in Fig. 1(b) without considering the resource bounds where the probability is computed to be 0.9728 after four time steps. However, the resource structure in standard PCTL is only able to model-check against 'expected accumulated resource after k time steps', 'expected instantaneous state resource at k time steps', and 'expected accumulated state resource before satisfying a formula'. Since these all compute the expectation of the resource at or after a certain number of time steps, standard PCTL is unable to determine if the accumulated resource within a path ever violates the bounds. As a result, the computation tree keeps branching from the state within the path that already went below threshold. Note that although the final resource at the end is above threshold, the mission



(a) Simple environment



(b) Probabilistic computation tree *without* resource constraint



(c) Probabilistic computation tree *with* resource constraint

Fig. 1. Subfigs. 1(b) and 1(c) show computation trees for 1(a), without and with resource constraints.

is considered to be unsuccessful if any state within the path does not satisfy the constraint.

The computation tree *with* a threshold constraint (Fig. 1(c)) shows that branching terminates at a state when the accumulated resource goes below zero. The probability of success in this case is 0.1536. From Fig. 1(c) it is shown intuitively that the successful path within four time steps is the one in which the agent stays in state s_1 for two time steps until its accumulated resource exceeds $+1$. Hence there is a need for a control policy structure and evaluation function that depend on the accumulated resource. This will be discussed in Sec. IV.

A. Resource Threshold-Probabilistic Computation Tree Logic (RT-PCTL)

The syntax for RT-PCTL is defined as

$$
\begin{aligned}
&\Phi ::= true | a | \neg\Phi | \Phi_1 \wedge \Phi_2 | \Phi_1 \vee \Phi_2 | \mathrm{A}\phi | \mathrm{E}\phi | P^{x:h}_{\sim\lambda}[\phi^k] \\
&\phi^k ::= \mathrm{X}\Phi | \mathrm{F}^{\leq k}\Phi | \Phi_1 \mathrm{U}^{\leq k}\Phi_2 \\
&\textit{where } \Phi \textit{ is a state formula,} \\
&\qquad \phi^k \textit{ is a path formula,} \\
&\qquad a \textit{ is an atomic proposition,} \\
&\qquad \sim \in \{<, \leq, \geq, >\}, \\
&\qquad x \in \mathbb{R} \textit{ and } \lambda \in [0,1] \textit{ and } k \in \mathcal{N}.
\end{aligned} \tag{1}
$$

A and E are *quantifiers over a path* and they state 'with all paths' and 'with at least one path' respectively. In this paper, the quantifiers A and E are used to evaluate *hard* and *soft* guarantees. X, F and U are *path-specific quantifiers*. $[\mathrm{X}p]$, $[\mathrm{F}^{\leq k}p]$ and $[p_1 \mathrm{U}^{\leq k} p_2]$ represent 'the property p is satisfied in the next time step', 'the property p is satisfied in future within k time steps' and 'the property p_1 holds until p_2 is satisfied within k time steps'. A transition from s to s' $(s \rightarrow s')$ is said to be valid if $P_{ss'} > 0$. $Path(s)$ is defined as a set of all infinite sequences of states starting from state s in which all consecutive transitions are valid. $\omega \in Path(s)$ is a path in the set. For the evaluation of state formulae Φ, we use the notation $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$ which denotes 'a set of states that satisfy the formula Φ'. We use the notation $P^{x:h_s}_{\sim\lambda}[\Phi]$ to specify probability inequality $[\sim \lambda]$, initial resource x and upper and lower resource bounds $h_s = [h^l_s, h^u_s]$ to a state formula Φ that is related to either $\mathrm{A}\phi^k$ or $\mathrm{E}\phi^k$. $Prob_s(x, h_s, \phi^k)$ indicates the probability of satisfying the path formula ϕ^k with initial resource of x over set of paths $Path(s)$.

The satisfaction relation $\models$ is defined for state formulae by:

$$
\begin{aligned}
s \models a &\iff a \in L(s) \\
s \models \neg\Phi &\iff s \not\models \Phi \\
s \models \Phi_1 \wedge \Phi_2 &\iff s \models \Phi_1 \textit{ and } s \models \Phi_2 \\
s \models \Phi_1 \vee \Phi_2 &\iff s \models \Phi_1 \textit{ or } s \models \Phi_2 \\
s \models \mathrm{A}\phi^{k+1} &\iff \omega' \in Path(s') \models \phi^k, \forall (s \rightarrow s') \\
s \models \mathrm{E}\phi^{k+1} &\iff \omega' \in Path(s') \models \phi^k, \exists (s \rightarrow s') \\
s \models P^{x:h}_{\sim\lambda}[\mathrm{A}\phi^{k+1}] &\iff Prob_{s'}(x', h_s, \phi^k) \sim \lambda, \forall (s \rightarrow s') \\
s \models P^{x:h}_{\sim\lambda}[\mathrm{E}\phi^{k+1}] &\iff Prob_{s'}(x', h_s, \phi^k) \sim \lambda, \exists (s \rightarrow s') \\
&\qquad \textit{and } Prob_s(x, h_s, \phi^{k+1}) \sim \lambda \\
&\quad \textit{where } x' = x + r_s + r_{ss'}
\end{aligned} \tag{2}
$$

Given a path ω in $\mathcal{M}$, the satisfaction relation is defined:

$$
\begin{aligned}
\omega \models \mathrm{X}\Phi &\iff \omega[1] \models \Phi \\
\omega \models \mathrm{F}^{\leq k}\Phi &\iff true \mathrm{U}^{\leq k}\Phi \\
\omega \models \Phi_1 \mathrm{U}^{\leq k}\Phi_2 &\iff \omega[i] \models \Phi_2 \wedge \omega[j] \models \Phi_1, \\
&\qquad \exists 0 \leq i \leq k, \forall 0 \leq j < i
\end{aligned} \tag{3}
$$

Two examples are shown below:

- $P^{2:[0,10]}_{<0.05}[\mathrm{AX}danger]$ = 'with all possible paths from the starting state, the probability of reaching the *danger* state in the next time step is less than 0.05, given the initial accumulated resource of 2, while maintaining the accumulated resource between 0 and 10'.
- $P^{0:[2,4]}_{>0.8}[E[\neg danger \mathrm{U}^{\leq 20s} goal]]$ = 'there exists at least one path from the starting state where the probability of reaching the goal state while avoiding the danger states within 20 seconds is greater than 0.8, given the initial accumulated resource of 0 while maintaining the accumulated resource between 2 and 4'.

B. Definition of Piecewise Function

This section defines the properties and operations of piecewise functions that are used later.

1) Piecewise Function: Suppose $f(x)$ is a piecewise function,

$$f(x) = \begin{cases} p_1^f & \text{if } x > c_1^f \\ \vdots & \\ p_k^f & \text{if } c_{k-1}^f \geq x > c_k^f \\ \vdots & \\ p_n^f & \text{if } c_{n-1}^f \geq x > c_n^f \\ 0 & \text{else} \end{cases} = \begin{cases} p_1^f & \text{if } x > c_1^f \\ \vdots & \\ p_k^f & \text{elseif } x > c_k^f \\ \vdots & \\ p_n^f & \text{elseif } x > c_n^f \\ 0 & \text{else} \end{cases}$$
$$\text{where } k \in \mathcal{N}, p_k^f = [0,1] \text{ and } c_{k+1}^f < c_k^f, \forall k \in \mathcal{N} \tag{4}$$

2) Addition:

$$f(x) + g(x) = \begin{cases} \vdots & \\ f(c) + g(c) & \text{elseif } x > c \\ \vdots & \\ 0 & \text{else} \end{cases} \tag{5}$$
$$\text{where } c \in (c^f \cup c^g)$$

3) Shift:

$$f(x + c) = \begin{cases} p_1 & \text{if } x > c_1 - c \\ \vdots & \\ p_n & \text{elseif } x > c_n - c \\ 0 & \text{else} \end{cases} \tag{6}$$

4) Multiplication:

$$c \cdot f(x) = \begin{cases} c \cdot p_1 & \text{if } x > c_1 \\ \vdots & \\ c \cdot p_n & \text{elseif } x > c_n \\ 0 & \text{else} \end{cases} \tag{7}$$

5) Conditioning:

$$f(x) \ominus c = \begin{cases} f(x) & \text{if } x > c \\ 0 & \text{else} \end{cases} \tag{8}$$

6) Merging:

$$f(x) \oplus g(x) = \begin{cases} f(x) & \text{if } x > c_n^f \\ g(x) & \text{else} \end{cases} \tag{9}$$

7) Bound Function: Returns zero if x is above c_u or below c_l; otherwise return $f(x)$.

$$\Gamma_{c_l}^{c_u}(f(x)) = \begin{cases} 0 & \text{if } x > c_u \\ f(x) & \text{elseif } x > c_l \\ 0 & \text{else} \end{cases} \tag{10}$$

IV. Performance Evaluation of Control Policy

To evaluate a control policy, we analytically solve a piecewise probability function (PPF) at each state. In this section, we show how to compute PPFs with respect to quantifiers X (next), U (until), and F (future).

A. PPF Solutions for Quantifiers X, U, and F

The quantifier X specifies a path property from a state where $[\mathrm{X}p]$ denotes 'property p holds in the next transition'. The solution for computing the PPF is shown in (11) for a *single-action control policy* $\pi(s)$. Note that $Prob_s(x, h, X\Phi)$ is a piecewise function defined in III-B that returns the probability of holding the property Φ in the next transition starting from the state s with the entering accumulated resource of x while maintaining the accumulated resource between $h_s = [h_s^l, h_s^u], \forall s \in S$. The algorithm for solving (11) is shown in Alg. 1. $P_{ss'}^{\pi(s)}$ denotes the transition probability from s to s' for an action $\pi(s)$.

$$\begin{aligned} & Prob_s(x, h_s, \mathrm{X}\Phi) \\ &= \begin{cases} 0 & \text{if } x > h_s^u - r_s \\ \sum_{s' \in S} P_{ss'}^{\pi(s)} \cdot Prob_{s'}(x', h_s, \mathrm{X}^0\Phi) & \text{if } x > h_s^l - r_s \\ 0 & \text{else} \end{cases} \\ &= \Gamma_{h_s^l - r_s}^{h_s^u - r_s}\Big(\sum_{s' \in S} P_{ss'}^{\pi(s)} \cdot Prob_{s'}(x', h_s, \mathrm{X}^0\Phi)\Big), \\ & \textit{where } Prob_s(x, h_s, \mathrm{X}^0\Phi) = \Gamma_{h_s^l - r_s}^{h_s^u - r_s}(1), \forall s \in Sat(\Phi), \\ & \qquad x' = x + r_s + r_{ss'} \end{aligned} \tag{11}$$

The quantifier U specifies the satisfaction of a property Φ along the path until it ends with another property Ψ, formally written as $[\Phi \mathrm{U} \Psi]$. The PPF is shown in (12) for a *single-action control policy* $\pi(s)$ and the algorithm is defined in Alg. 2. The formula $[\mathrm{F}^{\leq k}\Psi]$ is identical to $[true\mathrm{U}^{\leq k}\Psi]$.

$$\begin{aligned} & Prob_s(x, h_s, \Phi \mathrm{U}^{\leq k+1}\Psi), \forall s \in Sat(\Phi \wedge \neg\Psi) \\ & \quad = \Gamma_{h_s^l - r_s}^{h_s^u - r_s}\Big(\sum_{s' \in S} P_{ss'}^{\pi(s)} \cdot Prob_{s'}(x', h_s, \Phi \mathrm{U}^{\leq k}\Psi)\Big), \\ & \qquad\qquad \textit{where } x' = x + r_s + r_{ss'} \\ & Prob_s(x, h_s, \Phi \mathrm{U}^{\leq k}\Psi), \forall s \in Sat(\Psi) \\ & \quad = \Gamma_{h_s^l - r_s}^{h_s^u - r_s}(1) \\ & Prob_s(x, h_s, \Phi \mathrm{U}^{\leq k}\Psi), \forall s \in Sat(\neg\Phi \wedge \neg\Psi) \\ & \quad = 0 \end{aligned} \tag{12}$$

B. Piecewise Control Policy

The choice of action at a given state s, in our formulation, depends on the value of the accumulated resource x. We represent this as a *piecewise control policy* $\pi(s, x)$ where $\mathcal{A}(s)$ is a set of possible actions at state s:

$$\pi(s, x) = \begin{cases} a_1 \in \mathcal{A}(s) & \text{if } x > x_{a_1}^s \\ \vdots & \\ a_n \in \mathcal{A}(s) & \text{elseif } x > x_{a_n}^s \\ \emptyset & \text{else} \end{cases}. \tag{13}$$

Algorithm 1 Solve for $Prob_s(x, h_s, \mathrm{X}\Phi), \forall s \in S$

$Prob_s(x, h_s, \mathrm{X}^0\Phi) \leftarrow 0, \forall s \in Sat(\neg\Phi)$
$Prob_s(x, h_s, \mathrm{X}^0\Phi) \leftarrow \Gamma_{h_s^l - r_s}^{h_s^u - r_s}(1), \forall s \in Sat(\Phi)$
for $s \in S \backslash Sat(\Phi)$ **do**
  $Prob_s^a(x, h_s, X\Phi) \leftarrow \Lambda(s, a), \forall a \in \mathcal{A}(s)$
  $Prob_s(x, h_s, X\Phi) \leftarrow (Prob_s^{a_1}(\ldots) \ominus x_{a_1}^s) \oplus \ldots \oplus (Prob_s^{a_n}(\ldots) \ominus x_{a_n}^s)$
end for
return $Prob_s(x, h_s, \mathrm{X}\Phi), \forall s \in S$
where $\Lambda(s, a) = \Gamma_{h_s^l - r_s}^{h_s^u - r_s}(\sum_{s' \in S} P_{ss'}^a \cdot Prob_{s'}(x + r_s + r_{ss'}, h_s, \mathrm{X}^0\Phi))$

Algorithm 2 Solve for $Prob_s(x, h_s, \Phi \mathrm{U}^{\leq K}\Psi), \forall s \in S$

$Prob_s(x, h_s, \Phi \mathrm{U}^{\leq 0}\Psi) \leftarrow 0, \forall s \in Sat(\neg\Psi)$
$Prob_s(x, h_s, \Phi \mathrm{U}^{\leq 0}\Psi) \leftarrow \Gamma_{h_s^l - r_s}^{h_s^u - r_s}(1), \forall s \in Sat(\Psi)$
for $k = 1$ to K **do**
  for $s \in Sat(\Phi \wedge \neg\Psi)$ **do**
    $Prob_s^a(x, h_s, \Phi \mathrm{U}^{\leq k}\Psi) \leftarrow \Lambda(s, a, k-1), \forall a \in \mathcal{A}(s)$
    $Prob_s(x, h_s, \Phi \mathrm{U}^{\leq k}\Psi) \leftarrow (Prob_s^{a_1}(\ldots) \ominus x_{a_1}^s) \oplus \ldots \oplus (Prob_s^{a_n}(\ldots) \ominus x_{a_n}^s)$
  end for
end for
return $Prob_s(x, h_s, \Phi \mathrm{U}^{\leq K}\Psi), \forall s \in S$
where $\Lambda(s, a, k) = \Gamma_{h_s^l - r_s}^{h_s^u - r_s}(\sum_{s' \in S} P_{ss'}^a \cdot Prob_{s'}(x + r_s + r_{ss'}, h_s, \Phi \mathrm{U}^{\leq K}\Psi))$

For evaluation of the control policy, we define $Prob_s^a(x, h_s, \cdots^{\leq k} \cdots)$ as the PPF for the state s of taking action a at time k. and this function would come from IV-A. The PPF with respect to a *piecewise control policy* $\pi(s, x)$ is shown in (14). It is important to note that $Prob_s(x, h_s, \cdots^{\leq k} \cdots)$ and $Prob_s^a(x, h_s, \cdots^{\leq k} \cdots)$ should always be computed in the same iteration; they cannot be computed separately.

$$\begin{aligned} &Prob_s(x, h_s, \cdots^{\leq k} \cdots) \\ &\quad = (Prob_s^{\pi(s,x_1)}(x, h_s, \cdots^{\leq k} \cdots) \ominus x_{a_1}^s) \oplus \\ &\qquad \ldots \oplus (Prob_s^{\pi(s,x_n)}(x, h_s, \cdots^{\leq k} \cdots) \ominus x_{a_n}^s) \\ &\quad = \begin{cases} Prob_s^{\pi(s,x_1)}(x, h_s, \cdots^{\leq k} \cdots) & \text{if } x > x_{a_1}^s \\ \vdots & \\ Prob_s^{\pi(s,x_n)}(x, h_s, \cdots^{\leq k} \cdots) & \text{elseif } x > x_{a_n}^s \\ 0 & \text{else} \end{cases} \end{aligned} \tag{14}$$

C. Hard and Soft Guarantees

For a given control action at a state, there may be a number of paths that an agent may take due to uncertainty in transition. In RT-PCTL, we add *hard guarantee constraints* and *soft guarantee constraints* with quantifiers A and E to the conventional PCTL for richer expressivity of safety-critical requirements. Along with quantifiers $\phi^{\leq k} \in \{\mathrm{X}p, \mathrm{F}^{\leq k}p, p_1\mathrm{U}^{\leq k}p_2\}$, $[\mathrm{A}\phi^{\leq k}]$ and $[\mathrm{E}\phi^{\leq k}]$ define the hard and soft satisfaction guarantees that specify 'all transitions from the initial state should lead to states that satisfy the formula $\phi^{\leq k}$' and 'at least one transition from the initial state should lead to states that satisfy it'.

Algorithm 3 Solve for $P_{\sim\lambda}^{x:h_s}[\mathrm{A}\phi^{\leq k+1}]$

$P \leftarrow \{\emptyset\}$
Get $Prob_s(x, h_s, \phi^{\leq k}), \forall s \in S$
for $s \in S$ **do**
  if $Prob_{s'}(x + r_s + r_{ss'}, h_s, \phi^{\leq k}) \sim \lambda, \forall P_{ss'} > 0$ **then**
    $P \leftarrow s \cup P$
  end if
end for
return P

Algorithm 4 Solve for $P_{\sim\lambda}^{x:h_s}[\mathrm{E}\phi^{\leq k+1}]$

$P \leftarrow \{\emptyset\}$
Get $Prob_s(x, h_s, \phi^{\leq k}), \forall s \in S$
for $s \in S$ **do**
  if $Prob_{s'}(x + r_s + r_{ss'}, h_s, \phi^{\leq k}) \sim \lambda, \exists P_{ss'} > 0$ and $Prob_s(x, h_s, \phi^{\leq k+1}) \sim \lambda$ **then**
    $P \leftarrow s \cup P$
  end if
end for
return P

The solutions to computing the set of states that satisfy the hard guarantee and soft guarantee are shown in (15) and (16), where $\phi^{\leq k} \in \{\mathrm{X}p, \mathrm{F}^{\leq k}p, p_1\mathrm{U}^{\leq k}p_2\}$. Note that $Sat(P_{\sim\lambda}^{x:h_s}[\mathrm{A}\phi^{\leq k}]) \subseteq Sat(P_{\sim\lambda}^{x:h_s}[\mathrm{E}\phi^{\leq k}])$. This is shown algorithmically in Algs. 3 and 4.

$$\begin{aligned} Sat\left(P_{\sim\lambda}^{x:h_s}\left[\mathrm{A}\phi^{\leq k+1}\right]\right) = \{s \in S \mid \forall P_{ss'} > 0, \\ Prob_{s'}(x + r_s + r_{ss'}, h_s, \phi^{\leq k}) \sim \lambda\} \end{aligned} \tag{15}$$

$$\begin{aligned} Sat\left(P_{\sim\lambda}^{x:h_s}\left[\mathrm{E}\phi^{\leq k+1}\right]\right) = \{s \in S \mid Prob_s(x, h_s, \phi^{\leq k+1}) \sim \lambda \\ \textit{and } (Prob_{s'}(x + r_s + r_{ss'}, h_s, \phi^{\leq k}) \sim \lambda, \exists P_{ss'} > 0)\} \end{aligned} \tag{16}$$

D. Calculation Example

We illustrate the calculation of a PPF using the simple scenario shown earlier in Fig. 1(a). The agent can move left or right with probability 0.8 of moving as intended, and probability 0.2 of self-transition. Attempting to move past the left or right boundary always results in self-transition. The agent starts in state s_1 with entering resource value 0 and attempts to reach goal state s_3 while maintaining the value of accumulated resource between the resource bound

$h_s = [0, 5], \forall s \in S.$

$$\pi(s_1) = \textit{Right},$$
$$\pi(s_2) = \textit{Right}$$
$$Prob_1(x, h_s, \mathrm{F}^{\leq 4} s_3) = \begin{cases} 0.0000 & \text{if } x > +3.79 \\ 0.7680 & \text{if } x > +3.11 \\ 0.6400 & \text{if } x > +2.58 \\ 0.7936 & \text{if } x > +1.90 \\ 0.7680 & \text{if } x > +1.37 \\ 0.7936 & \text{if } x > +0.95 \\ 0.1536 & \text{if } x > -0.26 \\ 0.0256 & \text{if } x > -1.21 \\ 0 & \text{else} \end{cases} \quad (17)$$

A purely state-based policy for this example is shown in (17), where $Prob_1(x, h_s, \mathrm{F}^{\leq 4} s_3)$ and $\pi(s_1)$ are the PPF and policy at state s_1 within four time steps. $Prob_1(x, h_s, \mathrm{F}^{\leq 4} s_3)$ represents the PPF of the logic specification to reach state s_3 with accumulated resource x within four time steps with resource constraint h_s at state s_1. The policy is simply to move right at states s_1 and s_2. Since we constrain the accumulated resource to be above zero and below five, the agent is likely to fail if it enters state s_2 with zero accumulated resource; transitioning from state s_1 to state s_2 results in accumulated resource -0.95.

In contrast, evaluation of the piecewise control policy is shown in (18). Here, the action mapping varies with the initial accumulated resource and the agent attempts to go left until its accumulated resource exceeds $+1.0$. With such a policy, the probability of mission success has jumped from 0.1536 to 0.768. Note that the resource values can be any real number.

$$\pi(s_1, x) = \begin{cases} \textit{Right} & \text{if } x > +1.0 \\ \textit{Left} & \text{else} \end{cases}$$
$$\pi(s_2, x) = \textit{Right}, \forall x \in \mathbb{R}$$
$$Prob_1(x, h_s, \mathrm{F}^{\leq 4} s_3) = \begin{cases} 0.0000 & \text{if } x > +3.79 \\ 0.7680 & \text{if } x > +3.11 \\ 0.6400 & \text{if } x > +2.58 \\ 0.7936 & \text{if } x > +1.90 \\ 0.7680 & \text{if } x > +1.37 \\ 0.7936 & \text{if } x > +1.00 \\ 0.7680 & \text{if } x > -0.21 \\ 0.6400 & \text{if } x > -1.21 \\ 0 & \text{else} \end{cases} \quad (18)$$

V. Task Planning for an Autonomous Glider

Suppose there is an autonomous thermal glider in a hexagonal grid-based environment that gains or loses altitude based on instantaneous thermal wind energy. The glider has precise *a priori* knowledge of the time-invariant thermal energy distribution as shown in Fig. 2. The environment is discretised into a 10x10 grid with six possible glider orientation values. A state-direction pair is denoted (s, d). Each state is labelled with $\mathcal{S}$, $\mathcal{D}$, $\bar{\mathcal{S}}$ and $\mathcal{G}$ that denote *safe*, *danger*, *semi-safe* and *goal*. The task is formulated based on the labels. The glider fails its mission when: 1) it hits the boundary of the environment, 2) it enters a forbidden state, or 3) its altitude goes out of resource bounds $h_s = [0, 30], \forall s \in S$. The glider satisfies the mission when it completes its task without violating any constraints.

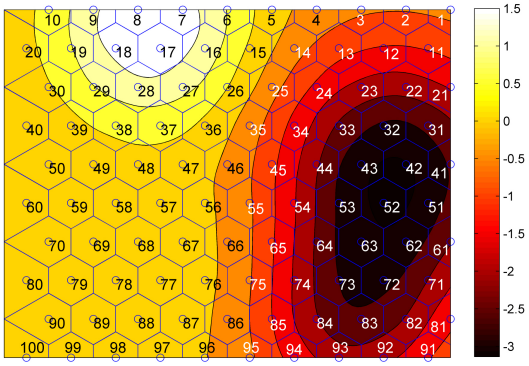
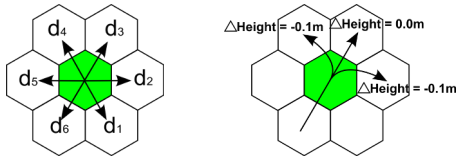


Fig. 2. Scenario environment (colour indicates change in altitude in metres).



(a) (b)

Fig. 3. Glider dynamics in a hexagonal grid where (a) shows possible transitions and (b) defines the glider's non-holonomic behaviour.

Fig. 3(a) shows the dynamics of the glider in this scenario. There are three possible actions defined relative to current orientation. The two actions that correspond to 60-degree turns lead to an altitude reduction of $0.1m$, whereas maintaining the previous direction does not incur any loss of altitude. Transitions are stochastic; the glider moves in its intended direction with probability 0.8 and moves 60 degrees to either side of the intended direction with probability 0.1.

For the purpose of evaluation, a piecewise control policy is generated based on a heuristic that is divided into *risky* and *conservative* actions based on the agent's altitude when it enters a region of interest. Note that in general, a control policy in RT-PCTL is capable of having any number of actions based on accumulated resources. The *risky* action gives the most direct path to the goal position without considering the energy distribution along the path, whereas the *conservative* action gives the greatest expected return of instantaneous altitude increase. The *risky* action is taken when the accumulated resource exceeds the amount of resource required to take the most probable path to the goal state, otherwise the *conservative* action is taken. We do not attempt to generate the optimal policy in this paper.

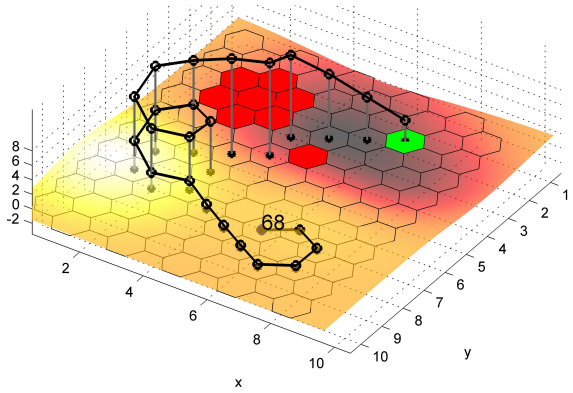


Fig. 4. The most probable path of a glider launched from minimum altitude at (s_{68}, d_1) to reach the *goal* (green hexagon) without entering *danger* states (red hexagons) while maintaining altitude between 0 and 30m.

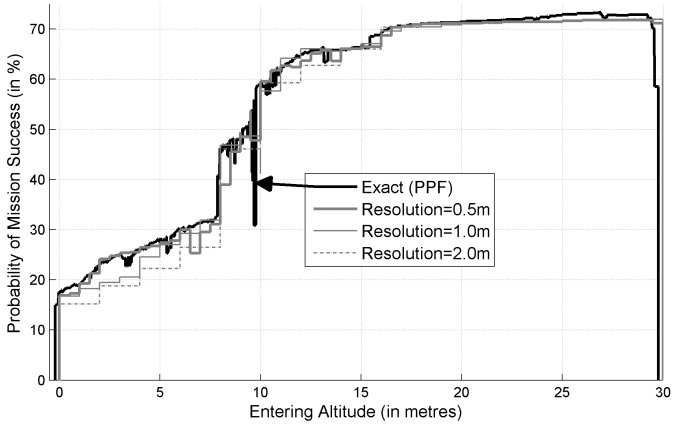


Fig. 5. Probabilities of mission success with respect to entering altitudes at state (s_{68}, d_1) after 30-step time horizon for approximation cases and PPF.

The heuristic policy is defined as:

$$\pi(s,x) = \begin{cases} \{d \in \mathcal{A}(s) \mid \text{'Most direct route to } \mathcal{G}\text{'}\} & \text{if } x > \alpha \\ \{d \in \mathcal{A}(s) \mid \max_d \sum_{s' \in S} P^d_{ss'} \cdot r_{s'}\} & \text{else} \end{cases}, \tag{19}$$

where $\mathcal{A}(s)$ is the set of actions available at state s and α is the negative sum of rewards along the most direct path to $\mathcal{G}$ from s that avoids forbidden states.

A. Reach the Goal and Avoid Danger

We consider an initial scenario where the glider is 'to reach the *goal* state within 30 time steps while avoiding any *danger* states, and maintaining minimum altitude of zero metres'. The RT-PCTL path formula for the specification is $[\neg\mathcal{D}\mathrm{U}^{\leq 30}\mathcal{G}]$.

The environment and most probable path from (s_{68}, d_1) are shown in Fig. 4. The PPF represents success probability over all possible paths from the given state while following the given piecewise control policy, not just for the path in Fig. 4.

Generally, the probability of success increases with altitude. However, there are certain altitude ranges where success probability decreases with increasing altitude. This is due to the heuristic control policy that is not capable of finding the optimal solution for this formulation.

Fig. 5 shows probability of mission success evaluated by approximating altitude using discretisation versus PPF based on real-valued altitude at (s_{68}, d_1). Probabilities were calculated using Alg. 2. In the discrete cases, accumulated reward was rounded to the nearest value discretised at uniform resolution ($0.5m$, $1.0m$, and $2.0m$) starting from $0.0m$. Discrete approximation evaluates success probability at discrete altitude values, whereas the PPF is a piecewise-constant function. This difference is important because success probability can change abruptly with altitude. In Fig. 5, the PPF value has a dip centred at $9.7m$ corresponding to a decision boundary in the policy. Above this range, the glider has sufficient energy to fly directly to the goal. Below this range, the policy directs the glider along a more energy-conservative path. Within this range, the policy takes the direct route but has high probability of failure. This is a good illustration of the benefit of the PPF representation because although the glider has reasonable direct control over lateral position, it has less direct control over altitude. The PPF captures safety-critical altitude conditions exactly, but discrete approximation in general does not.

We also illustrate the use of hard and soft guarantee constraints in this scenario. For the hard guarantee, we would like to find the set of states that satisfies the requirement that the probability of mission success is greater than 0.28 in *all* potential immediate stochastic transitions. The RT-PCTL formula for the requirement is written in (20) where 7 states satisfy the requirement, given resource bounds $h_s = [0, 30]$.

$$Sat\left(P^{0m:h_s}_{\geq 0.28}\left[\mathrm{A}\left[\neg\mathcal{D}\mathrm{U}^{\leq 30}\mathcal{G}\right]\right]\right) = \{(s_{29}, d_1), \ldots, (s_{38}, d_4)\} \tag{20}$$

Consider state (s_{38}, d_3) which belongs to the set in (20). The control policy at the state with altitude of zero metres is d_3 which leads to (s_{37}, d_2), (s_{28}, d_3) and (s_{29}, d_4) where the probabilities of satisfying the specification are 0.2843, 0.2841 and 0.3165 at time step 29. As expected from (20), all paths from the state (s_{38}, d_3) hold true for the mission requirement. As opposed to state (s_{38}, d_3), the state (s_{58}, d_3) does not satisfy the hard constraint although the state itself has the success probability of 0.2812, because one of the successor states has success probability less than 0.2387.

B. A Complex High-Level Mission Specification

In this scenario, the glider has a more complex high-level mission specification as shown in Fig. 6, where red represents *danger* and blue represents *semi-safe*. The mission is 'to reach the goal position within 30 time steps such that the glider always travels along safe states or semi-safe states if all next locations from the state are not dangerous'. The RT-PCTL path formula for the specification is $[(\mathcal{S} \vee (\bar{\mathcal{S}} \wedge \mathrm{AX}\neg\mathcal{D}))\mathrm{U}^{\leq 30}\mathcal{G}]$. The most probable path is shown in Fig. 6.

We also consider an example of a soft guarantee where the glider is to have probability greater than 0.20 of holding the

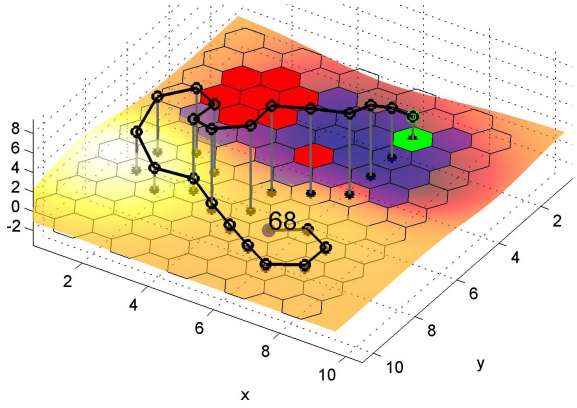


Fig. 6. The most probable path of a glider launched from minimum altitude at (s_{68}, d_1) to reach the *goal* (green hexagon) while traversing *safe* states (transparent hexagons) or *semi-safe* states (blue hexagons) if all successor states are not *danger* states and maintaining altitude above zero.

mission specification in *at least one direction*. The RT-PCTL formula is given in (21) and there are 36 states satisfying the requirement. Consider the state (s_{58}, d_3) which belongs to the set in (21). The successor states from the starting state have the success probabilities of 0.2061, 0.2064 and 0.1812. Therefore the soft guarantee requirement is met.

$$Sat\left(P_{\geq 0.20}^{0m:h_s}\left[\mathrm{E}\left[\left(\mathcal{S} \vee \left(\bar{\mathcal{S}} \wedge \mathrm{AX}\neg\mathcal{D}\right)\right)\mathrm{U}^{\leq 30}\mathcal{G}\right]\right]\right) = \left\{(s_{17}, d_2), \ldots, (s_{78}, d_4)\right\} \tag{21}$$

VI. Conclusions and Future Work

We have presented an extension to PCTL for systems with real-valued resource threshold constraints and stochastic transitions. We introduced the piecewise control policy and presented algorithms for model-checking a given policy against a formal specification and performance guarantee. We validated our theoretical results through a simulated example of motion planning among obstacles for an autonomous gliding aircraft.

The glider example demonstrates the significance of our results. We provide a level of confidence in the glider's ability to complete its mission without knowing in advance the exact path the glider will follow through discrete high-level states. Model-checking in our method provides a performance guarantee that applies to a piecewise control policy where real-valued energy resources are represented exactly.

RT-PCTL represents an important step towards the grand goal of complex mission specifications for stochastic systems, but many open problems remain. We have shown how to evaluate a given control policy, but generating an optimal piecewise control policy is an important area of future work. Understanding the scalability of model-checking with RT-PCTL and developing computationally efficient algorithms for formal verification is an open problem in general. We are also interested in extending RT-PCTL for dynamic environments where state labels change over time, which would be useful in tasks such as searching and tracking. Allowing for Gaussian-distributed and partially observable resource values are further major avenues of future work.

References

[1] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *Eur. J. Contr.*, 16(6):624–641, 2010.

[2] C. Baier and J.-P. Katoen. *Principles of model checking*. The MIT Press, 2008.

[3] A. Belta, C.and Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G.J. Pappas. Symbolic planning and control of robot motion. *IEEE Rob. Autom. Mag.*, 14(1):61–70, 2007.

[4] A. Bhatia, M.R. Maly, L.E. Kavraki, and M.Y. Vardi. Motion planning with complex goals. *IEEE Rob. Autom. Mag.*, 18(3): 55–64, 2011.

[5] A. Biere, C. Artho, and V. Schuppan. Liveness checking as safety checking. *Electron. Notes Theor. Comput. Sci.*, 66(2): 160–177, 2002.

[6] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and LaValle S.M. Controlling wild bodies using linear temporal logic. In *Proc. of Robotics: Science and Systems*, 2011.

[7] J.R. Burch and E.M. Clarke. Sequential circuit verification using symbolic model checking. In *Proc. of ACM/IEEE DAC*, pages 46–51, 1990.

[8] L.A. Cortes, P. Eles, and Z. Peng. Formal coverification of embedded systems using model checking. In *Proc. of EUROMICRO*, pages 106–113, 2000.

[9] X.C. Ding, M. Kloetzer, Y. Chen, and C. Belta. Automatic deployment of robotic teams. *IEEE Rob. Autom. Mag.*, 18(3): 75–86, 2011.

[10] X.C. Ding, S.L. Smith, C. Belta, and D. Rus. LTL Control in Uncertain Environments with Probabilistic Satisfaction Guarantees. *18th IFAC World Congress*, 18, 2011.

[11] E.A. Emerson and E. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Programming*, 2(3):241–266, 1982.

[12] M. Groesser. *Reduction methods for probabilistic model checking*. PhD thesis, TU Dresden, 2008.

[13] B. Johnson and H. Kress-Gazit. Probabilistic analysis of correctness of high-level robot behavior with sensor error. In *Proc. of Robotics: Science and Systems*, 2011.

[14] H. Kress-Gazit. Ensuring correct behavior: Formal methods for hardware and software systems [Special Issue]. *IEEE Rob. Autom. Mag.*, 18(3), 2011.

[15] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Translating structured english to robot controllers. *Adv. Robot.*, 22(12): 1343–1359, 2008.

[16] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu. Correct, reactive, high-level robot control. *IEEE Rob. Autom. Mag.*, 18 (3):65–74, 2011.

[17] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007.

[18] M. Lahijanian, J. Wasniewski, S.B. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Proc. of IEEE ICRA*, pages 3227–3232, 2010.

[19] N.R. Lawrance and S. Sukkarieh. Autonomous exploration of a wind field with a gliding aircraft. *J. Guid. Control Dynam.*, 34(3):719–733, 2011.

[20] G. Madl, S. Abdelwahed, and D.C. Schmidt. Verifying distributed real-time properties of embedded systems via graph transformations and model checking. *Real-Time Systems*, 33 (1):77–100, 2006.

[21] A. Pnueli. The temporal logic of programs. In *Proc. of FOCS*, pages 46–57, 1977.

[22] A. Undurti and J.P. How. An online algorithm for constrained POMDPs. In *Proc. of IEEE ICRA*, pages 3966–3973, 2010.