# Leveraging Temporal Structure in Safety-Critical Task Specifications for POMDP Planning

Jason Liu, Eric Rosen, Suchen Zheng, Stefanie Tellex, George Konidaris
Computer Science Department, Brown University

***Abstract*—Navigating a partially observable environment while satisfying temporal and spatial constraints is an essential safety feature of many robotic applications. For example, an autonomous drone needs to understand the command "Find the supermarket while avoiding the park" to avoid possible collisions with trees. Previous approaches chose to sacrifice generality for computational efficiency in large state spaces by designing action heuristics that do not apply across different tasks or used a value-iteration-based planner that does not scale well. Our approach automatically extracts structured rewards from linear temporal logic (LTL) task specifications to guide a sampling-based POMDP planner, named LTL-POMCP. We augment a partially observable Markov decision process (POMDP) with an LTL task specification then use LTL-POMCP to solve the resultant composite POMDP. Quantitative results from a classic POMDP domain show that LTL-POMCP can generalize to various LTL task specifications and scale to large state spaces. We then demonstrate the first end-to-end system from temporally-constrained natural language to robot policies in partially observable maps in simulation.**

## I. Introduction

Navigating partially observable environments by following natural language commands that specify goals and path constraints is an essential safety feature of robots interacting with humans. For example, in a search and rescue mission, first responders can command an autonomous drone by saying "Search for survivors while avoiding the explosion at location A." We can model this temporally constrained navigation problem as a partially observable Markov decision process (POMDP), whose reward is specified by a linear temporal logic (LTL) expression [13].

Previous work [2] solved this problem with a value-iteration-based planning algorithm and demonstrated its performance in small environments. It could not scale because full-width Bellman backups are intractable in large state and action spaces due to the curse of dimensionality and the curse of history. To overcome these challenges, Silver and Veness [15] proposed POMCP, a sampling-based planner. Instead of estimating the value function via iterative applications of the Bellman equation using an exact model, sampling-based methods use Monte-Carlo simulations to estimate action values from interactions with a generative model of the environment. However for different task specifications, POMCP requires different hand-specified action priors to bias the exploration. These heuristics use observations cached during simulations to decide the best action to take next. Our approach does not need domain experts to design heuristics to solve a task.

This work uses an LTL to specify a safety-critical navigation task and translates the LTL to a deterministic finite automaton (DFA) [14] that encodes terminal and subgoal rewards. The DFA and the environment POMDP are combined to construct an LTL-POMDP problem. To solve it in large domains, we propose LTL-POMCP, a sampling-based planner with an additional term added to its action value estimates to bias the sampling of actions that likely lead to subgoals specified by the DFA. During Monte-Carlo simulations, besides keeping track of action value estimates, visitation counts of states and actions, LTL-POMCP caches all DFA transitions occurred after taking an action in the current history state. It then uses the augmented action value estimates to select the next action.

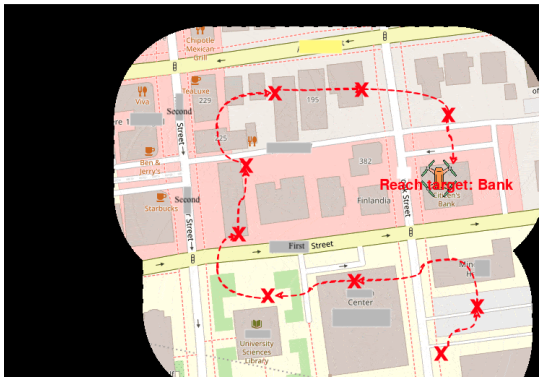



Fig. 1: An illustration of object search in the OpenStreetMap simulator. The natural language command is "Stay off the 2nd Street while looking for a bank." Its corresponding LTL formula is "G(!street2) & F(bank)." The agent has partial observability of the target bank, demonstrated by the fog-of-war effect, and perfect observation of the 2nd Street.

Results from a classic POMDP domain show that LTL-POMCP is more generalizable across task specifications than POMCP with hard-coded heuristics [15] and faster than a planner based on value iteration [2] in solving LTL-POMDPs with large state spaces. We then demonstrate the first end-to-end system from temporally-constrained natural language to robot behavior in partially observed maps [11].

The main contributions of this work are as follows.

- A POMDP formulation, LTL-POMDP, that automatically extracts structured rewards from LTL task specifications.
- A sampling-based POMDP planner, LTL-POMCP, that leverages the structured rewards, generalizes across tasks and scales well.
- An end-to-end system from temporally-constrained nat-

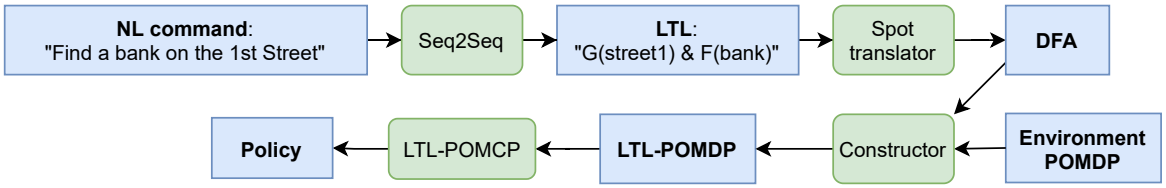



Fig. 2: **End-to-End System for PO-OSM.** Natural language is translated to an LTL then a DFA. The DFA and the environment POMDP are composed to construct an LTL-POMDP, which is solved by the LTL-POMCP online to produce a policy.

ural language to robot behavior in partially observed OpenStreetMap (PO-OSM) domain.

## II. Related Work and Background

A large body of research has investigated the usage of LTL task specifications in fully observable domains. [7][9][10][1][12] studied navigation in fully observed environments while enforcing temporal constraints. We consider a more challenging partially observable setting, where an agent must actively plan to gather information. [5] proposed to learn LTL constraints from multi-step demonstrations to facilitate robotic manipulation in fully observed domains. [16] required domain experts to define reward machines that specifies goals and temporal constraints. A reward machine is a fully observable version of LTL-POMDP.

Bouton et al. [2] solved the same LTL-POMDP problem with a value-iteration-based planner for small environments. [3] requires perfect local perception to partition the environment into known and unknown areas and does not maintain a probability distribution over states in the unknown area, thus the planning is only done in a fully observable area to solve an LTL task. Silver and Veness [15] solved large POMDPs with a sampling-based planner and heuristics defined for specific tasks. Our approach extracts subgoal rewards from LTLs to guide the action selection of a sampling-based POMDP planner. This work can be generalize to solve any LTL-POMDP problem with a generative model of the environment and noisy sensors.

**Linear Temporal Logic (LTL):** We use LTLs [13] to specify tasks because they can represent both goals and temporal constraints.

LTL has the following syntax:

$$\phi := \sigma \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid X\phi \mid F\phi \mid G\phi \mid \phi U \psi, \quad (1)$$

where $\phi$ and $\psi$ are LTL formulas; $\sigma \in \Sigma$ is an atomic proposition. $\neg, \wedge, \vee$ are logic connectives negation, conjunction and disjunction. LTL extends propositional logic with temporal operators, $X$ (next), $F$ (finally), $G$ (globally or always) and $U$ (until), applying to future time steps. We evaluate the satisfaction of an LTL formula on an infinite sequence $w = w_0 w_1 \ldots$, where $w_i \in 2^{\Sigma}$. $X\phi$ is satisfied by $w$ at step $i$ if $\phi$ is satisfied at the next step $i+1$. $F\phi$ is true at step $i$ if $\phi$ holds true at some future time $j \geq i$. $G\phi$ holds if $\phi$ is true for the entire sequence. $\phi U \psi$ is satisfied if $\phi$ holds true at least until $\psi$ becomes true, which must happen at the current or a future time. Table I shows examples of LTL formulas in the PO-OSM domain. For example, to satisfy "G(street1) & F(bank)," a robot needs to stay on the 1st Street while looking for a bank.

**Deterministic Finite Automaton (DFA):** We use the Spot library [6] to translate an LTL formula to an equivalent DFA [4]. A DFA is a 5-tuple $\mathrm{D} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, where $\mathcal{Q}$ is a finite set of states; $\Sigma$ is a finite set of atomic propositions; $\delta : \mathcal{Q} \times 2^{\Sigma} \to \mathcal{Q}$ is a deterministic transition function; $q_0 \in Q$ is the initial state; $\mathcal{F} = \mathcal{F}_{\text{success}} + \mathcal{F}_{\text{fail}} \subseteq \mathcal{Q}$ is a set of success and failure terminal states. A run on a finite sequence $w = w_0 w_1 \ldots w_n$ with $w_i \in 2^{\Sigma}$ produces a sequence of states $q_0 q_1 \ldots q_n$ with $q_t \in \mathcal{Q}$, where $q_0$ is the initial state, $q_n \in \mathcal{F}$ is a final state and $q_{t+1} = \delta(q_t, w_t)$. Table I shows some examples of LTL formulas and their corresponding DFAs.

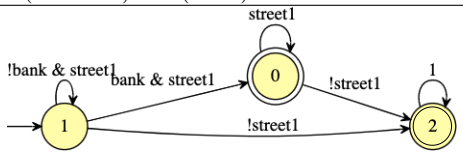
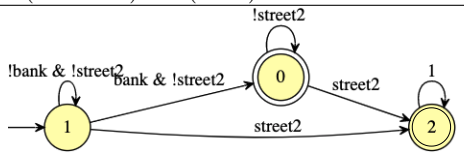
**Environment POMDP:** We model the environment as a Partially Observable Markov Decision Process (POMDP). A POMDP is defined by a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma)$. The dynamics $T(s, a, s') = P(s'|s, a)$ and $R(s, a) = E[r|s, a]$ determine the distribution of the next state $s' \in \mathcal{S}$ and the immediate reward after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. In POMDP, states cannot be fully observed. Instead the agent receives an observation $o \in \mathcal{O}$ based on an observation model $O(a, s', o) = P(o|a, s')$. A policy of a POMDP is define by $\pi(h) = a$, where $h$ is a history of actions and observations. Any POMDP has at least one optimal policy $\pi^*$ that maximizes $V^{\pi}(h) = E_{\pi}[\sum_{t}^{\infty} \gamma^{t-1} r_t | h]$. A belief state is a probability distribution over possible states given the history, $B(s, h) = P(s|h)$, and it is Markovian. We define our environment POMDP to be generative such that given a transition $(s, a, s')$, we can sample from models $T, R$ and $O$ to get the next state, an immediate reward and an observation. A belief state is represented by a set of particles.

## III. Technical Approach

This section provides technical details on how we augment an environment POMDP with a DFA to construct an LTL-POMDP. We will also describe LTL-POMCP, a sampling-based planner that leverages structured rewards provided by the DFA, generalizes across LTL tasks and scales well.

**LTL-POMDP:** We augment an environment POMDP with a DFA, so states of the resultant LTL-POMDP are Markovian and encoding progressions in the DFA. LTL-POMDP $= (\tilde{\mathcal{S}}, L, \mathcal{A}, \mathcal{O}, \tilde{T}, \tilde{O}, \tilde{R}, \gamma)$, where $\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Q}$ is a Cartesian product of environment POMDP and DFA state spaces; $L : \mathcal{S} \to 2^{\Sigma}$ is a labeling function that maps environment POMDP

TABLE I: Examples of LTL formulas and their corresponding DFAs.

| Language | Find a bank while staying on the 1st Street. | Avoid the 2nd Street while looking for a bank. |
|---|---|---|
| LTL | $G(\texttt{street1})\ \&\ F(\texttt{bank})$ | $G(!\texttt{street2})\ \&\ F(\texttt{bank})$ |
| DFA |  |  |



states to atomic propositions.

The transition probability of entering LTL-POMDP state $\tilde{s}' = (s', q')$ after taking action $a$ from state $\tilde{s} = (s, q)$ is

$$\tilde{T}(\tilde{s}, a, \tilde{s}') = \begin{cases} T(s, a, s'), & \text{if } q' = \delta(q, L(s')) \\ 0, & \text{otherwise} \end{cases}. \quad (2)$$

As shown in the Table I, an example LTL-POMDP transition can be that an agent takes an action in the environment to reach a bank while avoiding the 2nd Street, which induces the DFA transition from $q = 1$ to the goal $q = 0$.

The observation model of LTL-POMDP is

$$\tilde{O}(\tilde{s}, a, \tilde{s}') = O(s, a, s'). \quad (3)$$

The structured reward function is specified by the LTL progression

$$\tilde{R}(\tilde{s}, a, \tilde{s}') = \begin{cases} r_{\text{goal}}, & \text{if } q' \in \mathcal{F}_{\text{success}} \\ r_{\text{fail}}, & \text{if } q' \in \mathcal{F}_{\text{fail}} \\ r_{\text{subgoal}}, & \text{if } q' \in \mathcal{Q} - \mathcal{F} \wedge q' \neq q \\ r, & \text{otherwise} \end{cases} \quad (4)$$

where $r_{\text{goal}} \gg 0$, $r_{\text{fail}} \ll 0$, $r_{\text{subgoal}} > 0$ and $r < 0$.

**LTL-POMCP:** LTL-POMDPs model POMDP planning problems with temporally constrained task specifications. We introduce LTL-POMCP, a sampling-based planner that can solve LTL-POMDP problems in large environments.

We adopt the POMCP algorithm [15] with two modifications. In addition to the estimated $Q$-values $\hat{Q}(ha)$, visitation counts $N(h)$ and $N(ha)$, LTL-POMCP caches the frequencies of the DFA transitions occurred after taking action $a$ in the current history state $h$ during the Monte-Carlo simulation. We then augment the $Q$-value estimates with an additional third term as follows,

$$Q(ha) = \hat{Q}(ha) + \alpha\sqrt{\frac{\log N(h)}{N(ha)}} + \beta\sqrt{\frac{\log N(e_a)}{N(ha)}}, \quad (5)$$

where $e_a$ represents a DFA transition towards a final goal state in the future trajectory after taking the action $a$ from the current history state $h$; $\alpha$ and $\beta$ are coefficients. The LTL-POMCP algorithm leverages high-level subgoals encoded in the DFA and automatically favors the transitions leading to the DFA goal state without explicitly constructing preferred action sets for rollout as in [15]. Equation 5 balances exploring less taken actions and exploiting actions that have led to a DFA transitions and high rewards. The third term in Equation 5 diminishes to 0 asymptotically because logarithm grows slower than linear, and $N(e) \leq N(ha)$.

**Translating Natural Language to LTL:** The language model first uses a pretrained name entity recognizer (NER) by [8] to replace all landmark names from a natural language command by place holders, then feeds the masked language into a sequence to sequence (Seq2Seq) model with LSTM cells, and finally substitute the landmark names in the output LTL expression. With the help of NER, the Seq2Seq model only needs to memorize LTL templates, not landmark names. Because NER was pretrained on a very large dataset of landmark names, this language model can recognize places unseen in the training set. It takes 118 seconds, 443 data points and 10 epochs to train the Seq2Seq model to achieve 100% accuracy on the test set.

## IV. Experiments

The aim of our experiments is to test the hypothesis that the LTL-POMCP planner is more general than POMCP used with hard-coded action heuristics [15] and more scalable than value-iteration-based planners [2] in the RockSample domain. We also show an end-to-end system from temporally-constrained language to navigation policy in partially observed maps.

**RockSample:** A RockSample problem $RS(n, k)$ has $k$ rocks randomly placed in an $n \times n$ grid, $k + 5$ actions (i.e. 4 move, 1 pick up and $k$ sensing actions), 2 observations of rock types (i.e. good, bad) and deterministic transitions. The initial belief is uniformly distributed over rock types. Sensing accuracy decreases exponentially in the distance to a rock.

To compare generality, we measure the success rate and total reward of solving different tasks on the same RockSample domains using LTL-POMCP and Silver POMCP [15]. The first task requires the robot to pick up a good rock then go to exit area while avoiding bad rocks. The second task requires the robot to pick up a bad rock then exit while avoiding good rocks. We use LTL expressions, "G(! bad) & F(good & F(exit))" and "G(! good) & F(bad & F(exit))" representing both tasks respectively. As shown in Figure 3, because the action heuristics used by Silver POMCP are defined for the first task, it achieves higher success rate and more rewards. But for the second task, Silver POMCP performs even worse than basic POMCP without heuristics because the same action heuristics used to solve the first task direct the agent to pick up unfavorable rocks. LTL-POMCP can sovle both LTL tasks with comparable performance. This shows that LTL-POMCP is generalizable across different LTL task specifications.

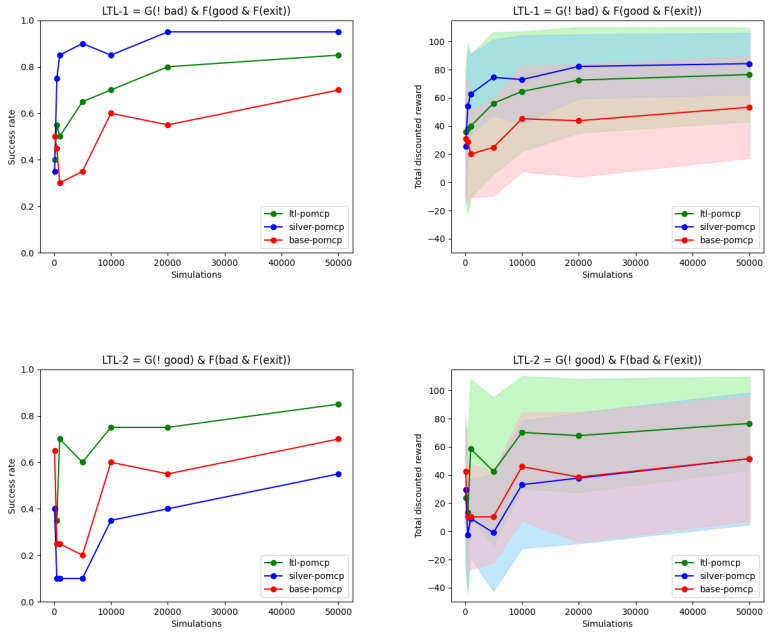



Fig. 3: The top row plots are the success rate and reward vs. the number of simulations for the LTL G(!bad) & F(good & F(exit)). The bottom row plots are for G(!good) & F(bad & F(exit)). Each data point is average over 20 runs.

To compare scalability of LTL-POMCP and SARSOP used in [2], we measure the planning time in large domain $RS(11, 11)$. An value-iteration-based-planner SARSOP provided with sparse LTL rewards [2] cannot produce a policy within 96 hours. LTL-POMCP can solve the problems constantly within 2 hours, and its speed can be further improved by using a compiled language and parallelization.

**Partially Observable OpenStreetMap (PO-OSM):** In PO-OSM, the locations of major landmarks, e.g. streets, are known, and the locations of small landmarks, e.g. banks, are unknown. It mimics the real world scenarios where some landmarks, like a new construction site or disaster area, are not stored in a map database, and an autonomous agent needs to locate or avoid them by following natural language commands from humans. We map landmarks from the OpenStreetMap database to a $20 \times 20$ grid. The state contains the agent's pose and target landmark locations. The initial belief is uniformly distributed over possible landmark locations. The agent can rotate in 4 cardinal directions and move forward. Every step, the agent receives a noisy observation of whether the target landmark is within its fan-shaped sensing range. We use a deterministic transition function and a non-deterministic observation model for computational reasons. They are also realistic assumptions of the drones because existing drones can reliably move around the environment but have less reliable sensors.

The end-to-end system from temporally-constrained natural language can consistently produce navigation policies to complete different tasks as shown in Table II.

TABLE II: Examples of LTL tasks that LTL-POMCP can solve and their corresponding natural language commands in PO-OSM.

| Language | LTL |
|---|---|
| Find a bank. | $F(\texttt{bank})$ |
| Find a store. | $F(\texttt{store})$ |
| Find a cafe. | $F(\texttt{cafe})$ |
| Stay on the 1st Street, and find a bank. | $G(\texttt{st1})\&F(\texttt{bank})$ |
| Find a store while staying on the 2nd Street. | $G(\texttt{st2})\&F(\texttt{store})$ |
| Stay on the 2nd Street while looking for a cafe. | $G(\texttt{st2})\&F(\texttt{cafe})$ |
| Avoid the 2nd Street while looking for a bank. | $G(!\texttt{st2})\&F(\texttt{bank})$ |
| Find a store and avoid the 1st Street. | $G(!\texttt{st1})\&F(\texttt{store})$ |
| Look for a cafe while avoiding the 1st Street. | $G(!\texttt{st1})\&F(\texttt{cafe})$ |
| Fly on the 1st Street until you find a bank. | $\texttt{st1}\mathcal{U}\texttt{bank}$ |
| Be on the 2nd Street until you find a store. | $\texttt{st2}\mathcal{U}\texttt{store}$ |
| Stay on the 2nd Street until you find a cafe. | $\texttt{st2}\mathcal{U}\texttt{cafe}$ |
| Avoid the 2nd Street until you find a bank. | $(!\texttt{st2})\mathcal{U}\texttt{bank}$ |
| Avoid the 1st Street until you find a store. | $(!\texttt{st1})\mathcal{U}\texttt{store}$ |
| Stay off the 1st Street until you find a cafe. | $(!\texttt{st1})\mathcal{U}\texttt{cafe}$ |

## V. Conclusion

We introduced a generalizable planner that can solve LTL-POMDP problems in large environments and demonstrated an end-to-end system from temporally-constrained natural language to robot behavior in the partially observed OpenStreetMap domain.

The automatic extraction of structured rewards from LTLs and the planner are generic, and they are are not limited to solve only navigation tasks. One interesting future work is to apply LTL-POMCP to mobile manipulation tasks specified by LTL.

## References


[1] Matthew Berg, Deniz Bayazit, Rebecca Mathew, Ariel Rotter-Aboyoun, Ellie Pavlick, and Stefanie Tellex. Grounding Language to Landmarks in Arbitrary Outdoor Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[2] Maxime Bouton, Jana Tumova, and Mykel J. Kochenderfer. Point-based methods for model checking in partially observable Markov decision processes. 2020. doi: https://doi.org/10.1609/aaai.v34i06.6563.

[3] Christopher Bradley, Adam Pacheck, Gregory Stein, Sebastian Castro, Hadas Kress-Gazit, and Nicholas Roy. Learning and planning for temporally extended tasks in unknown environments. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[4] J Richard Büchi. On a decision method in restricted second order arithmetic. In *The collected works of J. Richard Büchi*, pages 425–435. Springer, 1990.

[5] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Explaining multi-stage tasks by learning temporal logic formulas from suboptimal demonstrations. In *Proceedings of Robotics: Science and Systems (RSS) XVI*, 2020.

[6] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 — a framework for LTL and $\omega$-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, October 2016. doi: 10.1007/978-3-319-46520-3_8.

[7] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2020–2025. IEEE, 2005.

[8] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

[9] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341*, 2017.

[10] Yoonseon Oh, Roma Patel, Thao Nguyen, Baichuan Huang, Ellie Pavlick, and Stefanie Tellex. Planning with State Abstractions for Non-Markovian Task Specifications. In *Proceedings of Robotics: Science and Systems*, Freiburg, Germany, June 2019.

[11] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[12] Roma Patel, Ellie Pavlick, and Stefanie Tellex. Grounding language to non-markovian tasks with no supervision of task specifications. In *Proceedings of Robotics: Science and Systems*, June 2020.

[13] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE.

[14] Michael O Rabin and Dana Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.

[15] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[16] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2112–2121, 2018.