Human-in-the-Loop Control Synthesis for Multi-Agent Systems under Hard and Soft Metric Interval Temporal Logic Specifications*

Sofie Ahlberg[1] and Dimos V. Dimarogonas[1]

***Abstract*— In this paper we present a control synthesis framework for a multi-agent system under hard and soft constraints, which performs online re-planning to achieve collision avoidance and execution of the optimal path with respect to some human preference considering the type of the violation of the soft constraints. The human preference is indicated by a mixed initiative controller and the resulting change of trajectory is used by an inverse reinforcement learning based algorithm to improve the path which the affected agent tries to follow. A case study is presented to validate the result.**

I. INTRODUCTION

With the progress in the robotics and autonomous control fields we see an increase in robotic presence in environments populated by humans. This has increased the importance of human robot interaction (HRI) and Human-in-the-Loop planning and control. These include both physical interaction and communication, where it is important to create systems that are safe and receptive to human preference. To achieve safety, we need system designs with guarantees, such as those that can be achieved by formal methods, that aim at control synthesis from high level temporal logic specifications [1], [2], [3]. In this paper, we consider Metric Interval Temporal Logic (MITL) [4],[5], which can be represented by a timed automaton [6]. Our goal is to design a system that is safe, but also adaptive towards human input and the environment. To achieve this the standard control synthesis framework [7] should be extended to handle the case when a desired specification isn't completely satisfiable.

Different approaches have been suggested for solving this problem. In [8] a method for abstraction refinement to find control policies which could not be found in a sparser partitioning was suggested. In [9] a framework which gives feedback on why the specification is not satisfiable and how to modify it was presented. [10] instead treat the environment as stochastic and designs the controller such that the probability of satisfaction is maximized. Here, we will use the metric hybrid distance which we introduced in [11], to find the controller which minimizes the violation. We will also consider specifications consisting of hard and soft constraints, where the hard constraints must be satisfied.

To achieve adaptability towards the humans preference the system must attain the knowledge of what the human priorities and what consequences this should have on its behaviour. This was discussed in [12], where a control policy was created based on data of human decisions. In [13], a model of human workload information was used to optimize the systems behaviour to balance risk of stress due to full backlogs against risk of low productivity due to empty backlogs. Here, we will instead consider humans giving input to the controllers directly through the so-called mixed-initiative control [14]. The idea is to allow human input while still keeping the guarantees of safety which we acquired from the formal method-based synthesis. The same approach was used in [15] but without the added time constraints inherent to MITL. Here, the human preference is considered to be limited to in what manner the soft constraints should be violated, namely if time (deadlines) is higher prioritized than performing non-desired actions (entering states which should preferably be avoided) or vice versa. To convert the human control input into the desired knowledge we will use an inverse reinforcement learning (IRL)[16] approach.

This paper aims to blend the approach of [11] (Sec. IV) with the concepts of mixed initiative control (Sec. VII), inverse reinforcement learning (Sec. V) and re-planning algorithms (Sec. VI), in order to design a decentralized control synthesis framework for a multi-agent system which guarantees the satisfaction of hard constraints while maximizing the satisfaction of soft constraints with respect to human preference, and ensuring collision avoidance. The problem is formally stated in Section III, and the preliminaries and notation used through out the paper are given in Section II.

II. PRELIMINARIES AND NOTATION

A. Abstraction of Dynamics

In this paper, we consider a multi-agent system where each agent has controllable linear dynamics which can be abstracted into a Weighted Transition System (WTS) where the weights are the corresponding transition times.

Definition 1: A *Weighted Transition System* (WTS) is a tuple $T = (\Pi, \Pi_{init}, \rightarrow, AP, L, d)$ where $\Pi = \{\pi_i : i = 0, ..., M\}$ is a finite set of states, $\Pi_{init} \subset \Pi$ is a set of initial states, $\rightarrow \subseteq \Pi \times \Pi$ is a transition relation; the expression $\pi_i \rightarrow \pi_k$ is used to express transition from π_i to π_k, AP is a finite set of atomic propositions, $L : \Pi \rightarrow 2^{AP}$ is an labelling function and $d :\rightarrow\rightarrow \mathbb{R}_+$ is a positive weight assignment map; the expression $d(\pi_i, \pi_k)$ is used to express the weight assigned to the transition $\pi_i \rightarrow \pi_k$.

Definition 2: A *timed run* $r^t = (\pi_0, \tau_0)(\pi_1, \tau_1)...$ of a WTS T is an infinite sequence where $\pi_0 \in \Pi_{init}$, $\pi_j \in \Pi$, and $\pi_j \rightarrow \pi_{j+1}$ $\forall j \geq 1$ s.t.

*This work was supported by the H2020 ERC Starting Grand BUCOPH-SYS, the Swedish Foundation for Strategic Research, the Swedish Reasearch Council and the Knut and Alice Wallenberg Foundation.

[1]Sofie Ahlberg and Dimos V. Dimarogonas are with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden sofa@kth.se, dimos@kth.se

Manuscript 372 submitted to 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Received March 15, 2019.

- $\tau_0 = 0$,
- $\tau_{j+1} = \tau_j + d(\pi_j, \pi_{j+1}),\ \forall j \geq 1$.

B. MITL Specification

MITL is used to express the considered specifications.

Definition 3: The *syntax of MITL* over a set of atomic propositions AP is defined by the grammar

$$\phi := \top \mid ap \mid \neg\, \phi \mid \phi \wedge \psi \mid \phi\, \mathcal{U}_{[a,b]}\, \psi \tag{1}$$

where $ap \in AP$, $a, b \in [0, \infty]$ and ϕ, ψ are formulas over AP. The operators are *Negation* ($\neg$), *Conjunction* ($\wedge$) and *Until* ($\mathcal{U}$) respectively. Given a timed run $r^t = (\pi_0, \tau_0)(\pi_1, \tau_1), ...$ of a WTS, the semantics of the satisfaction relation is then defined as [5], [4]:

$$(r^t, i) \models ap \Leftrightarrow L(\pi_i) \models ap \text{ (or } ap \in L(\pi_i)), \tag{2a}$$

$$(r^t, i) \models \neg\phi \Leftrightarrow (r^t, i) \not\models \phi, \tag{2b}$$

$$(r^t, i) \models \phi \wedge \psi \Leftrightarrow (r^t, i) \models \phi \text{ and } (r^t, i) \models \psi, \tag{2c}$$

$$(r^t, i) \models \phi\, \mathcal{U}_{[a,b]}\, \psi \Leftrightarrow \exists j \in [a, b],\ s.t.\ (r^t, j) \models \psi \text{ and } \forall i \leq j, (r^t, i) \models \phi. \tag{2d}$$

From this we can define the extended operators *Eventually* ($\Diamond_{[a,b]}\phi = \top \mathcal{U}_{[a,b]}\phi$) and *Always* ($\Box_{[a,b]}\phi = \neg\Diamond_{[a,b]}\neg\phi$). The operators $\mathcal{U}_I$, $\Diamond_I$ and $\Box_I$, are bounded by the interval $I = [a, b]$, which indicates that the operator should be satisfied within $[a, b]$. If $b \neq \infty$, this implies that the operator is subject to some deadline. We will denote these as *temporally bounded* operators. All operators that are not included in the set of temporally bounded operators, are called *non-temporally bounded* operators. The operator $\mathcal{U}_I$ can be temporally bounded (if a deadline is associated to the second part of the formula) but contains a non-temporally bounded part. When we use the term *violating non-temporally bounded operators*, we refer to the non-temporally bounded part of an operator being violated. An example of this is $\phi = A\mathcal{U}_{\leq T}B$, indicating that A must hold until B holds, and that B must hold within T time units. Here, the non-temporally bounded operator is violated if $\neg A$ becomes true before B has become true, while the temporally bounded operator is violated if time T is exceeded before B becomes true. A formula ϕ which contains a temporally bounded operator will be called a temporally bounded formula. The same holds for non-temporally bounded formulas. An MITL specification ϕ can be written as $\phi = \bigwedge_{i\in\{1,2,...,n\}} \phi_i = \phi_1 \wedge \phi_2 \wedge ... \wedge \phi_n$ for some $n > 0$ and some subformulas ϕ_i. In this paper, the notation subformulas ϕ_i of ϕ, refers to the set of subformulas which satisfies $\phi = \bigwedge_{i\in\{1,2,...,n\}} \phi_i$ for the largest possible choice of n such that $\phi_i \neq \phi_j\ \forall i \neq j$. For each subformula ϕ_i, there are 3 possible temporal outcomes if ϕ_i is temporally bounded: satisfaction, violation, or uncertainty.

Example 1: $\phi_i = \Diamond_I A$ is satisfied if A holds at some $t \in I$, violated if $\neg A$ holds $\forall\ t \in I$, and uncertain if $\neg A$ holds for all $t \leq \tau$ where $\tau \in I$ is the current clock valuation.

If ϕ_i is non-temporally bounded there are only two possible temporal outcomes, depending on its properties:

Example 2: $\phi_i = \Diamond_{[0,\infty]} A$ is; satisfied if A holds at some $t \in [0, \infty]$, and uncertain if $\neg A$ holds for all $t \leq \tau$ where τ is the current clock valuation.

TABLE I: Operators categorized according to the *temporally bounded/non-temporally bounded* notation and Definition 4.

Operator	$b = \infty$	$b \neq \infty$
$\Box_{[a,b]}$	Non-temporally bounded, type II	Temporally bounded
$\Diamond_{[a,b]}$	Non-temporally bounded, type I	Temporally bounded
$\mathcal{U}_{[a,b]}$	Non-temporally bounded, type I	Temporally bounded

Example 3: $\phi_i = \Box_{[0,\infty]} A$ is: violated if $\neg A$ holds for some $t \in [0, \infty]$, and uncertain if A holds for all $t \leq \tau$ where τ is the current clock valuation.

To distinguish these non-temporally bounded formulas from each other we introduce *Type I* and *Type II* notation:

Definition 4: A non-temporally bounded formula ϕ is denoted as *Type I* if ϕ cannot be concluded to be violated at any time (since it can be satisfied in the future), and as *Type II* if ϕ cannot be concluded to be satisfied at any time (since it can be violated in the future). The resulting categorization of operators is given in Table I.

C. Hybrid Distance

The *hybrid distance* is a metric which shows the degree of violation of a run with respect to a given MITL formula. It was first introduced in [11] and will be used to find a least violating run with respect to some soft constraints. A plan can violate a MITL formula in two ways; i) by continuous violation, i.e. exceeding deadlines, or ii) by discrete violation, i.e. the violation of non-temporally bounded operators. We quantify these violations with a metric with respect to time:

Definition 5: The hybrid distance d_h is a satisfaction metric with respect to a MITL formula ϕ and a timed run $r^t = (\pi_0, \tau_0), (\pi_1, \tau_1), ..., (\pi_m, \tau_m)$, defined as: $d_h = hd_c + (1 - h)d_d$, where d_c and d_d are the *continuous and discrete distances* between the run and the satisfaction of ϕ, such that $d_c = \sum_{i\in X} T_i^c$, and $d_d = \sum_{j=0,1,...,m} T_j^d$, where X is the set of clocks (given next in Definition 7), T_i^c is the time which the run violates the deadline expressed by clock i, $T_j^d = 0$ if no non-temporally bounded operators are violated by the action $L(\pi_j)$ and $T_j^d = \tau_j - \tau_{j-1}$ otherwise, and $h \in [0, 1]$ is the weight assigning constant which determines the priority between continuous and discrete violations, where $h = 0.5$ yields equal importance.

To be able to calculate d_h we define its derivative:

Definition 6: $\Phi_H = (\dot{d}_c, \dot{d}_d)$, is a tuple, where $\dot{d}_c \in \{0, ..., n_c\}$ and $\dot{d}_d \in \{0, 1\}$, and $n_c = |X|$ is the number of time bounds associated with the MITL specification (or number of clocks).

D. Timed Automaton with Hybrid Distance

In [11], we introduced an extension of the timed Büchi automaton (TBA) [17] denoted Timed Automaton with hybrid distance or TAhd for short. The TAhd was used as a representation of a soft constraint given as a MITL specification. The definition of the TAhd is given by:

Definition 7: [11] A *Timed Automaton with hybrid distance* (TAhd) is a tuple $A_H = (S, S_0, AP, X, F, I_X, I_H, E, H, \mathcal{L})$ where $S = \{s_i : i = 0, 1, ...m\}$ is a finite set of locations, $S_0 \subseteq S$ is the set of initial locations, 2^{AP} is the alphabet (i.e. set

Manuscript 372 submitted to 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Received March 15, 2019.

of actions), where AP is the set of atomic propositions, $X = \{x_i : i = 1, 2, ..., n_c\}$ is a finite set of clocks (n_c is the number of clocks), $F \subseteq S$ is a set of accepting locations, $I_X : S \to \Phi_X$ is a map from location to clock constraints, $H = (d_c, d_d)$ is the hybrid distance, $I_H : S \to \Phi_H$ is a map from location to hybrid distance derivative (labelling each location with some derivatives, $\dot{d}_d$ and $\dot{d}_c$), where I_H is such that $I_H(s) = (d_1, d_2)$ where d_1 is the number of temporally bounded operators violated in s, and $d_2 = 0$ if no non-temporally bounded operators are violated in s and $d_2 = 1$ otherwise, $E \subseteq S \times \Phi_X \times 2^{AP} \times S$ is a set of edges, and $\mathcal{L} : S \to 2^{AP}$ is a labelling function mapping each location to a set of actions.

The notation $(s, g, a, s') \in E$ is used to state that there exists an edge from s to s' under the action $a \in 2^{AP}$ where the valuation of the clocks satisfy the guard $g = I_X(s) \subseteq \Phi_X$. The expressions $d^c(s)$ and $d^d(s)$ are used to denote the hybrid distance derivatives $\dot{d}_c$ and $\dot{d}_d$ assigned to s by I_H.

The clock constraints are defined as:

Definition 8: [17] A *clock constraint* Φ_x is a conjunctive formula of the form $x \bowtie a$, where $\bowtie \in \{<, >, \leq, \geq\}$, x is a clock and a is some non-negative constant. Let Φ_X denote the *set of clock constraints* over the set of clocks X.

Definition 9: An *automata timed run* $r^t_{A_H} = (s_0, \tau_0), ..., (s_m, \tau_m)$ of a TAhd, A_H, corresponding to the timed run $r^t = (\pi_0, \tau_0), ..., (\pi_m, \tau_m)$ of a WTS T, is a sequence where $s_0 \in S_0$, $s_j \in S$, and $(s_j, g_{j+1}, a_{j+1}, s_{j+1}) \in E \ \forall j \geq 1$ such that i) $\tau_j \models g_j$, $j \geq 1$, and ii) $L(\pi_j) \in \mathcal{L}(s_j)$, $\forall j$.

It follows from Definitions 7 and 9, that the continuous violation for the automata timed run is $d_c = \sum_{i=0,...,m-1} d^c(s_i)(\tau_{i+1} - \tau_i)$, and similarly, the discrete violation for the automata timed run is $d_d = \sum_{i=0,...,m-1} d^d(s_i)(\tau_{i+1} - \tau_i)$, and hence the hybrid distance, d_h, as defined in Definition 5, is equivalently given with respect to *an automata timed run* as

$$d_h(r^t_{A_H}, h) = \sum_{i=0}^{m-1} (h d^c(s_i) + (1-h) d^d(s_i))(\tau_{i+1} - \tau_i) \quad (3)$$

E. Product Automaton

The product of a WTS and a TAhd was presented in [11]:

Definition 10: Given a weighted transition system $T = (\Pi, \Pi_{init}, \Sigma, \rightarrow, AP, L, d)$ and a timed automaton with hybrid distance $A_H = (S, S_0, AP, X, F, I_X, I_H, E, H, \mathcal{L})$ their *Product Automaton* (P) is defined as $T^p = T \otimes A_H = (Q, Q^{init}, \rightsquigarrow, d, \mathcal{F}, AP, \mathcal{L}^p, I^p_X, I^p_H, X, H)$, where $Q \subseteq \{(\pi, s) \in \Pi \times S : L(\pi) \in \mathcal{L}(s)\} \cup \{(\pi, s) \in \Pi_{init} \times S_0\}$ is the set of states, $Q^{init} = \Pi_{init} \times S_0$ is the set of initial states, $\rightsquigarrow$ is the set of transitions defined such that $q \rightsquigarrow q'$ if and only if i) $q = (\pi, s)$, $q' = (\pi', s') \in Q$, ii) $(\pi, \pi') \in \rightarrow$, and iii) $\exists\ g, a$, s.t. $(s, g, a, s') \in E$, $d(q, q') = d(\pi, \pi')$ if $(q, q') \in \rightsquigarrow$, is a positive weight assignment map, $\mathcal{F} = \{(\pi, s) \in Q : s \in F\}$, is the set of accepting states, $\mathcal{L}^p(q) = L(\pi)$ is an observation map, $I^p_X(q) = I_X(s)$ is a map of clock constraints, and $I^p_H(q) = I_H(s)$ is a map of hybrid distance derivative constraints.

III. Problem Formulation

The problem considered in this paper is to, for each agent in a multi-agent system, i) find the plan which violates the given soft constraint the least, for some human preference, while satisfying the given hard constraint, ii) learn the human preference concerning the type of violation of the soft constraints based on human control input, and iii) avoid collisions with other agents by re-planning when the next target region is occupied. The input of each agent is assumed to be bounded with $|u_i| \leq u_{max},\ \forall i \in \{1, ..., N\}$.

The hybrid distance (d_h) is used as the measurement of violation, where $d_h = 0$ corresponds to complete satisfaction. The human preference is indicated by the value of h. This can be expressed as four sub-problems:

Problem 1: **Initial plan:** Given a WTS T and an MITL specification $\phi = \phi^{hard} \wedge \phi^{soft}$, find the timed run r^t of T that corresponds to the automata timed run $\hat{r}^t_{A_H}$ that satisfies: $\hat{r}^t_{A_H} = \arg\min_{r^t_{A_H}} d_h(r^t_{A_H}, h)$, where A_H is the TAhd that corresponds to ϕ and $h = 0.5$. That is, find the control policy which guarantees the satisfaction of ϕ^{hard}, and maximizes the satisfaction of ϕ^{soft}, given the preference h.

Problem 2: **Learning preference and updating plan:** Given a human control input u_h, update the estimation of h such that the resulting trajectory (up until this point in time) is optimal with respect to the hybrid distance. Given the updated value of h, find a new plan r^t_{new} (for the remainder of the task) such that $d_h(r^t_{A_H}, h)$ is minimized by the corresponding automata timed run $\overline{r}^t_{A_H}$. Assuming that the human has a value of h in mind and acts accordingly, the updated solution should thus satisfy $d_h(\overline{r}^t_{A_H}, h) < d_h(\hat{r}^t_{A_H}, h)$.

Problem 3: **Collision avoidance:** Given the location of all other agents in the system, find a new plan which doesn't include occupied states and otherwise follows the preferences of the human, if the imminent part of the trajectory crosses the location of another agent.

Problem 4: **Safety:** Design a control law such that the input from the human (u_h) can not cause the agent to violate the hard constraint.

IV. Offline Synthesis of Initial Plan

The solution to Problem 1 is performed offline and follows the outline we suggested in [11]. It is inspired by the standard 3 steps procedure for single agent control synthesis; i) expressing the temporal logic specification as an automaton, ii) constructing the product of the automaton and the transition system, and iii) implementing graph search to find the shortest path. The suggested control synthesis framework is de-centralized and for each agent the planning follows the steps:

1) Construct a Timed Automaton with Hybrid Distance ($TAhd$) which represents the MITL specification.
2) Construct a Product Automaton as the product of the $TAhd$ and a WTS representing the system dynamics.
3) Find the least violating path, i.e, the shortest path with respect to the hybrid distance, d_h, and for $h = 0.5$.

The difference between the solution suggested here and the solution presented in [11] appears in step 1, where we now

Manuscript 372 submitted to 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Received March 15, 2019.

consider hard constraints as well as soft which alters the construction of the TAhd. The details are given in below.

A. Constructing a Timed Automata with Hybrid Distance for hard and soft constraints

In this section we consider the construction of a TAhd when both soft and hard constraints are given. The construction follows the same method which we used in [11] where only soft constraints where given, with a modification to the construction of edges.

We start by describing the construction of locations. To do so we introduce the evaluation sets φ:

Definition 11: A evaluation set φ_i of a subformula ϕ_i contains the possible evaluations of the subformula:

$\{\phi_i^{unc}, \phi_i^{vio}, \phi_i^{sat}\}$	if ϕ_i is temporally bounded
$\{\phi_i^{unc}, \phi_i^{sat}\}$	if ϕ_i is non-temporally bounded, type I
$\{\phi_i^{unc}, \phi_i^{vio}\}$	otherwise

Next we introduce subformula evaluations ψ:

Definition 12: A subformula evaluation ψ of a formula ϕ is one possible outcome of the formula, i.e. a conjunction of elements of the evaluation sets: $\psi = \bigwedge_i \phi_i^{state}$, $\phi_i^{state} \in \varphi_i$. We will use Ψ to denote the set of all subformula evaluations ψ of a formula ϕ, i.e. all possible outcomes of ϕ at any time.

We can now construct the location set $S = \{s_i : i = 1, ..., |\Psi|\}$. Then $S_0 = s_j$ where $\psi_j = \bigwedge_i \phi_i^{unc}$, and $F = s_k$ where $\psi_k = \bigwedge_{i \in I} \phi_i^{sat} \wedge \bigwedge_{j \in J} \phi_j^{unc}$, where $I \cap J = 1, .., |\Psi|$ and J contains the indexes of all ϕ_j which are non-temporally bounded type II (i.e. cannot be evaluated as satisfied). The set of clocks X must include at least one clock for each temporally bounded ϕ_i, two if there is both a lower and an upper bound. I_X is easily constructed such that $s \rightarrow \Phi_X \in I_X$ if $\phi_i^{vio} \notin \psi$ where ϕ_i is temporally bounded by Φ_X. The hybrid distance derivative mapping $I_H(s) = (d_1, d_2)$ is constructed such that d_1 is equal to the number of clock constraints associated with the subformulas $\phi_i^{vio} \in \psi$, and $d_2 = 1$ if any non-temporally bounded subformula $\phi_i^{vio} \in \psi$ and $d_2 = 0$ otherwise. To construct the edges we first introduce some new definitions and notation:

Definition 13: The distance set of two subformula evaluations ψ and ψ' is defined as $|\psi - \psi'| = \{\phi_i : \phi_i^{state'} \neq \phi_i^{state}\}$. That is, it consists of all subformulas ϕ_i which are evaluated differently in the subformula evaluations.

We use $(\psi, g, a) \rightarrow \psi'$ to denote that all subformulas $\phi_i \in |\psi - \psi'|$ are i) uncertain in ψ and ii) it holds that ϕ_i is re-evaluated to $\phi_i^{state'} \in \psi'$ if action a occurs at time t satisfying guard g.

The edges can now be constructed in 4 steps;

i) Construct all edges corresponding to progress (the edges a TBA would have) such that: $(s, g, a, s') \in E$ if $(\psi, g, a) \rightarrow \psi'$.

ii) Construct edges which correspond to non-temporally bounded soft constraint/s no longer being violated such that: $(s, g, a, s') \in E$ if
- $\forall \phi_i \in |\psi - \psi'|$, $\phi_i \in \phi^{soft}$ and is non-temporally bounded, and $\phi_i^{vio} \in \psi$,
- $(s'', g, a, s') \in E$ for some s'' where $|\psi - \psi'| = |\psi - \psi''|$

or
- $\forall \phi_i \in |\psi - \psi'|$, $\phi_i \in \phi^{soft}$ and is non-temporally bounded, and $\phi_i^{vio} \in \psi$
- $(s', g, a', s) \in E$, where $a' = 2^{AP} \backslash a$.

iii) Construct edges which correspond to temporally-bounded soft constraint/s no longer being violated such that: $(s, g, a, s') \in E$ if
- $\exists \phi_i \in |\psi - \psi'|$, such that $\phi_i \in \phi^{soft}$ is temporally bounded, and $\phi_i^{vio} \in \psi$, $\phi_i^{sat} \in \psi'$, $\phi_i^{unc} \in \psi''$,
- where $(s'', g', a, s') \in E$, and $(s'', g, a, s) \in E$,
- and $g = g' \backslash \Phi_{X_i}$, where X_i is the set of clocks associated with ϕ_i s.t. $\phi_i^{unc} \in \psi'$ and $\phi_i^{vio} \in \psi$,
- and $\nexists \phi_i \in |\psi - \psi'|$ such that $\phi_i \in \phi^{hard}$

iv) Construct self-loops such that $(s, g, a, s) \in E$ if $\exists\ (g, a)$ s.t. $g \subseteq g'$, $a \subseteq a'$ where $(s', g', a', s) \in E$ for some s' and $(s, g, a, s'') \notin E$ for any s''.

B. Finding an Initial Plan

The initial plan is now found by constructing the product automaton of the TAhd and the WTS following definition 10 and applying the modified Dijkstra Algorithm 1. Here we have added some further modifications by adding the inputs; initial time and current violation metrics. These inputs are used to set the distance metrics of the initial state and are all zero-valued in the previous version of the algorithm presented in [11]. By allowing non-zero values the same algorithm can be used to re-plan when the mission has began and the time from start as well as violations are no longer zero when the graph search begins.

In [11] we showed that a solution to the algorithm is always found under the assumption that the temporally bounded part of the MITL formula is feasible on the given WTS when deadlines are disregarded. This result is however based on the fact that the TAhd was constructed to represent a soft constraint alone and does no longer apply when hard constraints are applied as well. The result can however be relaxed by adding the assumption that the hard constraint is feasible and does not contradict any eventually or until operators of the soft constraints when deadlines of the soft constraint are disregarded.

V. Learning Human Preference

In this section we consider Problem 2, i.e. learning the preferred value of h based on human control input u_h. The method is an inverse reinforcement learning (IRL) [16] approach and the estimated value of h is iteratively improved when new knowledge is given in the form of human input (i.e. when $u_h \neq 0$) under the assumption that the human is trying to help the system (i.e. u_h is chosen such that d_h is optimized for the true value of h). That is,

$$Cost(r_P^{t,*}, h^*) = \min_{r_P^t} Cost(r_P^t, h^*) \tag{4}$$

where $r_P^{t,*}$ is the timed run of P which the human would guide the agent through (and the optimal run w.r.t. d_h given

Manuscript 372 submitted to 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Received March 15, 2019.

Algorithm 1: $dijkstraHD()$ Dijkstra Algorithm with Hybrid Distance as cost function

Data: P, h, τ_0, d_c^0, d_d^0, d_h^0
Result: r_{hd}^{min}, d_h, d_c, d_d
Q =*set of states;* q_0 =*initial state;* $SearchSet = q_0$*;*
$d(q, q')$ =*weight of transition* $q \rightsquigarrow q'$ *in* P*;*
if $q = q_0$ **then**
  $dist(q) = \tau_0$, $d_h(q) = d_h^0$, $d_c(q) = d_c^0$, $d_d(q) = d_d^0$;
else
  $dist(q) = d_h(q) = d_c(q) = d_d(q) = \infty$;
end
for $q \in Q$ **do**
  $pred(q) = \emptyset$;
end
while *no path found* **do**
  Pick $q \in SearchSet$ *s.t.* $q = \arg\min(d_h(q))$*;*
  if $q \in F$ **then** *path found*
  else
    find all q' *s.t.* $q \rightsquigarrow q'$;
    for *every* q' **do**
      $d_h^{step} = (h\dot{d}_c(q) + (1-h)\dot{d}_d(q))d(q, q')$;
      if $d_h(q') > d_h(q) + d_h^{step}$ **then**
        update $dist(q')$, $d_h(q')$, $d_c(q')$, $d_d(q')$
        and $pred(q')$ *and add* q' *to* $SearchSet$;
        Remove q *from* $SearchSet$*;*
      end
    end
  end
end
while $q \neq q_0$ **do**
  use $pred(q)$ to iteratively form the path back to q_0;
  $\rightarrow r_{hd}^{min}$
end

$h = h^*$), and $Cost(r_P^t, h) = d_h(proj(r_P^t, A_H), h)$ where $proj(r_P^t, A_H)$ is the projection of the timed run of the product automaton P onto the TAhd A_H as defined below. We also define the projection onto the WTS T for later use.

Definition 14: The projections of a timed run of a product automaton $r_P^t = (\pi_1, s_1)(\pi_2, s_2), ..., (\pi_m, s_m)$ onto a TAhd A_H and a WTS T are defined as:

$$proj(r_P^t, A_H) = s_1, s_2, ..., s_m, \text{ and} \tag{5}$$

$$proj(r_P^t, T) = \pi_1, \pi_2, ..., \pi_m. \tag{6}$$

To determine the k estimate of h we suggest solving

$$h_k = \arg\min_{h \in [0,1]} \sum_{i=1}^{k} p(Cost(r_P^{t,h}, h) - Cost(r_P^{t,i}, h)) \tag{7}$$

$$p(x) = \begin{cases} x & \text{if } x \leq 0 \\ \infty & \text{if } x > 0 \end{cases} \tag{8}$$

$$r_P^{t,h} = \arg\min_{r_P^t \in R_P^t} Cost(r_P^t, h) \tag{9}$$

where $r_P^{t,i}$, $i = 1, .., k$ are the previously suggested paths (i.e. $r_P^{t,1}$ is the initial plan and the outcome of Section IV), $R_P^t = \{r_P^t = q_1, q_2, ...q_m : r_P^{t,0} = q_1, q_2, ..., q_l, l \leq m\}$ and $r_P^{t,0}$ is the timed run of P which has been followed from start up until the time of the human input. That is, R_P^t is the set of timed runs of P which can be followed given the up-to-date trajectory. The function $p(x)$ is used to ensure that $Cost(r_P^{t,h_k}, h_k) \leq Cost(r_P^{t,i}, h_k)$ for $i = 1, ..., k$, removing any solutions h_k for which a previously suggested run would be better than the optimal run given the initial trajectory. No loss of correct solutions occurs due to assumption (4). The solution to (7) is the h which maximizes how much d_h decreases due to the human input.

The optimal timed run w.r.t. hybrid distance and h_k is then

$$r_P^{t,k+1} = \arg\min_{r_P^t \in R_P} Cost(r_P^t, h_k). \tag{10}$$

That is, the timed run $r_P^{t,h}$ calculated in (9) for $h = h_k$. The new path to follow is then found by the projection of $r_P^{t,k+1}$ onto the WTS, i.e, $r_{A_H}^{t,k+1} = proj(r_P^{t,k+1}, A_H)$. The solution to (7) and (10) can be found by implementing Algorithm 2.

Algorithm 2: $irl4h()$: Finds h_k, $r_P^{t,k+1}$ and $r_{A_H}^{t,k+1}$

Data: $d_c(r_P^t)$ and $d_d(r_P^t)$ for $r_P^t = r_P^{t,i}$ for $i = 1, ..., k$ and $r_P^t = r_P^{t,0}$, P
Result: h_k, $r_P^{t,k+1}$, $r_{A_H}^{t,k+1}$
δ =design parameter for the step size of the optimization;
update Q^{init}, τ_0, d_c^0, d_d^0 and d_h^0;
for $h = 0, \delta, 2\delta, ..., 1$ **do**
  $dijkstraHD() \rightarrow r_P^{t,h}$ and h_d (Alg 1);
  $Cost(r_P^{t,h}, h) = h_d$;
  for $i = 1, ..., k$ **do**
    $Cost(r_P^{t,i}, h) = hd_c(r_P^{t,i}) + (1-h)d_d(r_P^{t,i})$;
    $p(i) = p(Cost(r_P^{t,0,h}, h) - Cost(r_P^{t,i}, h))$;
    (where p is defined as (8));
  end
end
$h_k = \arg\min \sum p(i)$;
$dijkstraHD() \rightarrow r_P^{t,k+1}$ and h_d (Alg 1);
$r_{A_H}^{t,k+1} = proj(r_P^{t,k+1}, A_H)$;

VI. Re-planning for Collision Avoidance

In this section we describe the solution to Problem 3. We will assume that the agents can share their current position with each other. This could be done either by shared knowledge or visual detection. Each agent monitors its imminent trajectory by determining if another agent is located within the next region $\pi \in \Pi$ in its planned path. If an agent discovers that its upcoming state is occupied it must stop and re-plan. The re-planing is done in two steps; i) update the product automaton according to the present conditions, and ii) perform Algorithm 1 on the updated product automaton (using the latest estimate of h according to Section V). The concept of the re-planning is as follows; mark all states $q \in Q$ which correspond to the occupied region $\pi \in \Pi$ as occupied by setting the weight of all outgoing edges to infinity (hence making it deadlock states), update q_0 to the current state (i.e. make the current state the state which we try to find a path from), and set the start

Manuscript 372 submitted to 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Received March 15, 2019.

time to the current time. The result is that Algorithm 1 will attempt to find a path from the current state, which does not include the occupied state, to an accepting state. With the suggested approach, the progress already made is saved and the actual time of the previous movement is considered.

In the event that no path under the new restrictions can be found, we suggest that the agent waits until the other agent moves, leaving the previously occupied state free. To avoid a deadlock in the system (which could occur if two agents waits for each other to move indefinitely) we suggest a maximum wait-time after which the agent attempts to find a temporary path for which it moves out of the way (allowing the other agent to pass). This is done by temporary updating the product automaton such that every state which i) does not correspond to the violation of the hard constraint, and ii) does not correspond to the current state of the WTS, are marked as accepted. This temporary task (moving out of the way) can then be solved as long as the hard constraint doesn't forbid all other transitions. To minimize the computation we only consider the neighbouring regions. If any of them is safe in the sense that the corresponding transition in the product automaton lead to a state from which an accepting state can be reached this will be the shortest re-routing, and if they are not, then there is no solution to the temporary task.

We denote the set of forbidden states (states which cannot reach the accepting states) as Q_T. Q_T can be determined indirectly by first finding $Q_T^{-1} = Q \backslash Q_T$ (the set of states which an accepting state can be reached from). Q_T^{-1} is found iteratively by: $q \in Q_T^{-1}$ if $q \rightsquigarrow q'$ and $q' \in Q_T^{-1}$, where initially $Q_T^{-1} = \mathcal{F}$. We can now apply the collision avoidance algorithm described in Algorithm 3, where we have made use of second part of Definition 14.

Remark 1: Assuming that the transition times of the WTS are over-approximations it is possible that a re-planned path has a lower hybrid distance than an initial path. The reason for this is that the already performed progress required less time than estimated. In this case the re-planning will have a start time which is lower than the estimated progress time at the corresponding state used by the offline algorithm. Based on this, one could argue that it would be better if each agent re-planned multiple times online regardless of whether an imminent state is occupied. The downside of this is loss of performance time (due to standing still while re-planning).

Remark 2: Further improvement of the re-planning could be done by finding a way to update the product automaton such that the occupied state is disregarded only in the near future (since the agent in the state would, most likely, eventually move).

Algorithm 3: $collAv()$ Collision Avoidance of agent i

Data: Position of agents: $x = x_1, x_2, ..., x_k$, current discrete plan $p_c = q_c, q_{c+1}, ..., q_{goal}$ (q_c :=current state), data for $dijsktraHD()$
Result: New discrete plan $p = q_c, q'_{c+1}, ..., q'_{goal}$
while *no path found* **do**
  for $j = 1 : k$ **do**
    if $x_j \in \pi_{c+1}$ *where* $\pi_{c+1} = proj(q_{c+1}, T)$ **then**
      $Occupied = True$;
    end
  end
  if $Occupied == True$ **then**
    update τ_0, d_c^0, d_d^0, d_h^0 and Q^{init};
    set $d(q, q') = \infty \ \forall q$ if $proj(q', T) = \pi_{c+1}$;
    $dijkstraHD()$ (alg 1);
    if *a path was found* **then**
      break;
    else
      wait for $t = \Delta T$;
      set $T_{wait} = T_{wait} + \Delta T$;
      check if π_{c+1} is free again;
    end
  end
  if $T_{wait} > T_{wait}^{max}$ **then**
    update τ_0;
    set $q \in \mathcal{F}$ if $proj(q, T) = \pi$ where $\pi \in \Pi_{neighboors}$, and $q \notin Q_T$;
    $dijkstraHD()$ (alg 1);
    break;
  end
end

VII. Restrictions on Human Control Input

We will now consider Problem 4, i.e, how to avoid violation of ϕ^{hard} when the human control input is non-zero. As in [15] we will use a mixed-initiative controller [14]:

$$u = u_r(x, \pi_s, \pi_g) + \kappa(x, \Pi) u_h(t) \tag{11}$$

for each transition $(\pi_s, \pi_g) \in \rightarrow$, where u_r is the control input from the system designed to follow the plan which was conceived in Section IV-VI, and u_h is the human input. The problem then becomes to design κ such that ϕ^{hard} is never violated. To solve the problem we follow the same idea as in [15], namely to design κ such that:

i it is zero if the agent is on the limit of entering an area which would violate ϕ^{hard},
ii it is one if the agent is outside of said limit with a given safety margin,
iii it is in the interval $[0, 1]$ if the agent is outside of the limit but inside of the safety margin, and
iv in the area in between the limit and the safety margin it decreases when nearing the limit and increases when nearing the safety margin.

This was achieved in [15] by choosing:

$$\kappa(x, \mathcal{O}_t) = \frac{\rho(d_t - d_s)}{\rho(d_t - d_s) + \rho(\varepsilon + d_s - d_t)} \tag{12}$$

where d_t is the minimum distance between the agent and any region within $\mathcal{O}_t$, $\rho(s) = e^{-1/s}$ for > 0 and $\rho(s) = 0$ for $s \leq 0$, $d_s > 0$ and $\varepsilon > 0$ are design parameters for safety, and $\mathcal{O}_t$ contains all regions $\pi \in \Pi$ which corresponds to a violating state $q \in Q$.

Unlike [15], here we must also consider the time constraints of ϕ^{hard}. Assuming that ϕ^{hard} has temporally

Manuscript 372 submitted to 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Received March 15, 2019.

bounded operators almost all states $\pi \in \Pi$ of the WTS will correspond to the violation of ϕ^{hard} for some time t (i.e. belong to $\mathcal{O}_t$). Hence, if we apply the solution in [15] directly we will be quite conservative, setting $\kappa = 0$ in almost all states. As a result, the learning algorithm wouldn't have enough data and the human wouldn't have enough impact.

To solve this problem we use the set Q_T (containing all states which cannot reach an accepting state) which we constructed in the previous section, and construct a new set $Q_T^t = \{(q,t) : q \in Q_T, t = \min(x \in I_X^P(q))\}$ containing all states corresponding to the violation of ϕ^{hard} paired with the corresponding violated deadline (i.e. the minimum time required to enter the state). We then redefine: $d_t = \min_{(q,t)\in Q_T^t} dist(x,(q,t))$ where $dist(x,(q,t)) = \|x - proj(q,T)\|$ if $t_0 + d(\pi_0, proj(q,T)) > t$ and $dist(x, proj(q,T)) = \infty$ otherwise, where t_0 and π_0 are the time and state of the WTS at the time of calculation. That is, $dist(x,(q,t))$ is the distance between the current location and the region corresponding to q if the deadline is violated (and the transition would lead to the violation of ϕ^{hard}), and ∞ otherwise. The resulting d_t is then the minimum distance to a violating state, and hence equation (12) can be applied without the aforementioned issue.

VIII. Case Study

A simulation with two agents, each following the dynamics in eq. (13), has been performed. Agent 1 is partially controlled by a human user, i.e. u_1 follows eq. (11), while agent 2 is fully autonomous (see eq. (14)).

$$\dot{x}_i = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix} x_i + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_i,\ i = 1,2 \tag{13}$$

$$u_1 = u_r(x, \pi_s, \pi_g) + \kappa(x, \Pi) u_h(t) \tag{14}$$

$$u_2 = u_r(x, \pi_s, \pi_g) \tag{15}$$

Agent 1 is tasked with visiting areas c and d, while agent 2 is tasked with visiting areas e and f, both with soft deadlines. Both agents should also try to avoid areas marked b while they are strictly forbidden to enter area a. The resulting MITL specifications are $\phi_1 = \phi_1^{hard} \wedge \phi_1^{soft} = (\square\neg a) \wedge (\square\neg b \wedge \lozenge_{\leq t_1} c \wedge \lozenge_{\leq t_2} d)$ and $\phi_2 = \phi_2^{hard} \wedge \phi_2^{soft} = (\square\neg a) \wedge (\square\neg b \wedge \lozenge_{\leq t_3} e \wedge \lozenge_{\leq t_4} f)$. The assumption $t_1 \geq t_2$ and $t_3 \geq t_4$ has been made to minimize the number of edges of the resulting TAhd's. The assumption can be made without loss of generality since the labels c, d, e and f can be chosen such that the assumption holds. The control input from the system, i.e. u_r, is determined following the method we suggested in [18], where for each transition in the WTS a controller is found such that: i) the velocity of the agent is positive in the direction of the transition for every x in the start region, and ii) the velocity of the agent is negative in the direction of any other edge on the given edge. That is, the controllers are designed to steer the agent towards the desired edge, while stopping the agent from exiting the starting region through any other edge. The transition times of the WTS's were also determined following [18] and are the maximum times required for each transition to be guaranteed. It follows that the transitions may actually be faster. Since the violation distances used during planning consider the transition times it also follows that they are worst case estimates, and that they may be smaller in implementation when the planned paths are followed. During the online re-planning when the system learns the value of h the real violation distances are considered for the trajectories which have been followed while the estimates are used for the planning of the future path.

TABLE II: Values of the violation distances: d_c, d_d and d_h for agent 1 and agent 2 in the case study. The table shows the estimated values used when constructing the initial and final plans and the real values calculated from the resulting trajectories when the plans had been followed. For agent 1 we have used $h = 1$ (the learnt human preference), and for agent 2, $h = 0.5$ (the initial setting) was used.

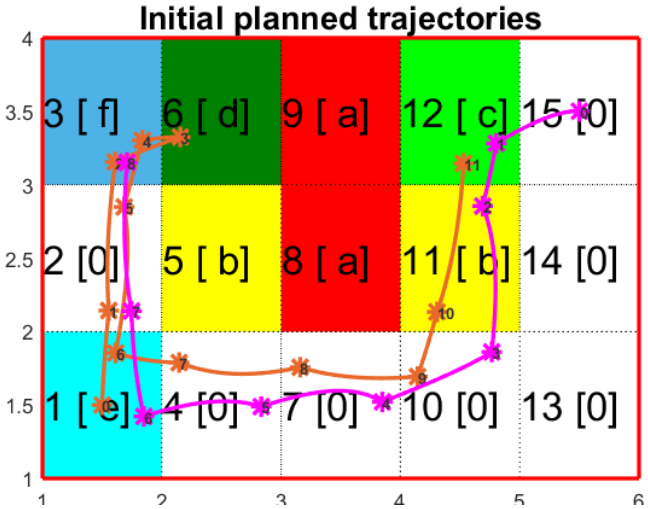
		Initial Plan/Trajectory		Final Plan/Trajectory	
		Estimates	Real Values	Estimates	Real Values
Ag. 1	d_c	0.0418	0.0405	0	0
	d_d	0.0418	0.0405	0.1137	0.1037
	d_h	0.0418	0.0405	0	0
Ag. 2	d_c	0.9351	0.7573	1.6571	1.2795
	d_d	0.0719	0.0664	0.0759	0.0668
	d_h	0.7099	0.4118	0.8665	0.6731

The workspace and the initial plans for each agent is illustrated in Figure 1a. During the online run the human user has a chance to apply control input every 0.015 time steps. Figure 1b illustrates one possible outcome. Here, the human decided to apply the control input $u_h = (0, -u_{max})$ during time $[0.09, 0.135]$ (step 6-9), at which point agent 1 was steered into region 5 instead of region 3. At this point the agent applied the Algorithm 2 to determine that the optimal h was 1, and re-planned accordingly. The next interesting event occurred at step 15 (or time$= 0.225$) where agent 1 and 2 blocked each other by being in the others goal region. This was solved by the agents applying the collision avoidance algorithm (Alg. 3), resulting in agent 2 moving out of the way (into region 13) and agent 1 waiting until region 10 was free. At step 21 (time$= 0.315$) agent 2 was once again blocked by agent 1, which hadn't left region 10 yet. Agent 2 once again applied Algorithm 3, resulting in agent 2 waiting until region 10 was free. The estimated values of the violation distances which where used for planning as well as the real violation distances from following the plans are given in Table II. As expected the real distances are a bit smaller than the estimates (due to the fact that the agents move faster than estimated by the abstraction). Comparing the initial plans with the final plans, the hybrid distance is decreased for agent 1 while it is increased for agent 2. The improvement of agent 1 is due to the path being planned with the correct value of h. The negative change for agent 2 is caused by the collision avoidance forcing the agent to take a longer route.

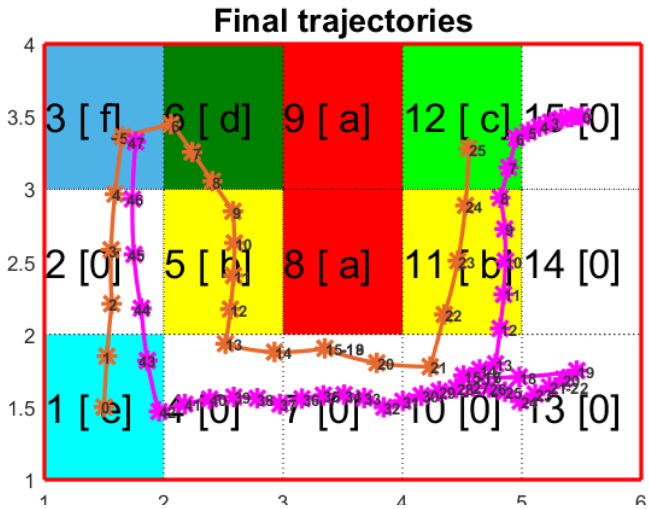
IX. Conclusions and Future Work

We have presented a decentralized control synthesis framework for a multi-agent system under hard and soft constraints given as MITL specifications. The framework uses mixed initiative control to allow a human-in-the-loop to affect the

Manuscript 372 submitted to 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Received March 15, 2019.



(a) Initial plans of the 2 agents with dynamics according to (13), tasked with the MITL specifications ϕ_1 and ϕ_2, where $t_1 = 0.5$, $t_2 = 0.9$, $t_3 = 0.01$ and $t_4 = 0.03$. For the initial planning both agents use $h = 0.5$. Agent 1 follows the orange trajectory and agent 2 follows the magenta trajectory. Agent 1 starts in region 1 and agent 2 starts in region 15.



(b) Final trajectories of agent 1 and 2, where human input has been given to agent 1 to change the transition $6 \to 3$ to $6 \to 5$, indicating that keeping deadlines is more important than avoiding b, i.e. $h = 1$. Collision avoidance has been applied by both agents at step 15, and again by agent 2 at step 21.

Fig. 1: Initial planned and final trajectories of the agents in the case study. Each number/star along the trajectories indicates one iteration where the human had a chance to change her control input, the time step in between is 0.015 time units. The tasks are to avoid the red areas and preferably avoid the yellow areas, while agent 1 should visit the green areas and agent 2 should visit the blue areas.

trajectories of the agents while keeping the guarantees of satisfaction for the hard constraints. The human input is used in an IRL approach to learn the value of a weight assigning constant which indicates the human preference considering the manner of violation of the soft constraints. A collision avoidance algorithm is used to ensure safety. The result is a control policy which guarantees satisfaction of hard constraints and maximizes the satisfaction of soft constraints with respect to human preference, while avoiding collisions.

Future work includes determining under which conditions agents should re-plan to optimize performance time, determining how the step size of h in the learning algorithm can be optimized, and implementing the framework on a robotic platform.

References

[1] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 89.

[2] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, Dec 2009.

[3] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343 – 352, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510980800455X

[4] J. Ouaknine and J. Worrell, "On the decidability of metric temporal logic," in *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*. IEEE, 2005, pp. 188–197.

[5] D. Souza and P. Prabhakar, "On the expressiveness of mtl in the pointwise and continuous semantics," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 1, pp. 1–4, 2007.

[6] D. Ničković and N. Piterman, "From mtl to deterministic timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 152–167.

[7] E. A. Gol and C. Belta, "Time-constrained temporal logic control of multi-affine systems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, pp. 21–33, 2013.

[8] P.-J. Meyer and D. V. Dimarogonas, "Compositional abstraction refinement for control synthesis," *Nonlinear Analysis: Hybrid Systems*, 2017, to appear.

[9] G. E. Fainekos, "Revising temporal logic specifications for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 40–45.

[10] J. Fu and U. Topcu, "Computational methods for stochastic control with metric interval temporal logic specifications," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 7440–7447.

[11] S. Andersson and D. V. Dimarogonas, "Human in the Loop Least Violating Robot Control Synthesis under Metric Interval Temporal Logic Specifications," *European Control Conference (ECC) 2018*, 2018.

[12] S. Carr, N. Jansen, R. Wimmer, J. Fu, and U. Topcu, "Human-in-the-loop synthesis for partially observable markov decision processes," in *2018 Annual American Control Conference (ACC)*, June 2018, pp. 762–769.

[13] R. Schlossman, M. Kim, U. Topcu, and L. Sentis, "Toward achieving formal guarantees for human-aware controllers in human-robot interactions," *arXiv preprint arXiv:1903.01350*, 2019.

[14] W. Li, D. Sadigh, S. S. Sastry, and S. A. Seshia, "Synthesis for human-in-the-loop control systems," in *Tools and Algorithms for the Construction and Analysis of Systems*, E. Ábrahám and K. Havelund, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 470–484.

[15] M. Guo, S. Andersson, and D. V. Dimarogonas, "Human-in-the-loop mixed-initiative control under temporal tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6395–6400.

[16] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[17] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[18] S. Andersson, A. Nikou, and D. V. Dimarogonas, "Control Synthesis for Multi-Agent Systems under Metric Interval Temporal Logic Specifications," *20th World Congress of the International Federation of Automatic Control (IFAC WC 2017)*, 2017.