Task-Oriented Active Perception and Planning in Environments with Partially Known Semantics

Mahsa Ghasemi [1] **Erdem Arinc Bulgur** [2] **Ufuk Topcu** [2]

Abstract

We consider an agent that is assigned with a temporal logic task in an environment whose semantic representation is only partially known. We represent the semantics of the environment with a set of state properties, called *atomic propositions* over which, the agent holds a probabilistic belief and updates it as new sensory measurements arrive. The goal is to design a joint perception and planning strategy for the agent that realizes the task with high probability. We develop a planning strategy that takes the semantic uncertainties into account and by doing so provides probabilistic guarantees on the task success. Furthermore, as new data arrive, the belief over the atomic propositions evolves and, subsequently, the planning strategy adapts accordingly. We evaluate the proposed method on various finite-horizon tasks in planar navigation settings where the empirical results show that the proposed method provides reliable task performance that also improves as the knowledge about the environment enhances.

1. Introduction

An autonomous agent should simultaneously perceive its surroundings and plan for a given task according to its current knowledge. Machine learning algorithms have been hugely successful in dealing with different perception tasks such as semantic segmentation and object detection while extensive studies of control problems have led to numerous planning algorithms with provable theoretical guarantees. On the other hand, algorithms for joint perception and planning with the effectiveness of machine learning and the provable guarantees from control theory do not yet exist.

[1]Department of Electrical and Computer Engineering, University of Texas at Austin [2]Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin. Correspondence to: Mahsa Ghasemi <mahsa.ghasemi@utexas.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Joint perception and planning pose two fundamental challenges. The first challenge is designing an interface that not only enables the planning to use the perception outputs but also provides a way for the perception to ask for information-gathering strategies. The second challenge is the well-known trade-off of *exploration* versus *exploitation*, i.e., to what extent the agent should search for new information or to what extent it can rely on its current knowledge to take actions.

We argue that a pivotal factor to overcome these challenges is to associate the agent's knowledge and the available information to the task progress. Building this association requires compositional and structured reasoning over the task. Examples of structured task descriptions and compositional reward functions are sub-goal sequences (Singh, 1992), linear temporal logic (Wen et al., 2015; Li et al., 2017; Hahn et al., 2019), and reward machines (Icarte et al., 2018; 2019). Temporal logic specifications are interpretable by humans, can model a variety of tasks including non-Markovian ones, have a structured representation as an automaton, and are suitable for automated verification. Here, we focus on a subclass of temporal logic specifications, namely syntactically co-safe linear temporal logic (scLTL) (Kupferman & Vardi, 2001) that is tailored for specifying finite-horizon tasks.

In this paper, we study a problem of joint perception and planning under uncertainty. In particular, we consider an agent that interacts with an environment captured by a Markov decision process (MDP) (Puterman, 2014) and aims to successfully perform a task expressed by an scLTL specification. Our main contribution is to employ the structured form of a temporal logic specification to develop a *task-oriented* perception and planning. A temporal logic specification formally describes a task by imposing constraints on the task-related attributes of the states, called *atomic propositions*, that the agent visits. For instance, the temporal logic representation of a reach-avoid task is *"avoid the blocked states until reaching to the target state,"* where the set of atomic propositions is $\{$*blocked,target*$\}$. We model the semantic knowledge of the agent about the environment, called *belief*, through distributions over the atomic propositions. As new perception outputs arrive, the agent updates

its belief in a Bayesian or frequentist way, depending on its access to a reliable observation model. The proposed algorithm relies on online replanning and can dynamically switch between exploration and exploitation modes.

The problem in this paper is related to multi-tasking simultaneous localization and mapping (SLAM) (Guez & Pineau, 2010) where the agent may have side tasks such as maximizing a reward function in addition to localization and mapping. Nevertheless, divergent from the conventional SLAM setting which represents a map via a set of landmarks, we consider a map that consists of a set of task-related state attributes. Another related modeling framework is that of POMDPs (Sharan & Burdick, 2014) as they naturally represent state uncertainty. Nevertheless, in order for a POMDP to capture the perception uncertainty, the state variable must also include the environment's configuration (e.g., position of the obstacles) in addition to the agent's configuration (e.g., position of the agent). This state extension significantly increases the dimension of the state space and makes the solution computationally prohibitive (i.e., will result in a curse of dimensionality). Additionally, quantitative analysis of a POMDP with a temporal logic task is in general an undecidable problem (Chatterjee et al., 2016).

2. Related Work

There exist several formalisms for modeling decision-making under imperfect perception. Bai et al. (2014) modeled an integrated perception and planning problem using POMDPs and developed a value iteration algorithm over generalized policy graphs. In (Ghasemi & Topcu, 2019), the authors proposed a perception-aware point-based POMDP solver for settings where an agent can simultaneously take a planning action and a perception action. In (Benenson et al., 2006), the authors integrated SLAM with a partial motion planner for autonomous navigation. Fu et al. (2016) considered a robot with a temporal logic task in a probabilistic map obtained from a semantic SLAM. By defining a notion of δ-confident labeling function around the mean of the probabilistic map, they reformulated the stochastic control problem into a deterministic shortest path problem.

Temporal logic planning under imperfect perception has been studied from different perspectives. Jones et al. (2013) proposed a new type of logic, called distribution temporal logic, to enable expressions over belief-based predicates. In (Ding et al., 2011), the authors considered an agent moving over a graph where the truth values of the predicates over the nodes depend on known probabilities. There is also a family of solutions that rely on sampling methods (Vasile & Belta, 2013). In another related work, da Silva et al. (2019) propose a synthesis algorithm for probabilistic temporal logic over reals specifications in the belief space. In (Montana et al., 2017), the authors proposed a sampling-based solution to temporal logic planning under imperfect state information. To tackle the curse of history, their method relies on constructing a transition system by sampling from a feedback-based information roadmap. A relevant work to ours is that of Kress-Gazit et al. (2009) where they considered uncertainty in the environment propositions and proposed to design a reactive controller offline such that it can satisfy the task for all admissible environments. However, if the number of environment propositions are large, the controller should be able to cope to all admissible environments which is intractable.

Similar to our proposed approach, many solutions resort to replanning techniques. In (Wongpiromsarn et al., 2009), the authors tackled the state explosion problem of controller synthesis by proposing an iterative receding-horizon planning. For a subclass of temporal logic formulas, Livingston et al. (2012) introduced a method to locally patch a nominal controller once a change in the environment is detected. In (Lahijanian et al., 2016), the authors proposed an iterative replanning strategy that can relax the constraints imposed by the task if the discovery of a new obstacle deems the task unrealizable. Fu and Topcu (2015; 2016) designed an alternating active sensing and planning strategy for temporal logic tasks. Their approach is also an online replanning one that only calls for sensing when no strategy can be found for the current belief. In contrast, in this work sensing and planning occur simultaneously. Another related work is that of (Guo et al., 2013) where the agent generates a plan according to its initial knowledge and after new real-time information is gathered, it revises the plan. However, their model is deterministic as opposed to the stochastic model in our work, and the information is perfect and infrequent.

3. Problem Formulation

We now introduce the agent model, the environment model, the observation model and the task specification language that we use in the formal problem statement.

3.1. Agent Model

We model the interaction between the agent and the environment by a Markov decision process (MDP).

Definition 1. *A Markov decision process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, s_{init}, \mathcal{A}, \mathcal{T})$, where $\mathcal{S}$ is a finite discrete state space, $s_{init} \in \mathcal{S}$ is an initial state, $\mathcal{A}$ is a finite discrete action space, and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a probabilistic transition function such that, for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}$, $\sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') = 1$.*

The agent's decisions create a path over the MDP. Let $*$ to denote the Kleene star. A finite path $\mu = s_0 a_1 s_1 a_2 s_2 \ldots \in a_n s_n \in (\mathcal{S} \times \mathcal{A})^*$ is a finite sequence of states and actions such that $s_0 = s_{init}$ and, for all $i \in \mathbb{Z}$, $\Pr(s_i, a_{i+1}, s_{i+1}) >$

0.

A policy π is a mapping from the history $h(t) = (s_0, a_1, s_1, a_2, \ldots, s_t)$ of states and actions to a distribution over the action space. A memoryless policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a policy whose output action a_t depends only on the current state s_t. A deterministic policy $\pi : (\mathcal{S} \times \mathcal{A})^* \to \mathcal{A}$ is a policy that outputs an action deterministically.

A policy π when implemented on an MDP $\mathcal{M} = (\mathcal{S}, s_{init}, \mathcal{A}, \mathcal{T})$, induces a Markov chain. For a memoryless deterministic policy π, let $\pi(s)$ indicate the action chosen by the policy at state s. Then, the induced Markov chain is $\mathcal{M}^\pi = \mathcal{M} = (\mathcal{S}^\pi, {s_{init}}^\pi, \mathcal{T}^\pi)$ where $\mathcal{S}^\pi = \mathcal{S}$, ${s_{init}}^\pi = s_{init}$, and $\mathcal{T}^\pi : \mathcal{S} \times \mathcal{S} \to [0, 1]$ is such that

$$\mathcal{T}^\pi(s, s') = \mathcal{T}(s, \pi(s), s').$$

3.2. Environment Model

The agent perceives its environment through a set of binary-valued properties associated with the states, called atomic propositions. If an atomic proposition p holds true at a state s, we denote that by $s \models p$. A time-varying labeling function captures the belief of the agent about the truth values of the atomic propositions.

Definition 2. *An environment model is a tuple $\mathcal{E} = (\mathcal{S}, \mathcal{AP}, \bar{\mathcal{L}})$ where $\mathcal{S}$ is a finite, discrete state space, $\mathcal{AP}$ is a set of atomic propositions, and $\bar{\mathcal{L}} : \mathcal{S} \to 2^{\mathcal{AP}}$ is a deterministic labeling function that captures the true state of the environment. The agent's belief at time t is a probabilistic labeling function $\mathcal{L}_t : \mathcal{S} \times 2^{\mathcal{AP}} \to [0, 1]$. For a state s and a subset of atomic propositions $P \subseteq \mathcal{AP}$, $\mathcal{L}_t(s, P)$ assigns the probability of the event that P holds true at s, i.e., $Pr(\bigcap_{p \in P} s \models p)$.*

Notice that at each time step t and for every state $s \in \mathcal{S}$, $\sum_{P \subseteq \mathcal{AP}} \mathcal{L}_t(s, P) = 1$. $\mathcal{L}_0$ is the agent's prior belief which for example, may be an uninformative prior distribution. We assume that the truth values of the propositions are mutually independent in each state,

$$\forall s \in \mathcal{S} \quad \text{and} \quad \forall P \subseteq \mathcal{AP} : \qquad Pr(\bigcap_{p \in P} s \models P) = \prod_{p \in P} Pr(s \models p),$$

and also between every pair of different states,

$$\forall s_1, s_2 \in \mathcal{S}, s_1 \neq s_2 \quad \text{and} \quad \forall P_1, P_2 \subseteq \mathcal{AP} : \\ Pr(s_1 \models P_1 \wedge s_2 \models P_2) = Pr(s_1 \models P_1) Pr(s_2 \models P_2).$$

This independence assumption facilitates the update of the labeling function over time. Nevertheless, so long as the joint distribution model is known, the updates can be computed.

3.3. Observation Model

At each time step, the agent's perception module processes a set of sensory measurements regarding the atomic propositions. While the measurements may be from multiple sensing units, for ease of notation, we consider their joint model by a general observation function.

Definition 3. *Let $\mathcal{Z}(s_1, s_2, p) \in \{True, False\}$ denote the perception output of the agent at state s_1 for the atomic proposition p at state s_2. The joint observation model of the agent is $\mathcal{O} : \mathcal{S} \times \mathcal{S} \times \mathcal{AP} \times \{True, False\} \to [0, 1]$ where $\mathcal{O}(s_1, s_2, p, b)$ represents the probability that $\mathcal{Z}(s_1, s_2, p) = True$ if the truth value of p is according to b.*

In particular, an accurate observation model is the one for which the output probability of $\mathcal{O}(s_1, s_2, p, b)$ is one for $b = True$ and zero for $b = False$.

In the Bayesian framework, the observation model is used for the update of the agent's belief. Nevertheless, in the absence of such observation model, one can perform the update in a frequentist way.

3.4. Task Specification Language

We use syntactically co-safe linear temporal logic (scLTL) (Kupferman & Vardi, 2001) to specify finite-horizon tasks for the agent.

Definition 4. *The formulas in scLTL are inductively defined according to the following grammar:*

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

where the truth symbols, negation, and conjunction ($\wedge$) are borrowed from the propositional logic. $p \in \mathcal{AP}$ is an atomic proposition and φ, φ_1, and φ_2 are scLTL formulas. The temporal operators $\bigcirc$, $\mathcal{U}$, and $\Diamond$ correspond to Next, Until, and Eventually, respectively.

Let $\Sigma = 2^{\mathcal{AP}}$ denote the finite alphabet composed of all possible valuations of the propositions. A letter $\sigma \in \Sigma$ is interpreted as the valuation that assigns value $True$ to all $p \in \sigma$ and $False$ to all $p \in \mathcal{AP} \setminus \sigma$. A finite word $w = \sigma_0 \sigma_1 \sigma_2 \ldots \sigma_n \in \Sigma^*$ is a finite sequence of letters. An scLTL formula φ generates a language over finite words such that a word w is included in the language if it satisfies φ, denoted by $w \models \varphi$. See (Baier & Katoen, 2008) for the complete set of rules that determine the semantics of scLTL specifications.

The language defined by an scLTL formula can equivalently be represented through a deterministic finite automaton (Kupferman & Vardi, 2001).

Definition 5. *A deterministic finite automaton (DFA) is a tuple $\mathcal{D} = (\mathcal{Q}, q_{init}, \Sigma, \delta, \mathcal{F})$, where $\mathcal{Q}$ is a finite set of*

states, $q_{init} \in \mathcal{Q}$ is an initial state, Σ is an input alphabet, $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ is a deterministic transition function, and $\mathcal{F} \subseteq \mathcal{Q}$ is a set of accepted states.

A run $\rho = q_0 q_1 q_2 \dots q_n \in Q^*$ on $\mathcal{D}$ is a finite sequence of states such that $q_0 = q_{init}$ and, for all $i \in \mathbb{Z}$, there exists $\sigma_i \in \Sigma,\ q_{i+1} = \delta(q_i, \sigma_i)$. ρ is an accepting run of $\mathcal{D}$ if and only if $q_n \in \mathcal{F}$. Let $w = \sigma_0 \sigma_1 \sigma_2 \dots \sigma_n \in \Sigma^*$ be a finite word. w is included in the language defined by a DFA $\mathcal{D}$ if the corresponding run of w on $\mathcal{D}$, indicated by $\rho(w)$, is accepting.

Consider an environment model $\mathcal{E} = (\mathcal{S}, \mathcal{AP}, \bar{\mathcal{L}})$ and a finite path $\mu = s_0 a_1 s_1 a_2 s_2 \dots \in a_n s_n \in (\mathcal{S} \times \mathcal{A})^*$ over an MDP $\mathcal{M} = (\mathcal{S}, s_{init}, \mathcal{A}, \mathcal{T})$. Let $w(\mu) = \sigma_0 \sigma_1 \sigma_2 \dots \sigma_n \in \left(2^{\mathcal{AP}}\right)^*$ indicate the word generated by μ according to $\bar{\mathcal{L}}$, i.e., $\sigma_i = \{p \,|\, p \in \bar{\mathcal{L}}(s_i)\}$. μ satisfies an scLTL task φ if and only if the run induced by μ, i.e., $\rho(w(\mu))$, on the DFA representation of φ is accepting.

3.5. Problem Statement

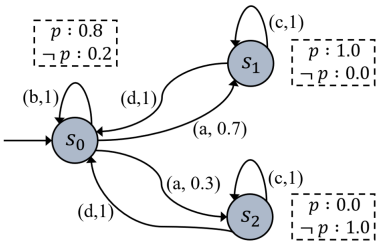
We consider an agent whose interaction with the environment is captured by an MDP. The agent is tasked with an scLTL specification that can be successfully completed within finite time. The agent is unaware of the environment state. However, it starts with a prior knowledge and over time, gathers observations that can be used to further revise its belief. The formal definition of the problem is stated next.

Problem 1. *Given an MDP $\mathcal{M} = (\mathcal{S}, s_{init}, \mathcal{A}, \mathcal{T})$, an environment model with unknown labeling function $\mathcal{E} = (\mathcal{S}, \mathcal{AP}, \_)$, an observation model $\mathcal{O}$, and an scLTL task φ, find a policy π that maximizes the probability of satisfying the task conditioned on the true labeling function, i.e.,*

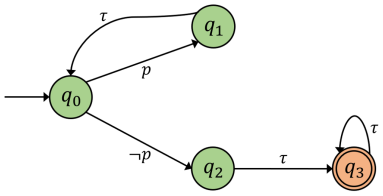
$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, Pr(\mathcal{M}^\pi \models \varphi \,|\, \bar{\mathcal{L}}).$$

4. Proposed Algorithm

Before introducing the algorithm, it is necessary to first describe the challenges of having a probabilistic view of the environment and, in particular, atomic propositions. In a setting where the agent is uncertain about the valuations of all atomic propositions over the whole environment, there may be up to $2^{|\mathcal{S}||\mathcal{AP}|}$ possibilities for how the environment is configured. In this case, computing policies that can account for all possible configurations, as offline reactive synthesis does (Baier & Katoen, 2008), is indeed computationally intractable. Additionally, if the environment is not dynamically changing, then such a comprehensive policy that accounts for all possible configurations, is not even needed. Another fundamental challenge is the fact that the nature of the probabilistic perception differs from the stochasticity of the agent model. Therefore, as seen in Example 1, one cannot combine the belief probabilities on the perception side with the transition probabilities of the MDP.



(a) An MDP. The edge labels represent an action and a transition probability, respectively, while the node labels capture agent's knowledge about the value of property p at each state.



(b) A DFA with knowledge uncertainty. The edge labels represent properties' valuations that lead to a transition where τ denotes any valuation (tautology). Node q_3 is the accepting state.

Figure 1. The MDP transitions behave in a stochastic way while the uncertainty in knowledge does not.

Example 1 (Stochastic transition versus knowledge uncertainty)**.** *Consider the simple MDP and DFA in Figure 1. The DFA accepts any path on the MDP whose induced word is in the form of $(p\tau)^* \neg p \tau$. For instance, if the true labels are $s_0 \models p$, $s_1 \models p$, and $s_2 \models \neg p$, then the path*

$$s_0 \xrightarrow{a} s_2 \xrightarrow{c} s_2 \xrightarrow{d} s_0$$

on the MDP generates the run

$$q_0 \xrightarrow{p} q_1 \xrightarrow{\neg p} q_0 \xrightarrow{\neg p} q_2 \xrightarrow{p} q_3$$

on the DFA which is accepting and satisfies the task.

A key observation is that the nature of the probabilities of the MDP transitions are distinct from that of the agent's belief. A stochastic transition means that if the agent takes the same action at the same state multiple times, every time the next state is determined by the given probabilities. Therefore, a path like

$$s_0 \xrightarrow{a} s_1 \xrightarrow{d} s_0 \xrightarrow{a} s_2$$

is possible on the MDP. On the other hand, the true labels of the states are fixed and the distribution of the agent's belief does not translate into similar behavior. For instance, the path

$$s_0 \xrightarrow{b} s_0 \xrightarrow{b} s_0 \xrightarrow{b} s_0$$

on the MDP cannot generate the run

$$q_0 \xrightarrow{p} q_1 \xrightarrow{p \vee \neg p} q_0 \xrightarrow{\neg p} q_2 \xrightarrow{p \vee \neg p} q_3$$

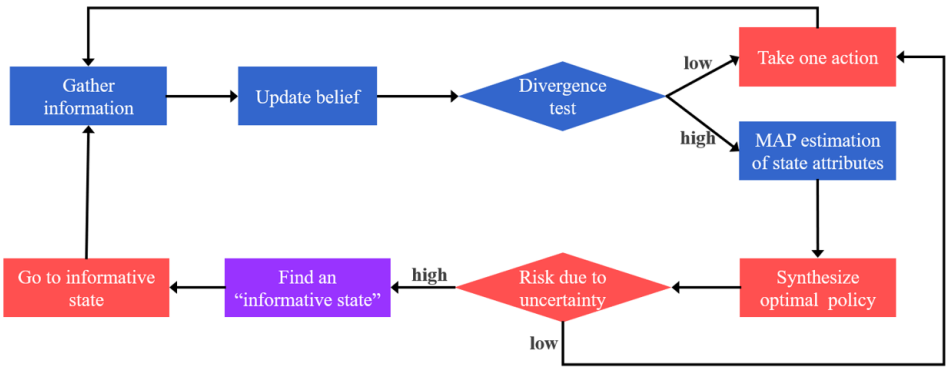


Figure 2. The schematic of the perception-planning loop. The blue blocks refer to pure perception modules and the red blocks refer to pure planning modules, and the purple block indicates a combined perception and planning module of the algorithm.

on the DFA. Since in this run, the truth value assignment of property p at state s_0 is inconsistent.

The outline of the proposed algorithm is illustrated in Figure 2. At each state, the agent gathers some perception outputs and uses them to update its belief. If the updated belief is significantly different from the previous one, the agent must replan. Otherwise, it will continue with its previous policy and takes a step according to it. To check the significance of the added information, we compute the divergence between the prior and posterior beliefs. If the threshold (a hyperparameter given to the algorithm) is exceeded, the agent first estimates the most probable configuration of the environment. Then, the agent applies a synthesis algorithm with the estimated environment model. This synthesis algorithm outputs a strategy that maximizes the probability of satisfying the given temporal logic task. The given strategy induces a Markov chain that is used to calculate the risk due to perception uncertainty. If the risk is lower than a threshold (another hyperparameter given to the algorithm), the agent uses the computed policy to take a step. Otherwise, it will find an active perception strategy to reduce its perception uncertainty. We now proceed to explain different stages of the proposed algorithm.

4.1. Information Processing

Consider the agent's state to be s_t at time t. The agent will receive new perception outputs according to the observation model $\mathcal{O}(s_t, ., ., .)$ for all states and atomic propositions.

The agent employs the observations to update its learned model of the environment in a Bayesian approach. For ease of notation, let $\mathcal{L}_t \equiv \mathcal{L}_t(s,p)$ and $\mathcal{O}(b) \equiv \mathcal{O}(s_t, s, p, b)$. Given the prior belief of the agent $\mathcal{L}_{t-1}$ and the received observations $\mathcal{Z}$, the posterior belief follows

$$Pr(s \models p | \mathcal{Z}(s_t,s,p) = True) = \frac{\mathcal{L}_{t-1}\mathcal{O}(True)}{\mathcal{L}_{t-1}\mathcal{O}(True) + (1-\mathcal{L}_{t-1})\mathcal{O}(False)},$$

$$Pr(s \models p | \mathcal{Z}(s_t,s,p) = False) = \frac{\mathcal{L}_{t-1}(1-\mathcal{O}(True))}{\mathcal{L}_{t-1}(1-\mathcal{O}(True)) + (1-\mathcal{L}_{t-1})(1-\mathcal{O}(False))},$$

for all $s \in \mathcal{S}$ and $p \in \mathcal{AP}$. Depending on the truth value observed for p, $\mathcal{L}_t(s,p)$ will be updated according to one of the above expressions. Besides, for any $P \subseteq \mathcal{AP}$,

$$\mathcal{L}_t(s,P) = \prod_{p \in \mathcal{AP}} \mathcal{L}_t(s,p).$$

4.2. Divergence Test on the Belief

If the agent's knowledge about the environment configuration has not significantly changed from its knowledge in the previous state, then the agent will continue with its previous policy. Nevertheless, if its knowledge has significantly changed, the agent will synthesize a new policy. We use Jensen-Shannon divergence to quantify the change in the belief distribution between two consecutive time steps. The cumulative Jensen-Shannon divergence over the states and the propositions can be expressed as

$$\begin{aligned} D_{\mathcal{JSD}}(\mathcal{L}_{t-1} \| \mathcal{L}_t) &= \frac{1}{2} D_{\mathcal{KL}}(\mathcal{L}_{t-1} \| \mathcal{L}_m^t) + \frac{1}{2} D_{\mathcal{KL}}(\mathcal{L}_m^t \| \mathcal{L}_t) \\ &= \frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{AP}} \mathcal{L}_{t-1}(s,p) \log \frac{\mathcal{L}_{t-1}(s,p)}{\mathcal{L}_m^t(s,p)} \\ &\quad + (1-\mathcal{L}_{t-1}(s,p)) \log \frac{1-\mathcal{L}_{t-1}(s,p)}{1-\mathcal{L}_m^t(s,p)} \\ &\quad + \frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{AP}} \mathcal{L}_m^t(s,p) \log \frac{\mathcal{L}_m^t(s,p)}{\mathcal{L}_t(s,p)} \\ &\quad + (1-\mathcal{L}_m^t(s,p)) \log \frac{1-\mathcal{L}_m^t(s,p)}{1-\mathcal{L}_t(s,p)}, \end{aligned}$$

where $\mathcal{L}_m^t = 1/2\,(\mathcal{L}_{t-1} + \mathcal{L}_t)$ is the average distribution. One of the input parameters to the algorithm is a threshold γ_d on the above divergence. If γ_d is not exceeded, the agent uses its previous policy π_{t-1} to pick an action and transitions according to its outcome. Otherwise, it has to synthesize a new policy.

For the policy synthesis step, the agent first estimates the most probable environment configuration from the distribution dictated by its updated belief. Let $\hat{\mathcal{L}}_t : \mathcal{S} \to 2^{\mathcal{AP}}$ indicate the agent's inference of the environment configuration at time t. The maximum a posteriori estimation is fairly simple as it decomposes into finding the mode of the posterior distribution for each property at each state. For binary-valued atomic propositions that follow Bernoulli distributions, the inference turns into picking the more probable outcome for each property at each state, i.e.,

$$\hat{\mathcal{L}}_t(s) = \{p \in \mathcal{AP} \,|\, \mathcal{L}_t(s,p) \geq 0.5\}.$$

4.3. Policy Synthesis

The first step to synthesize a policy π_t on the MDP that maximizes the probability of satisfying a temporal logic formula is to construct a product of the MDP with the automaton representing the formula. The product system is an extended MDP that not only captures the state evolution according to the agent model but also keeps track of the task progress according to the automaton.

Definition 6. *Given an MDP $\mathcal{M} = (\mathcal{S}, s_{init}, \mathcal{A}, \mathcal{T})$, an environment model $\mathcal{E} = (\mathcal{S}, \mathcal{AP}, \mathcal{L})$, and a DFA $\mathcal{D} = (\mathcal{Q}, q_{init}, \Sigma, \delta, \mathcal{F})$ the product MDP is defined as $\mathcal{M}_\mathcal{D} = \mathcal{M} \otimes \mathcal{D} = (\mathcal{S}_\mathcal{D}, s_{init\mathcal{D}}, \mathcal{A}_\mathcal{D}, \mathcal{T}_\mathcal{D})$ where $\mathcal{S}_\mathcal{D} = \mathcal{S} \times \mathcal{Q}$, $s_{init\mathcal{D}} = (s_{init}, q_{init})$, $\mathcal{A}_\mathcal{D} = \mathcal{A}$, and $\mathcal{T}_\mathcal{D} : \mathcal{S}_\mathcal{D} \times \mathcal{A}_\mathcal{D} \times \mathcal{S}_\mathcal{D} \to [0,1]$ is defined such that $\mathcal{T}_\mathcal{D}((s,q),a,(s',q')) = \mathcal{T}(s,a,s')$ if $\delta(q,\mathcal{L}(s')) = q'$, and 0 otherwise.*

Finding an optimal policy, i.e., a policy that maximizes the probability of realizing a temporal logic task, translates into a reachability criterion on the product MDP. Let $\mathcal{F}_\mathcal{D}^\mathcal{M} = \mathcal{S} \times \mathcal{F}$ denote the equivalent accepting states on the product MDP. The agent must find a policy that with high probability reaches to $\mathcal{F}_\mathcal{D}^\mathcal{M}$.

A well-known result from temporal logic synthesis states that there exists an optimal memoryless deterministic policy on the product MDP (Baier & Katoen, 2008). Hence, one can restrict the search space to that of memoryless deterministic policies and formulate

$$\pi_t = \arg\max_{\pi \in \Pi_{nm,d}} Pr(\mathcal{M}_\mathcal{D}^\pi \models \Diamond \mathcal{F}_\mathcal{D}^\mathcal{M} \,|\, \hat{\mathcal{L}}_t), \qquad (1)$$

where $\Pi_{nm,d}$ is the set of memoryless and deterministic policies. To find π_t in (1), we use a linear programming approach (Baier & Katoen, 2008). The optimal value of the linear program is the maximum probability of reaching to the accepting states. In order to find the corresponding optimal policy, it suffices to find the actions for which the constraints are active. If there are more than one action with active constraint for a state, any of those actions can be chosen arbitrarily.

4.4. Risk Assessment of Imperfect Perception

To assess the risk of the computed policy due to the perception uncertainties, we now factor in the probabilistic belief of the agent over the environment properties. In particular, we first generate the induced Markov chain $\mathcal{M}_\mathcal{D}^{\pi_t}$ by applying the policy π_t over the product MDP $\mathcal{M}_\mathcal{D}$ (see Section 3). Next, we verify the induced Markov chain with the uncertain labels against the task specification. We develop an algorithm (described in the supplementary material) via a computation graph that yields the exact probability of the task realization. However, due to the complexities explained in Example 1, such quantitative analysis has exponential complexity, as formalized in the next theorem.

Theorem 1. *Let $\mathcal{M}_\mathcal{D}^{\pi_t}$ to be a Markov chain with n states and $\mathcal{L}_t$ to be a fully probabilistic labeling function (i.e., all labels are uncertain) over m atomic propositions. Quantitative verification of $\mathcal{M}_\mathcal{D}^{\pi_t}$ against a reachability specification has a complexity of $O(n2^{nm})$.*

Proof. *See the supplementary material for the proof.*

Since the complexity of an exact quantitative analysis is prohibitive, we instead propose a statistical verification. More specifically, we approximate the expected value of the probability of the task realization over all possible instances of the environment

$$\mathbb{E}_{\mathcal{L} \sim Dist(\mathcal{L})}\left[Pr\left(\mathcal{M}_\mathcal{D}^{\pi_t} \models \varphi\right)\right]$$

by an empirical expectation with N samples

$$\hat{\mathbb{E}}_{\mathcal{L} \sim Dist(\mathcal{L})}\left[Pr\left(\mathcal{M}_\mathcal{D}^{\pi_t} \models \varphi\right)\right] = \frac{1}{N}\sum_{i=1}^{N} Pr\left(\mathcal{M}_\mathcal{D}^{\pi_t} \models \varphi | \mathcal{L}_i\right),$$

where $\mathcal{L}_i$ are samples drawn from $Dist(\mathcal{L}) = \mathcal{L}_t$. By the application of Hoeffding's inequality, we establish the following concentration result.

Theorem 2. *Let $f(\mathcal{L}_i) = Pr\left(\mathcal{M}_\mathcal{D}^{\pi_t} \models \varphi | \mathcal{L}_i\right)$ denote the output of verification for an environment modeled by $\mathcal{L}_i$. The empirical expectation of $\mathbb{E}_{\mathcal{L} \sim Dist(\mathcal{L})}\left[f(\mathcal{L})\right]$ with N samples has the following concentration bound*

$$Pr\left(\left|\mathbb{E}_{\mathcal{L} \sim Dist(\mathcal{L})}\left[f(\mathcal{L})\right] - \frac{1}{N}\sum_{i=1}^{N} f(\mathcal{L}_i)\right| \geq \epsilon\right) \leq 2\exp(-2N\epsilon^2).$$

It is worth noting that $Pr\left(\mathcal{M}_\mathcal{D}^{\pi_t} \models \varphi | \mathcal{L}_i\right)$ itself is the output of a system of linear equations (Baier & Katoen, 2008) that

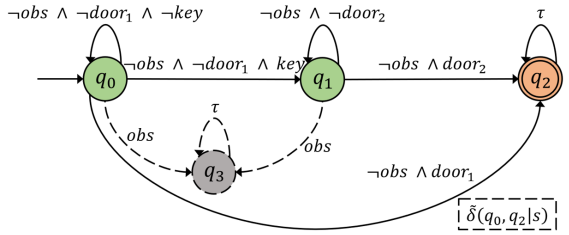


Figure 3. Total probabilistic finite automaton for the formula $\varphi = (\neg obs\, \mathcal{U}\, door_1) \vee ((\neg door_2\, \mathcal{U}\, key) \wedge (\neg obs\, \mathcal{U}\, door_2))$. State q_3, a sink state, and the transitions to it have been added to make the automaton *total*. The transitions between the states of the automaton are probabilistic where the probabilities depend on the belief over the states' propositions. For example, $\bar{\delta}(q_0, q_2|s)$ indicates how $\mathcal{L}_t(s, .)$ determines the probability of the transition between q_0 and q_2.

depend on the sampled labeling. Therefore, further characterization of $f(\mathcal{L}_i)$ such as bounding its higher moments is very difficult. $Pr\left(\mathcal{M}_{\mathcal{D}}^{\pi_t} \models \varphi | \mathcal{L}_i\right)$ can also be computed via statistical verification techniques by sampling paths over the Markov chain (Agha & Palmskog, 2018).

For a policy π_t, we define a risk parameter

$$\mathcal{R}(\mathcal{M}_{\mathcal{D}}, \pi_t, \mathcal{L}_t, \varphi) = \Big| Pr(\mathcal{M}_{\mathcal{D}}^{\pi_t} \models \varphi \,|\, \hat{\mathcal{L}}_t) - \mathbb{E}_{\mathcal{L} \sim Dist(\mathcal{L})} \left[Pr\left(\mathcal{M}_{\mathcal{D}}^{\pi_t} \models \varphi\right) \right] \Big|, \tag{2}$$

which accounts for the variation in the task realization guarantee of the policy with respect to the perception uncertainty. Another input parameter to the proposed perception and planning algorithm is a threshold γ_r on the risk due to perception uncertainty. If γ_r is not exceeded, the agent acts according to the computed policy π_t. Otherwise, it takes an active perception strategy as explained next.

4.5. Active Perception Strategy

We develop an algorithm to compute an active perception strategy in the form of a sequence of actions that the agent follows to reduce its perception uncertainty. We consider three criteria for computing an active perception strategy. First, such a strategy should enable the agent to reduce its uncertainty about the value of the propositions that affect the task progress. These propositions are the ones that enable the transitions from the current stage of the task, i.e., state of the automaton, to the next ones. For example in Figure 3, if the agent is at state q_1, the propositions that matter are obs and $door_2$. To measure the uncertainty reduction, we use expected entropy of the said propositions over the whole state space. Second, an active perception strategy must not affect the stage of the task and so, the agent has to remain in the same state of the automaton. Third, after the agent completes the sequence of actions, it should be able to return to the point from which it started the active perception strategy.

Based on these criteria, we propose Algorithm 1 to construct an active perception strategy. This algorithm takes a bound $C_{\mathcal{A}}$ on the number of actions and uses that to construct a tree of depth $C_{\mathcal{A}}$. Each node of the tree has four parameters: distribution over the states of the MDP, distribution over the states of the DFA, expected entropy reduction, and reachability probability back to the root node. The distribution over the states of the MDP depends on its transitions while the distribution over the states of the DFA depend on the agent's belief over the propositions, as shown in Figure 3. Once the tree is generated, the algorithm picks the best node using a hyperparameter β that weighs safety versus information quality. Safety refers to the ability of the agent to remain in the same state of the automaton as well as its ability to return to the root node while information quality refers to the amount of entropy reduction. The sequence of actions leading to the optimum node with respect to a combination of safety and information quality results in the active perception strategy. After following this sequence of actions, the perception-planning loop starts over. Further details of different steps as well as analysis of the proposed algorithm are provided in the supplementary material.

5. Simulation Results

We evaluate the proposed task-oriented active perception and planning algorithm in two simulation domains. The first one is a discretized planar environment and the second one is an urban environment in the AirSim (Shah et al., 2017) platform. Simulation videos, additional simulation results, and link to the implementation files are provided in the supplementary material.

5.1. Planar Navigation with Finite-Horizon Tasks

In the first set of simulations, we consider an agent that navigates in a discretized 2D environment and has a finite-horizon task. For instance, the task encoded as a DFA in Figure 3 asks the agent to either go to the state where $door_1$ is located or find a key and go to the state where $door_2$ is located, while avoiding the obstacles. We implemented different versions of the task-oriented perception and planning algorithm to evaluate the effect of each module on the performance.

Table 1 reports the results for a reach-avoid task in an environment with 64 states and with randomly generated obstacles and target. In the table, *No perc.* refers to a baseline scenario where the agent estimates the environment configuration with its prior knowledge and plans according to that. *Perc. w/ no update* is a perception strategy that incorporates

Algorithm 1 Constructing an Active Perception Strategy

Input: A product MDP $\mathcal{M}_{\mathcal{D}} = (\mathcal{S}_{\mathcal{D}}, s_{init\mathcal{D}}, \mathcal{A}_{\mathcal{D}}, \mathcal{T}_{\mathcal{D}})$, set of atomic propositions $\mathcal{AP}$, an observation model $\mathcal{O}$, a bound on number of actions $C_{\mathcal{A}}$, and a weighting parameter β
Output: A sequence of actions
Initialize $G = (V, E)$ with $V = \{v_0 = (b_0^s = [(s_t, 1)], b_0^q = [(q_t, 1)], \Delta e_0 = 0, r_0 = 1)\}$ and $E = \{\}$, $V_{exp} = v_0$, and $k = 0$
for $j = 0, \ldots, C_{\mathcal{A}}$ **do**
  $V_{add} = \{\}$
  for $v \in V_{exp}$ **do**
    for $a \in \mathcal{A}$ **do**
      $k \leftarrow k + 1$
      Compute belief b_k^s over the next state of the MDP
      Compute belief b_k^q over the next state of the DFA
      Compute expected entropy reduction Δe_k
      Compute reachability probability r_k back to s_t
      $v_k = (b_k^s, b_k^q, \Delta e_k, r_k)$
      $V_{add} \leftarrow V_{add} \cup v_k$
      $E \leftarrow E \cup \{(v, v_k)\}$
    end for
  end for
  $V \leftarrow V \cup V_{add}$
  $V_{exp} \leftarrow V_{add}$
end for
Return the sequence of actions leading to $v^* = \text{argmax}_{l \in [k]} \Delta e_l + \beta(b_l^q(q_t) + r_l)$

only the most recent perception output. *Perc. w/ update* is perception with a Bayesian update, as described in Section 4. Except the first algorithm, all the other ones have a replanning module, however, the ones with *div.* replan only if the divergence threshold over the change in the belief is exceeded. *info.* means that active perception is enabled and hence, the agent will perform active perception strategies when the risk due to perception uncertainty is high. The results show that adding the divergence test, reduces the number of policy synthesis steps. Furthermore, the divergence test reduces the success rate. On the other hand, adding the active perception module increases the success rate. Figure 4 depicts the results from a risk assessment step for the MDP with 64 states and 2 atomic propositions. Even though the size of the sampling space is large (2^{64}), it can be seen that the empirical expectation of the reachability probability quickly converges with about 20 samples.

5.2. Drone Navigation in Simulated Urban Environment

In the AirSim (Shah et al., 2017) simulator, we designed an urban environment and tasked a drone to fly from an initial state to a specific flagged building while avoiding collision with other entities of the environment. The drone is equipped with 4 cameras and 4 depth sensors. The perception module processes the cameras' readings as well as the depth measurements to map the semantic labels to a discretized model of the environment. We applied the proposed perception and planning scheme. However, in contrast to the previous simulation scenario, an observation model does not exist here. Therefore, we used a frequentist update rule for the agent's belief. Details of the simulation setting, as well as the recordings of the resulting behavior of the drone are presented in the supplementary material.

Table 1. Results of planar navigation under a reach-avoid task using different versions of the proposed algorithm. Success is the percentage of runs that complete the task. #Step is the average length of the runs and #Plan is the number of times that the agent synthesizes a new policy.

Algorithm	Success	#Step	#Plan
No perc.	0%	50	1
Perc. w/ no update + replan	0%	38.4	38.4
Perc. w/ update + replan	84%	21.8	21.8
Perc. w/ update + div.	80%	22.8	14.8
Perc. w/ update + replan + info.	92%	19.4	19.4
Perc. w/ update + div. + info.	86%	22.6	14.6

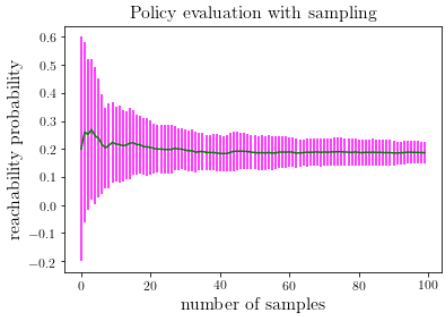


Figure 4. Statistical verification of an induced Markov chain with uncertain atomic propositions.

6. Conclusion

In this paper, we studied settings where an agent aims to satisfy a task expressed as a syntactically co-safe linear temporal logic specification under partial knowledge about the state of the environment. We modeled the partial knowledge of the agent about the environment through probabilistic semantic variables of the states. We proposed a task-oriented perception and planning algorithm that integrates the process of information gathering with that of control design. The proposed algorithm enables the agent to process new information from the perception module and plan according to the best interpretation of the current knowledge. Furthermore, it equips the agent with a module to assess the risk

due to imperfect perception and if the risk is identified as high, it generates a task-related active perception strategy.

As part of future work, we plan to relax the independence assumption of the atomic propositions and instead, consider spatial and causal relations between them. Inclusion of such relations will increase the complexity of different steps of the algorithm including belief update, the maximum a posteriori inference over the belief, as well as finding an active perception strategy. However, it will enable the agent to incorporate side knowledge in learning the environment model.

Acknowledgements

This work was supported in part by ARL grant ACC-APG-RTP W911NF, DARPA grant D19AP00004, and ONR grant N00014-18-1-2829. We thank the reviewers for their valuable feedback.

References

Agha, G. and Palmskog, K. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 28(1):1–39, 2018.

Bai, H., Hsu, D., and Lee, W. S. Integrated perception and planning in the continuous space: A POMDP approach. *The International Journal of Robotics Research*, 33(9): 1288–1302, 2014.

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

Benenson, R., Petti, S., Fraichard, T., and Parent, M. Integrating perception and planning for autonomous navigation of urban vehicles. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 98–104. IEEE, 2006.

Chatterjee, K., Chmelik, M., and Tracol, M. What is decidable about partially observable Markov decision processes with ω-regular objectives. *Journal of Computer and System Sciences*, 82(5):878–911, 2016.

da Silva, R. R., Kurtz, V., and Lin, H. Active perception and control from temporal logic specifications. *IEEE Control Systems Letters*, 3(4):1068–1073, 2019.

Ding, X. C. D., Smith, S. L., Belta, C., and Rus, D. LTL control in uncertain environments with probabilistic satisfaction guarantees. *IFAC Proceedings Volumes*, 44(1): 3515–3520, 2011.

Fu, J. and Topcu, U. Integrating active sensing into reactive synthesis with temporal logic constraints under partial observations. In *2015 American Control Conference (ACC)*, pp. 2408–2413. IEEE, 2015.

Fu, J. and Topcu, U. Synthesis of joint control and active sensing strategies under temporal logic constraints. *IEEE Transactions on Automatic Control*, 61(11):3464–3476, 2016.

Fu, J., Atanasov, N., Topcu, U., and Pappas, G. J. Optimal temporal logic planning in probabilistic semantic maps. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3690–3697. IEEE, 2016.

Ghasemi, M. and Topcu, U. Perception-aware point-based value iteration for partially observable Markov decision processes. In *Proceedings of the International Joint Conference on Artificial Inteligence (IJCAI)*, pp. 2371–2377, 2019.

Guez, A. and Pineau, J. Multi-tasking SLAM. In *2010 IEEE International Conference on Robotics and Automation*, pp. 377–384. IEEE, 2010.

Guo, M., Johansson, K. H., and Dimarogonas, D. V. Revising motion planning under linear temporal logic specifications in partially known workspaces. In *2013 IEEE International Conference on Robotics and Automation*, pp. 5025–5032. IEEE, 2013.

Hahn, E. M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., and Wojtczak, D. Omega-regular objectives in model-free reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 395–412. Springer, 2019.

Icarte, R. T., Klassen, T., Valenzano, R., and McIlraith, S. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pp. 2107–2116, 2018.

Icarte, R. T., Waldie, E., Klassen, T., Valenzano, R., Castro, M., and McIlraith, S. Learning reward machines for partially observable reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 15523–15534, 2019.

Jones, A., Schwager, M., and Belta, C. Distribution temporal logic: Combining correctness with quality of estimation. In *52nd IEEE Conference on Decision and Control*, pp. 4719–4724. IEEE, 2013.

Kress-Gazit, H., Fainekos, G. E., and Pappas, G. J. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009.

Kupferman, O. and Vardi, M. Y. Model checking of safety properties. *Formal Methods in System Design*, 19(3): 291–314, 2001.

Lahijanian, M., Maly, M. R., Fried, D., Kavraki, L. E., Kress-Gazit, H., and Vardi, M. Y. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Transactions on Robotics*, 32(3): 583–599, 2016.

Li, X., Vasile, C.-I., and Belta, C. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3834–3839. IEEE, 2017.

Livingston, S. C., Murray, R. M., and Burdick, J. W. Backtracking temporal logic synthesis for uncertain environments. In *2012 IEEE International Conference on Robotics and Automation*, pp. 5163–5170. IEEE, 2012.

Montana, F. J., Liu, J., and Dodd, T. J. Sampling-based reactive motion planning with temporal logic constraints and imperfect state information. In *Critical Systems: Formal Methods and Automated Verification*, pp. 134–149. Springer, 2017.

Puterman, M. L. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.

Sharan, R. and Burdick, J. Finite state control of POMDPs with LTL specifications. In *Proceedings of American Control Conference*, pp. 501–508. IEEE, 2014.

Singh, S. P. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the National Conference on Artificial Intelligence*, number 10, pp. 202. JOHN WILEY & SONS LTD, 1992.

Vasile, C. I. and Belta, C. Sampling-based temporal logic path planning. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4817–4822. IEEE, 2013.

Wen, M., Ehlers, R., and Topcu, U. Correct-by-synthesis reinforcement learning with temporal logic constraints. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4983–4990. IEEE, 2015.

Wongpiromsarn, T., Topcu, U., and Murray, R. M. Receding horizon temporal logic planning for dynamical systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 5997–6004. IEEE, 2009.