JMLR: Workshop and Conference Proceedings 21:1–16, 2012 The 11th International Conference on GI

Integrating Grammatical Inference into Robotic Planning*

Jane Chandlee JANEMC@UDEL.EDU
Jie Fu JIEFU@UDEL.EDU
Konstantinos Karydis KKARYD@UDEL.EDU
Cesar Koirala KOIRALA@UDEL.EDU
Jeffrey Heinz HEINZ@UDEL.EDU
Herbert Tanner BTANNER@UDEL.EDU
University of Delaware

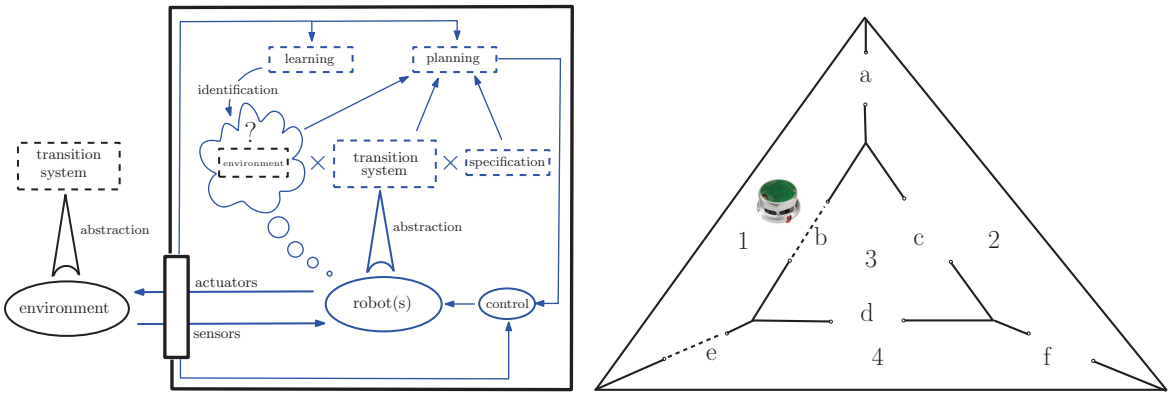
Editors: Jeffrey Heinz, Colin de la Higuera, and Tim Oates

1. Introduction

This paper shows how grammatical inference (GI) and game-theoretic techniques can be jointly utilized for robotic planning. The planning problem is to find a sequence of robot maneuvers so that a desired task is completed; the maneuvers themselves are assumed to be implemented by some existing low-level controllers. The challenge here is that the environment in which the robot is operating is unknown, dynamic, and possibly adversarial, and the role of GI is to enable the robot to learn from experience and improve its planning capability over time. Our hypothesis is that under certain technical conditions, this improvement indeed takes place and can be demonstrated by the robot being able to devise strategies that are *guaranteed* to accomplish the task—despite the dynamic, potentially adversial environment.

Research in robotic planning largely addresses the problem of planning when the environment is static (Piterman and Pnueli, 2006; Belta et al., 2007; Lahijanian et al., 2010; Wongpiromsarn et al., 2010; Kress-Gazit et al., 2011; LaViers et al., 2011). As for learning, in robotics it has been used primarily for adjusting parameters in the robot's models or control laws during the robot's interaction with its environment. The learning methodologies are based primarily on *reinforcement learning*, applied to a wide variety of problems, including multi-agent coordination (Matarić, 1997), walking (Byl and Tedrake, 2009), humanoid robots (Peters et al., 2003), varying-terrain wheeled robot navigation (Brunskill et al., 2009), and unmanned aerial vehicle control (Abbeel et al., 2010). The use of grammatical inference as a learning mechanism in robotics has been limited (with a few exceptions, see Luzeaux (1996); Dean et al. (1992); Rivest and Schapire (1993); Rieger (1995); Schmill et al. (2000); Krishnaswamy et al. (2011); Chen et al. (2012)). The aforementioned learning techniques operate typically on discrete models, like a Markov chain or a transition system. Other research on planning has focused on establishing formal (rather than heuristic) relationships between the concrete domain of continuous system

* We thank the anonymous reviewers for useful questions and suggestions. This research is supported by grant #1035577 from the United States National Science Foundation.



(*a*) The architecture. (*b*) The room visiting game.

Figure 1: The architecture of robotic planning with a module for grammatical inference (a) and the application example in the form of a noncooperative game (b).

dynamics in which behaviors are expressed in terms of differential equations, and the discrete world of automata and transition systems (Tanner et al., 2012; Fainekos et al., 2009; Kloetzer and Belta, 2007; Pola et al., 2008). These relationships are needed to ensure that policies computed at the discrete level can always be implemented at the finer level of continuous dynamics. At the higher level, a planner computes a discrete plan which ensures the completion of the task or the satisfaction of a specification expressed in some formal logic, and then the formal abstraction interfaces translate the plan into control laws for the lower-level continuous dynamics.

This paper assumes the actual low-level dynamics are given, and the challenge is to compose them temporally so that a specification is met in the presence of unknown—but rule-governed—environment dynamics. The proposed solution utilizes GI within the overall architecture depicted in Figure 1(*a*). We envision a *flexible* and *modular* robotic system which includes a grammatical inference module. The robot interacts with its environment through its sensors and actuators. Both the robot and its environment are modeled as hybrid dynamical systems (represented as ovals) and are assumed to admit discrete abstractions in the form of some finite-state system (dashed rectangles). The robotic system (enclosed inside the solid rectangle) has a certain objective, encoded in the form of a task specification. The robot, equipped with an abstraction of its dynamics, the specification, and its perception (theory of mind) of its environment, plans its actions and implements them via some control loops. The same sensory data used by the robot's low level controllers serve as input to the GI module. With this information, the robot refines the abstract model of its environment through a GI algorithm which learns the discrete environment dynamics under some learning criterion. An important contribution of this paper is to present and analyze such a system in the context of well-known and well-understood GI algorithms.

This architecture is illustrated and analyzed with respect to a simple scenario introduced in §3. In this scenario, the robot's task is to visit every room in a building. In Figure 1(*b*) there are four rooms marked with numbers from 1 to 4. The rooms are connected through doors marked with the letters a,...,f. Doors can be open or closed (like e and b in Figure 1(*b*)) operated by an adversary of the robot. There are constraints on how the adversary

operates the doors which are unknown to the robot. Once the robot has learned the nature of these constraints, it outwits the agent whenever a strategy for completing the task exists. The term *strategy* indicates our approach to the planning problem from the viewpoint of game theory: the scenario can be formulated as a two-player zero-sum game and the robot and the adversary are both players. The game theoretic analysis offers algorithms for the robot to make the best planning decisions. This analysis is presented in §3 and its interaction with the GI module is described in §4.

2. Preliminaries

2.1. Languages, Automata and Game Theory

Let Σ denote a fixed, finite alphabet, and Σ^n, $\Sigma^{\leq n}$, Σ^*, Σ^ω be sequences over this alphabet of length n, of length less than or equal to n, of any finite length, and of infinite length, respectively. The *empty string* is denoted λ, and the *length of string* w is denoted $|w|$. A *language* L is a subset of Σ^*. For all $w = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*$, the *shuffle ideal* of w is defined as $SI(w) := \Sigma^*\sigma_1\Sigma^*\sigma_2\cdots\Sigma^*\sigma_n\Sigma^*$. A string u is a *factor* of string w iff $\exists x, y \in \Sigma^*$ such that $w = xuy$. If in addition $|u| = k$ then u is a *k-factor* of w. The function F_k maps words to the set of k-factors within them: $F_k(w) := \{u : u \text{ is a } k\text{-factor of } w\}$. This function is extended to languages as $F_k(L) := \bigcup_{w\in L} F_k(w)$.

A *positive text* S of a language L is a total function $S : \mathbb{N} \to L \cup \{\#\}$ (# is a 'pause') such that for every $w \in L$, there exists $n \in \mathbb{N}$ such that $S(n) = w$. Let S_i denote the first i elements of S. A *learner* ϕ is an algorithm which maps finite initial portions of positive texts to *grammars*. The learner ϕ *identifies a class of languages* $\mathcal{L}$ *in the limit from positive data* iff for all $L \in \mathcal{L}$, for all positive texts S for L, there is some $i \in \mathbb{N}$ such that for all $j > i$, $\phi(S_j) = \phi(S_i)$ is a grammar generating exactly L. A language class with such a ϕ is *identifiable in the limit from positive data*.

A *semiautomaton (SA)* is a tuple $A = \langle Q, \Sigma, T\rangle$ where Q is the set of states, Σ is the set of alphabet symbols and the transition relation is $T : Q \times (\Sigma \cup \{\lambda\}) \to Q$. The transition relation is expanded recursively in the usual way. If $T(q, w) \neq \emptyset$, we write $T(q, \sigma)\downarrow$; otherwise $T(q, \sigma)\uparrow$. A word w is *admissible* in A if there exist $q_1, q_2 \in Q$ such that $T(q_1, w) = q_2$. A *finite state acceptor (FSA)* is a tuple $\mathcal{A} = \langle A, I, F\rangle$ where $A = \langle Q, \Sigma, T\rangle$ is a SA and $I, F \subseteq Q$ are the initial and final states, respectively. The language accepted by the FSA is $L(\mathcal{A}) = \{w \mid T(I, w) \cap F \neq \emptyset\}$. A *discrete event system (DES)* (Cassandras and Lafortune, 1999) is an FSA equipped with an *active event function* $\Gamma : Q \to 2^\Sigma$, which singles out the labels of the defined outgoing transitions at each state: $\Gamma(q) = \{\sigma \mid T(q, \sigma)\downarrow, \sigma \in \Sigma\}$. The activation function is practically redundant, since all the information is included in the transition function; its introduction does, however, simplify notation occasionally. For this reason we will abuse the definition of a SA at places and implicitly equip this machine with an activation function as well.

A *game G(X)* on Σ is a set $X \subset \Sigma^\omega$. A *play* in the game is a word $x = w_0w_1\ldots \in \Sigma^\omega$ in which two players alternate so that player 1 plays w_0, player 2 plays w_1, etc. Player 1 *wins* the play if $x \in X$; otherwise player 2 wins. Given a word $u \in \Sigma^*$, $G_u(X)$ denotes the game that starts at u.A *strategy* for player 1 is a function $f : (\Sigma^2)^* \to \Sigma$ from the set of words of even length into Σ. A strategy for player 2 is a function $g : (\Sigma^2)^*\Sigma \to \Sigma$ from the set of words of odd length into Σ. Strategy f is a *winning strategy* for player 1 if for any infinite

word $x \in \Sigma^\omega$ such that for all $n \geq 0$, $w_{2n} = f(w_0 \dots w_{2n-1})$, $x \in X$. A game is *determined* if one of the players has a winning strategy.

2.2. Hybrid Dynamical Systems

A *hybrid system* is a dynamical process involving states that take values in a continuous domain and evolve according to some differential equations, and states that take discrete values and change based on some discrete logic. A hybrid system H is defined as a tuple of objects (Lygeros et al., 2003) that includes the domains of continuous and discrete variables, the subsets of initial states in those domains, the description of the family of continuous dynamics parameterized by the discrete states, and rules for resetting continuous and discrete states and switching between the members of the family of continuous dynamics.

The present analysis focuses on a specific class of hybrid systems wherein the continuous dynamics have specific (set) attractors, the shape and location of which are dependent on a finite set of parameters that are selected by the system's supervisor (Tanner et al., 2012). By judiciously selecting the parameters, a controller activates a specific sequence of continuous and discrete transitions, steering the hybrid system H from a given initial state to a final desired state. Tanner et al. (2012) describe an abstraction process which derives a SA which is *observably bisimilar* (Stirling, 1996) to H; in other words, the sequences of discrete modes executed by H can be matched by words admissible in the SA, modulo some subset of symbols in Σ that are thought of as *silent*, and vice versa. Moreover, the abstract system that is derived in Tanner et al. (2012) is deterministic in transitions, as ensured by the convergence of vector fields for each discrete mode.

3. Game Theoretic Analysis

3.1. Constructing the game

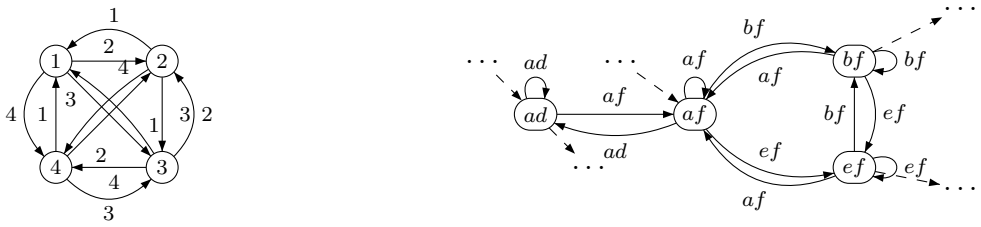
In a game theory formulation the behaviors of the two competing players can be modeled as SAs. Let these SAs be $A_1 = \langle Q_1, \Sigma_1, T_1 \rangle$ for player 1 and $A_2 = \langle Q_2, \Sigma_2, T_2 \rangle$ for player 2.

In the example case study considered here, there are two players: the robot and its adversary. Time is discretized into turns, and agents alternate taking turns and making their moves. During its turn, the robot chooses an adjacent room with an open door to visit. The adversary can open exactly one door and close exactly one other, provided that the two doors closed belong to a predetermined list. This list is a parameter of the game, allowing different configurations of the game to be played (example configurations are shown in Table 1).

Opposite	Only opposite doors can be closed at any time: $\{a,d\}$, $\{a,e\}$, $\{a,f\}$, $\{b,f\}$, $\{c,e\}$, $\{e,f\}$
Adjacent	Only adjacent doors can be closed at any time: $\{a,b\}$, $\{a,c\}$, $\{b,c\}$, $\{b,d\}$, $\{b,e\}$, $\{c,d\}$, $\{c,f\}$, $\{d,e\}$, $\{d,f\}$
General	Any pair of doors can be closed at any time

Table 1: Rules for the adversary (controlling the doors).

The SA for the robot is denoted A_1 and is shown in Figure 2(a), while the behavior of the adversary in the Opposite configuration (Table 1) is captured by A_2 depicted in Figure 2(b). In A_1, there is one state for each room and each transition is labeled with



(a) The robot SA, A_1. (b) The adversary SA (part of) A_2.

Figure 2: Semiautomata for the robot (left) and for a fragment of the adversary (right).

the room the robot is going into. In A_2, the states correspond to pairs of doors currently closed, and a transition label indicates the pair of doors that are subsequently closed.

In the robotics literature, task specifications are typically given in higher-level formalisms which are translated into a Kripke structure (Belta et al., 2007), essentially a finite SA with marked initial states, which is also equipped with a function labeling each state with a set of atomic propositions that are true there (Clarke Jr. et al., 1999). Different task specifications result in different types of games, such as a reachability game where $X \in (\Sigma_1 \cup \Sigma_2)^*$, or a Büchi game where $X \in (\Sigma_1 \cup \Sigma_2)^\omega$.

In the example scenario, the robot needs to visit all four rooms in any order; this specification is expressed as the union of shuffle ideals of the permutations of 1234.[1] A fragment of the FSA $\mathcal{A}_s$ representing this specification is shown in Figure 3. With these

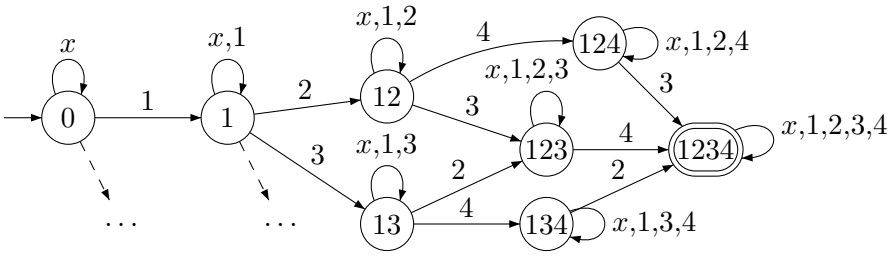


Figure 3: Fragment of the specification automaton $\mathcal{A}_s$, in which $x = \Sigma_2$.

three ingredients in place—A_1, A_2 and $\mathcal{A}_s$—we define what we call the *turn-based product*, through which we construct a bipartite graph that expresses the moves players can make in alternation. The standard product of the bipartite graph with the specification yields the representation of the game.

Note that a move by one player may influence a move of the other; this is captured by the interacting functions $U_i : Q_i \times Q_j \to 2^{\Sigma_j}, (i,j) \in \{(1,2),(2,1)\}$, which for each player i, single out the transitions that *the other player cannot* take. For example, when the adversary closes a door, the robot cannot go through it on its next turn.

1. This whole operation of constructing such an FSA can be fully automated.

Definition 1 (Turn-based product) *Given two semiautomata* $A_1 = \langle Q_1, \Sigma_1, T_1 \rangle$ *and* $A_2 = \langle Q_2, \Sigma_2, T_2 \rangle$ *with interacting functions* U_1, U_2*, the turn-based product* $P = A_1 \circ A_2$ *can be obtained as follows: (i) for each* $A_i, i = 1, 2$*, add a state* 0 *and transitions* $T_i(0, \lambda) = q_i, \forall q_i \in Q_i$*. The set of states in* P *is* $Q_p = (Q_1 \cup \{0\}) \times (Q_2 \cup \{0\}) \times \{0, 1\}$*; (ii)* $\Sigma_1 \cup \Sigma_2$ *is the alphabet; and (iii)* T *is the transition relation defined as follows:*

- $T\big((0,0,c),\sigma\big) = \begin{cases} (T_1(0,\lambda\sigma),0,0) & \text{if } c=1, \sigma \in \Sigma_1 \\ (0,T_2(0,\lambda\sigma),1) & \text{if } c=0, \sigma \in \Sigma_2 \end{cases}$
- $T\big((q,0,0),\sigma\big) = \big(q, T_2(0,\lambda\sigma), 1\big)$ *if* $\sigma \in \Sigma_2$
- $T\big((0,q,1),\sigma\big) = \big(T_1(0,\lambda\sigma), q, 0\big)$ *if* $\sigma \in \Sigma_1$
- *For* $q_1 \in Q_1$ *and* $q_2 \in Q_2$,[2]
  $T\big((q_1,q_2,c),\sigma\big) = \begin{cases} (T_1(q_1,\sigma),q_2,0) & \text{if } c=1, \sigma \in \Gamma(q_1;A_1) \cap \{\Sigma_1 \setminus U_2(q_2,q_1)\} \\ (q_1,T_2(q_2,\sigma),1) & \text{if } c=0, \sigma \in \Gamma(q_2;A_2) \cap \{\Sigma_2 \setminus U_1(q_1,q_2)\} \end{cases}.$

The construction of Definition 1 is an important step for defining a two-player turn-based game as a run in an FSA. The turn-based product involves a binary variable, or "coin" c which keeps track of whose turn it is: $c = 1$ if player 1 is to play, $c = 0$ if player 2 moves next. A fragment of the $A_1 \circ A_2$ for our example is shown in Figure 4. The machine

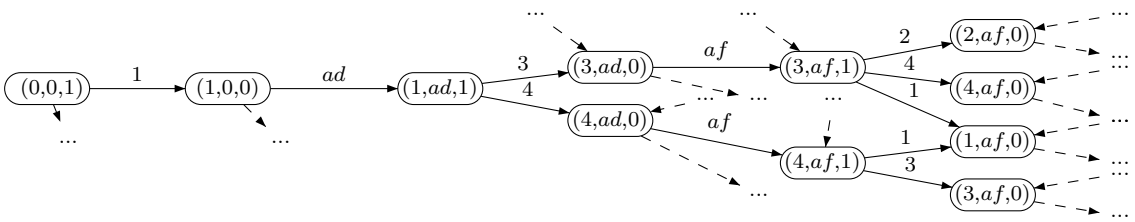


Figure 4: Fragment of turn-based product $P = A_1 \circ A_2$ for the robot and its adversary.

being at state $(r, d_1 d_2, c)$ is interpreted as the robot being in room r, doors d_1 and d_2 being closed, and c telling us whose turn it is to play. We define a set of *legitimate initial states* of the turn-based product as $I \subseteq Q$, which essentially is the set of possible configurations the game can be initialized from. By taking the set of legitimate initial states I as the set of initial states and setting all states final, we can obtain a non-deterministic FSA: $(A_1 \circ A_2)^0 = \langle A_1 \circ A_2, I, Q_p \rangle$.[3] Then, we can embed the winning condition into the game by taking the intersection of $(A_1 \circ A_2)^0$ and the specification FSA $\mathcal{A}_s$. The outcome of this operation is the *game automaton*.

Definition 2 (Game automaton) *The game automaton is a FSA and is defined as* $\mathcal{G} = \langle Q, \Sigma, T, Q_0, F \rangle = (A_1 \circ A_2)^0 \times \mathcal{A}_s$ *where* $(A_1 \circ A_2)^0 = \langle A_1 \circ A_2, I, Q_p \rangle$*,* $A_i = \langle Q_i, \Sigma_i, T_i \rangle$ $i = 1, 2$ *represent the players' behaviors,* I *is the set of legitimate initial states and the set of final states includes all states in* $A_1 \circ A_2$*. The FSA* $\mathcal{A}_s = \langle Q_s, \Sigma, T_s, q_{0s}, F_s \rangle$ *encodes the winning conditions for player 1.*

A fragment of the game automaton for the room visiting game is shown in Figure 5.

2. In what follows, $\Gamma(q; A)$ denotes the $\Gamma(q)$ map of A.
3. $(A_1 \circ A_2)^0$ is non-deterministic because of the existence of multiple initial states and is deterministic in transitions.

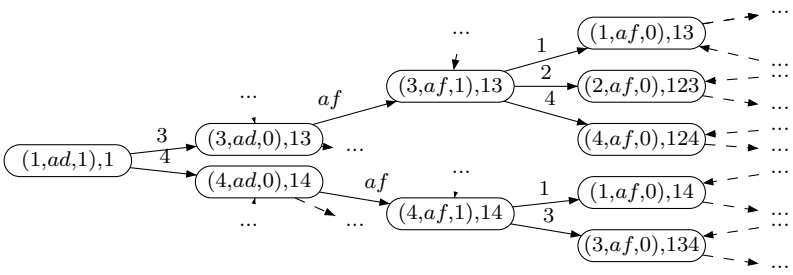


Figure 5: Fragment of the game automaton $\mathcal{G} = (A_1 \circ A_2)^0 \times \mathcal{A}_s$ for the door-robot game, where the set of legitimate initial states is $I = \{(q_1, q_2, 1) \mid q_1 \in Q_1, q_2 \in Q_2\}$, i.e. the game can start with the robot in any room and any pair of opposite doors closed, and the robot is to make a move first.

3.2. Computing the winning strategy

Since the task specification is described as a regular set, the game is a reachability game (Thomas, 2002) and is determined. The control synthesis problem has been converted into computing the winning strategy for player 1 in the game. The following result is obtained by adapting the computation of strategy in a two-player zero-sum game in (Mazala, 2001).

Let $\mathcal{G} = \langle Q, \Sigma, T, Q_0, F \rangle$ be a game automaton where $Q_0 \subseteq Q$ is the set of possible initial states for the game and $F \subseteq Q$ is the subset of the states in which player 1 has won the game. Let Γ be its event activation function. Then the *attractor* of F, denoted $\mathsf{Attr}(F)$, is the largest set of states $W \supseteq F$ in $\mathcal{G}$ from which player 1 can force the play into F. It is defined recursively as follows. Let $W_0 = F$ and define

$$W_{i+1} = W_i \cup \{q = \big((q_1, q_2, 1), q_s\big) \in Q \mid \text{ for some } \sigma \in \Gamma(q), T(q, \sigma) \in W_i\} \\ \cup \{q = \big((q_1, q_2, 0), q_s\big) \in Q \mid \text{ for all } \sigma \in \Gamma(q), T(q, \sigma) \in W_i\} \ . \quad (1)$$

Since $\mathcal{G}$ is finite, there exists the smallest $m \in \mathbb{N}$ such that $W_{m+1} = W_m$. Then $\mathsf{Attr}(F) = W_m$. On the other hand, since $\mathcal{G}$ is determined, the complement of $\mathsf{Attr}(F)$ forms a *trap* for player 1; it contains all the states from which player 2 can prevent player 1 from winning the game. By construction, therefore, the following theorem is proved:

Theorem 3 *Player 1 has a winning strategy iff* $\mathsf{Attr}(F) \cap Q_0 \neq \emptyset$.

If $\mathsf{Attr}(F) = W_m \cap Q_0 \neq \emptyset$, we can ensure that for a particular initial state $q_0 \in \mathsf{Attr}(F) \cap Q_0$, there exists a winning strategy WS_1 for player 1: $\mathsf{WS}_1 : Q \to 2^{\Sigma_1}$ defined as $\mathsf{WS}_1(q) = \{\sigma \mid q = \big((q_1, q_2, 1), q_s\big), \sigma \in \Gamma(q), T(q, \sigma) \in \mathsf{Attr}(F)\}$. This winning strategy is not necessarily optimal, i.e., the one involving the least number of moves. To compute an optimal winning strategy, partition W_m into a set of subsets V_i, $i = 0, \ldots, m$ in the following way: let $V_0 = W_0 = F$ and set $V_i := W_i \setminus W_{i-1}$, for all $i \in \{1, \ldots, m\}$. The sets V_i partition the attractor into layers.

Lemma 4 *For each* $q \in V_{i+1}$, $i = 0, \ldots, m-1$, *there exists at least one* $\sigma \in \Gamma(q)$ *such that* $q' = T(q, \sigma) \in V_i$. *Moreover, if at* q *it is player 2's turn, then for each* $\sigma \in \Gamma(q)$, *there is a* $j \leq i$ *such that* $T(q, \sigma) \in V_j$.

Proof Let $q \in V_{i+1}$. According to (1), either (i) it is player 1's turn and so $\exists\, \sigma \in \Gamma(q) : T(q,\sigma) \in W_i$, or (ii) it is player 2's turn and $\forall\, \sigma \in \Gamma(q), T(q,\sigma) \in W_i$. Consider player 1's turn and suppose there exists $k < i$, $T(q,\sigma) \in V_k$. Then according to (1), q belongs to V_{k+1}. But since the sets V_i partition the states of the attractor, $V_{k+1} \cap V_{i+1} = \emptyset$ since $k \neq i$. This contradicts the assumption that q is also in V_{i+1}. Thus $T(q,\sigma) \notin W_{i-1}$ and belongs instead to V_i. When it is player 2's turn, the same argument shows there exists at least one σ such that $T(q,\sigma) = V_i$. Additionally, if it is player 2's move at state q, and $\sigma \in \Gamma(q)$ with $T(q,\sigma) \notin V_i$, then it follows that $T(q,\sigma) \in V_j$ for some $j < i$ because $T(q,\sigma) \in W_i$. ■

Lemma 4 suggests that if the current state is in player 1's attractor, then the move of player 1, and all moves of player 2, will lead to a state with rank strictly decreasing by *at least* 1. It implies that at this stage the best player 2 can do is to slow down the victory of player 1 by selecting an action that results in a state with level decreased by 1.

Proposition 5 *Suppose* $q_0 = \big((q_1, q_2, 1), q_{s0}\big)$ *and that* $q_0 \in V_k$ *for some* $k \leq m$. *Then player 1 wins the game in at most* k *turns following the strategy* WS_1^*, *according to which*

$$\mathsf{WS}_1^*(q) = \{\sigma \mid T(q,\sigma) \in V_{i-1},\ q \in V_i,\ i \geq 1\} \quad . \tag{2}$$

Proof Given a state $q = \big((q_1, q_2, 1), q_s\big) \in V_i$, player 1 adhering to WS_1^* will choose σ^* such that $T(q,\sigma^*) = q^*$ for some $q^* \in V_{i-1}$ (Lemma 4). At q^* it is player's 2 turn. Any move also forces the game automaton to a state $q'' \in V_j$ for $j < i-1$. Again by Lemma 4, player 2 can only slow player 1's victory by selecting a σ such that $j = i-2$. It follows inductively that player 1 wins in at most k turns. ■

It is suggested that WS_1^* ensures player 1 will win in a minimal number of steps, which is crucial if a cost is associated with each action. Let us see how Proposition 5 applies to the robotic case study we consider here, by looking at Figure 5. The winning set of states is $F = \{(q, 1234) \in Q \mid q \in A_1 \circ A_2\}$; $\mathsf{Attr}(F)$ is obtained by computing the fixed-point of (1). Space limitations prevent us from enumerating all the states in $\mathsf{Attr}(F)$, so we will only give a winning path for the robot according to the winning strategy WS_1^*, assuming that the rules for the adversary correspond to the Opposite game configuration (Table 1) and that the initial states lies in $Q_0 \cap \mathsf{Attr}(F) = \{\big((1, ad, 1), 1\big), \big((1, ce, 1), 1\big), \big((2, ad, 1), 2\big), \big((2, bf, 1), 2\big), \big((4, ce, 1), 4\big), \big((4, bf, 1), 4\big)\}$.

Let us arbitrarily select $q_0 = \big((1, ad, 1), 1\big) \in \mathsf{Attr}(F) \cap Q_0$. Following WS_1^* of (2) the robot's fastest route (play) to victory is

$$(1, ad, 1, 1) \xrightarrow{4} (4, ad, 0, 14) \xrightarrow{ae} (4, ae, 1, 14) \xrightarrow{2} (2, ae, 0, 124) \xrightarrow{ce} (2, ce, 1, 124) \xrightarrow{1} (1, ce, 0, 124) \xrightarrow{ef} (1, ef, 1, 124) \xrightarrow{3} (3, ef, 0, 1234) \quad .$$

On the other hand, in the cases of the Adjacent and General configurations (Table 1), the robot cannot win no matter what the initial state is since in both cases $\mathsf{Attr}(F) \cap Q_0 = \emptyset$. In these game configurations, the robot player, even with perfect knowledge of the moves the adversary can and cannot make, can never win.

4. The Grammatical Inference Module

4.1. Overview

The previous section outlined a methodology for computing optimal strategies (if they exist) in noncooperative, adversarial games when the behavior of both players can be captured in the form of SAs. The computation of a winning strategy hinges on complete knowledge of these behaviors. In this section, this assumption is relaxed such that only one player—the adversary—has full knowledge of the rules of the game; the robotic player does not.

Instead, the robot is equipped with a GI module. This module itself is straightforward because any GI algorithm (de la Higuera, 2010) can be plugged into the modular archictecture developed here. Prior knowledge, if available, can help select the particular algorithm.

The behavior of the unknown environment becomes a positive learning text for the GI module. Inference over this text yields an abstract model of the environment's dynamics (Figure 1(*a*)). This model can then be used to recompute the game and the attractor as described in §3. It is therefore guaranteed that the robot's theory of mind for the unknown adversary will eventually converge to the true abstract model of the adversary, provided (i) the true model lies within the class of models inferrable by the GI module, and (ii) the unknown environment's behavior suffices for a correct inference to be made (in other words, a characteristic sample for the target language is observed).

4.2. Theory of Mind

The *theory of mind* refers to the ability of an agent to infer the mental (read: hidden, unobservable) states of others (Frith and Frith, 2003; Premack and Woodruff, 1978). When the robot begins planning, it is unaware of the capabilities of its adversary. Its *initial theory of mind* about its adversary is that the latter is absent, or if present, cannot act at least in a way that affects how it accomplishes its task.

In our toy example, this means the unsuspecting robot initially assumes that its environment is static. If the game begins with doors e and b closed then the robot's "theory of mind" is that the doors e and b will remain closed and every other door will remain open. Hence in the robot's mind, the environment is a SA with only one state eb with a self-loop labeled eb. The robot uses this hypothesized model for the adversary to compute the game, based on which a strategy is computed. As outside observers, we see the robot entertains a false belief about its environment. Therefore, the robot can easily make a move which keeps it inside its *hypothesized* attractor, but which actually takes it outside the *true* attractor. And once this mistake has been made, the adversary will win because it knows the true nature of the game, and can therefore prevent the robot from visiting all four rooms.

This is where GI enters the picture. Our approach is to use GI to enable the robotic player to incrementally construct an increasingly more accurate model of the behavior of its adversary. We expect that as this model becomes more accurate, the planning efficacy of the player increases. After a sufficient period of observation, it should be able to devise strategies that enable it to succeed regardless of the play of its adversary, at least from those initial starting positions where winning strategies for the robot exist.

4.3. Learning to play using strictly local models

A language L is *Strictly k-Local* (SL_k) iff there exists a finite set $S \subseteq F_k(\rtimes\Sigma^*\ltimes)$, such that $L = \{w \in \Sigma^* : F_k(\rtimes w\ltimes) \subseteq S\}$. Languages which are SL_k form a subclass of the regular languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011) with some interesting and useful properties. For example, it is decidable a) whether the language of a regular grammar is Strictly k-Local, and b) if so what the value k is (Trahtman, 1998).

By combining the results of Caron (1998) and Trahtman (1998), we verified for all sets of rules in Table 1 that the behavior of the adversary is a Strictly 2-Local (SL_2) language. (This is obvious from Figure 2(*a*) which is clearly a Myhill graph.)

Garcia et al. (1990) proved the following:

Theorem 6 (Garcia et al. (1990)) *For every k, Strictly k-Local languages are identifiable in the limit from positive data.*

The algorithm by Garcia et al. (1990) essentially constructs a prefix tree of the observed strings, and then merges those states that have the same incoming path of length $k-1$. A functionally equivalent algorithm described in set-theoretic terms also exists (Heinz, 2010).

The implications of this theorem for the example scenario are that a robot equipped with a Strictly 2-Local learner will eventually develop a true model of its adversarial environment (assuming the adversary's behavior is a positive text).

4.4. Simulation

Figure 6 serves as an illustration of the motion planning of a robot incorporated with a GI. Through interacting with the environment, the robot employs GI to update its theory of mind for the adversary A_2 and then consequently the game $\mathcal{G}$ using the products, with which a winning strategy is computed that determines its own action. We allow the game to be played repeatedly with random initial game states. The convergence of GI ensures that the robot's strategy will converge to the true winning strategy.

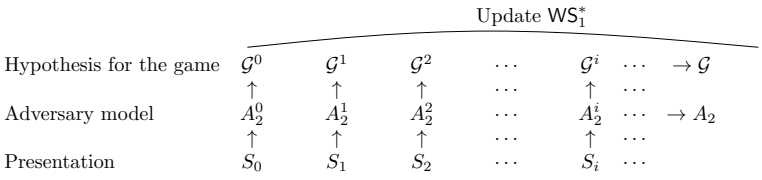


Figure 6: Motion planning with a grammatical inference module. The superscript i indicates that the environment model and the game are updated as the game is played repeatedly.

Algorithm 1 shows how the simulation is implemented. In the simulation, the robot and the adversary play intelligently: they move within their attractor if such a move is available. If no such move exists for the robot, the game is restarted as the robot resigns. Otherwise, if the state is in the robot's attractor, the environment makes a move to slow down the victory of the robot.[4] When a game is restarted, the configuration stays the same,

4. More intelligent choices exist which take us beyond the scope of this paper. For example, if the adversary maintained a theory of what the robot player believed, it could choose to open and close doors in a manner that would make the robot player take longer to converge to the true model of the adversary.

Algorithm 1: Motion planning of the robot with a GI module through repeated games.
input : The abstract model A_1, task specification $\mathcal{A}_s$, and $S_0 = \lambda$
output: A semiautomaton A_2

Let $i = 1$, starting the first game with state $q = (q_1, q_2, 1, q_{0s})$ and $S_1 = q_2$. The upper limit on the number of turns in repeated games is set to N.

while $t \leq N$ **do**
  $A_2^i \leftarrow$ `BuildEnvModel` (S_i); ; `// Constructing` A_2^i `from presentation` S_i `with GI.`
  $\mathcal{G}^i = \langle Q, \Sigma, T, Q_0, F \rangle \leftarrow$ `ComputeGame` $(A_2^i,\ A_1,\ \mathcal{A}_s)$
    $\mathsf{Attr}(F) \leftarrow$ `Attractor` $(\mathcal{G}^i, F)$; $\mathsf{WS}_1^* \leftarrow$ `WinningStrategy` $(\mathcal{G}^i, \mathsf{Attr}(F))$
    $\mathsf{Attr}(Q \setminus F) \leftarrow$ `Attractor` $(\mathcal{G}^i, Q \setminus F)$; $\mathsf{WS}_2^* \leftarrow$ `WinningStrategy` $(\mathcal{G}^i, \mathsf{Attr}(Q \setminus F))$;
    `// The strategy played by the adversary environment is computed simuliarly by taking` $Q \setminus F$ `as the target states.`
  if $q \in \mathsf{Attr}(F)$ **then**
    if $q \notin F$ **then**
      $a \leftarrow$ `RobotMove` $(\mathsf{WS}_1^*,\ q)$; $q \leftarrow T(q, a)$; $t \leftarrow t + 1$;
      if $q \notin F$ **then**
        $b \leftarrow$ `EnvMove` $(\mathsf{WS}_2^*,\ T(q, a))$; $S_{i+1} \leftarrow$ `AddEnvMove` (S_i, b)
        $i \leftarrow i + 1$; $t \leftarrow t + 1$; $q \leftarrow T(q, b)$
      end
    else Restart the game with a random initial state.
      $q = (q_1, q_2, 1, q_{0s}) \leftarrow$ `Choice` (Q_0); $S_{i+1} \leftarrow$`AddEnvMove` $(S_i,\ q_2)$
    end
  else The robot resigns, and the game is restarted.
    $q = (q_1, q_2, 1, q_{0s}) \leftarrow$ `Choice` (Q_0); $S_{i+1} \leftarrow$`AddEnvMove` $(S_i,\ q_2)$
  end
end

the robot is randomly placed in an initial room and a permissible pair of doors is randomly closed, but the model of the adversary (A_2) is passed forward.

The robotic simulations use the Khepera II$^{\text{TM}}$miniature wheeled mobile robot, whose position and orientation feedback is provided by a motion capture system (VICON$^{\text{TM}}$). The robot's wheels are internally controlled by PID loops. This enables simple motion primitives (move straight, turn in place). This behavior can be formally captured as a hybrid system of the form described by Tanner et al. (2012) (§2.2), which affords finite abstractions.

Three robots each played 300 games of the Opposite Door configuration. Robot 1 had no prior knowledge of the adversary and no GI module. Robot 2 only differed from robot 1 in that it was equipped with a GI module. Robot 3 only differed from robot 1 in that it had complete prior knowledge of the adversary. Robot 1 lost every game; robots 2 and 3 won 79 and 82 games, respectively. In this toy example, the incorporation of an appropriate GI module allows robots to nearly reach their maximal potential.

5. Discussion

Our approach accomplishes two goals: a) it offers a game-theoretic framework for robotic planning in an unknown, dynamic, but rule-governed environment, and b) it demonstrates how the inclusion of a GI module makes a difference in terms of the robot's successful

outcomes. Moreover, the theory presented is *modular* and *flexible*, in the sense that as long as the hypothesized abstract models of the environment, the robot model, and the task specification are finite-state, the GI learning algorithm is guaranteed to work.

An additional desideratum would be to show that the theoretical analysis also provides *practical* procedures. In this brief discussion, we explain why we are optimistic.

- The time complexity for computing the attractor $\mathsf{Attr}(F)$ is $\mathcal{O}(|Q| \times \Sigma)$.
- The example game $\mathcal{G}$ consisted of 1008 states. While a concern about the size of $\mathcal{G}$ becoming too large when more complex problems are considered is reasonable, we believe there are representations of the game that enable the computation of the attractor (or at least the next move) without requiring the utilization of the full state space.
- While § 4 suggests that the game automaton $\mathcal{G} = (A_1 \circ A_2)^0 \times \mathcal{A}_s$ has to be recomputed with every new hypothesis on A_2, we anticipate the number of product operations actually required will remain quite limited since portions of $\mathcal{G}$ can be precompiled. This is partly because A_1 and $\mathcal{A}_s$ are known in advance and fixed. Indeed, aspects of A_2 that are known can also be precompiled. In our ongoing simulations, we avoid computing the product at every instance by using a larger, more general representation of the game with many of the transitions turned off.

The interaction of an adaptive agent with its environment as defined by a probablistic process can alternatively be modeled as a Markov Decision Process (Watkins and Dayan, 1992), in which reinforcement learning (RL) is applied for the planning of the agent. In the case of an agent-environment scenario, there are minimax Q-Learning (Littman, 1994), Nash-Q (Hu and Wellman, 2003), Friends-or-Foe Q (Littman, 2001) and other extensions that can be used to decide what action of the agent will maximize the reward over time. In Q-Learning, the convergence has probability 1 if the sampling of state and action pairs happens infinitely often. In the case of an adversarial environment, certain conditions have to be enforced or assumed in order to prove convergence. Our formulation of the agent-environment interaction is fundamentally different because (a) the environment is not probablistic and hence is not a Markov game; and consequently (b) the target of learning is different: here we aim to learn a (class of) languages while RL is a stochastic process. Moreover (c) it is unclear how to ensure that the agent will win in the shortest number of actions in the setting of RL. Yet, this requirement can be crucial in agent-environment interactions if costs are associated actions in the dynamical system.Although the problem addressed in this paper can probably be reformulated so that RL is applicable, this paper aims to introduce an alternative method, which, although in a preliminary stage, builds a bridge between GI and the symbolic planning and control of systems.

6. Conclusion

This paper has demonstrated that the utilization of grammatical inference (GI) in a game-theoretic framework for the purpose of robotic planning can guarantee that the desired task for the robot can be completed even when its environment is unknown, dynamic, and possibly even adversarial. The conditions under which such a result can be achieved are that the robot dynamics can be abstracted in the form of a semiautomaton, the task can be expressed by a finite state acceptor, and the unknown environment dynamics are rule-based

and equivalent at an abstract level to a language belonging to a learnable class of languages. It has been shown in related literature that there are nontrivial families of models for each category (robot, task, and environment) that fulfill these requirements. When they do, a GI algorithm allows the robot to create a model for its environment and refine it in the course of observation, in such a way that it gradually converges to an accurate representation of its world. As soon as this happens, the computation of optimal strategies for the completion of the task becomes possible.

References

Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *International Journal of Robotics Research*, 29(13): 1608–1639, 2010.

Calin Belta, Antonio Bicchi, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George Pappas. Symbolic planning and control of robot motion. *IEEE Robotics Automation Magazine*, 14(1):61–70, 2007.

Emma Brunskill, Bethany R. Leffler, Lihong Li, Michael L. Littman, and Nicholas Roy. Provably efficient learning with typed parametric models. *Journal of Machine Learning Research*, 10:1955–1988, 2009.

Katie Byl and Russ Tedrake. Metastable walking machines. *International Journal of Robotics Research*, 28(8):1040–1064, 2009.

Pascal Caron. Langage: A Maple package for automaton characterization of regular languages. In Derick Wood and Sheng Yu, editors, *Automata Implementation*, volume 1436 of *Lecture Notes in Computer Science*, pages 46–55. Springer Berlin / Heidelberg, 1998.

Christos Cassandras and Stéphan Lafortune. *Introduction to Discrete Event Systems*. Kuwer, 1999.

Yushan Chen, Jana Tøumová, and Calin Belta. LTL robot motion control based on automata learning of environmental dynamics. In *Proceedings of the IEEE Conference on Robotics and Automation*, (to appear), 2012.

Edmund M. Clarke Jr., Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, 1999.

Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

T Dean, K. Basye, L. Kaelbling, E. Kokkevis, O. Maron, D. Angluin, and S. Engelson. Inferring finite automata with stochastic output functions and an application to map learning. In *Proceedings of the 10th National Conference on Artificial Intelligence*, San Jose, CA, 1992.

Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, February 2009.

U. Frith and C.D. Frith. Development and neurophysiology of mentalizing. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, (358):459–473, 2003.

Pedro Garcia, Enrique Vidal, and José Oncina. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, pages 325–338, 1990.

Jeffrey Heinz. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906, Uppsala, Sweden, July 2010.

Junling Hu and Michael P. Wellman. Nash q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 4:1039–1069, dec 2003. ISSN 1532-4435.

M. Kloetzer and C. Belta. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, 23(2):320–331, 2007.

H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu. Correct, reactive, high-level robot control. *Robotics Automation Magazine, IEEE*, 18(3):65 –74, sept. 2011.

Kavita Krishnaswamy, Jennifer Sleeman, and Tim Oates. Real-time path planning for a robotic arm. In *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments*, PETRA '11, pages 11:1–11:4, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0772-7.

M. Lahijanian, J. Wasniewski, S.B. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3227 –3232, may 2010.

A. LaViers, Yushan Chen, C. Belta, and M. Egerstedt. Automatic sequencing of ballet poses. *Robotics Automation Magazine, IEEE*, 18(3):87 –95, sept. 2011. ISSN 1070-9932.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann, 1994.

Michael L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328. Morgan Kaufmann, 2001.

D. Luzeaux. Machine learning applied to the control of complex systems. In *Proceedings of the 8th International Conference on Artificial Intelligence and Expert Systems Applications*, Paris, France, 1996.

J. Lygeros, K.H. Johansson, S.N. Simić, and S.S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

Maja J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

René Mazala. Infinite games. In *Automata, Logics, and Infinite Games*, pages 23–42, 2001.

Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, 1971.

Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2003.

Nir Piterman and Amir Pnueli. Synthesis of reactive(1) designs. In *In Proc. Verification, Model Checking, and Abstract Interpretation (VMCAI'06*, pages 364–380. Springer, 2006.

Giordano Pola, Antoine Girard, and Paulo Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508—2516, 2008.

D. Premack and G. Woodruff. Does the chimpanzee have a theory of mind? *Behav. Brain. Sci*, pages 515–526, 1978.

A. Rieger. Inferring probabilistic automata from sensor data for robot navigation. In *Proceedings of the MLnet Familiarization Workshop and Third European Workshop on Learning Robots*, pages 65–74, 1995.

R.L. Rivest and R.E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103:299–347, 1993.

James Rogers and Geoffrey Pullum. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342, 2011.

Matthew D. Schmill, Tim Oates, and Paul R. Cohen. Learning planning operators in real-world, partially observable environments. In *in Proceedings of ICAPS*, pages 246–253. AAAI Press, 2000.

Colin Stirling. Modal and temporal logics for processes. In Faron Moller and Graham Birtwistle, editors, *Logics for concurency: structure vs automata*. Springer, 1996.

Herbert Tanner, Jie Fu, Chetan Rawal, Jorge Piovesan, and Chaouki Abdallah. Finite abstractions for hybrid systems with stable continuous dynamics. *Discrete Event Dynamic Systems*, 22:83–99, 2012. ISSN 0924-6703.

Wolfgang Thomas. Infinite games and verification (extended abstract of a tutorial). In *Proceedings of the 14th International Conference on Computer Aided Verification*, CAV '02, pages 58–64, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43997-8.

A. N. Trahtman. A polynomial time algorithm for local testability and its level. *International Journal of Automation and Computing*, 9(1):31–39, 1998.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4): 279–292, 1992.

Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon control for temporal logic specifications. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, HSCC '10, pages 101–110, New York, NY, USA, 2010. ACM.