Control Synthesis for Multi-Agent Systems under Metric Interval Temporal Logic Specifications ⋆

Sofie Andersson * **Alexandros Nikou** *
Dimos V. Dimarogonas *

* *ACCESS Linnaeus Center, School of Electrical Engineering and KTH Center for Autonomous Systems, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden.*
E-mail: {sofa, anikou, dimos}@kth.se

Abstract: This paper presents a framework for automatic synthesis of a control sequence for multi-agent systems governed by continuous linear dynamics under timed constraints. First, the motion of the agents in the workspace is abstracted into individual Transition Systems (TS). Second, each agent is assigned with an individual formula given in Metric Interval Temporal Logic (MITL) and in parallel, the team of agents is assigned with a collaborative team formula. The proposed method is based on a correct-by-construction control synthesis method, and hence guarantees that the resulting closed-loop system will satisfy the desired specifications. The specifications considers boolean-valued properties under real-time bounds. Extended simulations has been performed in order to demonstrate the efficiency of the proposed methodology.

Keywords: Reachability analysis, Verification and abstraction of hybrid systems, Multi-agent systems, Control design for hybrid systems, Modelling and control of hybrid and discrete event systems, Temporal Logic

1. INTRODUCTION

Multi-agent systems are composed by $N \geq 2$ number of agents which interact in an environment. Cooperative control for multi-agent systems allows the agents to collaborate on tasks and plan more efficiently. In this paper, the former is considered by regarding collaborative team specifications which requires more than one agent to satisfy some property at the same time. The aim is to construct a framework that will start from an environment and a set of tasks, both local (i.e. specific to an individual agent) and global (i.e. requires collaboration between multiple agents), and yield the closed-loop system that will achieve satisfaction of the specifications, by control synthesis.

The specification language that has been introduced to express such tasks is Linear Temporal Logic (LTL) (see e.g., [Loizou and Kyriakopoulos 2004]). The general framework that is used is based on a three-steps procedure ([Kloetzer and Belta 2008, Kress-Gazit et al. 2007]): First the agent dynamics is abstracted into a Transition System. Second a discrete plan that meets the high level task is synthesized. Third, this plan is translated into a sequence of continuous controllers for the original system.

Control synthesis for multi-agent systems under LTL specifications has been addressed in [Kloetzer et al. 2011, Guo and Dimarogonas 2015, Kantaros and Zavlanos 2016]. Due to the fact that we are interested in imposing timed constraints to the system, the aforementioned works cannot be directly utilized. Timed constraints have been introduced for the single agent case in [Gol and Belta 2013, Raman et al. 2015, Fu and Topcu 2015, Zhou et al. 2016] and for the multi-agent case in [Karaman and Frazzoli 2008, Nikou et al. 2016b]. Authors in [Karaman and Frazzoli 2008] addressed the vehicle routing problem, under Metric Temporal Logic (MTL) specifications. The corresponding approach does not rely on automata-based verification, as it is based on a construction of linear inequalities and the solution of a resulting Mixed-Integer Linear Programming (MILP) problem. In our previous work [Nikou et al. 2016b], we proposed an automatic framework for multi-agent systems such that each agent satisfies an individual formula and the team of agents one global formula.

The approach to solution suggested in this paper follows similar principles as in [Nikou et al. 2016b]. Here however, we start from the continuous linear system itself rather than assuming an abstraction, by adding a way to abstract the environment in a suitable manner such that the transition time is taken explicitly into account. The suggested abstraction is based on the work presented in [Gol and Belta 2013], which considered time bounds on facet reachability for a continuous-time multi-affine single agent system. Here, we consider multi-agent systems and suggest an alternative time estimation and provide a proof for its validity. Furthermore, we present alternative product definitions, compared to the work presented in [Nikou et al. 2016b]. The definitions suggested here requires a smaller number of states and hence, a lower computational demand. The drawback of the suggested definitions is an increased risk of a false negative result and a required modification to the applied graph-search-algorithm. However, this will have no effect on the fact that the method is correct-by-construction. The method, in its entirety, has been implemented in simulations, demonstrating the satisfaction of the specifications through the resulting controller.

⋆ This work was supported by the H2020 ERC Starting Grand BUCOPHSYS, the Swedish Research Council (VR), the Swedish Foundation for Strategic Research (SSF) and the Knut och Alice Wallenberg Foundation.

The contribution of this paper is summarized in four parts; (1) it extends the method suggested in [Nikou et al. 2016b] with the ability to define the environment directly as a continuous linear system rather than treating the abstraction as a given, (2) it provides for a less computationally demanding alternative, (3) simulation results which support the claims are included, (4) it considers linear dynamics in contrast to the already investigated (in [Nikou et al. 2016b]) single integrator. Due to space limitations, we refer to [Andersson et al. 2017] for detailed proofs of the claimed results as well as a more detailed example.

This paper is structured as follows. Section 2 introduces some preliminaries and notations that will be applied throughout the paper, Section 3 defines the considered problem and Section 4 presents the main result, namely the solution framework. Finally, simulation result is presented in Section 5, illustrating the framework when applied to a simple example, and conclusions are made in Section 6.

2. PRELIMINARIES AND NOTATION

In this section, notations and preliminary definitions from formal methods that are required for this paper are introduced.

Given a set S, we denote by $|S|, 2^S$ its cardinality and the set of all its subsets respectively. Let $A \in \mathbb{R}^{n\times m}, B \in \mathbb{R}^n$ be a matrix and a vector respectively. Denote by $[A]_{ij}$ the element in the i-th row and j-th column of matrix A. Similarly, denote by $[B]_i$ the i-th element of vector B.

An *atomic proposition ap* is a statement over the system variables that is either true ($\top$) or false ($\bot$).

Definition 1. A *p-dimensional rectangle* $R_p(a,b) \subset \mathbb{R}^p$ is characterized by two vectors a, b, where $a = (a_1, a_2, ..., a_p)$, $b = (b_1, b_2, ..., b_p)$ and $a_i < b_i, \forall i = 1, 2, ..., p$. The rectangle is then given by $R_p(a,b) = \{x \in \mathbb{R}^p : a_i \leq x_i \leq b_i, \forall i \in \{1, 2, .., p\}\}$,

Definition 2. A *Weighted Transition System* (WTS) is a tuple $T = (\Pi, \Pi_{init}, \Sigma, \rightarrow, AP, L, d)$ where

- $\Pi = \{r_i : i = 0, ..., M\}$ is a set of states,
- $\Pi_{init} \subset \Pi$ is a set of initial states,
- $\Sigma = \{\sigma_i : i = 0, ..., l\}$ is a set of inputs,
- $\rightarrow \subset \Pi \times \Sigma \times \Pi$ is a transition map; the expression $r_i \xrightarrow{\sigma_j} r_k$ is used to express transition from r_i to r_k under the action σ_j,
- AP is a set of observations (atomic propositions),
- $L : \Pi \rightarrow 2^{AP}$ is an observation map and
- $d : \rightarrow \rightarrow \mathbb{R}_+$ is a positive weight assignment map; the expression $d(r_i, \sigma_j, r_k)$ is used to express the weight assigned to the transition $r_i \xrightarrow{\sigma_j} r_k$.

Definition 3. A *timed run* $r^t = (r(0), \tau_0)(r(1), \tau_1)...$ of a WTS T is an infinite sequence where $r(0) \in \Pi_{init}$, and $r(j) \in \Pi$, $r(j) \xrightarrow{\sigma_i} r(j+1)\ \forall j \geq 1$ s.t.

- $\tau_0 = 0$,
- $\tau_{j+1} = \tau_j + d(r(j), \sigma_i, r(j+1))$, $\forall j \geq 1$,

for some $\sigma_i \in \Sigma$.

Definition 4. A *timed word* produced by a timed run is an infinite sequence of pairs $w(r^t) = (L(r(0)), \tau_0)(L(r(1)), \tau_1)...$, where $r^t = (r(0), \tau_0)(r(1), \tau_1)...$ is the timed run.

Definition 5. The *syntax of MITL* over a set of atomic propositions AP is defined by the grammar

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

where $ap \in AP$, $a, b \in [0, \infty]$ and ϕ, ψ are formulas over AP. The operators are *Negation* ($\neg$), *Disjunction* ($\vee$) and *Until* ($\mathcal{U}$) respectively. The extended operators *Eventually* ($\Diamond$) and *Always* ($\Box$) are defined as:

$$\Diamond_{[a,b]}\phi := \top \mathcal{U}_{[a,b]} \phi, \tag{2a}$$

$$\Box_{[a,b]}\phi := \neg \Diamond_{[a,b]} \neg \phi. \tag{2b}$$

Given a timed run $r^t = (r(0), \tau_0)(r(1), \tau_1), ...$ of a WTS, the semantics of the satisfaction relation is then defined as:

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

Definition 6. A *clock constraint* Φ_x is a conjunctive formula on the form $x \bowtie a$, where $\bowtie \in \{<, >, \leq, \geq\}$, x is a clock and a is some non-negative constant. Let $\Phi_{\mathcal{X}}$ denote the *set of clock constraints*.

The TBA was first introduced in [Alur and Dill 1994] and is defined as

Definition 7. A *Timed Büchi Automaton* (TBA) is a tuple $A = (S, S_0, \mathcal{X}, I, E, F, AP, \mathcal{L})$ where

- $S = \{s_i : i = 0, 1, ..., M\}$ is a finite set of locations,
- $S_0 \in S$ is the set of initial locations,
- $\mathcal{X}$ is a finite set of clocks,
- $I : S \rightarrow \Phi_{\mathcal{X}}$ is a map labelling each state s_i with some clock constraints $\Phi_{\mathcal{X}}$,
- $E \subseteq S \times \Phi_{\mathcal{X}} \times 2^{\mathcal{X}} \times S$ is a set of transitions and
- $F \subset S$ is a set of accepting locations,
- AP is a finite set of atomic propositions,
- $\mathcal{L}$ is a labelling function, labelling every state with a subset of atomic propositions.

A state of A is a pair (s, v) where $s \in S$ is a location and v is a clock valuation that satisfies the clock constraint $I(s)$. The initial state of A is a pair $(s_0, (0, 0, ..., 0))$, where $s_0 \in S_0$ and the null-vector $(0, 0, ..., 0)$ is a vector of $|\mathcal{X}|$ number of valuations $v_i = 0$. For the semantics and examples of the above TBA definition we refer the reader to [Nikou et al. 2016a].

It has been shown in [Alur et al. 1996] that any MITL formula can be algorithmically translated to a TBA such that the language that satisfies the MITL formula is also the language that produces accepting runs by the TBA. The TBA expresses all possibilities, both satisfaction and violation of the MITL formula. All timed runs which result in the satisfaction of the MITL formula are called accepting:

Definition 8. An *accepting run* is a run for which there are infinitely many $j \geq 0$ s.t. $q_j \in F$, i.e. a run which consists of infinitely many accepting states.

In motion-planning, the movement of an agent can be described by a timed run. For the multi-agent case, the movement of all agents can be collectively described by a collective run. The definition is

Definition 9. [Nikou et al. 2016b] The *collective timed run* $r_G = (r_G(0), \tau_G(0))(r_G(1), \tau_G(1))...$ of N agents, is defined as follows

- $(r_G(0), \tau_G(0)) = (r_1(0), ..., r_N(0), \tau_G(0))$
- $(r_G(i+1), \tau_G(i+1)) = (r_1(j_1), ..., r_N(j_N), \tau_G(i+1))$, for $i \geq 0$ where $(r_G(i), \tau_G(i)) = (r_1(i), ..., r_N(i), \tau_G(i))$ and
  - $l = \underset{k \in I}{\operatorname{argmin}}\{\tau_k(i_k + 1)\}$,
  - $\tau_G(i+1) = \tau_l(i_l + 1)$,

$\cdot\ r_k(j_k) = \begin{cases} r_l(i_l+1), & \text{if } k=l \\ r_k(i_l), & \text{otherwise.} \end{cases}$

3. PROBLEM DEFINITION

3.1 System Model

Consider N agents performing in a bounded workspace $X \subset \mathbb{R}^n$ and governed by the dynamics

$$\dot{x}_m = A_m x_m + B_m u_m,\ \ m \in \mathcal{I},$$
$$x_m(0) = x_m^0, x_m \in X, \tag{3}$$

where $\mathcal{I} = \{1, ..., N\}$ is a set containing a label for each agent.

3.2 Problem Statement

The problem considered in this paper consists of synthesizing a control input sequence, $u_m, m \in \mathcal{I}$, such that each agent satisfies a local individual MITL formula ϕ_m over the set of atomic propositions AP_m. At the same time, the team of agents should satisfy a team specification MITL formula ϕ_G over the set of atomic propositions $AP_G = \bigcup_{m\in\mathcal{I}} AP_m$.The problem can be defined as:

Problem 1. Synthesize a sequence of individual timed runs $r_1^t, ..., r_N^t$ such that the following holds:

$$\left(r_G \models \phi_G\right) \wedge \left(r_1^t \models \phi_1 \wedge ... \wedge r_N^t \models \phi_N\right), \tag{4}$$

where the collective run r_G was defined in Definition 9.

Remark 1. Initially it might seem that if a run r_G that satisfies the conjunction of the local formulas i.e., $r_G \models r_1^t \wedge \ldots \wedge r_N^t$ can be found, then the Problem 1 is solved in a straightforward centralized way. This does not hold since by taking into account the counterexample in [Nikou et al. 2016b, Section III], the following holds:

$$r_G^t \models \bigwedge_{k\in\mathcal{I}} \varphi_k \nLeftrightarrow r_1^t \models \varphi_1 \wedge \ldots \wedge r_N^t \models \varphi_N. \tag{5}$$

4. PROPOSED SOLUTION

The solution approach involves the following steps:

(1) For each agent, we abstract the continuous-time linear system (3) into a WTS which describes the possible movements of the agent considering the dynamics and limitations of the state space (section 4.1).
(2) For each agent, we construct a local Büchi Weigthed Transition System (BWTS) out of its WTS and a TBA representing the local MITL specification. The accepting timed runs of the local BWTS satisfy the local specification (section 4.2).
(3) Next, we construct a product BWTS out of the local BWTSs. The accepting timed runs of the product BWTS satisfy all local specifications (section 4.3).
(4) Next, we construct a global BWTS out of the product BWTS and the TBA representing the global MITL specification. The accepting runs of the global BWTS satisfy both the global specification and all local specifications (section 4.4).
(5) Finally, we determine the control input by applying a graph-search algorithm to find an accepting run of the global BWTS and projecting this accepting run onto the individual WTSs (section 4.5).

The computational complexity of the proposed approach is discussed in Section 4.6.

4.1 Constructing a WTS

In this section we consider the abstraction of the environment into a WTS. The definition of a WTS was given in Section 2. The abstraction is performed for each agent $m \in \mathcal{I}$, resulting in N number of WTSs.

Following the idea of [Gol and Belta 2013], we begin by dividing the state space X_m into p-dimensional rectangles, defined as in Definition 1 such that formula (6) is satisfied for each rectangle.

$$ap_i = (\top,\ \forall x \in R_p(a,b)) \text{ or}$$
$$ap_i = (\bot,\ \forall x \in R_p(a,b)), \forall ap_i \in AP_m. \tag{6}$$

The set of states $\Pi = \{r_0, r_1, ..., r_M\}$ of the WTS is then defined as the set of rectangles $\mathcal{R} = \{R_p(a^0, b^0), R_p(a^1, b^1), ..., R_p(a^M, b^M)\}$. From this, the definition of the initial state Π_{init}, transitions $\rightarrow$ and labelling L follows directly:

$$\Pi_{init} = \{r_i \in \Pi | x_m^0 \in R_p(a^i, b^i)\} \tag{7}$$

$$r_i \rightarrow r_j \text{ iff } R_p(a^i, b^i) \text{ and } R_p(a^j, b^j) \text{ have a common edge,} \tag{8}$$

$$L(r_i) = \{ap_i \in AP_m | ap_i = \top\ \forall x \in R_p(a^i, b^i)\} \tag{9}$$

The set Σ is given as the set of control inputs which induce transitions. In particular, a control input must be defined for each possible transition such that it guarantees the transition, that is no other transition can be allowed to occur and the edge of which the transition goes through must be reachable. These conditions on control inputs are required both to ensure that the synthesized path is followed and to guarantee that the following time estimation holds. A suggested low-level controller for a transition $r_k \rightarrow r_l$ in direction i, based on [Gol and Belta 2013], is given by

$$\begin{aligned} \max_{u_m \in U_m}\ & [\dot{x}_m]_i \\ \text{s. t.}\ & [\dot{x}_m]_j \leq -\epsilon, \forall j \neq i, j = \{1, ..., p\}, \text{if } [x_m]_j = b_j^k, \\ & [\dot{x}_m]_j \geq \epsilon, \forall j \neq i, j = \{1, ..., p\}, \text{if } [x_m]_j = a_j^k, \\ & [\dot{x}_m]_i \geq \epsilon > 0. \end{aligned} \tag{10}$$

where $U_m = [-u_{max}, u_{max}]$ is some bound on u_m and ϵ is a robustness parameter. The idea is to maximize the transition speed, under the conditions that the speed in direction j is negative at the edge with norm direction j, for $j \neq i$.

Finally, the weights d are assigned as the maximum transition times. These times are given according to Theorem 1 below. The theorem depends on the assumption $B_m u_m = B_{m1} x_m + B_{m2}$, where B_{m1} and B_{m2} are matrices of dimension $N \times N$ and $N \times 1$ respectively. The assumption corresponds to u_m being affine.

Theorem 1. The maximum time $T^{max}(r_k, r_l)$ required for the transition $r_k \rightarrow r_l$ to occur, where $R_p(a^k, b^k)$ and $R_p(a^l, b^l)$ share the edge e_{kl}, $\overline{e_{kl}}$ is the edge located opposite to e_{kl} in $R_p(a^k, b^k)$, i is the direction of the transition, and assuming that e_{kl} is reachable from all points within r_k, is defined as:

$$T^{max}(r_k, r_l) = \ln\left(\frac{([A_m^*]_{ii} x^0 + C_m^* + [B_m^*]_i)}{([A_m^*]_{ii} x^1 + C_m^* + [B_m^*]_i)}\right) \frac{1}{[A_m^*]_{ii}}$$

where

$$C_m^* = \sum_{\substack{j=1 \\ j\neq i}}^{n} \min_{[x_m]_j \in r_k} \left([A_m^*]_{ij} [x_m]_j\right),$$

and $x^0 = x_i(0) \in \overline{e_{kl}}$, $x^1 = x_i(T^{max}) \in e_{kl}$, $A_m^* = A_m + B_{m1}$ and $B_m^* = B_{m2}$, where $\dot{x}_m = A_m x_m + B_m u_m = A_m x_m + B_{m1} x_m + B_{m2}$.

See Figure 1 for illustration of the variables of Theorem 1 in 2 dimensions. The full proof of the theorem can be viewed in [Andersson et al. 2017].

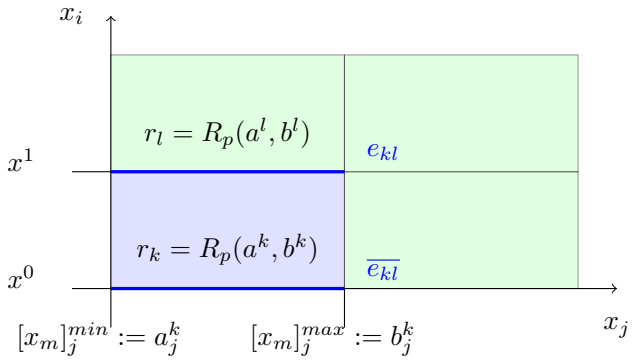


Fig. 1. 2D-illustration of the variables in Theorem 1.

Remark 2. If $C_m^* = 0$ or $[\dot{x}_m]_j = 0$ $\forall j$, then T^{max} is the maximal time required for the transition to occur. Otherwise T^{max} is an over-approximation.

Finally, the weights of the WTS are defined as

$$d(r_i, r_j) = T^{max}(r_i, r_j) \text{ where } (r_i, \sigma, r_j) \in \rightarrow, \quad (11)$$

for $\sigma = u_m(r_i, r_j)$.

4.2 Constructing a Local BWTS

Next, a local BWTS is constructed out of the WTS and a TBA representing the local MITL specification for each agent. As stated in Section 2 any MITL formula can be represented by a TBA [Alur et al. 1996]. Approaches for the translation were suggested in [Maler et al. 2006], [Brihaye et al. 2013] and [Ničković and Piterman 2010]. Note that the time-intervals considered by the MITL formulas must be on the form $\leq a$ due to the over-approximation of time in the abstraction. The local BWTS is defined as:

Definition 10. Given a weighted transition system $T = (\Pi, \Pi_{init}, \Sigma, \rightarrow, AP, L, d)$ and a timed Büchi automaton $A = (S, S_{init}, \mathcal{X}, I, E, F, AP, \mathcal{L})$ their *local BWTS* is defined as $T^p = T \otimes A = (Q, Q^{init}, \rightsquigarrow, d^p, F^p, AP, L^p, I^p, C)$ with:

- $Q \subseteq \{(r, s) \in \Pi \times S : L(r) = \mathcal{L}(s)\}$,
- $Q^{init} = \Pi_{init} \times S_{init}$
- $q \rightsquigarrow q'$ iff
  - $q = (r, s)$, $q' = (r', s') \in Q$
  - $(r, r') \in \rightarrow$ and
  - $\exists\, \gamma, R$, s.t. $(s, \gamma, R, s') \in E$,
- $d^p(q, q') = d(r, r')$ if $(q, q') \in \rightsquigarrow$,
- $F^p = \{(r, s) \in Q : s \in F\}$ and
- $L^p(r, s) = L(r)$
- $I^p(q) = I(s)$
- $C = \{c_1, ..., c_M\}$
  $c_i = \{(q, q') \mid \exists\ R \ s.t.\ (s, R, s') \in E \text{ and } x_i \in R\}$

where $M = |\mathcal{X}|$.

It follows from the construction and automata-based LTL model checking theory [Baier and Katoen 2007] that all possible runs of the local BWTS correspond to a possible run of the WTS. Furthermore, all accepting states of the local BWTS corresponds to accepting states in the TBA. This is formalized in Lemma 1.

Lemma 1. An accepting timed run $r_k^t = (q_k(0), \tau_k(0))(q_k(1), \tau_k(1))...$ of the local BWTS projects onto the timed run $r^t = (r(0), \tau(0))(r(1), \tau(1))...$ of the WTS that produces the timed word $w(r^t) = (L_k(r(0)), \tau(0))(L_k(r(1)), \tau(1))...$ accepted by the TBA. Also, if there is a timed run that produces an accepting timed word of the TBA, then there is an accepting timed run of the local BWTS.

4.3 Constructing a Product BWTS

The definition of the Product BWTS is:

Definition 11. Given N local BWTSs $T_1^p, ..., T_N^p$, defined as in Definition 10, and $M_k = |\mathcal{X}_k|$ for $k \in \{1, .., N\}$, the *product BWTS* $T_G = T_1^p \otimes ... \otimes T_N^p$, where $T_i^p = (Q_i, Q_i^{init}, \rightsquigarrow_i, d_i^p, F_i^p, AP_i, L_i^p, I_i^p, C_i, M_i)$ and $T_G = (Q_G, Q_G^{init}, \rightarrow_G, d_G, F_G, AP_G, L_G, I_G, C_G, M)$ is defined as:

- $Q_G \subseteq Q_1 \times ... \times Q_N$
- $Q_G^{init} = Q_1^{init} \times ... \times Q_N^{init}$
- $(q_G, q_G') \in \rightarrow_G$ iff
  - $q_G = (q_1, ..., q_N) \in Q_G$,
  - $q_G' = (q_1', ..., q_N') \in Q_G$,
  - $\exists q_k' \in Q_k$ s.t. $(q_k, q_k') \in \rightsquigarrow_k$, $\forall\ k \in \mathcal{I}$,
- $d_G(q_G, q_G') = d_{max} = \max_{i=\mathcal{I}}(d_i^p)$, if $(q_G, q_G') \in \rightarrow_G$,
- $F_G = \{(q_1, ..., q_N) \in Q_G \text{ s.t. } q_k \in F_k^p, \forall k \in \mathcal{I}\}$,
- $AP_G = \bigcup_{k \in \mathcal{I}} AP_k$,
- $L_G(q_1, .., q_N) = \bigcup_{k \in \mathcal{I}} L_k^p(q_k)$,
- $I_G(q_G) = \bigcup_{k \in \mathcal{I}} I_k^p(q_k)$,
- $C_G = \{C^1, ..., C^N\}$, $C^i = \{c_1^i, ..., c_{M_i}^i\}$
  $c_k^i = \begin{array}{l}\{(q_G, q_G'), q_G = (q_1, ..., q_N), q_G' = (q_1', ..., q_N') \\ s.t\ \ (q_i, q_i') \in c_k, c_k \in C_i\}\end{array}$
- $M = \{M_1, .., M_N\}$

It follows from the construction that an accepting collective run of the product BWTS corresponds to accepting runs of each local BWTS. Formally

Lemma 2. An accepting collective run r_G of the product BWTS projects onto an accepting timed run r_k^t of a local BWTS, for each $k \in I$. Moreover, if there exists an accepting timed run for every local BWTS, then there exists an accepting collective run.

Remark 3. Note that the definition does not allow for the agents to start transitions at different times. This causes an overestimation of required time which increases the risk for false negative result. An alternative definition which allows the mentioned behaviour was suggested in [Nikou et al. 2016b]. However, the definition suggested here requires less number of states and hence less computational time.

4.4 Constructing a Global BWTS

Finally, a global BWTS is constructed from the product BWTS and a TBA representing the global MITL specification.

Definition 12. Given a product BWTS $T_G = (Q_G, Q_G^{init}, \rightarrow_G, d_G, F_G, AP_G, L_G, I_G, C_G, M)$ and a global TBA $A_G = (S_G, S_G^{init}, \mathcal{X}_G, I_G, E_G, \mathcal{F}_G, \mathcal{L}_G)$, with $M_G = |\mathcal{X}_G|$, their *global BWTS* $\hat{T}_G = T_G \otimes A_G = (\hat{Q}_G, \hat{Q}_G^{init}, \rightsquigarrow_G, \hat{d}_G, \hat{F}_G, AP_G, \hat{L}_G)$ is defined as:

- $\hat{Q}_G \subseteq \{(q, s) \in Q_G \times S_G \text{ s.t. } L_G(q) = \mathcal{L}_G(s)\} \times Z_0 \times ... \times Z_N \times \{1, 2\}$, where $Z_i = \{z_1^i, ..., z_{M_i}^i\}$ for $i = 1, ..., N$ and $Z_0 = \{z_1^0, ..., z_{M_G}^0\}$
- $\hat{Q}_G^{init} = Q_G^{init} \times S_G^{init} \times \{1, .., 1\} \times ... \times \{1, ..., 1\} \times \{1, 2\}$, where $\{1, ..., 1\} \times ... \times \{1, ..., 1\}$ consists of $N + 1$ sets, where the first set contains M_G ones, and the remaining sets contains M_i ones each,
- $(q_G, q_G') \in \rightsquigarrow_G$ iff
  - $q_G = (q, s, Z_0, ..., Z_N, l) \in \hat{Q}_G$,
  - $q_G' = (q', s', Z_0', ..., Z_N', l') \in \hat{Q}_G$,
  - $(q, q') \in \rightarrow_G$,

· $\exists \gamma, R$ s.t. $(s, \gamma, R, s') \in E_G$ s.t,
For all $i \in \{1, ..., N\}$, Z_i and Z_i' are such that

$$z_k^i = 0 \text{ and } z_k^{i'} = 1, \text{ if } (q, q') \in c_k^i$$
$$z_k^{i'} = z_k^i, \qquad \text{otherwise}$$

Z_0 and Z_0' are such that

$$z_k^0 = \begin{cases} 0 \text{ if } x_k \in R \\ 1 \text{ otherwise} \end{cases}$$

$$z_k^{0'} = \begin{cases} 1 \text{ if } x_k \in R \\ z_k^0 \text{ otherwise} \end{cases}$$

· $l' = \begin{cases} 1, \text{ if } l = 1 \text{ and } q \in F_G \\ \quad \text{or } l = 2 \text{ and } s \in \mathcal{F}_G \\ 2, \text{ otherwise} \end{cases}$

- $\hat{d}_G(q_G, q_G') = d_G(q, q')$ if $(q_G, q_G') \in \rightsquigarrow_G$,
- $\hat{F}_G = \{(q, s, Z_0, ..., Z_N, 1) \in \hat{Q}_G \text{ s.t. } q \in F_G\}$ and
- $\hat{L}_G(q, s, Z_0, ..., Z_N, l) = L_G(r)$.
- $I(q_G) = I_G(q) \cup I(s)$

It follows from the construction that an accepting run of the global BWTS corresponds to an accepting run of the product BWTS as well as an accepting run of the TBA representing the global specification. Formally

Lemma 3. An accepting timed run r_G^t of the global BWTS projects onto an accepting collective run r_G of the product BWTS that produces a timed word $w(r_G^t)$ which is accepted by the TBA representing the global specification. Also, if there exists an accepting collective run that produces a timed word accepted by the TBA, then there is an accepting timed run r_G^t of the global BWTS.

4.5 Control Synthesis

The controller can now be designed by applying a modified graph-search algorithm (such as a modified Dijkstra) to find an accepting run of the global product. The modification of the algorithm includes a clock valuation when considering a transition. A sketch of the modification is given in [Andersson et al. 2017]. The idea is to calculate the clock valuation for each clock given the predecessors of the current state, if a valuation does not satisfy the clock constraint the transition is not valid. When the algorithm is complete the accepting run is projected onto the WTSs following Lemma 1, Lemma 2 and Lemma 3. Finally, the set of controllers are given as the sequences of control inputs which induces the timed runs $(r_1^t, r_2^t, ...r_N^t)$ which in turn produce accepted timed words of all local TBAs as well as of the global TBA.

4.6 Complexity

The framework proposed in this paper requires at most

$$|\hat{T}_G| = \prod_{i=1}^{N}(|T_i| \times |A_i| \times 2^{M_i}) \times |A_G| \times 2^{M_G} \times 2 \qquad (12)$$

number of states. The method suggested in [Nikou et al. 2016b] requires

$$|\hat{T}_G| = \prod_{i=1}^{N}\left(|T_i| \times |A_i| \times (C_{max}^i + 1)^{M_i}\right) \times |A_G| \times 4 \times (C_{max}^G + 1)^{M_G}$$

number of states, where all possible clock values are integers in the set $[0, C_{max}^i]$ and $[0, C_{max}^G]$ for the local and global TBA's respectively. Hence the number of states required in the proposed framework is a factor $\frac{\prod_{i=1}^{N}(C_{max,i}+1)^{M_i} \times (C_{max,G}+1)^{M_G}}{2^{\sum_{i=1}^{N}(M_i)+M_G}}$ less.

Fig. 2. Environment considered in the example. The circles represents the initial states of each agent.

5. SIMULATION RESULTS

Consider $N = 2$ agents with dynamics in the form:

$$\dot{x} = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u, \qquad (13a)$$

$$\dot{x} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} u, \qquad (13b)$$

evolving in a bounded workspace $X = \{(x_1, x_2) : 1 \leq x_1 \leq 5.5, 1 \leq x_2 \leq 4\}$, divided into 6 rooms and a corridor in accordance with the partition illustrated in Figure 2. Each agent is assigned with the local MITL formula $\phi_L = \Diamond_{0.1} r_2 \wedge r_2 \rightarrow \Diamond_{0.3} r_6$ ('Eventually, within 0.1 time units, the agent must be in room 2, and if the agent enters room 2 it must then enter room 6 within 0.3 time units.'). Furthermore, they are assigned with the global MITL formula $\phi_G = \Diamond_1(a_1 = r_1 \wedge a_2 = r_2)$ ('Eventually, within 1 time units, agent 1 must be in room 1 and agent 2 must be in room 2, at the same time.'). The initial positions of each agent is indicated in Figure 2.

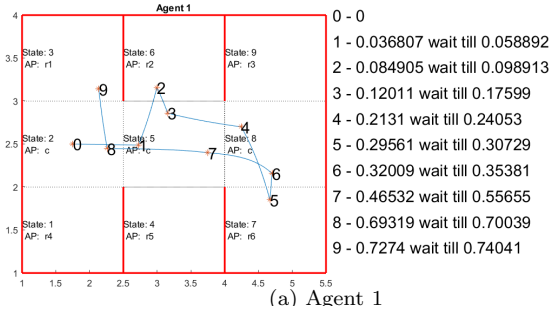
Remark 4. As can be seen in figure 2, some walls have been added to the environment. Transitions through these are forbidden. This is handled by the abstraction since the edges on which the walls are placed aren't reachable.

MATLAB was used to simulate the problem by constructing all transition systems and applying a modified Dijkstra algorithm to find an accepting path as well as a control sequence that satisfies the specifications. The result was a global BWTS with 248832 number of states and an accepting path of 9 transitions. The projection of the accepting run onto each WTS, yields the paths illustrated in Figure 3, which shows the evolution of each closed-loop system for the given initial positions. The figure was constructed by implementing the built-in function *ode45* for the determined controllers in each state. The switching between controllers is position-triggered; namely the switching from controller u_{ij} to u_{jk} is performed when the agent has entered far enough into state j, where "far enough" was defined as 5 iterations of *ode45* upon exiting the previous state. A comparison between the estimated time distances and the actual times for each transition is given in table 1. That is, the worst case transition times yields that ϕ_1, ϕ_2 and ϕ_G are all satisfied by the real run, where $\phi_1 = \Diamond_{0.0989} r_2 \wedge r_2 \implies \Diamond_{0.2084} r_6$, $\phi_2 = \Diamond_{0.0589} r_2 \wedge r_2 \implies \Diamond_{0.2484} r_6$, $\phi_G = \Diamond_{0.7404}(a_1 = r_1 \wedge a_2 = r_2)$ and r_{real} is the actual run. From this, it is clear that the given path will satisfy the MITL formulas.

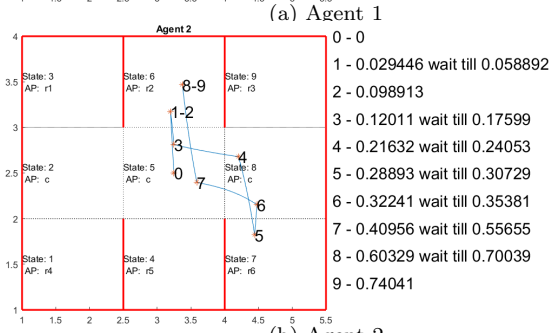
The simulation presented in this section was run in MATLAB on a laptop with a Core i7-6600U 2.80 GHz processor, the runtime was approximately 30min.

Table 1. The worst case estimation and the actual required time for each joined transition. The actual time is the time required for all agents to transition. *These transitions require agent 2 to stay in place, hence the actual time is here defined as the time agent 1 requires to complete the transition.⋆Numbered in order of transitions, see figure 3.

Position⋆	Agent 1	Agent 2	Worst Case Time Estimation	Actual Time
0	2	5	0	0
1	5	6	0.0589	0.0368
2	6	6	0.04	0.026*
3	5	5	0.0771	0.0212
4	8	8	0.0645	0.0403
5	7	7	0.0668	0.0551
6	8	8	0.0465	0.0151
7	5	5	0.2027	0.1115
8	2	6	0.1438	0.1366
9	3	6	0.04	0.027*



(a) Agent 1



(b) Agent 2

Fig. 3. Illustration of the paths of each agent in the example. The numbers 0-9 represent the end of each joined transition. The actual arrival time at each location as well as the time the agent is required to wait till it is guaranteed that all other agents have transitioned, is noted to the right of the figure. It is notable that both agents finish all transitions on less time than the worst case estimation. Hence, run-time can be reduced by allowing the agents to communicate to each other.

6. CONCLUSIONS AND FUTURE WORK

A correct-by-construction framework to synthesize a controller for a multi-agent system following continuous linear dynamics such that some local MITL formulas as well as a global MITL formula are satisfied, has been presented. The method is supported by result of the simulations in the MATLAB environment. Future work includes communication constraints between the agents.

REFERENCES

Alur, R. and Dill, D.L. (1994). A theory of timed automata. *Theoretical computer science*, 126(2), 183–235.

Alur, R., Feder, T., and Henzinger, T.A. (1996). The benefits of relaxing punctuality. *Journal of the ACM (JACM)*, 43(1), 116–146.

Andersson, S., Nikou, A., and Dimarogonas, D.V. (2017). Control Synthesis for Multi-Agent Systems under Metric Interval Temporal Logic Specifications. *https://arxiv.org/abs/1703.02780v1*.

Baier, C. and Katoen, J.P. (2007). *Principles of model checking*. MIT press.

Brihaye, T., Estiévenart, M., and Geeraerts, G. (2013). On mitl and alternating timed automata. In *Formal Modeling and Analysis of Timed Systems*, 47–61. Springer.

Fu, J. and Topcu, U. (2015). Computational methods for stochastic control with metric interval temporal logic specifications. *CoRR*, abs/1503.07193.

Gol, E.A. and Belta, C. (2013). Time-constrained temporal logic control of multi-affine systems. *Nonlinear Analysis: Hybrid Systems*, 10, 21–33.

Guo, M. and Dimarogonas, D. (2015). Multi-Agent Plan Reconfiguration Under Local LTL Specifications. *The International Journal of Robotics Research*, 34(2), 218–235.

Kantaros, Y. and Zavlanos, M. (2016). A Distributed LTL-Based Approach for Intermittent Communication in Mobile Robot Networks. *American Control Conference (ACC), 2016*, 5557–5562.

Karaman, S. and Frazzoli, E. (2008). Vehicle Routing Problem with Metric Temporal Logic Specifications. *47th IEEE Conference on Decision and Control (CDC 2008)*, 3953–3958.

Kloetzer, M., Ding, X.C., and Belta, C. (2011). Multi-Robot Deployment from LTL Specifications with Reduced Communication. *50th IEEE Conference on Decision and Control (CDC 2011)*, 4867–4872.

Kloetzer, M. and Belta, C. (2008). A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53(1), 287–297.

Kress-Gazit, H., Fainekos, G.E., and Pappas, G.J. (2007). Where is waldo? sensor based temporal logic motion planning. *mag*.

Loizou, S. and Kyriakopoulos, K. (2004). Automatic Synthesis of Multi-Agent Motion Tasks Based on LTL Specifications. *43rd IEEE Conference on Decision and Control (CDC 2004)*, 1, 153–158.

Maler, O., Nickovic, D., and Pnueli, A. (2006). From mitl to timed automata. In *Formal Modeling and Analysis of Timed Systems*, 274–289. Springer.

Ničković, D. and Piterman, N. (2010). *From MTL to deterministic timed automata*. Springer.

Nikou, A., Boskos, D., Tumova, J., and Dimarogonas, D.V. (2016a). Cooperative Planning for Coupled Multi-Agent Systems under Timed Temporal Specifications. *http://arxiv.org/pdf/1603.05097v2.pdf*.

Nikou, A., Tumova, J., and Dimarogonas, D.V. (2016b). Cooperative task planning of multi-agent systems under timed temporal specifications.

Raman, V., Donzé, A., Sadigh, D., Murray, R., and Seshia, S. (2015). Reactive Synthesis from Signal Temporal Logic Specifications. *18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015)*, 239–248.

Zhou, Y., Maity, D., and Baras, J.S. (2016). Timed Automata Approach for Motion Planning Using Metric Interval Temporal Logic. *European Control Conference (ECC 2016)*.