Human in the Loop Least Violating Robot Control Synthesis under Metric Interval Temporal Logic Specifications*

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

***Abstract*— Recently, multiple frameworks for control synthesis under temporal logic have been suggested. The frameworks allow a user to give one or a set of robots high level tasks of different properties (e.g. temporal, time limited, individual and cooperative). However, the issue of how to handle tasks, which either seem to be or are infeasible, remains unsolved. In this paper we introduce a human to the loop, using the human's feedback to determine preference towards different types of violations of the tasks. We introduce a metric of violation called *hybrid distance*. We also suggest a novel framework for synthesizing a least violating controller with respect to the *hybrid distance* and the human feedback. Simulation result indicate that the suggested framework gives reasonable estimates of the metric, and that the suggested plans correspond to the expected ones.**

I. INTRODUCTION

The introduction of humans in the control loop, especially when the intended task is infeasible, is of great interest since it allows the human to react immediately and approve plans which would otherwise be discarded due to violations. Several schemes based on human in the loop or mixed-initiative have been considered. In [1], the human takes the role as a supervisor assigning types of tasks to individual robots in a multi-robot system. This gives the human direct impact on the priority between different types of tasks. [2] considers cooperative tasks, where human and robot produces separate control inputs, and suggest an adaptive control scheme that combine the inputs while avoiding oscillatory behaviour. Allowing the human control of the input signal raises the question of the impact on the inherited guarantees of task satisfaction caused by the modifications to the plan. This was investigated in [3], where a control scheme was suggested that only lets the human modify the plan in such a way that the guarantees remain. The control scheme is built on navigation functions which drives the human input to zero if a safety constraint is about to be violated. In this paper we instead limit the human's impact to indicate which guarantees should be kept, rather than giving direct input to the plan. To this end, we suggest an automata based control scheme with tasks given as Metric Interval Temporal Logic (MITL).

An advantage with using temporal logic for specifications is its similarities to structural English [4]. The literature on temporal logic is rich and includes [5], [6] and [7]. Multiple control synthesis frameworks for temporal logic specifications has been suggested, considering different branches of logic for both single- and multi-agent systems. In [8] an automata-based method to synthesize a controller for a single-agent system under Linear Temporal Logic (LTL) was presented. This idea was applied to a multi-agent system under Metric Interval Temporal Logic (MITL), in [9], adding time-constraints to the specification. One suggestion of a timed abstraction for this framework was given in [10], which also suggested complexity improving modifications to the products. However, none of these frameworks consider how to handle an infeasible specification. This problem has been approached by using formula revision in papers such as [11] and [12], where the idea of closeness between formulas is used to revise the formula into a satisfiable specification with as small changes as possible. Another approach which have been investigated is abstraction refinement [13], where the partitioning of the environment is refined in an attempt to find previously hidden paths. A third approach is to consider how well a formula is satisfied. This is done in [14] and [15], where metrics are introduced to find an approximate or robust solution to the control synthesis. It allows the user to find a solution that is within an error margin of the specification.

In this work, we suggest a cooperative framework for a single-agent system and a human user considering MITL specifications. The purpose is to find the plan which is closest to satisfying the specification. A metric defining the distance between a plan and the satisfaction of a specification with respect to human feedback is provided in Section II-C. We suggest a method which finds the plan with smallest distance in Section IV. It follows that a solution is always given for all specifications if the reachability parts of the task corresponds to reachable areas in the environment. The human feedback consists in prioritizing between the possible violations of the specification and is further described in Section IV-D.

II. PRELIMINARIES AND NOTATION

A. Abstraction of Dynamics

In this paper, the abstraction of the dynamics and environment is assumed to be given as a weighted transition system.

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

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

[1]Sofie Andersson and Dimos V. Dimarogonas are with the department of Automatic Control, School of Electrical Engineering, KTH Royal Institute of Technology, Sweden sofa@kth.se, dimos@kth.se

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.

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

B. MITL Specification

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], [6]:

$$(r^t, i) \models ap \Leftrightarrow L(\pi_i) \models ap \text{ (or } ap \in L(\pi_i)), \tag{2a}$$
$$(r^t, i) \models \neg\phi \Leftrightarrow (r^t, i) \not\models \phi, \tag{2b}$$
$$(r^t, i) \models \phi \wedge \psi \Leftrightarrow (r^t, i) \models \phi \text{ and } (r^t, i) \models \psi, \tag{2c}$$
$$(r^t, i) \models \phi\, \mathcal{U}_{[a,b]}\, \psi \Leftrightarrow \exists j \in [a, b],\ s.t.\ (r^t, j) \models \psi \text{ and } \forall i \leq j, (r^t, i) \models \phi. \tag{2d}$$

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.

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

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.

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

In this section we introduce the novel metric *hybrid distance*, which shows the degree of violation of a run with respect to a given MITL formula. Later we will use the metric to find a least violating run. 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:

$$d_h = h d_c + (1 - h) d_d \tag{3}$$

where d_c and d_d are the *continuous and discrete distances* between the run and the satisfaction of ϕ:

$$d_c = \sum_{i \in X} T_i^c \qquad d_d = \sum_{j=0,1,...,m} T_j^d$$

X is the set of clocks, T_i^c is the time which the run violates the deadline expressed by clock i, T_j^d is defined as:

$$T_j^d = \begin{cases} \tau_{j+1} - \tau_j & \text{if } (r^t, j) \not\models \phi_i \\ 0 & \text{otherwise,} \end{cases}$$

where ϕ_i is a non-temporally bounded subformula of ϕ and $h \in [0, 1]$ is a 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 is the number of time bounds associated with the MITL specification.

Clock constraints are used to express the time constraints of ϕ in the timed automata representation:

Definition 7: [16] 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.

D. Timed Automaton with Hybrid Distance

In this section, we introduce an extension of the timed Büchi automaton [16] with the hybrid distance included:

Definition 8: 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 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 $g = I_X(s) \in \Phi_X$. 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.

Definition 9: An *automata timed run* $r^t_{A_H} = (s_0, \tau_0)(s_1, \tau_1)...(s_m, \tau_m)$ of a TAhd, A_H, corresponding to a timed run $r^t = (\pi_0, \tau_0), (\pi_1, \tau_1), ..., (\pi_m, \tau_m)$ of a WTS T, is a sequence where $s_0 \in S_0$, $s_j \in S$, and $(s_j, g_{j+1}, a_{j+1}, s_{j+1}) \in E \ \forall j \geq 1$ (for some a_{j+1} and g_{j+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 8 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

$$d_h(r^t_{A_H}) = \sum_{i=0}^{m-1} (h d^c(s_i) + (1-h) d^d(s_i))(\tau_{i+1} - \tau_i) \quad (4)$$

E. Human Feedback

The human feedback $H_f : (r^t, d_c, d_d) \to \mathbb{F}$ is a mapping from the tuple (r^t, d_c, d_d), where r^t is a suggested path, and d_c and d_d are the corresponding distances, to the set $\mathbb{F}$:

Definition 10: The *human feedback* takes values in the set $\mathbb{F} = \{d_c^+, d_c^-, d_c^0, abort\}$, where d_c^+, d_c^- and d_c^0 correspond to giving higher priority to d_d, giving greater priority to d_c and approving the priority (and the plan) respectively; $abort$ indicates that both the values of d_c and d_d are too big to satisfy the human's preferences.

An evaluation function $eval$ is defined for the purpose of comparing two timed runs with each other. The function is used in order to determine if a suggested path is an improvement compared to a previous path, with respect to the hybrid distance and a given human feedback element.

Definition 11: Given two timed runs r_1^t and r_2^t, the corresponding values of the continuous and discrete distances d_c^1, d_c^2, d_d^1, d_d^2, and human feedback $f \in \mathbb{F}$ given as response on r_1^t, d_c^1 and d_d^1, we define the *evaluation* of these two runs as

$$eval(r_1^t, r_2^t, f) = \begin{cases} d_d^1 - d_d^2 & \text{if } f = d_c^+ \\ d_c^1 - d_c^2 & \text{if } f = d_c^- \\ 0 & \text{if } f \in \{d_c^0, abort\} \end{cases}$$

III. Problem Formulation

The problem considered in this paper is to find the plan which violates a given MITL specification the least, for some human preference. Hybrid distance is used as the measurement of violation, where $d_h = 0$ corresponds to complete satisfaction and $d_h \geq 0$. The human preference is indicated by the choice of h. The result is two sub problems:

Problem 1: Given a WTS T and an MITL specification ϕ, find the timed run r^t of T that corresponds to the automata timed run $r^t_{A_H}$ that satisfies:

$$r^t_{A_H} = \arg\min_{r^t_{A_H}} d_h(r^t_{A_H})$$

where A_H is the TAhd that corresponds to ϕ.

Problem 2: Given a human feedback $f \in \mathbb{F}$, update h such that the new solution of Problem 1, r^t_{new}, satisfies $eval(r^t_{old}, r^t_{new}, f) > 0$, where r^t_{old} is the previously found solution, if such a solution exists.

IV. Control Synthesis Framework

The solution to Problems 1 and 2, 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 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 by finding the shortest path with respect to the hybrid distance, d_h, and a given h.
4) Update h in accordance with human feedback and repeat step 3-4 until a plan is approved/aborted.

The details of the proposed solution are further described in Sections IV-A, IV-B, IV-C and IV-D below.

A. Constructing a Timed Automata with Hybrid Distance

In this section we consider the construction of a TAhd. The construction is roughly based on the LTL to automata translation in [7]. Considering the set of locations, it follows from Section II-B that the formula ϕ can be partitioned into

subformulas ϕ_i such that $\phi = \bigwedge_{i\in\{1,...,n\}} \phi_i$ for some $n > 0$. Each subformula ϕ_i can be evaluated as $\phi_i^{state} \in \varphi_i$, where

$$\varphi_i = \begin{cases} \{\phi_i^{vio}, \phi_i^{sat}, \phi_i^{unc}\} & \text{if } \phi_i \text{ is temporally bounded} \\ \{\phi_i^{vio}, \phi_i^{unc}\} & \text{if } \phi_i \text{ is non-temporally bounded of Type I} \\ \{\phi_i^{sat}, \phi_i^{unc}\} & \text{if } \phi_i \text{ is non-temporally bounded of Type II} \end{cases}$$

Based on this we introduce $\Psi = \prod_{i\in\{1,...,n\}} \varphi_i$, and construct the set of locations such that there exists a location s for each possible $\psi \in \Psi$. The initial location is then defined as the location where each subformula is uncertain, i.e. no progress has been made. The accepting location is defined as the location where each temporally-bounded subformula and each non-temporally bounded subformula of Type I are satisfied, while all non-temporally bounded subformulas of Type II are uncertain, i.e. satisfaction of ϕ.

Algorithm 1: Construct set of locations S, initial location S_0 and accepting location F of a TAhd

Data: MITL specification: ϕ
Result: Corresponding set of locations: S, S_0, F
$\Phi = \{\phi_i : \phi = \bigwedge_i \phi_i\}$;
for *each* $\phi_i \in \Phi$ **do**
  if ϕ_i *is temporally bounded* **then**
    $\varphi_i = \{\phi_i^{sat}, \phi_i^{vio}, \phi_i^{unc}\}$;
  else
    $\varphi_i = \{\phi_i^{sat}, \phi_i^{unc}\}$ if ϕ_i is Type I;
    $\varphi_i = \{\phi_i^{vio}, \phi_i^{unc}\}$ if ϕ_i is Type II;
  end
end
$\Psi = \prod_i \varphi_i$;
$S = \{s_i : i = 0, ..., n\}$, where n is the number of $\psi \in \Psi$, that is we create one location s for each $\psi \in \Psi$;
$S_0 = s_0$, where s_0 corresponds to $\psi_0 = \bigcap_i \phi_i^{unc}$;
$F = s_F$, where s_F corresponds to $\psi_F = \bigwedge_{i\in I} \phi_i^{sat} \wedge \bigwedge_{j\in J} \phi_j^{unc}$, where $i \in I$ are the indexes of subformulas that are either temporally bounded or of Type I, and $j \in J$ are the indexes of subformulas that are of Type II;

The clock constraints are defined such that each temporally bounded operator corresponds to one clock. A location s is mapped to a clock constraint if it includes the corresponding temporally bounded operator. The hybrid distance derivatives mapping maps a location s to $(\dot{d}_c, \dot{d}_d)$, where $\dot{d}_c = k$ is the number of temporally bounded operators violated in s, $\dot{d}_d = 0$ if no non-temporally bounded operators are violated and $\dot{d}_d = 1$ otherwise.

The edges are constructed in five sets; E_1, E_2, E_3, E_4, E_5, in Algorithm 2. E_1 corresponds to progress of the MITL formula and contains the edges of a standard timed Büchi automaton. E_2 contains edges from locations which corresponds to discrete violations, and represents the progress which occurs simultaneously as the current discrete violation. E_3 contains edges from locations which corresponds to continuous violations. These edges are equivalent to the edges from the location's predecessors with the exception of the removal of clock constraint/s corresponding to the deadline/s violated when entering the location. E_4 corresponds to self-loops, i.e. transitions from and to the same location. They are defined such that all combinations of actions $a \in 2^{AP}$ and guards $g \in \Phi_X$ present in the ingoing edges are handled by outgoing edges. This ensures that there are no deadlocks in the automaton. E_5 contains the edges which corresponds to going back when discrete violations stop. This set is constructed last in order to determine the actions and guards of the edges based on the other sets. The motivation behind the subsets is to use their properties in the construction.

Algorithm 2: Construct edges E

Data: MITL specification: ϕ, set of locations S, set of actions 2^{AP}, mapping of clock constraints I_X, and mapping of hybrid distance derivative I_H
Result: Corresponding set of edges: E
$\mathbf{E_1}$: $(s, g, a, s') \in E_1$ *if* ψ' corresponding to s' is satisfied when a is performed under g in s;
$\mathbf{E_2}$: $(s, g, a, s') \in E_2$ *if* i) $(s'', g', a', s) \in E_1$, ii) a non-temporally bounded operator is violated in s, iii) $(s'', g, a, s') \in E_1$, and iv) it holds $\forall$ temporally bounded ϕ_j that the state of ϕ_j is identical in s'' and s' (e.g. if ϕ_j is satisfied in s'' it is satisfied in s');
$\mathbf{E_3}$: $(s, g, a, s') \in E_3$ *if* i) $(s'', g', a', s) \in E_1$, ii) a temporally bounded operator ϕ_i is violated in s, and iii)$(s'', g, a, s') \in E_1$ where ϕ_i is satisfied in s';
$\mathbf{E_4}$: $(s, g, a, s) \in E_4$ *if* either *a)* $(g, a) = \bigcup_i (g_i, a_i)$, where (g_i, a_i) are the guard/action tuples of all ingoing edges to s in $E_1 \cup E_2 \cup E_3$, or *b)* i) $s = s_0$, ii) $(g, a) = \Phi_X \times 2^{AP} \backslash \bigcup_i (g_i, a_i)$, where (g_i, a_i) are the guard/action tuples of all outgoing edges from s_0 in $E_1 \cup E_2 \cup E_3$;
$\mathbf{E_5}$: $(s, g, a, s') \in E_5$ *if* i) $\exists (s', g', a', s) \in E_1$ ii) a non-temporally bounded operator ϕ_i is violated in s, iii) ϕ_i is uncertain in s' iv) it holds for all temporally bounded ϕ_j that the state of ϕ_j is identical in s and s' (e.g. if ϕ_j is satisfied in s it is satisfied in s') v) $(g, a) = \Phi_X \times 2^{AP} \backslash \bigcup_i (g_i, a_i)$, where (g_i, a_i) are the guard/action tuples of all outgoing edges from s', i.e. $\exists (s', g_i, a_i, s_i) \in E_1 \cup E_2 \cup E_3 \cup E_4$ for some s_i;
$\mathbf{E}$: $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$;

B. Constructing a Product Automata

The construction of a product of a WTS and a TAhd is similar to the product of a WTS and a TBA (which definition can be found in [10] and [9]). The only modification needed is the consideration of the mapping of the hybrid distance derivatives through simple projection:

Definition 12: 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 $T^p = T \otimes A_H = (Q, Q^{init}, \rightsquigarrow, d, \mathcal{F}, AP, \mathcal{L}^p, I_X^p, I_H^p, X, H)$, where $Q \subseteq \{(\pi, s) \in \Pi \times S : L(r) \in \mathcal{L}(s)\} \cup \{(\pi, s) \in \Pi_{init} \times S_0\}$ 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

- $q = (\pi, s)$, $q' = (\pi', s') \in Q$
- $(\pi, \pi') \in \rightarrow$ and
- $\exists\, g, a$, 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, $I_X^p(q) = I_X(s)$ is a map of clock constraints, and $I_H^p(q) = I_H(s)$is a map of hybrid distance derivative constraints.

C. Finding the Least Violating Path with Human Feedback

The path in P that corresponds to the smallest value of d_h can be found by using a Dijkstra algorithm, where the cost function is defined as the hybrid distance. The idea of the suggested algorithm is given in Algorithm 3. The distance with respect to time is used when finding successors.

Algorithm 3: Dijkstra Algorithm with Hybrid Distance as cost function

Data: Product Automata, weight assignment constant h
Result: Shortest path with respect to hybrid distance r_{hd}^{min}, corresponding distances d_h, d_c, and 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** $dist(q) = d_h(q) = d_c(q) = d_d(q) = 0$;
else $dist(q) = d_h(q) = d_c(q) = d_d(q) = \infty$ **for** $q \in Q$ **do** $pred(q) = \emptyset$;
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**
      $\%d_h^{step} = d_h$ *for transition* $q \rightsquigarrow q'$
      $d_h^{step} = (h\dot{d}_c(q) + (1-h)\dot{d}_d(q))d(q, q')$;
      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
$r_{hd}^{min} = q$;
while $q \neq q_0$ **do**
  $q = pred(q)$;
  $r_{hd}^{min} = [q \quad r_{hd}^{min}]$;
end

Remark 1: In Algorithm 3, we assume that the agent move from q to q' at the end of the transition time. Hence, d_h^{step} considers the hybrid distance derivative of the previous state q rather than the successor q'.

Theorem 1: If $\exists$ some $\pi \in \Pi$ for every $w \in 2^{AP}$ that are considered by a reachability operator in the MITL specification ϕ, such that $w \in L(\pi)$, and if π is reachable from Π_{init}, then Algorithm 3 will always have a solution. Here, w is a word (a combination of atomic propositions), and the reachability operators are *eventually* and *until*, the operators which requires a word to be reached at some point.

Proof: If $\exists$ some $\pi \in \Pi$ for a reachability operator such that $w \in L(\pi)$, then it follows that $\exists\, q \in Q$ such that $q = (\pi, s)$, where s is a location in the $TAhd$ corresponding to the satisfaction of the reachability operator. Furthermore, q is reachable from Q^{init} if π is reachable from Π_{init}. It follows that $\exists$ a state $q' = (\pi', s') \in Q$, which is reachable from Q^{init}, where s' corresponds to the satisfaction of all reachability operators in ϕ. By definition $q' \in \mathcal{F}$, and hence Algorithm 3 will have a solution. ■

D. Human Robot Feedback

To incorporate the human feedback in the system it must be translated into a deterministic response. The response should be such that d_c^+ leads to a decrease in d_d (if possible), and d_c^- leads to a decrease in d_c (if possible). The response to the remaining feedback, d_c^0 and $abort$, should be to end the synthesis. In this paper, we suggest that the system responds as described in Algorithm 4. The idea is simply to decrease or increase h with an increment δ in order to adjust the value of $d_h = hd_c + (1-h)d_d$. We only consider $h \in [0, 1]$, to avoid violation having positive impact. The increment $\delta > 0$ should be chosen small enough to avoid that possible paths are missed. However, decreasing δ will result in a greater number of runs of Algorithm 3.

Algorithm 4: Algorithm for handling feedback from human user

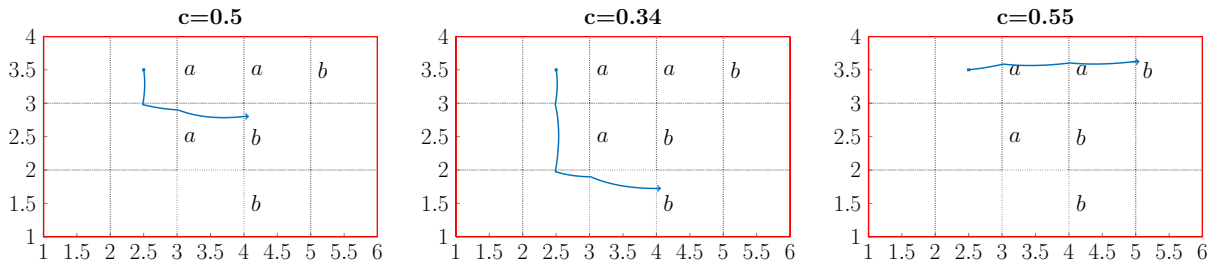
if *feedback*=d_c^0 **then** *Implement controller*;
else if *feedback*=*abort* **then** *Ask for a new task*;
else if *feedback*=d_c^+ **then**
  while *no new path is found and* $h \geq 0$ **do**
    $h = h - \delta$; *find new path*;
  end
  if $h < 0$ **then** $\nexists$ *path with smaller* d_d;
  else *suggest new path to human*;
else if *feedback*=d_c^- **then**
  while *no new path is found and* $h \leq 1$ **do**
    $h = h + \delta$; *find new path*;
  end
  if $h > 1$ **then** $\nexists$ *path with smaller* d_c;
  else *suggest new path to human*;
end

V. Case Study

To illustrate the suggested framework, simulations have been performed in MATLAB. The simulations consider a single agent with the dynamics given in (5), in the environment illustrated in Figure 1, the abstraction of the dynamics was performed as in [10] and considers worst case transition times.

$$\dot{x} = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u \quad x_0 = (2.5, 3.5) \qquad (5)$$
$$x_1 \in [1, 6], x_2 \in [1, 4] \quad |u| \in [-20, 20]$$

The MITL task $\phi = \Box_{[0,\infty]} \neg a \wedge \Diamond_{[0,0.01]} b$ (avoid a and reach b within 0.01s) was given as input. The only word considered by a reachability operator is hence $w = \{b\}$, and



(a) Final path for feedback d_c^0, i.e. $h = 0.5$. The algorithm weighs d_c and d_d equally and chooses a path that has both small discrete and small continuous violations.

(b) Final path for feedback d_c^+, i.e. $h < 0.5$. The algorithm favours d_d and chooses a path that only has continuous violations.

(c) Final path for feedback d_c^-, i.e. $h > 0.5$. The algorithm favour d_c and chooses a path that only has discrete violations.

Fig. 1: Suggested paths satisfying ϕ as close as possible with respect to the hybrid distance and human feedback.

TABLE II: Hybrid distance as estimated by the control synthesis and as calculated from the resulting trajectory, for the paths suggested in the case study.

Path	h	d_c	d_d	Estimated d_h	Real d_h
1	0.5	0.09	0.036	0.064	0.042
2	0.34	0.16	0	0.055	0.041
3	0.55	0.067	0.067	0.067	0.062

since there exists states in the environment where b holds, it follows that the control synthesis will give at least one suggested path. The construction of the product automaton was performed in 3s, and the graph search in 47ms on a laptop with a Core i7-6600U 2.80 GHz processor. The increment δ was set to 0.01. Three different paths were suggested based on the human feedback; one where d_c and d_d were weighed equally, one where d_d was prioritized and one where d_c was prioritized. The resulting paths are illustrated in Figure 1, where the determined control sequences where implemented. The switches between the controllers were performed based on position, i.e. on the edge between states. Hence, the transitions times are in reality shorter than as suggested by the synthesis, resulting in less violation than predicted. The resulting values of h, d_d, d_c and d_h determined both by the synthesis and from the final trajectory are given in Table II. Neither increasing h above 0.55 nor decreasing it below 0.34, results in any new paths.

VI. CONCLUSIONS

In this paper a novel hybrid distance metric is introduced, and is associated to deriving the least violating path with respect to an MITL formula given by a human user. A framework for finding the path with the lowest value for this metric with respect to human feedback is suggested. The presented case study illustrates that the framework gives reasonable estimations of the metric, and that the resulting path suggestion follows the expected behaviour. Current efforts focus on experimental validation of the proposed algorithm.

REFERENCES

[1] M. Cao, A. Stewart, and N. E. Leonard, "Integrating human and robot decision-making dynamics with feedback: Models and convergence analysis," in *2008 47th IEEE Conference on Decision and Control*, Dec 2008, pp. 1127–1132.

[2] V. Okunev, T. Nierhoff, and S. Hirche, "Human-preference-based control design: Adaptive robot admittance control for physical human-robot interaction," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, Sept 2012, pp. 443–448.

[3] S. G. Loizou and V. Kumar, "Mixed initiative control of autonomous vehicles," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 1431–1436.

[4] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating structured english to robot controllers," *Advanced Robotics*, vol. 22, no. 12, pp. 1343–1359, 2008.

[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] 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.

[7] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of model checking*. MIT press, 2008.

[8] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 55–64, 2011.

[9] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas, "Cooperative planning for coupled multi-agent systems under timed temporal specifications," *arXiv preprint arXiv:1603.05097*, 2016.

[10] S. Andersson, A. Nikou, and D. 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.

[11] 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.

[12] M. Lahijanian and M. Kwiatkowska, "Specification revision for markov decision processes with optimal trade-off," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 7411–7418.

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

[14] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.

[15] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

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