EXAMENSARBETE INOM TEKNIKOMRÅDET
TEKNISK FYSIK
OCH HUVUDOMRÅDET
ELEKTROTEKNIK,
AVANCERAD NIVÅ, 30 HP
*STOCKHOLM, SVERIGE 2016*

# Automatic Control Design Synthesis under Metric Interval Temporal Logic Specifications

**SOFIE ANDERSSON**

**KTH**
**SKOLAN FÖR ELEKTRO- OCH SYSTEMTEKNIK**

# Automatic Control Design Synthesis under Metric Interval Temporal Logic Specifications

Master Thesis

Sofie Andersson
sofa@kth.se
System Control and Robotics
Automatic Control Department, EES
Engineering Physics, KTH

Supervisor: Alexandros Nikou
Examiner: Prof. Dimos Dimarogonas

**Abstract**

The problem of synthesizing controllers for motion planning of multi-agent systems under Linear Temporal Logic (LTL) high-level specifications has been of great interest and has been widely studied over the last years. However, LTL cannot handle time constraints as specifications. The time aspect would allow more complicated and specific tasks and it is therefore desirable to incorporate. This work aims to determine how control synthesis for a continuous linear system can be performed based on Metric Interval Temporal Logic (MITL), which is able to handle desired time constraints to high-level specifications. Firstly, a control design synthesis method for a single-agent, based on previous work within both the field of LTL and MITL is presented. Secondly, a control design synthesis method for multi-agent systems considering both local an global MITL specifications is presented. Extended simulations has been performed in MATLAB environment demonstrating the two proposed methodologies. The result shows that the methods guarantee that the MITL specifications are satisfied, for all cases for which a solution is found.

**Sammanfattning**

Problemet gällande regulator syntetisering för rörelse planering av fler-agents system under Linear Temporal Logic (Linjär Temporal Logik=LTL) hög-nivå specifikationer har varit av stort intresse och har studerats brett under de senaste åren. LTL kan emellertid inte hantera tidsbegränsingar som specifikationer. Tidsaspekten skulle tillåta mer komplicerade och specifika uppgifter. Det är därför önskvärt att inkorporera. Målet med det här arbetet är att fastställa hur regulator syntetisering för ett kontinuerligt, linjärt system kan utföras utgående från Metric Interval Temporal Logic (Metrisk Intervall Temporal Logic =MITL), en gren av Temporal Logik som kan hantera de önskvärda tidsbegränsningarna för hög-nivå specifikationer. Först presenteras en metod för att syntetisera regulatorer för en-agents system. Metoden är baserad på tidigare arbeten inom fälten LTL och MITL. Sedan presenteras en metod för att syntetisera regulatorer för fler-agents system som önskas uppfylla såväl lokala som globala MITL specifikationer. Utbredda simulationer har genomförsts i MATLAB miljö för att demonstrera de två föreslagna metoderna. Resultatet visar att metoderna garanterar att MITL specifikationerna är uppfyllda för alla fall för vilka en lösning hittas.

**Acknowledgements**

Firstly, I would like to express my gratitude to Prof. Dimos Dimarogonas for giving me the opportunity to explore the field of temporal logic as a Master Thesis student at the control department. It has been a challenging and inspiring time and I greatly appreciate your support. Secondly, I want to thank my supervisor, Alexandros Nikou, for hours of rewarding discussion, tips and encouragement along the way and support throughout the process. You have given me more confidence in my work and made my time on the project more fun. Finally, I direct my appreciation towards my family who has always been supportive of my studies. Without your encouragement and support throughout my years as a student (in primary school, upper secondary school as well as here at KTH) I doubt I would be where I am today.

*Sofie Andersson*
*May 2, 2016*
*Stockholm*

**Erkännanden**

Först vill jag uttrycka min tacksamhet till Prof. Dimos Dimarogonas, för att han gav mig möjligheten att utforska fältet temporal logik som Master Examensstudent vid regleravdelningen. Det har varit en utmanande och inspirerande tid och jag uppskattar ditt stöd. Sedan vill jag tacka min handledare Alexandros Nikou, för timmar av givande diskussioner, tips och uppmuntran längs vägen och stöd igenom processen. Du har ökat mitt förtroende för mitt arbete och gjort min tid inom projektet roligare. Tillsist riktar jag min uppskattning mot min familj, som alltid varit stöttande av mina studier. Utan er uppmuntran och ert stöd under mina år som student (i grundskolan, gymnasiet såväl som här på KTH) tvivlar jag att jag varit där jag är idag.

*Sofie Andersson*
*Maj 2, 2016*
*Stockholm*

# Contents

# List of Figures



# List of Tables

# 1 Introduction

This master thesis will consider automatic control design synthesis based on high level temporal logic motion planning. The main purpose is to study how to design control input for a continuous linear system such that the controlled system satisfies a temporal logic formula. Temporal logic consists of mathematical formulas which express properties that a system is desired to satisfy. The formulas are built by atomic propositions, logic connectives and temporal modal operators. Atomic propositions are statements which can be true or false and which considers the system variables [1]. An example of an atomic proposition is *"The robot is in room 1"*, where the system is the robot motion and room 1 is a subset of the area the robot can move around in. The example is expressed as in equation (1).

$$\phi_1 = r_1 \tag{1}$$

Logic connectives are operators which, when applied to the atomic propositions, describes other areas of the system's state space as a function of the named propositions [1]. An example of a logic connective is *"The robot is either in room 1 or in room 2."*, here the logic connective is the *or* which is expressed as a disjunction ($\vee$). The example is expressed as in equation (2).

$$\phi_2 = r_1 \vee r_2 \tag{2}$$

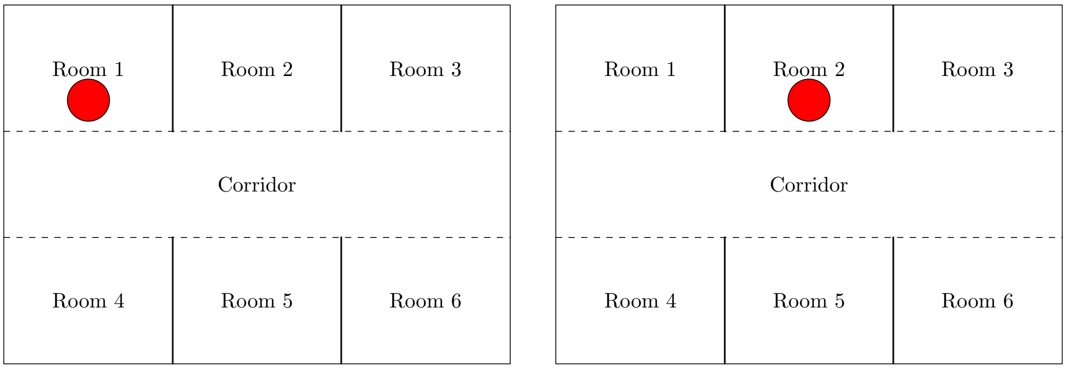
Other logic connectives includes *negation* ($\neg$), *conjunction* ($\wedge$) and *implies* ($\Rightarrow$). Temporal modal operators describe present and future events with respect to the atomic propositions [1]. An example of a temporal modal operator is *"The robot will eventually be in room 2."*, where the temporal modal operator is *eventually* ($\Diamond$). The example is expressed as (3).

$$\phi_3 = \Diamond r_2 \tag{3}$$

Other temporal modal operators includes *next* ($\bigcirc$), *always* ($\Box$) and *until* ($\mathcal{U}$). Three simple examples of implemented temporal logic are illustrated in figures 1 and 2. The examples consider a robot which is moving around 6 rooms through a corridor. In figure 1a, the robot stands still in room 1. This example satisfies the atomic proposition $r_1$ and the satisfied formulas include $\Box r_1$. In figure 1b, the robot stands still in room 2. This example satisfies the atomic proposition $r_2$. Furthermore it satisfies the formula $r_1 \vee r_2$, composed of the atomic propositions $r_1$ and $r_2$ and the logic connective $\vee$. In figure 2, the robots starts in room 1, moves through the corridor to room 2 followed by room 6 and finally return to room 1 where it stops. Throughout the run, different atomic propositions hold at different point in time. The run itself, satisfies formulas such as $\Diamond r_2$, $r_2 \mathcal{U} c$, $\neg \Diamond r_5$ among others.

Temporal logic consists of several types such as Linear Temporal Logic (LTL), Metric Interval Temporal Logic (MITL) and Signal Temporal Logic (STL). Which connectives and operators are included in each type is defined by the grammar and semantics of the type, which is presented in section 2. Up until now the focus area within research have been LTL. The subject of formal control design based on LTL have been widely studied in papers such as [1], [2], [3], [4], [5], [6], [7], [8] and [9], motivating a shift of focus to new areas such as MITL or STL. LTL considers discrete time [10], as illustrated in tables 1 and 2. While MITL and STL both considers real-time [11], [12]. Adding a time aspect to the problem would increase the possibilities regarding the specifications given to a system. For instance, it would allow for language such as *"Remain within room 1 for all time in the time interval 5 to 10 time-units."* ($\Box_{[5,10]} r_1$), a developed form of *"Remain within room 1, always."* ($\Box r_1$).

In this report, all three mentioned temporal logics will be presented; including grammar, semantics and some easy to follow examples based on motion planning. This is done in section 2, which also includes a comparison between the subjects. Based on this comparison, MITL have been chosen as the topic of study in this master thesis. The problem definition and preliminaries of the single-agent problem and the multi-agent problem are presented in section 3 and section 6 respectively, and the approach to the problems, i.e. the solutions are described in section 4 and section 7. Examples and results from MATLAB simulations are presented in sections 5 and 8. Finally, the result is summarized and evaluated in section 9, and conclusions regarding the thesis as well as future work are presented in section 10.




(a) The robot is in room 1. Hence, formulas $\phi_1$ and $\phi_2$ holds, while formula $\phi_3$ doesn't.

(b) The robot is in room 2. Hence formulas $\phi_2$ and $\phi_3$ holds while formula $\phi_1$ doesn't.

Figure 1: Example of two very simple runs of a motion planning system. The system consists of a robot which moves around in 6 rooms through a corridor. The atomic proposition set consists of the set $R = \{r_i | i = 1, 2, 3, 4, 5, 6\}$ which considers if the robot is in a given room. The red circle represents the robot.

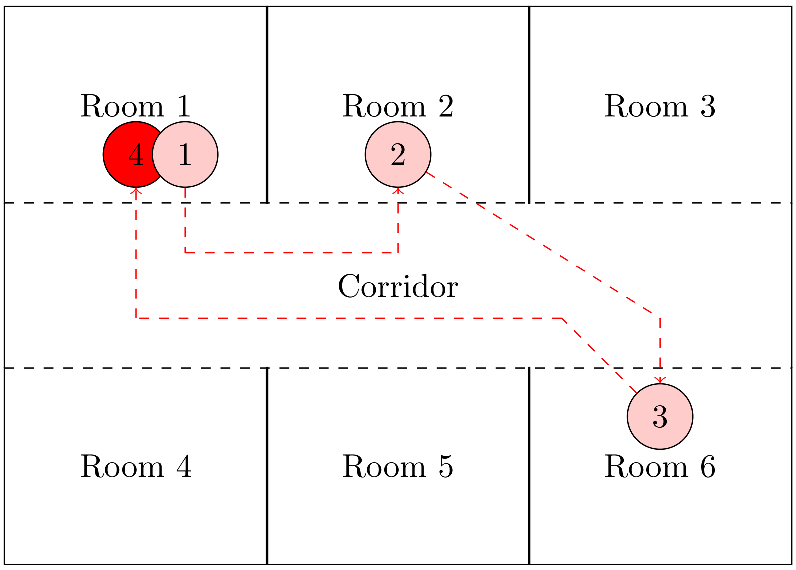



Figure 2: The figure illustrates a slightly more complicated run of the same system as introduced in 1. The robot moves according to the arrows and numbers, starting and ending in room 1. The run satisfies formula $\phi_3$. The other two formulas $\phi_1$ and $\phi_2$ are satisfied at some points in time, but not throughout the entire run.

# 2 Temporal Logic

In this section, the topics of LTL, MITL and STL are presented. The grammar, semantics and terminology of the three temporal logic versions are described in sections 2.1, 2.2 and 2.3, and the differences are discussed in 2.4, ending with a motivation of the conclusion to base this master thesis on MITL.

## 2.1 Linear Temporal Logic

The grammar of LTL is defined according to equation (4), and includes *true, atomic proposition, negation, disjunction, until* and *next* [10].

$$\phi := \top \mid \pi \mid \neg\,\phi \mid \phi \vee \psi \mid \phi\,\mathcal{U}\,\psi \mid \bigcirc\,\phi \tag{4}$$

The semantics of an LTL formula is defined as a language $Words(\phi)$ which contains all infinite words over the alphabet, $2^{\Pi}$, that satisfy $\phi$ [10]. The language is defined in accordance with Definition 2.1.1. The properties of the satisfaction relation ($\models$) are defined in Definition 2.1.2.

**Definition 2.1.1.** Let $\phi$ be an LTL formula over $\Pi$. The linear-time property induced by $\phi$ is defined by:

$$Words(\phi) = \{\sigma \in 2^{\Pi} | \sigma \models \phi\} \tag{5}$$

where $\models \subseteq 2^{\Pi} \times LTL$ is the satisfaction relation.

**Definition 2.1.2.** LTL semantics of the satisfaction relation is defined as:

$$\begin{aligned}
\sigma &\models \top \\
\sigma &\models \pi \quad &&\Leftrightarrow \quad \pi \in \sigma_0,\ (\sigma_0 \models \pi) \\
\sigma &\models \phi \wedge \psi \quad &&\Leftrightarrow \quad \sigma \models \phi\ and\ \sigma \models \psi \\
\sigma &\models \neg\phi \quad &&\Leftrightarrow \quad \sigma \not\models \psi \\
\sigma &\models \bigcirc\phi \quad &&\Leftrightarrow \quad \sigma_1\sigma_2... \models \phi \\
\sigma &\models \phi\,\mathcal{U}\,\psi \quad &&\Leftrightarrow \quad \exists j \geq 0, \sigma_j\sigma_{j+1}... \models \psi\ and\ \sigma_i\sigma_{i+1}... \models \phi, \forall\ i\ s.t.\ 0 \leq i < j
\end{aligned} \tag{6}$$

where $\sigma = \sigma_0\sigma_1\sigma_2.. \in 2^{\Pi}$ is an infinite word (see Definition 2.1.3) over $2^{\Pi}$ which satisfies $\phi$ and $\Pi = \{\pi_i | i = 0, ...n\}$ is a set of atomic propositions $\pi_i$.

From the grammar in equation (4); *eventually, always, false, conjunction, implies, equivalence* and *parity*(exclusive or), can be deducted in accordance with equation (7).

$$\begin{aligned}
\Diamond\phi &= \top\,\mathcal{U}\,\phi \\
\Box\phi &= \neg\Diamond\neg\phi \\
\bot &= \neg\top \\
\phi \wedge \psi &= \neg(\neg\phi \vee \neg\psi) \\
\phi \Rightarrow \psi &= \neg\phi \vee \psi \\
\phi \Leftrightarrow \psi &= (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi) \\
\phi \oplus \psi &= (\phi \wedge \neg\psi) \vee (\neg\phi \wedge \psi)
\end{aligned} \tag{7}$$

In all temporal logics, there are some terminology which is used. This terminology includes *words* and *runs* among other. The definitions of these terms are given below in Definition 2.1.3 and Definition 2.1.4 .

**Definition 2.1.3.** A **word** $\sigma$ is an infinite string $\sigma_0\sigma_1...$, where $\sigma_i \in 2^{\Pi}\ \forall\, i \geq 0$.

**Definition 2.1.4.** A **run** of $\sigma$ in an non-deterministic Büchi Automaton (NBA) (see Definition 2.1.6) is an infinite sequence of states s.t. $q_0 \in Q_0$ and $q_i \xrightarrow{\sigma_i} q_{i+1}$, $\forall\, i \geq 0$. Where $Q_0$ is the set of initial states.

When approaching control problems with LTL, transition systems are considered. Transition systems are a representation of systems just as automata and state space equations can be. The definition of a transition system is given in Definition 2.1.5 [10]. Examples of words included in the alphabet of a transition system and LTL formulas satisfied by a transition system is given in Example 2.1.

**Definition 2.1.5.** A transition system is a tuple $TS = (\Pi, \Pi_{init}, \Sigma, \rightarrow, AP, L)$, where

- $\Pi = \{r_i | i = 0, ..., n\}$ 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 actions,
- $\rightarrow \subseteq \Pi \times \Sigma \times \Pi$ is a transition relation, the expression $\delta(r_i, \sigma_j) = r_k$ i used to express transition from $r_i$ to $r_k$ under the action $\sigma_j$,
- $AP = \{ap_i | i = 0, ..., m\}$ is a set of atomic propositions and
- $L : \Pi \rightarrow 2^{AP}$ is a labelling function.

As mentioned above, another representation is automata. The definition of a non-deterministic automaton is given in Definition 2.1.6 [10]. LTL formulas can be translated into automata, using the fact that some states are accepting, creating an automaton which is accepting of all runs which satisfy the LTL formula it is built for. Definitions of accepting words and accepting runs are given in Definition 2.1.8 and Definition 2.1.7, also accepting language in Definition 2.1.9 [13]. An example of a automaton constructed from a temporal logic formula and accepting words/runs are given in section 2.2 in Example 2.2.

**Definition 2.1.6.** A non-deterministic Büchi Automaton is a tuple $A = (S, S_{init}, E, F, AP, \mathcal{L})$ where

- $S = \{s_i | i = 0, ..., n\}$ is a finite set of states,
- $AP = \{ap_i | i = 0, ..., l\}$ is a finite set of inputs, called an alphabet,
- $E \subseteq S \times AP \times S$ is a transition relation,
- $S_{init} \subseteq S$ is a set of initial states and
- $F \subseteq S$ is a set of accepting states,
- $\mathcal{L}$ is a labelling function, labelling some set of atomic propositions to each state.

**Definition 2.1.7.** 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.

**Definition 2.1.8.** An accepting string is a string $\sigma$ which has an accepting run in $A$.

**Definition 2.1.9.** An accepted language $L(A)$ is a set of all accepting strings of $A$.

**Example 2.1.** Returning to the example with the robot moving around 6 rooms, the system can be translated into the transition system presented in figure 3, assuming that the robot starts in room 1 and that the controllers which induces the transitions are $a, b, ..., f$ according to the figure. The language of the system includes any combination of $a, b, .., f$ (any word) starting with $a$ and otherwise only containing any of the combinations $(ba)^n$, $(ac)^n$, $(cb)^n$, $(de)^n$, $(ea)^n$ and/or $(fd)^n$, as well as a possible end letter. An example of a word included in the language is: *aaccbe*, which would take the robot from room 1 ($q_1$[1]), through the corridor ($q_0$) to room 4 ($q_4$), back through the corridor to room 5 ($q_5$)and finally back through the corridor to room 3 ($q_3$). Furthermore, the system would satisfy LTL formulas such as: $\phi = r_4 \Rightarrow (\bigcirc c)$, where c is the corridor. The formula translates to *"The robot being in room 4 implies that the next room it enters will be the corridor."*.

[1]Here $q_i$ is used to represent states in the transition system instead of $r_i$, this is done in order to avoid confusion between states and rooms.

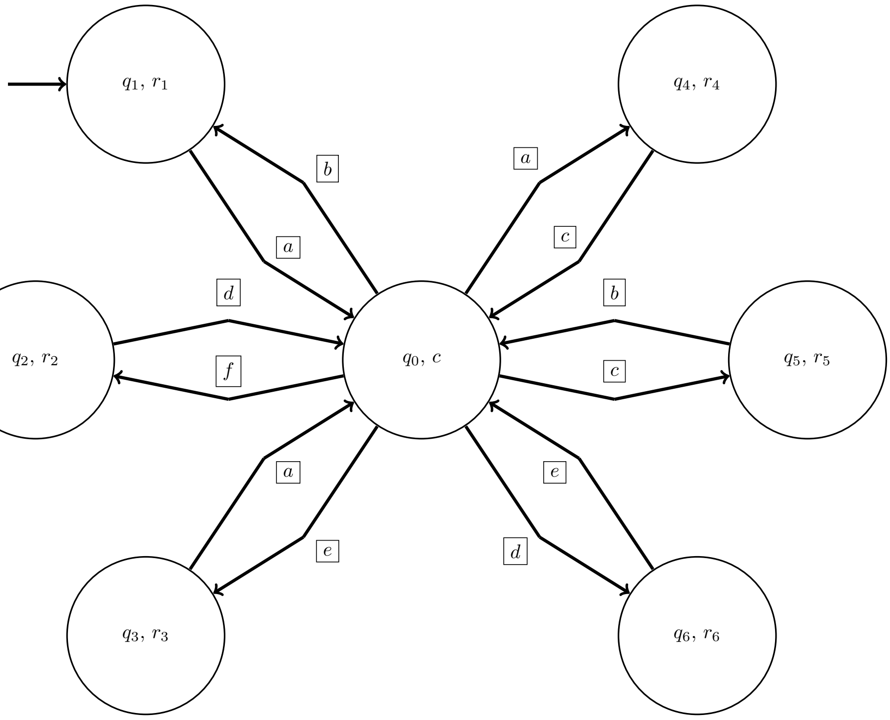



Figure 3: Transition system of a robot moving through 6 rooms $q_1, .., q_6$ by a hallway $q_0$. The robot starts in room 1.

## 2.2 Metric Interval Temporal Logic

This section contains definitions and examples considering both MITL as well as the timed aspects of automata and transition systems. Previous work within the fields which have been used as a basis for this section, includes [14], [7], [15] and [12]. In MITL real-time is considered rather than discrete-time. Therefore, this section is initialized by some definitions regarding timed terms. Namely, *time sequence*, *timed word* and *timed language*, the definitions follow [13].

**Definition 2.2.1.** A time sequence $\tau = \tau_0\tau_1...$ is an infinite sequence of time values which satisfies

- $\tau_i \in I \subset \mathbb{Q}_+$,
- $\tau_i < \tau_{i+1}$, $\forall i \geq 0$ and
- $\exists i \geq 1$, $s.t.\ \tau_i > t$, $\forall t \in I$.

**Definition 2.2.2.** A timed word $w$ over the set $\Pi$ is a finite sequence $w = (\sigma(0), \tau_0)(\sigma(1)\tau_1)...(\sigma(n)\tau_n)$, where $\sigma = \sigma(0)\sigma(1)...\sigma(n)$ is a finite word over $2^\Pi$ (see Definition 2.1.3) and $\tau = \tau_0\tau_1...\tau_n$ is a time sequence (see Definition 2.2.1).

**Definition 2.2.3.** A timed language $L$ over $\Pi$ is a set of timed words, i.e. $L = \{w_i | i = 0, .., n\}$.

The MITL grammar is defined as equation (8) [12], translating to *true, proposition, negation, disjunction* and *until*.

$$\phi := \top \mid p \mid \neg\, \phi \mid \phi \vee \psi \mid \phi\, \mathcal{U}_{[a,b]}\, \psi \tag{8}$$

The semantics of MITL is illustrated in Definition 2.2.4.

**Definition 2.2.4.** Let $\phi$ be an MITL formula over $\Pi$ and $\tau(s, I)$ be a timed state sequence (*timed word*). The semantics of the satisfaction relation is then defined as:

$$\begin{aligned}
\tau \models \pi \quad &\Leftrightarrow \quad \pi \in s_0\ (s_0 \models \pi) \\
\tau \models \neg\phi \quad &\Leftrightarrow \quad \tau \not\models \phi \\
\tau \models \phi \wedge \psi \quad &\Leftrightarrow \quad \tau \models \phi\ and\ \tau \models \psi \\
\tau \models \phi\, \mathcal{U}_I\, \psi \quad &\Leftrightarrow \quad \exists t \in I,\ s.t.\ \tau^t \models \psi\ and\ \forall t' \in (0,t), \tau^{t'} \models \phi
\end{aligned} \tag{9}$$

The grammar in (8) can be extended to include *eventually, always, false* and *conjunction*, as illustrated in equation (10).

$$\begin{aligned}
\Diamond_{[a,b]}\phi \quad &= \quad \top\, \mathcal{U}_{[a,b]}\, \phi \\
\Box_{[a,b]}\phi \quad &= \quad \neg\Diamond_{[a,b]}\neg\phi \\
\bot \quad &= \quad \neg\top \\
\phi \wedge \psi \quad &= \quad \neg(\neg\phi \vee \neg\psi)
\end{aligned} \tag{10}$$

As for the LTL, the system which is evaluated can be represented by a transition system. However, in order to take in consideration the time aspect which MITL includes, weights are added to transitions. These weights corresponds to the time a transition takes. The definition of a weighted transition system is given in Definition 2.2.5 [16].

**Definition 2.2.5.** A weighted transition system is a tuple $T = (\Pi, \Pi_{init}, \Sigma, \rightarrow, AP, L, d)$ where

- $\Pi = \{r_i | i = 0, ..., n\}$ 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,
- $\rightarrow: \Pi \times \Sigma \rightarrow 2^\Pi$ is a transition map, the expression $\delta(r_i, \sigma_j) = r_k$ i used to express transition from $r_i$ to $r_k$ under the action $\sigma_j$,
- $AP$ is a set of observations,
- $L : \Pi \rightarrow AP$ is an observation map and
- $d : \Pi \times \Sigma \rightarrow \mathbb{R}_+$ is a positive weight assignment map.

Corresponding to the runs, defined in section 2.1, there are timed runs, taking in consideration whether some clock-constraints are fulfilled. The definition is given in Definition 2.2.6.

**Definition 2.2.6.** A timed run $r^t = (r(0), \tau_0)(r(1), \tau_1)...(r(n)\tau_n) \in \Pi \times I$ for a transition system $T$ (see Definition 2.2.5) is a finite sequence where $r(0)r(1)...r(n)$ is an untimed run (see Definition 2.1.4) and $\tau_0\tau_1...\tau_n$ is a time sequence s.t.

- $\tau_0 = 0$
- $\tau_{i+1} = \tau_i + d(r(i), r(i+1)),\ \forall i \in \{0, 1, ..., n-1\}$,

where $d(r(i), r(i+1))$ is the transition weight for the transition between the state corresponding to $r(i)$ to the state which corresponds to $r(i+1)$, i.e. the time the transition needs.

Finally, the MITL formula can be translated into a timed automaton (see section 4.2 for details). The timed automaton includes clocks and clock-constraints. Before presenting the definition of a timed Büchi automaton, the definitions of clock constraints and clock valuation will be considered. The definition of clock constraints is given in Definition 2.2.7 [17].

**Definition 2.2.7.** Let $C$ be a finite set of clocks $C = \{c_1, c_2, ..., c_M\}$, a set of clock constraints $\Phi_C$ over $C$ is then defined as:

$$\Phi_C := \top|\bot|c \bowtie k|c - c' \bowtie k|\Phi_1 \wedge \Phi_2|\Phi_1 \vee \Phi_2,$$

where $k \in \mathbb{N}$ is a non-negative integer, $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ is an comparison operator and $c, c' \in C$ are clocks.

The clock valuations are defined as Definition 2.2.8 [18].

**Definition 2.2.8.** A clock valuation (or interpretation) $v$ for a set of clocks $C$, assigns a real value to each clock and hence maps from $C$ to $\mathbb{R}_+ \cup \{0\}$. $v + \delta$ denotes the valuation which maps every clock $c$ to the value $v(c) + \delta$. $v[R := 0]$ denotes the valuation for $C$ which assigns 0 to each $c \in R \subseteq C$, and agrees with $v$ over the rest of the clocks.

Now, we proceed with the definition of a timed Büchi automaton, which is given in Definition 2.2.9 [13].

**Definition 2.2.9.** A timed Büchi Automaton (TBA) is a tuple
$A = (S, S_0, X, I, E, F, AP, \mathcal{L})$ where

- $S = \{s_i | i = 0, 1, ...\}$ is a finite set of locations,
- $S_0 \in S$ is the set of initial locations,
- $2^{AP}$ is the alphabet or set of actions ($AP$ is the set of atomic propositions),
- $X$ is a finite set of clocks,
- $F \in S$ is a set of accepting locations,
- $I : S \to \Phi_X$ is a map labelling each state $s_i$ with some clock constraint,
- $E \subseteq S \times \Phi_X \times 2^X \times S$ is a set of transitions and
- $\mathcal{L}$ is a labelling function, labelling some set of atomic proposition to each state.

A state of $A$ is a pair $(s, v)$ where $s \in S$ is a location and $v$ is a valuation that satisfies $I(s)$. 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 $|X|$ number of valuations $v_i = 0$.

Similarly to the accepting word and accepting runs of the BA constructed from the LTL formula, there are accepting timed words and accepting timed runs for the TBA. An example of accepting timed words and accepting timed runs are given in Example 2.2.

**Example 2.2.** Consider the timed automata $A_\phi$ illustrated in figure 4. The automata consists of 3 states; $s_0$, $s_1$ and $s_2$, where $s_0$ is the initial state and $s_1$ is the accepting state. The accepting words of $A_\phi$, are the words which results in the system visiting the accepting state $s_1$ infinitely often. Similarly, the accepting runs of $A_\phi$, are the runs which visits the accepting state infinitely many times. An example of an accepting word of $A_\phi$ is:

$$(0, \{\neg a\})(t', \{a\})$$

where $t' \leq b$. The corresponding accepting run is :

$$(s_0, 0) \overset{0,\{\neg a\}}{\longrightarrow} (s_0, 0) \overset{t',\{a\}}{\longrightarrow} (s_1, 0)$$

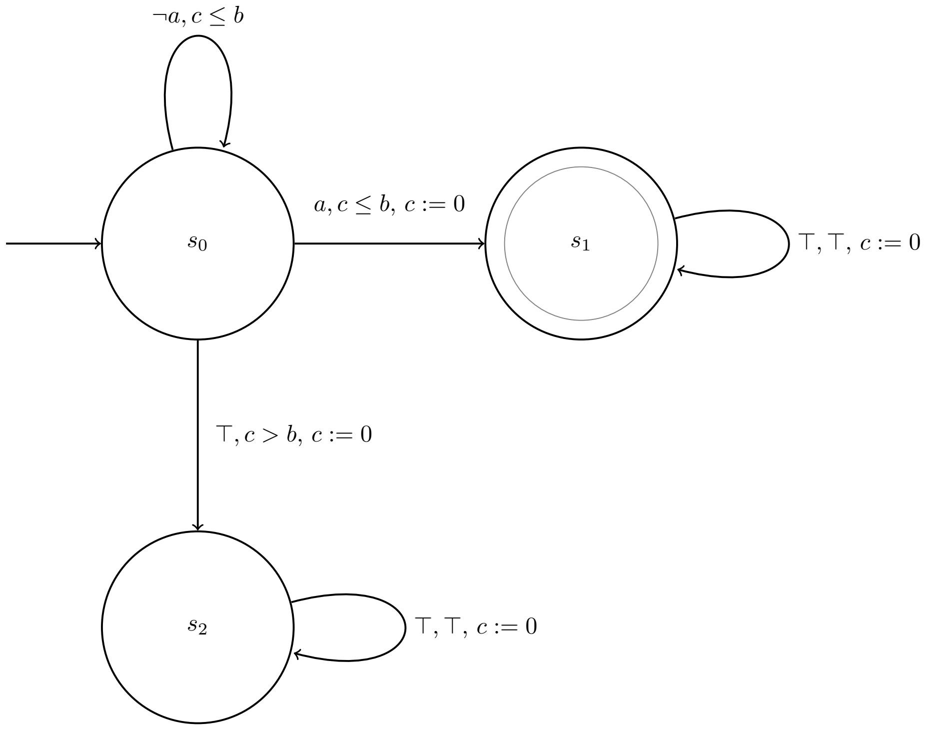



Figure 4: Illustration of the timed automata $A_\phi$, constructed of the MITL formula $\phi = \Diamond_{\leq b} a$, where $a$ is an atomic proposition i.e. $\Pi = \{a\}$.

Examples of a non-accepting words of $A_\phi$ is:

$$(0, \{\neg a\})(t'', \{a\})$$

where $t'' > b$, and

$$(\tau, (\{\neg a\})^w)$$

for any infinite time sequence $\tau$. In the first example the system transition to state $s_2$ due to the clock-constraint $c \leq b$ being broken, in the other example the atomic proposition $a$ is never fulfilled. The corresponding runs of the words are:

$$(s_0, 0) \xrightarrow{0,\{\neg a\}} (s_0, 0) \xrightarrow{t'',\{a\}} (s_2, 0)$$

and

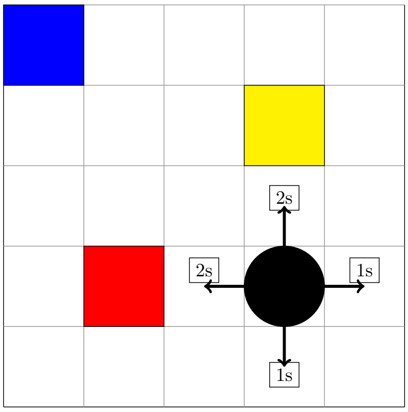
$$(s_0, 0) \xrightarrow{\tau[0 \ldots i-1],(\{\neg a\})^i} (s_0, 0) \xrightarrow{\tau[i],\{\neg a\}} (s_2, 0) \xrightarrow{\tau[i+1 \ldots],(\{\neg a\})^w} (s_2, 0)$$

where $\tau[i]$ is the $i$th element of the sequence $\tau$, $\tau[i..j]$ is the elements between $i$ and $j$ and $\tau[i]$ is the first element which is greater than $b$.

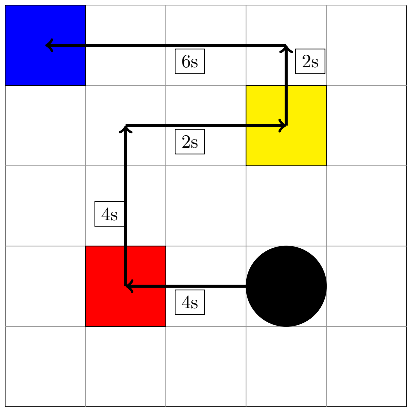
The accepting words corresponds to the sequences of atomic propositions and time which satisfies the MITL formula $\phi = \Diamond_{\leq b} a$, which the automaton is constructed of.

Notice: All accepting words are those with a prefix $(0, \{\neg a\})(\tau, (\{\neg a\})^n)(t', \{a\})$, for some $t' \leq b$, where $\tau$ is a finite time sequence of length $n$ and $t' > \tau_j$, $\forall j \leq n$.

An example of a weighted transition system which is evaluated by an MITL formula follows in Example 2.3.




(a) The robot (illustrated as a black dot), can move within the $5 \times 5$ area. Moving one square upwards or to the left demands 2s, while moving one square downwards or to the right only demands 1s.




(b) Example of a run which fulfils the MITL formula given in Example 2.3.

Figure 5: Motion planning example of a robot moving through a partitioned space. The figures illustrate costs of movements and a possible run.

**Example 2.3.** Considering the system illustrated in figure 5a, a robot is moving within a partitioned area of the size $5 \times 5$. Movements upwards or to the left costs the robot 2s, while movements downwards or to the right only costs 1s. Now, consider the MITL formula

$$\phi = \Diamond_{\leq 5s} red \wedge \neg blue\, \mathcal{U}_{\leq 10s}\, yellow \wedge \Box(yellow \Rightarrow \Diamond_{\leq 12s} blue)$$

The formula states

- that the robot must reach the red square within 5s,
- that it mustn't go to the blue square until it has been at the yellow square,
- that it must reach the yellow square within 10s and
- that it always must go to the blue square within 12s if it enters the yellow square.

Assuming the robot starts at the square which it is located at in the figure, the MITL formula can be satisfied. An example of a run which satisfy the formula is given in figure 5b. Here the robot reaches the red square by 4s ($4 \leq 5$ - ok!), it doesn't enter the blue square until it has been in the yellow square, it enters the yellow square by 10s ($10 \leq 10$ - ok!), and finally the blue square within 8s of entering the yellow square ($8 \leq 12$ - ok!).

## 2.3 Signal Temporal Logic

Previous work within STL include papers such as [12], [19], [20], [21] and [22]. This section is based on the information presented in those papers. The grammar of STL is given by equation (11) and includes *true, atomic proposition, negation, disjunction* and *until.* The grammar can be extended to include *eventually, always, conjunction* and *false* in accordance with equation (12).

$$\phi := \top \mid \mu \mid \neg\, \phi \mid \phi \vee \psi \mid \phi\, \mathcal{U}_{[a,b]}\, \psi \tag{11}$$

$$
\begin{aligned}
\Diamond_{[a,b]}\phi &= \top\,\mathcal{U}_{[a,b]}\,\phi \\
\Box_{[a,b]}\phi &= \neg\Diamond_{[a,b]}\neg\phi \\
\phi\wedge\psi &= \neg(\neg\phi\vee\neg\psi) \\
\bot = \neg\top
\end{aligned} \tag{12}
$$

Where the value of $\mu$ is determined by the underlying signal $x$; $\mu \equiv f(x) \sim c$, where $f$ is a scalar-valued function over $x$, $\sim\in \{<, >, \leq, \geq, =, \neq\}$ and $c$ is a constant real number. The boolean semantics of the satisfaction relation is given by Definition 2.3.1.

**Definition 2.3.1.** The boolean semantics of STL is defined as

$$
\begin{aligned}
(x,t)\vDash\mu &\Leftrightarrow x \text{ satisfies } \mu \text{ at time } t \\
(x,0)\vDash\phi &\Leftrightarrow x\vDash\phi \\
(x,t)\vDash\neg\mu &\Leftrightarrow (x,t)\nvDash\mu \\
(x,t)\vDash\phi\wedge\psi &\Leftrightarrow (x,t)\vDash\phi\ and\ (x,t)\vDash\psi \\
(x,t)\vDash\phi\vee\psi &\Leftrightarrow (x,t)\vDash\phi\ or\ (x,t)\vDash\psi \\
(x,t)\vDash\Box_I\phi &\Leftrightarrow \forall t'\in I+t,\ (x,t')\vDash\phi \\
(x,t)\vDash\phi\,\mathcal{U}_I\,\psi &\Leftrightarrow \exists t'\in I+t,\ s.t.\ (x,t')\vDash\psi\ and \\
&\qquad \forall t''\in\left[t,t'\right],\ (x,t'')\vDash\phi
\end{aligned} \tag{13}
$$

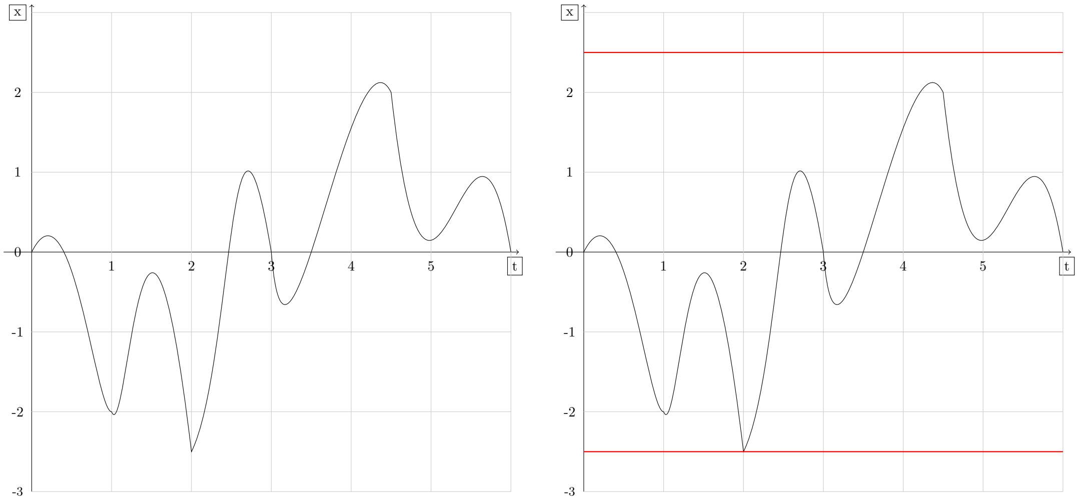
The new aspect of STL, which MITL is lacking is the possibility to measure how close the signal is to not fulfil $\mu$. This measurement is expressed by $\rho$. The value of $\rho$ is determined by the signal $x$ the atomic proposition $\mu$ and the time $t$. The semantics of $\rho$, also called the quantitative semantics, are given by Definition 2.3.2.

**Definition 2.3.2.** The quantitative semantics of STL is defined as

$$
\begin{aligned}
\rho(\mu,x,t) &= f(x(t)) \\
\rho(\neg\mu,x,t) &= -\rho(\mu,x,t) \\
\rho(\phi\wedge\psi,x,t) &= min(\rho(\phi,x,t),\rho(\psi,x,t)) \\
\rho(\phi\vee\psi,x,t) &= max(\rho(\phi,x,t),\rho(\psi,x,t)) \\
\rho(\Box_I\phi,x,t) &= \underset{t'\in I}{min}(\rho(\phi,x,t')) \\
\rho(\phi\,\mathcal{U}_I\,\psi,x,t) &= \underset{t'\in I}{max}\left(min\left(\rho(\psi,x,t'),\underset{t''\in[t,t']}{min}\rho(\phi,x,t'')\right)\right)
\end{aligned} \tag{14}
$$

An example of a signal which is evaluated by some STL formulas follows in Example 2.4.

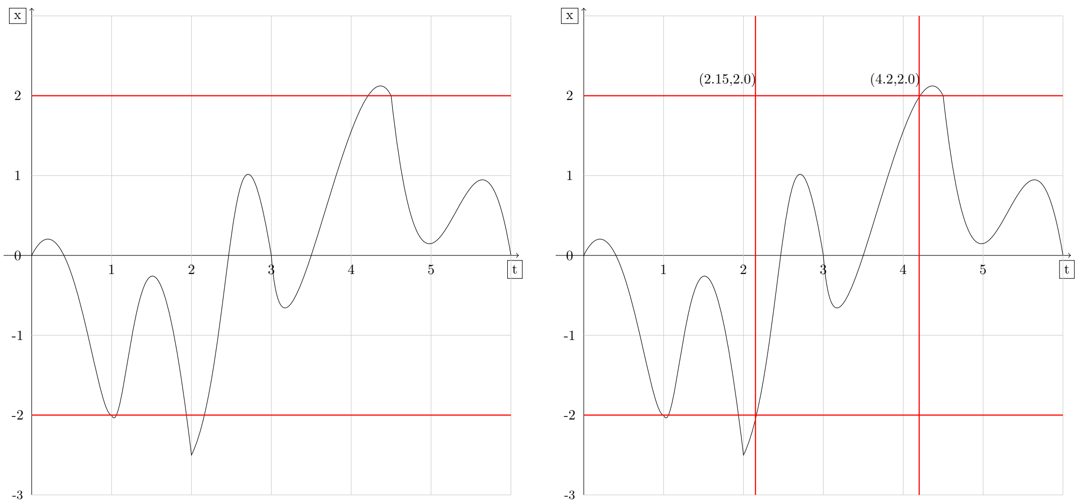
**Example 2.4.** An example of a system which can be evaluated by an STL formula is given in figure 6a. The figure illustrates a signal $x$ evolving over time. It is clear from figure 6b, that the system satisfies the STL formula $\Box(|x| < 3)$. While it does not satisfy $\Box(|x| < 2)$ (see figure 7a) for all t. However, as illustrated in figure 7b, it does satisfy the formula for some t, hence the STL formula $\Box_{[2.15,4.2]}(|x| < 2)$ is satisfied.




(a) A signal $x$ evolving over time $t$, defining a system.

(b) The absolute value of the signal never exceeds 3, and hence satisfy $\square(|x| < 3)$.

Figure 6: Example of a signal under evaluation of an STL formula.




(a) The absolute value of the signal does exceed 2, and hence doesn't satisfy $\square(|x| < 2)$.

(b) The absolute value of the signal doesn't exceed 2 at the time interval $[2.15, 4.2]$, and hence satisfy $\square_{[2.15,4.2]}(|x| < 2)$.

Figure 7: Example of how the real-time could be applied in order to satisfy the softer version of a temporal logic formula.

## 2.4 Comparison

The main differences between the already studied LTL and the possible areas of study, STL and MITL, are illustrated in table 1. An example is given in table 2. It follows that MITL is an extension of LTL which includes time-constraints and that STL is a further extension which, besides including time-constraints, also predicates over real-values compared to LTL and MITL which predicates over boolean. When approaching the problem at hand the considered methods would therefore differ.

Table 1: Properties that differ between LTL, MITL and STL, [23].

| | Predicates over | Time property |
|---|---|---|
| LTL | Boolean | Discrete-time |
| MITL | Boolean | Real-time |
| STL | Real-value | Real-time |

Table 2: Example of differences conserning the expression possibilities of LTL, MITL and STL.

| | Example | |
|---|---|---|
| LTL | $p\,\mathcal{U}\,q$ | At some point in the future, q will be true, until then p will be true. |
| MITL | $p\,\mathcal{U}_{[1,5]}\,q$ | At some point in the time interval 1 to 5 time-units, q will be true, until then p will be true. |
| STL | $(x(t) < 2)\,\mathcal{U}_{[1,5]}\,(y(t) > 5)$ | At some point in the time interval 1 to 5 time-units, y will be greater than 5, until then x will be smaller than 2. |

The approach to the problem using LTL is described by scheme 8. The scheme is a remake of the image*"Temporal Logic-Based Planning: Hierarchical Approach"* in [24]. In short, the LTL formula and the continuous system is abstracted into a joint discrete model by creating an automata product of a Büchi automaton representing the LTL formula and a discrete model abstracted directly from the system. The control input is then designed based on accepted runs of the automata product.

Due to STL predicating over real-values, it is not possible to translate an STL formula to an automaton. This would not be an issue for the MITL approach. To solve the problem based on STL, the considered approach would become an optimization problem where the system is considered as the cost function and the STL formula as conditions. Due to the authors preference towards automaton, the area of MITL has been chosen.

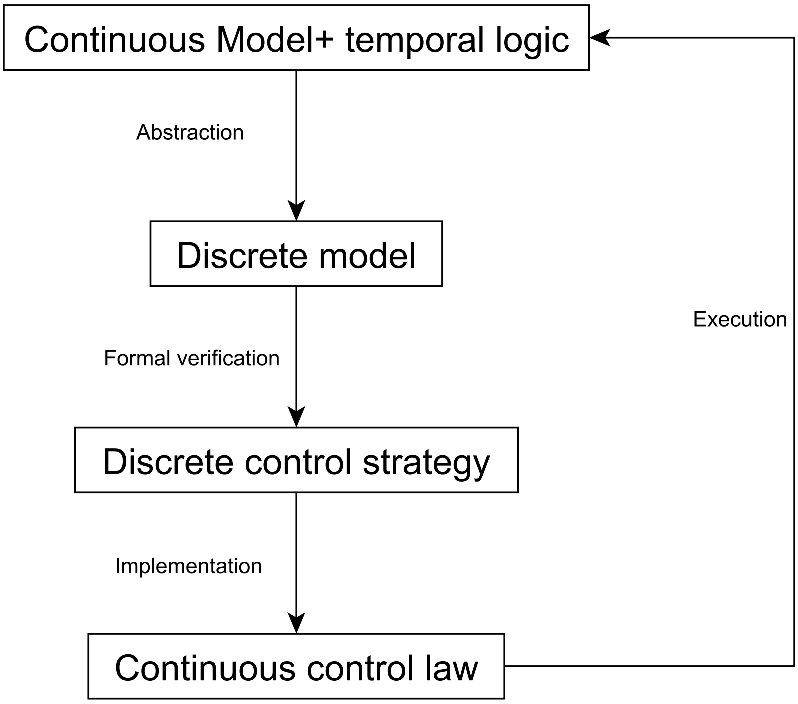



Figure 8: Remake of the scheme *"Temporal Logic-Based Planning: Hierarchical Approach"* in [24] by Jana Tumova.

# 3 Problem Definition 1

The problem considered in this master thesis is finding a control input for the continuous linear system (15), which fulfil the MITL formula $\phi$. (15) is assumed to be controllable and stabilizable.

$$\begin{aligned} \dot{x} &= Ax &+ Bu \\ &x \in X &u \in U \end{aligned} \tag{15}$$

# 4 Solution Approach 1

The intended approach to the problem is illustrated in scheme 9. The approach has been constructed based on previous work such as [23], [2], [1] and [3], the idea being to adapt the approach towards the LTL problem such that it suits the MITL issue. Each step of the approach is described in more detail in sections 4.1, 4.2, 4.3 and 4.4.

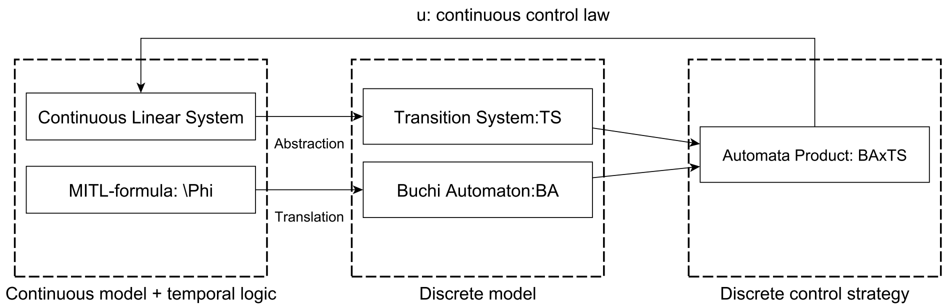



Figure 9: Scheme describing the MITL approach to the problem.

## 4.1 Abstraction of the Continuous System to a Transition System

Assuming that $x \in R_N(a^{\mathbb{X}}, b^{\mathbb{X}}) \subset \mathbb{R}^N$ in (15), that is that the state space of the system can be divided into rectangles of dimension N (see Definition 4.1.1), the following approach towards abstracting the environment into a weighted transition system is suggested, it follows the theory presented in [16].

**Definition 4.1.1.** An $N$-dimensional rectangle $R_N(a, b) \subset \mathbb{R}^N$ is characterized by two vectors $a, b$, where $a = (a_1, a_2, ..., a_N)$, $b = (b_1, b_2, ..., b_N)$ and $a_i < b_i$, $\forall i = 1, 2, ..., N$. The rectangle is then given by

$$R_N(a, b) = \{x \in \mathbb{R}^N | \forall i \in \{1, 2, .., N\} : a_i \leq x_i \leq b_i\} \tag{16}$$

That is, the vector $a$ includes the points in each dimension which the rectangle's first vertex is positioned in and the $b$ vector includes the points in each dimension which the rectangle's last vertex is positioned in.

Firstly, the state space $x$ is divided into rectangles in accordance with the atomic propositions which are considered. Namely, if $AP = \{ap_i | i = 0, ..., l\}$ is the set of atomic propositions then the partition follows equation (17). Which ensures that there is always a distinct answer regarding if an atomic proposition is true or false within a rectangle, i.e. it eliminates the possibility of an atomic proposition being true in part of a rectangle and false in the other part. Now, if the MITL formula is constructed of the atomic propositions $ap_i$, it will be possible to determine if a run in the partitioned state space satisfies the formula. The first step in abstracting system (15) to a weighted transition system, is then to define the states q in Definition 2.2.5 as the rectangles $R_N(a^{\mathbb{X}}, b^{\mathbb{X}})$.

$$[ap_i] = \cup_{j=1}^{d_i} R_N(a^{j,ap_i}, b^{j,ap_i}) \subset R_N(a^{\mathbb{X}}, b^{\mathbb{X}}),\ d_i \in \mathbb{N} \tag{17}$$

The next step is to include the time aspect in the abstraction. [16] suggest a solution by introducing the *Facet Reachability Problems*, which considers whether a closed-loop system can reach determined facets of a rectangle. Namely, is it possible to design a control input such that the system can exit one rectangle and enter another? A theorem determining when the problem is solvable, i.e. when such a controller can be designed, is presented in Theorem 1, introduced in [16].

**Theorem 1.** *Let $R_N(a,b)$ be a rectangle and $\varepsilon \subset \mathcal{F}(a,b)$ be a non-empty subset of its facets. $\exists$ a multi-affine feedback controller $k : R_N(a,b) \to U$ s.t. all the trajectories of the closed-loop system* (15)*, originating in $R_N(a,b)$, leave it through a facet from the set $\varepsilon$ in finite time if:*

$$n_F^\top(Av + Bk(v)) \leq 0, \forall F \in \mathcal{F}_v \backslash \varepsilon,\ \forall v \in \mathcal{V}(a,b), \tag{18}$$

*and*

$$0 \notin Conv(\{Av + Bk(v) | v \in \mathcal{V}(a,b)\}) \tag{19}$$

*where Conv denotes the convex hull and $\mathcal{V}(a,b)$ is the vertexes (corners) of the rectangle.*

Equation (18) states that the closed-loop system (15) must move away from the facets which are not approved. While equation (19) includes that the system must always evolve (the speed of the system can't be 0). Note, equation (19) contains more information than this. Theorem 1, states that it is possible to design a controller such that the system always exit a rectangle through a determined facet if equation (18) and equation (19) are satisfied. If there is only one approved facet, i.e if $\varepsilon = \{F\}$, condition (19) can be simplified to equation (20), stating that the system must move towards the approved facet.

$$n_F^\top(Av + Bk(v)) > 0, \forall v \in \mathcal{V}(a,b) \tag{20}$$

[16] continue by proposing that the system will leave the rectangle through the given facet in time less than or equal to $T^F$, where $T^F$ is defined according to equation (21), where $i$ corresponds to the outer normal $e_i$ which the particular facet has and $s_F$ and $\overline{s_F}$ are defined according to (22).

$$T^F = ln\left(\frac{s_F}{\overline{s_F}}\right)\frac{b_i - a_i}{s_F - \overline{s_F}} \tag{21}$$

$$s_F = \min_{v \in \mathcal{V}(F)} (h(v) + Bk(v))_i \quad \overline{s_F} = \min_{v \in \mathcal{V}(\overline{F})} (h(v) + Bk(v))_i \tag{22}$$

The idea behind $T^F$ is to calculate the time it would take for the system to reach the facet, assuming that it starts at the point the furthest away from it, i.e. on the opposite facet, and that it moves towards the facet at the slowest possible rate given the determined $u$. That is, choosing $T^F$ as the maximum time required for the transition to occur. For a continuous linear system (15) this corresponds to solving the problem

$$\begin{aligned} \dot{x}_i &= (Ax)_i + (Bu)_i \\ x(0)_i = x_0 \qquad & x(t_1)_i = x_1 \end{aligned} \tag{23}$$

where $i$ is the norm direction of the facet, for $t_1$. Which gives the time it will take for the system (15) to evolve from $x_0$ to $x_1$ in direction $i$. Hence, if $x_1$ is a point along the facet, $t_1$ is the time it will take for the system to reach the facet from the point $x_0$. Now, assuming that $u$ is linear, i.e. $Bu = B_1x + B_2$ system (23) can be rewritten to (24).

$$\dot{x}_i = ((A + B_1)x)_i + (B_2)_i = (A^*x)_i + B_i^* = \sum_{j=1}^{n} a_{ij}^* x_j + B_i^* \tag{24}$$

Finally, by introducing $C_i^* = B_i^* + \sum_{j=1, j\neq i}^{n} a_{ij}^* x_j$, the system can be further simplified to (25).

$$\begin{aligned} \dot{x}_i &= a_{ii}^* x_i + C_i^* \\ x(0)_i = x_0 \qquad & x(t_1)_i = x_1 \end{aligned} \tag{25}$$

The equation is solved by separating $x_i$ from $t$ as shown in (26), assuming that $C_i^*$ can be treated as a constant. The assumption is directly valid if $u$ is designed such that the dependence $\dot{x}_i$ has of $x_j$ for $j \neq i$ is cancelled out, i.e. if $a_{ij}^* = a_{ij} + (B_1)_{ij} = 0$. If this is not true, i.e. if there is a desire to choose $(B_1)_{ij}$ differently, the assumption would still be indirectly valid. The motivation for this is that the dependence on $x_j$ will be linear, this corresponds to solving the problem as if $C_i^*$ were constant for two cases - $x_j = x_{j,max}$ and $x_j = x_{j,min}$ where $min$ and $max$ are the smallest and biggest value $x_j$ can have in the rectangle - and then using the maximum of the two solutions.

$$\begin{aligned} \frac{dx_i}{dt} &= a_{ii}^* x_i + C_i^* \rightarrow \\ \int dt &= \int \left( \frac{1}{a_{ii}^* x_i + C_i^*} \right) dx_i \rightarrow \\ t + k &= \frac{ln(a_{ii}^* x_i + C_i^*)}{a_{ii}^*} \end{aligned} \tag{26}$$

Now, $k$ can be determined using $x(0)_i = x_0$, giving the result shown in (27), and using $x(t_1)_i = x_1$, $t_1$ can be determined as (28).

$$k = \frac{ln(a_{ii}^* x_0 + C_i^*)}{a_{ii}^*} \tag{27}$$

$$t_1 = \frac{ln(a_{ii}^* x_1 + C_i^*) - ln(a_{ii}^* x_0 + C_i^*)}{a_{ii}^*} \tag{28}$$

Finally, $T^F$ is the maximum time it will take for the system to reach the facet $(x_1)$. This corresponds to $t_1$, when $x_0$ is one of the points which is the furthest away from $x_1$, i.e. when $x_0$ is a point on the opposite facet $\overline{F}$, and when the system evolves at the slowest possible speed, i.e.when $C_i^*$ is minimized. Hence, $T^F$, for a continuous linear system can be defined as (29), where $s_F$ and $s_{\overline{F}}$ are defined as (30), and $a_{ii}^*$ is the $i \times i$th element of the matrix $A + B_1$. Here $A$ is the matrix determining the open-loop dependence on $x$, and $B_1$ is the matrix determining the added dependence of $x$ from the closed-loop.

$$T^F = ln \left( \frac{s_F}{s_{\overline{F}}} \right) \frac{1}{a_{ii}^*} \tag{29}$$

$$s_F = \underset{x \in F}{a_{ii}^* x_i + C_i^*} \qquad s_{\overline{F}} = \underset{x \in \overline{F}}{a_{ii}^* x_i + C_i^*} \tag{30}$$

Furthermore, [16] states that the time bound can be minimized by using the controller which maximizes $n_F^\top (Av + Bu_v)$, i.e. which maximizes the speed of which the system moves towards the given facet. More precisely the optimization problem given by equation (31) must be solved for all vertexes in a rectangle for a given facet. In a 2 dimensional case this results in 4 problems for each approved facet in each rectangle.

$$\begin{aligned} &\max_{u_v \in U} \left( n_F^\top (Av + Bu_v) \right) \\ &n_{F'}^\top (Av + Bu_v) \leq -\epsilon, \quad \forall F' \in F_v \setminus F \\ &\qquad\qquad u_v \in U \qquad\quad \epsilon > 0 \end{aligned} \tag{31}$$

Now, the time can be incorporated into the weighted transition system by setting the weights $d$ for each transition according to $T^F$, that is as the maximal time the system will need to finish the transition. Also, the inputs $\sigma$ can be set to the control-input $u$ which will cause the transition.

As for the remaining properties of the weighted transition system; $\rightarrow$ corresponds to the allowed transitions i.e. the approved facets, $AP = AP$ (the set of atomic propositions) and $L$ is the function that maps which atomic propositions that holds in each state (rectangle). An example is given in Example 4.1.

**Example 4.1.** Let the continuous linear system to be controlled be:

$$\dot{x} = x + u \tag{32}$$

where $x \in [(1,1)^\top, (5,6)^\top] \subset \mathbb{R}^2$ and $u \in [(-7,-7)^\top, (6,6)^\top] \subset \mathbb{R}^2$. Furthermore, let the MITL formula to be satisfied $\phi$ be over the atomic proposition set $AP = \{ap_0, ap_1, ap_2, ap_3\}$. Where $ap_i$ is defined as:

$$\begin{aligned} ap_0 :&\quad x_1 > 4, \quad x_2 < 3 \\ ap_1 :&\quad x_1 > 4, \quad x_2 > 3 \\ ap_2 :&\quad x_1 < 3, \quad x_2 < 3 \\ ap_3 :&\quad x_1 < 3, \quad x_2 > 3 \end{aligned} \tag{33}$$

The state space of the linear system (32) is then illustrated in figure 10a. By applying equation (17) on the state space, with $ap_i$ as defined in (33), the partition illustrated in figure 10b follows. This corresponds to a weighted transition system $T = (\Pi, , \Pi_{init}, \Sigma, \rightarrow, AP, L, d)$ with 5 states $r_i$ according to equation (34).

$$\begin{aligned} &\Pi = \{r_i | i = 0, 1, .., 4\} \\ &r_0 = R_2((1,1),(3,3)) \quad r_1 = R_2((3,1),(5,3)) \\ &r_2 = R_2((1,3),(5,4)) \quad r_3 = R_2((1,4),(3,6)) \\ &r_4 = R_2((3,4),(5,6)) \end{aligned} \tag{34}$$

It also follows that the observation set $AP$ is equal to the atomic proposition set $AP$ and that the observation map $L$ is described by equation (35).

$$\begin{aligned} L(r_0) &= ap_2 \\ L(r_1) &= ap_3 \\ L(r_2) &= \emptyset \\ L(r_3) &= ap_0 \\ L(r_4) &= ap_1 \end{aligned} \tag{35}$$

Left to define is now $\Sigma$, $\rightarrow$ and $d$. This is done by considering one rectangle at a time and solving the facet reachability problem for each approved facet of that rectangle. Starting with state $r_0 = R_2((1,1),(3,3))$ we must solve the optimization problem of equation (36) for each vertex of the rectangle (i.e. each corner), for each approved facet.

$$\begin{aligned} &\max_{u_v \in U} n_F^\top (x + u_v) \\ &n_{F'}^\top (x + u_v) \leq -\epsilon \quad \epsilon > 0 \end{aligned} \tag{36}$$

Due to the definition of the state space, there are two possible facets which the system is allowed to transition through, $F^*$ and $F^{**}$ which is illustrated in figure 11a. Hence the optimization problem must be solved 8 times. First, considering $F^*$, yields a transition $\delta(r_0, \sigma_0) = r_2$, if the facet reachability problem is solvable. It is simple to see that both condition (18) and (19) are fulfilled for some $u$. Namely, $x + u > 0$ in both direction $x_1$ and $x_2$ for some u, and 0 is not in the convex hull of the rectangle. Now, by solving the optimization problem for each corner of the rectangle one can conclude that $u_2$ must be greater than $-1$ in order to ensure that the system doesn't move in the wrong direction, also $u_1$ must be greater than $-1$ at the facet opposite $F^{**}$ but less than $-3$ along $F^{**}$. One possibility could then be to set $u_2 = u_{max} = 6$ and $u_1 = -x_1$. This would then yield $\sigma_0 = (-x_1, 6)$. Furthermore, solving equations (22) yields $s_{F^*} = 9$ and $s_{\overline{F}^*} = 7$, which when together with equation (21) gives a maximal time of $T^{F^*} = ln(9/7) \approx 0.25$. Hence, $d(\delta(r_0, \sigma_0)) = 0.25$.

Following the same theory, each transition in the direction of $x_1$ or $x_2$ from a rectangle of size $2 \times 2$ will result in the same maximal time ($\approx 0.25$). Furthermore, transitions in the direction of $-x_1$ from a rectangle of size $2 \times 2$, will need $T^F = ln(2) \approx 0.7$ and transitions in direction $-x_2$ from named rectangle will cost $T^F = ln(3) \approx 1.1$. Finally, the transitions out of the rectangle of size $4 \times 1$ will yield a maximal time of $T^F = ln(10/9) \approx 0.1$ in direction $x_2$ and $T^F = ln(4/3) \approx 0.3$ in direction $-x_2$. The final non-deterministic weighted transition system is given in equations (37), (38), (39), (40), (41), (42) and (43).

$$T = (\Pi, \Pi_{init}, \Sigma, \rightarrow, AP, L, d) \tag{37}$$

$$\begin{aligned}\Pi = \quad & \{r_0, r_1, r_2, r_3, r_4\} = \{R_2((1,1),(3,3)), R_2((3,1),(5,3)), \\ & R_2((1,3),(5,4)), R_2((1,4),(3,6)), R_2((3,4),(5,6))\}\end{aligned} \tag{38}$$

$$\Sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \{(-x_1, 6), (6, -x_2), (-6, -x_2), (-x_1, -6)\} \tag{39}$$

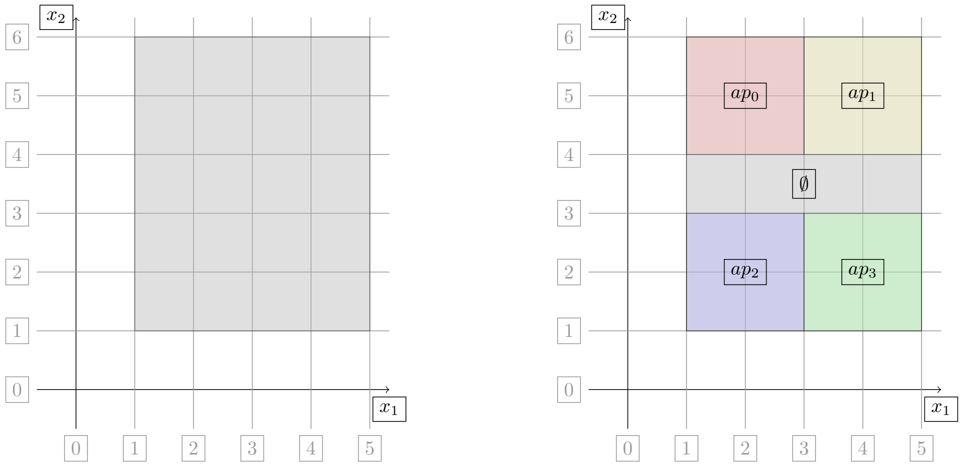
$$\begin{aligned}\delta(r_0, \sigma_0) = r_2 \qquad & \delta(r_0, \sigma_1) = r_1 \\ \delta(r_1, \sigma_0) = r_2 \qquad & \delta(r_1, \sigma_2) = r_0 \\ \delta(r_2, \sigma_0) \in \{r_3, r_4\} \quad & \delta(r_2, \sigma_3) \in \{r_0, r_1\} \\ \delta(r_3, \sigma_1) = r_4 \qquad & \delta(r_3, \sigma_3) = r_2 \\ \delta(r_4, \sigma_2) = r_3 \qquad & \delta(r_4, \sigma_3) = r_2\end{aligned} \tag{40}$$

$$AP = \{ap_0, ap_1, ap_2, ap_3\} \tag{41}$$

$$\begin{aligned}L(r_0) = ap_2 \quad & L(r_1) = ap_3 \\ L(r_2) = \emptyset \quad & L(r_3) = ap_0 \\ L(r_4) = ap_1 \quad &\end{aligned} \tag{42}$$

$$\begin{aligned}d(\delta(r_0, \sigma_0)) \approx 0.25 \quad & d(\delta(r_1, \sigma_0)) \approx 0.25 \\ d(\delta(r_0, \sigma_1)) \approx 0.25 \quad & d(\delta(r_3, \sigma_1)) \approx 0.25 \\ d\delta(r_1, \sigma_2)) \approx 0.7 \quad & d(\delta(r_4, \sigma_2)) \approx 0.7 \\ d(\delta(r_3, \sigma_3)) \approx 1.1 \quad & d(\delta(r_4, \sigma_3)) \approx 1.1 \\ d(\delta(r_2, \sigma_0)) \approx 0.1 \quad & d(\delta(r_2, \sigma_3)) \approx 0.3\end{aligned} \tag{43}$$

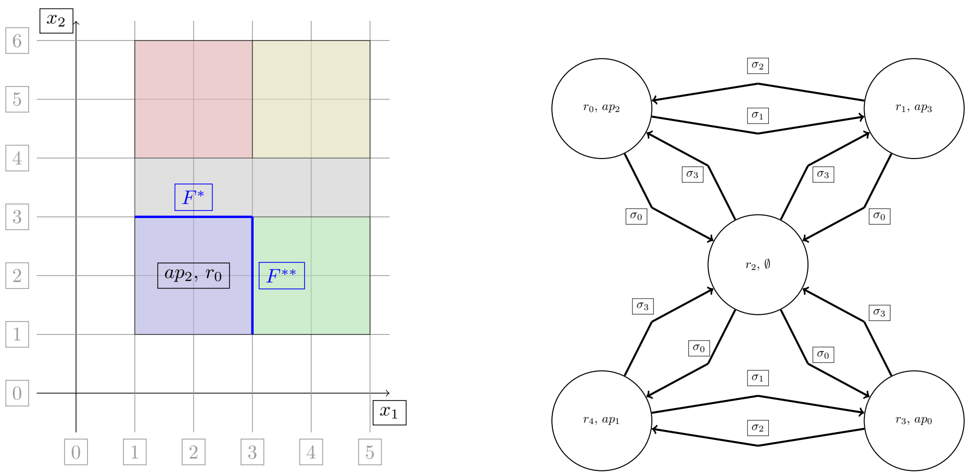
The weighted transition system is illustrated in figure 11b.




(a) The grey area represents the state space of the continuous linear system (32) in Example 4.1.

(b) The figure illustrates the partition of the state space of system (32), done according to equation (17), for Example 4.1.

Figure 10: The state space and rectangular partition of Example 4.1.




(a) The blue marked edges of rectangle $R_2((1,1),(3,3))$ are the facets which the system is allowed to exit through in Example 4.1.

(b) The figure illustrates the weighted transition system (37), which the continuous linear system in Example 4.1 can be abstracted to.

Figure 11: The facets of one of the rectangles of the partitioned system and the final weighted transition system of Example 4.1.

## 4.2 Translation of the MITL Formula to a Timed Büchi Automaton

In this section, the step of translating an MITL formula $\phi$ to a timed Büchi automaton (TBA) is described. Approaches towards translating an MITL formula into a timed automata has been presented in [14], [25], [26], [27] and [28]. The construction described in [26] and [27] regards MTL formulas rather than MITL, however since MITL is a subset of MTL, the method applies here as well. The main result of [25] is the corollary given in Corollary 1. The statement is supported and extended by the results of [27] presented in Corollary 2, as well as the results of [28] presented in Definition 4.2.1. The latter results extends the former by stating complexity of the automata.

**Corollary 1.** *MITL formulas can be transformed into timed automata using a simple procedure.*

**Corollary 2.** *For every MTL formula $\phi$ with $m$ propositions, $n$ unbounded temporal operators and inputs of bounded variability $k$, there exists*

- *...a non-deterministic timed automaton with* $2m\left\lceil \frac{k \cdot fut(\phi)}{2} \right\rceil + 1$ *clocks and* $\left(\left(2\left\lceil \frac{k \cdot fut(\phi)}{2} \right\rceil + 1\right)^m + 1\right)(2 \cdot 4^n + 1)$ *states that accepts the language of $\phi$.*
- *...a deterministic timed automaton with* $2m\left\lceil \frac{k \cdot fut(\phi)}{2} \right\rceil + 1$ *clocks and* $\left(\left(2\left\lceil \frac{k \cdot fut(\phi)}{2} \right\rceil + 1\right)^m + 1\right) \cdot 2^{2^{O(nlogn)}}$ *states that accepts the language of $\phi$.*

*where $fut(\phi)$ is a measurement of the time demanded to check if $\phi$ holds, the semantics are defined as:*

$$\begin{aligned}
fut(\phi) = 0, & \qquad p \textit{ is a proposition.} \\
fut(\phi_1 \vee \phi_2) &= max\left(fut(\phi_1), fut(\phi_2)\right) \\
fut(\neg\phi) &= fut(\phi) \\
fut(\phi_1 \mathcal{U}_I \phi_2) &= \begin{cases} a + 2 + max(fut(\phi_1), fut(\phi_2)), & I = (a, \infty) \\ b + max(fut(\phi_1), fut(\phi_2)), & I = (a, b) \textit{ or } I = [b, b] \end{cases}
\end{aligned}$$

**Definition 4.2.1.** For all MITL formulas $\phi$, $B_\phi$ has $M(\phi)$ clocks and $O((|\phi|)^{(m \cdot |\phi|)})$ locations, where $m = \max\limits_{I \in I_\phi} \left\{2 \times \left\lceil \frac{inf(I)}{|I|} \right\rceil + 1, \left\lceil \frac{sup(I)}{|I|} \right\rceil + 1\right\}$, and $I_\phi$ is the set of time intervals included in $\phi$.

The result of previous work clearly states that all MITL formulas can be translated into timed Büchi automata. Now, for the construction itself. The overall idea is as follows:

1. Define the initial location $s_0$ as the initial copy of the MITL formula: $\phi_{init}$.
2. Consider all possible initial actions which could yield a satisfying run and create one location for each such action. I.e. if the formula is $\phi = a \vee b$, the initial actions which could yield satisfying runs are $a$ and $b$. Create edges and define invariants and clock constraints accordingly.
3. Create a non-accepting state which handle all other possible actions. In the example above this would be $\neg(a \vee b)$.
4. Iterate over step 2 and 3 considering the locations created in step 2 rather than the initial location. When performing step 3 there is no need to create new non-accepting locations, it is enough to create new edges to the already existing non-accepting location. As for step 2, it might not always be a need to create a new location here either, in some cases a better solution is to create a transition back to itself or to another already existing location.
5. Mark the locations at the end of a formula, i.e. the locations which the system will remain in if the formula is satisfied, as accepting.
6. Add transitions to the non-accepting state and the accepting state, handling the time after the MITL formula, i.e. when the time bound has exceeded. These transitions must be constructed such that the suffix of infinite words doesn't affect the acceptance. For example the TBA constructed of the MITL formula $\Box_{\leq b} a$ must not include transitions between accepting and non-accepting states affected by whether $a$ holds for $t > b$. This is generally done by adding transitions from the state to itself for all atomic propositions when $t$ is outside the interval.

7. Define one or two clocks $x \in X$ for each bounded temporal operator in the MITL formula, i.e for each clock constraint. If the interval which is bounding the operator includes 0 or $\infty$, one clock is enough.
8. Define the labelling function in accordance with the created locations.

The statements regarding the creation of new locations in step 4 is of great importance. If the approach is followed without taking this in to consideration, the end result can be an infinitely growing automaton. For example the formula $\Box_{\leq b}\Diamond_{\leq a}\phi$ will have an infinite set of states if a new location was created for each action, while it is sufficient with two locations otherwise. To ensure that the construction is correct one can determine the accepting language of the TBA. If (and only if) the accepting language of the automaton is identical to the accepting language of the MITL formula, the construction is correct. Note that there are multiple automata which corresponds to the same formula.

A simple example of the translation was given already in Example 2.2 in section 2.2, where the TBA constructed of $\phi = \Diamond_{\leq b}a$ was used to illustrate accepting and non-accepting runs and words. Some other examples are presented in Example 4.2

**Example 4.2.** Consider the MITL formulas

$$\phi_1 = \Box_{\leq b}a$$

$$\phi_2 = a\mathcal{U}_{\leq b}c$$

and

$$\phi_3 = \Box_{\leq b}(a \rightarrow \bigcirc c)$$

The formulas can be transformed into timed Büchi automata by following the steps above.

First, consider $\phi_1$. Define the initial state of $A_{\phi_1}$ as the initial copy of $\phi_1$ and name it $s_0$. Now the possible actions which can yield an accepting run is $a$, hence there should be a transition from $s_0$ guarded by $a$. Also, there should be a transition corresponding to the negation of $a$: $\neg a$ to a non-accepting state. We therefore create the preliminary locations $s_1$ and $s_2$, where $s_1$ is the non-accepting state and $s_2$ is the potentially accepting state. Next, we consider the possible actions from $s_2$. Once again, the only possible action is $a$. Hence, it is clear that $s_2$ demands the same edges and guards as $s_0$. We can therefore merge $s_0$ and $s_2$ without changing the acceptance. It is clear, that the same will be true for each iteration as long as the clock constraint $t \leq b$ holds. Hence, a transition from $s_0$ back to itself is defined for $a, t \leq b$, i.e. when $a$ holds and the time constraint is fulfilled. Now, we consider what happens when the time has exceeded $b$. At this point in time, either the system is in location $s_0$ and the formula has been fulfilled, or the system has transitioned to $s_1$. In the former case the transitions of the automaton should be constructed such that all runs remain in an accepting state if it is located in $s_0$ at this point of time. This is defined by creating a transition from $s_0$ to itself for all atomic propositions ($\top$) when time $b$ has exceeded. For the latter case, a run which reaches $s_1$ should never be able to reach an accepting state. Hence, a transition from $s_1$ to itself for all atomic propositions and all time is created. Finally, we can conclude that $s_0$ is the accepting state (as well as the initial state) and there is need of one clock $x$ evaluated over the clock constraint $\leq b$. The finished TBA is illustrated in figure 12.

Following the same procedure for $\phi_2$ and $\phi_3$ yields the result illustrated in figures 13 and 14 respectively. It is clear that the TBA's have the same accepting language as the corresponding MITL formula $\phi_i$.

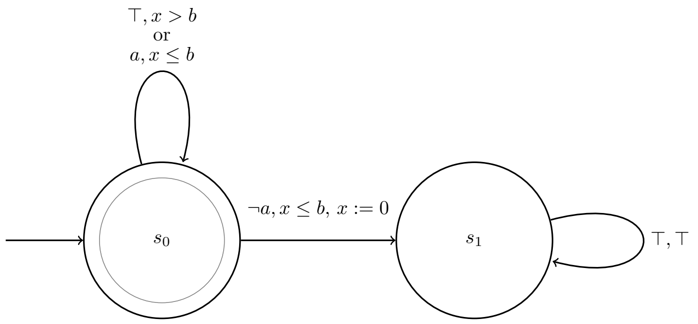



Figure 12: The timed Büchi automaton $A_{\phi_1}$ constructed of the MITL formula $\phi_1 = \Box_{\leq b} a$. The initial and accepting location is $s_0$. A transition from $s_0$ to $s_1$ will occur only if $a$ doesn't hold at some point in the time interval $[0, b]$ which corresponds to $\phi_1$ not being fulfilled.

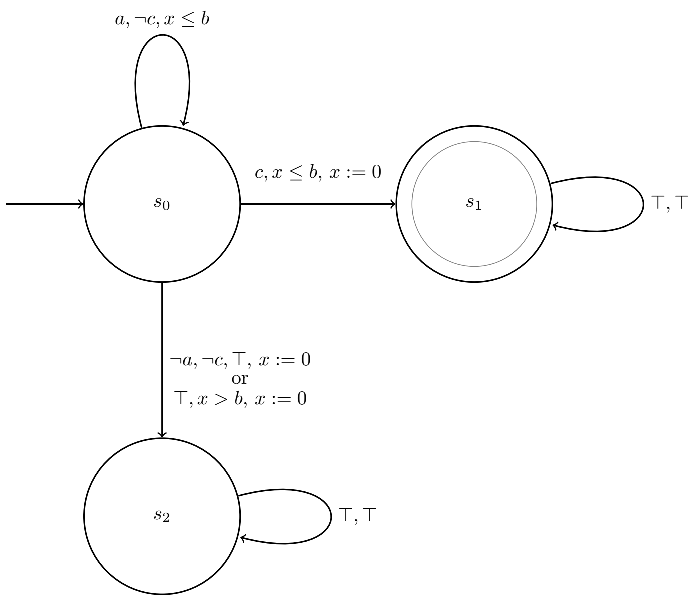



Figure 13: The timed Büchi automaton $A_{\phi_2}$ constructed of the MITL formula $\phi_2 = a\mathcal{U}_{\leq b} c$. The initial location is $s_0$ and the accepting location is $s_1$. A transition $s_0 \to s_1$ will occur if $c$ holds within the time interval, while a transition $s_0 \to s_2$ will occur if either; neither $a$ nor $c$ holds within the time interval or the time interval expires. Hence the TBA is accepting of words with the prefix $a^n c^{m+1}$, where $n$ and $m$ are some non-negative integers. That is, $c$ must hold at some point within the time interval and until it does $a$ must hold. Hence, it has the same accepting language as $\phi_2$

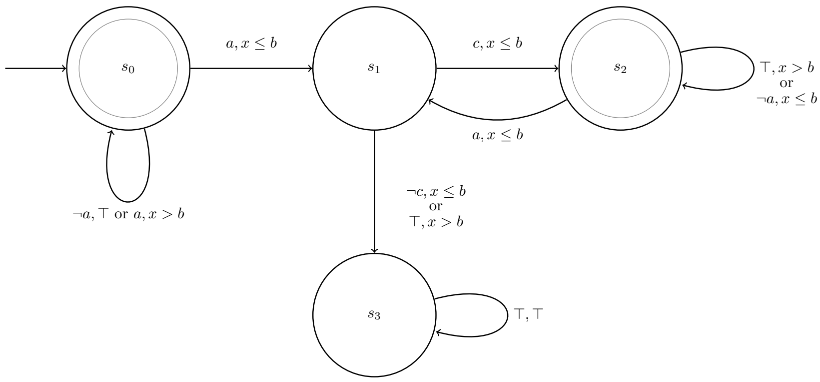



Figure 14: The timed Büchi automaton $A_{\phi_3}$ constructed of the MITL formula $\phi_3$. The initial location is $s_0$ and the accepting locations are $s_0$ and $s_2$. A transition $s_0 \rightarrow s_1$ will occur only if $a$ holds within the time interval. Furthermore, the system can never stay in $s_1$; transition $s_1 \rightarrow s_2$ will occur if $c$ holds and $s_1 \rightarrow s_3$ will occur if $c$ doesn't hold. Finally, the transition $s_2 \rightarrow s_1$ will occur if $a$ holds ones more within the time interval. Hence the accepting language consists of words such that either $a$ never holds within the time interval or $a$ is always followed by $c$. This corresponds to the accepting language of $\phi_3$.

## 4.3 Automata Product

In this section, the construction of the automata product is described. The construction results in a Büchi Weighted Transition System (BWTS), and follows Definition 4.3.1 [13].

**Definition 4.3.1.** Given a weighted transition system $T = (\Pi, \Pi_{init}, \Sigma, \to, AP, L, d)$ and a timed Büchi automaton $A = (S, S_{init}, X, I, E, F, AP, \mathcal{L})$ with $M = |X|$ and $c_{max}$ as the largest constant in $A$, their BWTS is defined as $T^p = T \otimes A = (Q, Q_{init}, \rightsquigarrow, d_p, F_p, AP, L_p)$ with:

- $Q \subseteq \{(r, s) \in \Pi \times S : L(r) = \mathcal{L}(s) \times \mathbb{T}_{\infty}^{M}$,
- $Q_{init} = \Pi_{init} \times S_{init} \times \{0\} \times ... \times \{0\}$, where $Q_{init} \subseteq Q$ and $\{0\} \times ... \times \{0\}$ consists of $M$ factors, i.e. there is one factor $\{0\}$ for each clock in $A$,
- $q \rightsquigarrow q'$ iff
  - $q = (r, s, v_1, ..., v_M) \in Q$, and $q' = (r', s', v_1', ..., v_M')$ where $v_i$ and $v_i'$ are clock valuations (see Definition 2.2.8),
  - $r \to r'$ and
  - $\exists\, \gamma, R$, s.t.
    - $(s, \gamma, R, s') \in E$,
    - $v_1, ..., v_M \models \gamma$,
    - $v_1', ..., v_M' \models I(s')$ and
    - $$v_i' = \begin{cases} 0, & \text{if } x_i \in R \\ v_i + d(r, r'), & \text{if } x_i \notin R \text{ and} \\ & v_i + d(r, r') \leq c_{max} \\ \infty, & \text{otherwise} \end{cases}$$
- $d_p(q, q') = d(r, r')$ if $q \rightsquigarrow q'$,
- $F_p = \{(r, s, v_1, ..., v_M) \in Q : s \in F\}$ and
- $L_p(r, s, v_1, ..., v_M) = L(r)$

A simple example is given in Example 4.3.

**Example 4.3.** Consider a continuous linear system $\dot{x} = x + u$ of two dimensions evolving in the state space $x \in \{(1, 1)(2, 3)\}$ from the initial position $x_0$, with the control input $u$ limited by $U = \{(-4, -4), (4, 4)\}$. Furthermore, the system should satisfy the MITL formula $\phi = \Diamond_{\leq a} b$ over the atomic proposition set $AP = \{b\}$, where $b$ holds for $x_2 > 2$. Finally, $x_0$ is such that $x_2 < 2$. By following the steps presented in section 4.1 the system can be abstracted to the weighted transition system

$$T = (\Pi, \Pi_{init}, \Sigma, \to, AP, L, d)$$

where

- $\Pi = \{r_0, r_1\} = \{R_2((1, 1), (2, 2)), R_2((1, 2), (2, 3)),$
- $\Pi_{init} = r_0,$
- $AP = \{b\},$
- $\to = \{(r_i, r_i), (r_0, r_1), (r_1, r_0)\} = \{\sigma_0, \sigma_1, \sigma_2\},$
- $d(r_0, r_1) = ln(6/5)$, $d(r_1, r_0) = ln2$ and $d(r_i, r_i) = 0$ and
- $L(r_0) = \emptyset$ and $L(r_1) = b$

The resulting WTS is illustrated in figure 15. Furthermore, in accordance with section 4.2, $\phi$ can be translated into the timed Büchi automaton

$$A = (S, S_{init}, X, I, E, F, AP, \mathcal{L})$$

where

- $S = \{s_0, s_1, s_2\},$
- $S_{init} = \{s_0\},$
- $X = \{x\},$

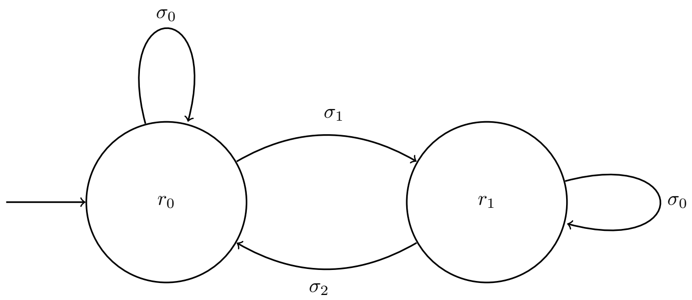



Figure 15: Weighted transition system abstracted from the continuous linear system of Example 4.3

- $I(s_0) : x \leq a$,
- $E = \{(s_0, x \leq a, x := 0, s_1), (s_0, x \leq a, \emptyset, s_0), (s_0, x > a, x := 0, s_2), (s_1, \top, \emptyset, s_1), (s_2, \top, \emptyset, s_2)\}$,
- $F = \{s_1\}$,
- $AP = \{b\}$ and
- $\mathcal{L}(s_1) = \mathcal{L}(s_2) = \top$ and $\mathcal{L}(s_0) = \emptyset$

The resulting TBA is illustrated in figure 16. Now, the automata product as defined in Definition 4.3.1 yields that the system can be expressed as the BWTS

$$T^p = (Q, Q_{init}, \rightsquigarrow, d_p, F_p, AP, L_p)$$

where

- $Q = \{q_0, q_1, q_2, q_3, q_4\} = \{(s_0, r_0), (s_1, r_1), (s_2, r_0), (s_1, r_0), (s_2, r_1)\}$,
- $Q_{init} = (q_0, 0) = (r_0, s_0, 0)$,
-

$$\begin{array}{lcl}
q_0 \rightsquigarrow q_1 & d_p(q_0, q_1) = ln(6/5) & (v' = 0) \\
q_0 \rightsquigarrow q_2 & d_p(q_0, q_2) = 0 & (v' = 0) \\
q_0 \rightsquigarrow q_4 & d_p(q_0, q_4) = ln(6/5) & (v' = 0) \\
q_2 \rightsquigarrow q_4 & d_p(q_2, q_4) = ln(6/5) & (v' = v + d) \\
q_4 \rightsquigarrow q_2 & d_p(q_4, q_2) = ln(2) & (v' = v + d) \\
q_1 \rightsquigarrow q_3 & d_p(q_1, q_3) = ln(2) & (v' = v + d) \\
q_3 \rightsquigarrow q_1 & d_p(q_3, q_1) = ln(6/5) & (v' = v + d)
\end{array}$$

- $F_p = \{(q_1, 0), (q_3, t)\}$, for all $t$, and
- $L_p(q_0) = L_p(q_2) = L_p(q_3) = \emptyset$ and $L_p(q_1) = L_p(q_4) = b$

The result is illustrated in figure 17.

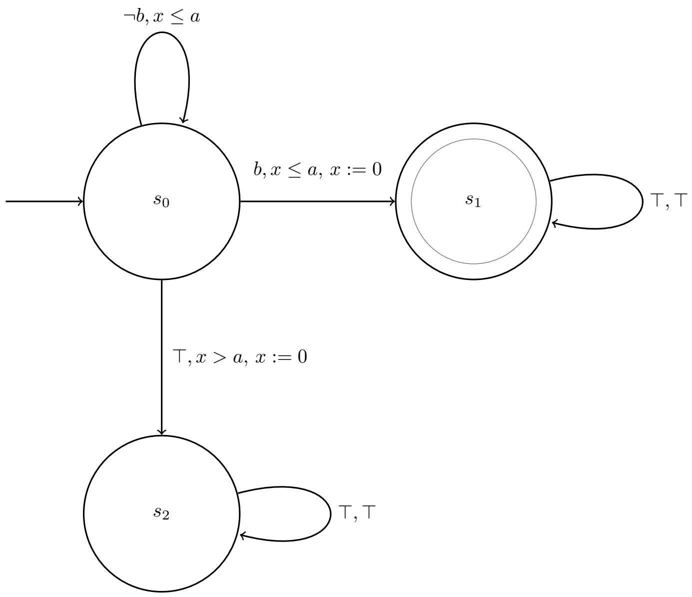



Figure 16: Timed Büchi automaton which has the same accepted language as the runs which satisfies $\phi = \Diamond_{\leq a} b$.

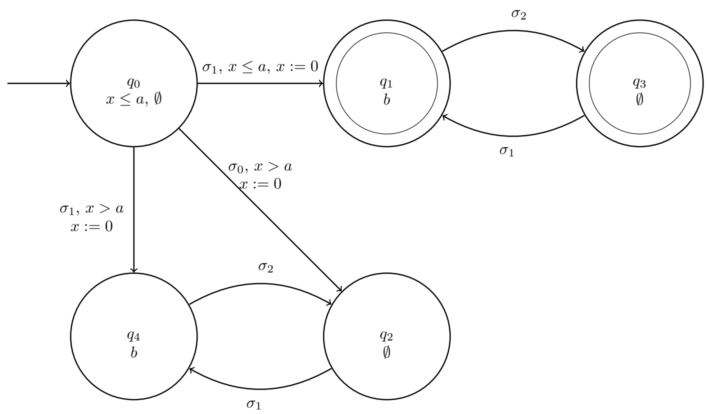



Figure 17: Resulting BWTS of Example 4.3, i.e. the product of the WTS in figure 15 and the TBA in figure 16.

## 4.4 Control Design

This section describes how to design the controller based on the Büchi weighted transition system which was constructed in the previous section (section 4.3).

The control design is fairly straight forward. As stated in section 4.2 the TBA constructed of an MITL formula has the same accepting language as the formula. Furthermore, the WTS abstracted from a continuous linear system has the same evolution as the system itself. Now, the automata product of the TBA and the WTS has the same evolution as the WTS, while having the same accepting language as the TBA. The result is hence a transition system for which all accepting runs, correspond to the runs in the original system that satisfies the MITL formula. Hence, the controller can be designed by finding an accepting run of the BWTS.

The accepting run of the BWTS is found by graph search algorithms such as Depth-First Search (DFS) [29] or Dijkstra's algorithm [30]. The DFS algorithm searches for a value in a graph by exploring each path as far as possible before backtracking, starting at the root. Dijkstra adds all graph-nodes which are successors of the initial node to a search-pool and then starts by determining if the node closest to the initial node is accepting, if not it continues with the node which is second in closeness, and so on. Along the way Dijsktra checks and updates the total node-distances if closer paths are found and adds the successors of each tested node to the search-pool. Since the BWTS can be viewed as a graph both the DFS and Dijkstra's algorithm can be directly implemented to search for an accepting run. When an accepting run is found the algorithm can be cancelled since there is no need to find more than one accepting run.

**Example 4.4.** Considering Example 4.3 in section 4.3, an example of an accepting run is:

$$r = q_0 \overset{\sigma_1, ln(6/5)}{\longrightarrow} q_1$$

if $ln(6/5) \leq a$. If $ln(6/5) > a$, there is no accepting run. Assuming $ln(6/5) < a$, the control input needed for the transition is $u = [-x_1, 4]$, which is the control input calculated during the abstraction. That is, applying $u$ guarantees that the transition $q_0 \to q_1$ occurs within $ln(6/5)$time units. Hence, we can conclude that the closed-loop system in Example 4.3 will satisfy $\phi = \Diamond_{\leq a} b$ for all $a \geq ln(6/5)$ for $u = [-x_1, 4]$.

# 5 Implementation 1

In this section, an example is presented covering all the steps of the solution. The example has been simulated in MATLAB implementing the method presented in section 4.

Consider the example first discussed in section 1, a robot moving through 6 rooms and a corridor. Let the motion of the robot follow the system equation (44), where $u$ is defined according to equation (45) and $x$ is bounded in accordance with equation (46). Also, assume that the rooms have walls between them, only allowing the robot to change room by going through the corridor.

$$\dot{x} = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} x + u \tag{44}$$

$$u = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} x + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \in U = [-20, 20] \tag{45}$$

$$x \in \{(1,1), (5.5, 4)\} \tag{46}$$

Furthermore, let $AP = \{r_1, r_2, r_3, r_4, r_5, r_6, c\}$ be the set of atomic propositions which is considered, where $r_i$ holds in room $i$ and $c$ holds in the corridor. Now, design the control input $u$ such that the closed-loop system satisfies the MITL formula $\phi = \Diamond_{\leq a_1} r_2 \wedge (r_2 \to \Diamond_{\leq a_2} r_6)$ (*"Reach room 2 within $a_1$ time units and; if room 2 is entered, always go to room 6 within $a_2$ time units."*).

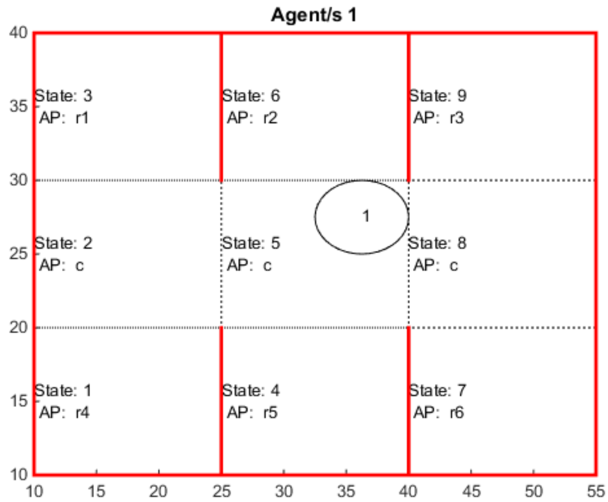



Figure 18: Partition constructed by the MATLAB scripts with the settings of Problem 1 as defined in section 5. The circle with the 1 inside represents the initial state.

## 5.1 Constructing the WTS

First, let's construct a weighted transition system (WTS) of the system starting with partitioning the state space. The information above yields a partition of the system consistent with the figure of the rooms presented in figure 1 in section 1. The partition is defined as equation (47).

$$\begin{aligned}
r_1 &= R_2((1,3)(2.5,4)) \quad r_2 = R_2((2.5,3)(4,4)) \quad r_3 = R_2((4,3)(5.5,4)) \\
r_4 &= R_2((1,1)(2.5,2)) \quad r_5 = R_2((2.5,1)(4,2)) \quad r_6 = R_2((4,1)(5.5,2)) \\
c &= R_2((1,2)(5.5,3))
\end{aligned} \tag{47}$$

The partition will (as in Example 4.1) yield a non-deterministic WTS. To avoid this the partition of $c$ can be further refined. This can be done by dividing $c$ into three sub-rectangles $c_1$, $c_2$ and $c_3$ which all have the same width as the rooms. Hence, $c_i$ is defined as equation (48).

$$c_1 = R_2((1,2)(2.5,3)) \quad c_2 = R_2((2.5,2)(4,3)) \quad c_3 = R_2((4,2)(5.5,3)) \tag{48}$$

Implementing the MATLAB script containing the construction-steps of section 4.1, the same result is achieved (however with another state numbering), which is illustrated in figure 18.

By following the linear abstraction presented in section 4.1, we conclude that the system can be written as:

$$\dot{x} = A^*x + B = \begin{bmatrix} 2 + a_{11} & 1 + a_{12} \\ a_{21} & 2 + a_{22} \end{bmatrix} x + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

To follow the suggested solution the non-diagonal elements of $A^*$ must be zero in all directions $e_j$ which is in the norm-direction of the facet. I.e. for transitions in the direction of $x_2$, $a_{21}$ must be 0 and for transitions in the direction of $x_1$, $a_{12} = -1$.

Now, we must solve the optimization problem

$$\max_{u \in U}(n_F^\top(A^*x + B))$$
$$n_{F'}^\top(A^*x + B) < -\epsilon, F' \in F * \backslash F$$

where $F*$ is the set of all facets of a rectangle.

We start with room 1. The only facet which the robot can exit through is the one shared with $c_1$, which is the only edge of the rectangle that isn't closed off by a wall. Since the direction of the transition is $-e_2$, $a_{21} = 0$ for $C_2^*$ to be a constant. Hence, the problem becomes:

$$(2 + a_{11})x_1 + (1 + a_{12})x_2 + b_1 > \epsilon, \quad x_1 = 1$$
$$(2 + a_{11})x_1 + (1 + a_{12})x_2 + b_1 < -\epsilon, \quad x_1 = 2.5$$
$$\max_{u \in U}(-((2 + a_{22})x_2 + b_2))$$

The result from the MATLAB script gave the following solution. One solution for the first two equations is $a_{11} = -5.4699$, $a_{12} = -1$ and $b_1 = 8.1748$. Which in turn yields $B_1 u_1 = u_1 = -5.4699x_1 - x_2 + 8.1748$, (since $B = I$). Now, the third equation is maximized throughout the rectangle if all the remaining control input is used at all time, i.e. if $a_{22} = 0$ and $b_2 = -u_{max,left}$, where $u_{max,left}$ is what is left to use of the control input. With $u_1$ as above and the limit being $\sqrt{u_1^2 + u_2^2} \leq 20^2$, it is simple to calculate that $u_{max,left} = -19.2289$. Hence, the resulting control signal for the transition is:

$$\overline{u_1} = \begin{bmatrix} -5.4699 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 8.1748 \\ -19.2289 \end{bmatrix}$$

Note, that this is only the optimal solution when the assumption that $C^*$ can be treated as a constant is made.

Now, $C_2^* = -19.2289$ and $a_{22}^* = 2$. From here it follows that $s_F = -a_{22}^* \cdot x2|_{x_2 \in F} - C_2^* = -2 \cdot 3 - (-19.2289) = 13.2289$ and $s_{\overline{F}} = -a_{22}^* \cdot x2|_{x_2 \in \overline{F}} - C_2^* = -2 \cdot 4 - (-19.2289) = 11.2289$ and finally the maximal time the transition will take is given by:

$$T_1^F = ln(\frac{s_F}{s_{\overline{F}}})\frac{1}{a_{22}^*} = ln(\frac{13.2289}{11.2289})\frac{1}{2} = 0.082$$

Following the same steps for room 2 and room 3 yields:

$$\overline{u_2} = \begin{bmatrix} -6.4314 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 17.2258 \\ -18.1039 \end{bmatrix} \quad T_2^F = 0.0903 \quad \text{From room 2}$$
$$\overline{u_3} = \begin{bmatrix} -3.6181 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 8.3993 \\ -16.3631 \end{bmatrix} \quad T_3^F = 0.1072 \quad \text{From room 3}$$

For room 4, room 5 and room 6 the direction of the transitions change to $e_2$, but otherwise the steps are the same. The difference in the calculation is hence choosing $b_2$ as positive instead of negative, due to the maximization problem changing sign. The result of the calculations are:

$$\overline{u_4} = \begin{bmatrix} -5.4957 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 8.2392 \\ 19.2289 \end{bmatrix} \quad T_4^F = 0.0450 \quad \text{From room 4}$$
$$\overline{u_5} = \begin{bmatrix} -7.1724 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 20.1895 \\ 18.1039 \end{bmatrix} \quad T_5^F = 0.0474 \quad \text{From room 5}$$
$$\overline{u_6} = \begin{bmatrix} -5.4399 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 18.4196 \\ 16.3631 \end{bmatrix} \quad T_6^F = 0.0517 \quad \text{From room 6}$$

For the corridor 6 external transitions towards the rooms and 4 internal transitions between $c_i$ are considered. The transitions towards the room follows the same calculations as before yielding:

$$\overline{u_7} = \begin{bmatrix} -5.4701 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 8.1752 \\ 19.2289 \end{bmatrix} \quad T_7^F = 0.0413 \quad \text{To room 1}$$

$$\overline{u_8} = \begin{bmatrix} -5.4686 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 8.1716 \\ -19.2289 \end{bmatrix} \quad T_8^F = 0.0704 \quad \text{To room 4}$$

$$\overline{u_9} = \begin{bmatrix} -6.3982 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 17.0928 \\ 18.1039 \end{bmatrix} \quad T_9^F = 0.0433 \quad \text{To room 2}$$

$$\overline{u_{10}} = \begin{bmatrix} -6.2329 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 16.4314 \\ -18.1039 \end{bmatrix} \quad T_{10}^F = 0.0765 \quad \text{To room 5}$$

$$\overline{u_{11}} = \begin{bmatrix} -8.0926 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 33.0091 \\ 16.3631 \end{bmatrix} \quad T_{11}^F = 0.0468 \quad \text{To room 3}$$

$$\overline{u_{12}} = \begin{bmatrix} -10.5954 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 46.7750 \\ -16.3631 \end{bmatrix} \quad T_{12}^F = 0.0882 \quad \text{To room 6}$$

Finally, the transitions within the corridor is calculated by maximizing in direction $\pm e_1$ and choosing $u_2$ s.t. there can't be transitions towards the rooms. Starting with the transition from $c_1$ to $c_2$, the problem becomes (here we assume $a_{12} = -1$):

$$\begin{aligned} (2 + a_{22})3 + a_{21}x_1 + b_2 &< -\epsilon \\ (2 + a_{22})2 + a_{21}x_1 + b_2 &> \epsilon \\ \underset{u \in U}{max}((2 + a_11)x_1 &+ b_1) \end{aligned}$$

Implementing the problem in MATLAB yielded that $a_{22} = -9.9894$ and $b_2 = 23.4681$ is a solution. With the same argument as before we choose $b_1 = 18.9143$ which results in:

$$\overline{u_{13}} = \begin{bmatrix} 0 & -1 \\ 0 & -9.9894 \end{bmatrix} x + \begin{bmatrix} 18.9143 \\ 23.4681 \end{bmatrix} \quad T_{13}^F = 0.0670 \quad c_1 \text{ to } c_2$$

The remaining controllers and time limits are calculated following the same steps resulting in:

$$\overline{u_{14}} = \begin{bmatrix} 0 & -1 \\ 0 & -9.4712 \end{bmatrix} x + \begin{bmatrix} 18.9143 \\ 21.9137 \end{bmatrix} \quad T_{14}^F = 0.0591 \quad c_2 \text{ to } c_3$$

$$\overline{u_{15}} = \begin{bmatrix} 0 & -1 \\ 0 & -7.6498 \end{bmatrix} x + \begin{bmatrix} -18.9143 \\ 16.4494 \end{bmatrix} \quad T_{15}^F = 0.1214 \quad c_2 \text{ to } c_1$$

$$\overline{u_{16}} = \begin{bmatrix} 0 & -1 \\ 0 & -8.2872 \end{bmatrix} x + \begin{bmatrix} -18.9143 \\ 18.3616 \end{bmatrix} \quad T_{16}^F = 0.1607 \quad c_3 \text{ to } c_2$$

The abstracted WTS corresponding to the system can therefore be defined as:

$$\begin{aligned} T &= (\Pi, \Pi_{init}, \rightarrow, \Sigma, AP, L, d) \\ \Pi &= \{r_1, r_2, r_3, r_4, r_5, r_6, c_1, c_2, c_3\} \\ \Pi_{init} &= \{c_1, c_2, c_3\} \\ AP &= \{r_1, r_2, r_3, r_4, r_5, r_6, c\} \end{aligned} \tag{49}$$

where the transitions are:

$$\begin{aligned} \rightarrow \quad = \{&(r_1, \overline{u_1}, c_1), (r_4, \overline{u_4}, c_1), (c_1, \overline{u_7}, r_1), (c_1, \overline{u_8}, r_4), (c_1, \overline{u_{13}}, c_2), (c_2, \overline{u_9}, r_2), (r_2, \overline{u_2}, c_2), (c_2, \overline{u_{10}}, r_5), \\ &(r_5, \overline{u_5}, c_2), (c_2, \overline{u_{14}}, c_3), (c_3, \overline{u_{11}}, r_3), (r_3, \overline{u_3}, c_3), (c_3, \overline{u_{12}}, r_6), (r_6, \overline{u_6}, c_3), (c_3, \overline{u_{16}}, c_2), (c_2, \overline{u_{15}}, c_1)\} \end{aligned}$$

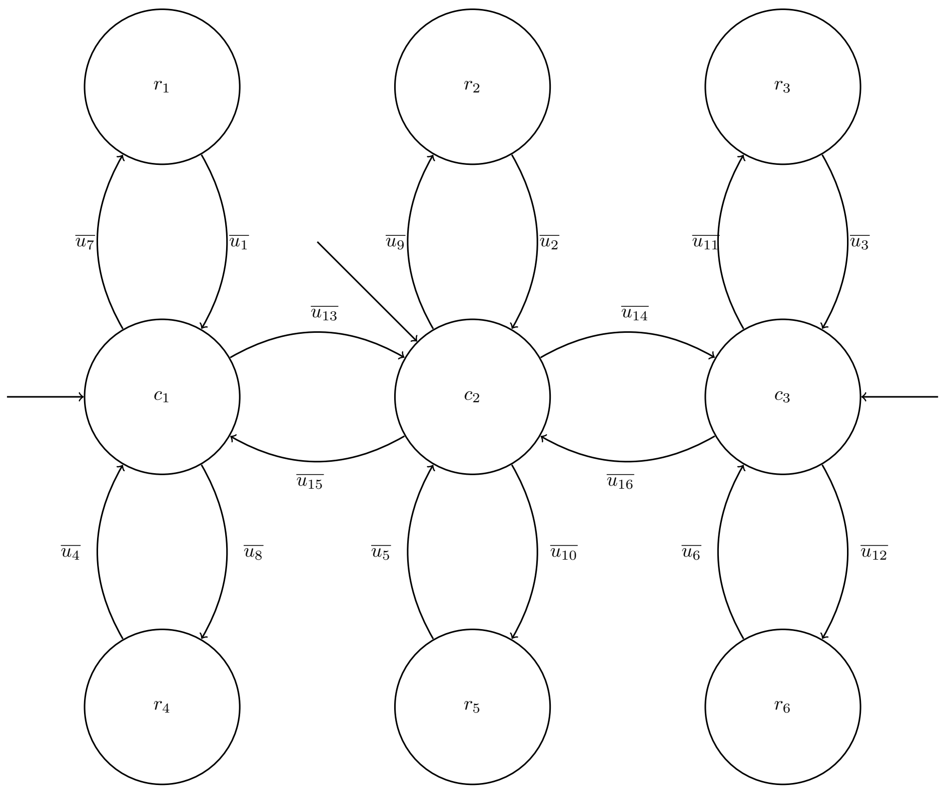



Figure 19: Weighted transition system constructed from the continuous linear system (44).

The weights $d$ for respective transition are $T_i^F$, where $i$ correspond to the index of the control signal $\sigma_i = u_i$ which is applied to induce the transition. Finally, the labelling function $L$ is defined as:

$$L(r_i) = r_i, \quad \forall i \in \{1, 2, ..., 6\}$$
$$L(c_i) = c, \quad \forall i \in \{1, 2, 3\}$$

The resulting WTS is illustrated in figure 19.

## 5.2 Constructing the TBA

Now, let's construct a timed Büchi automaton from the MITL formula.

$$\phi = \Diamond_{\leq a_1} r_2 \wedge r_2 \rightarrow \Diamond_{\leq a_2} r_6 \tag{50}$$

Since the construction of a TBA from an MITL formula presented in section 4.2 only consists of guidelines, rather than a detailed method, this step cannot be performed in MATLAB. The implementation has instead been performed by constructing the TBA manually and defining the already constructed TBA as input for the following steps.

We start the construction of the TBA by defining the initial location $s_0$ as the initial copy of the formula $\phi_{init}$. Now, let's consider time less than $a_1$. Either the robot reaches room 2, which would satisfy the first part of the formula, or there still exist the possibility that it will do so within the time limit. We therefore define location $s_1 = \Diamond_{\leq a_1} r_2$. Furthermore, we define an edge from $s_0$ to itself for all positions the robot can have that is not room 2 within the time interval. Now, let's

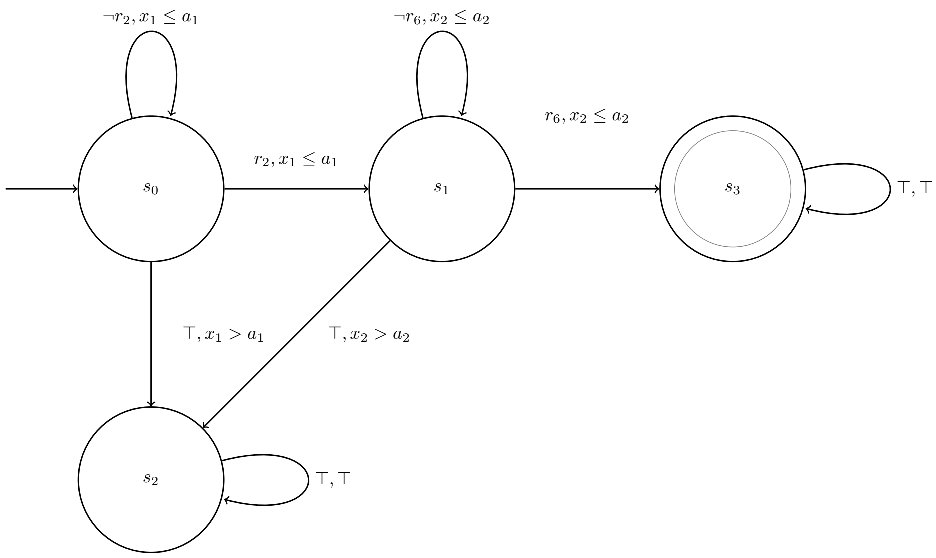



Figure 20: Timed Büchi automaton constructed from the MITL formula $\phi = \Diamond_{\leq a_1} r_2 \wedge (r_2 \rightarrow \Diamond_{\leq a_2} r_6)$.

consider time greater than $a_1$. If the system is still in location $s_0$, i.e. if the robot hasn't reached room 2 yet, the formula can no longer be satisfied. We therefore define $s_2 = \neg\Diamond_{\leq a_1} r_2$. Now, consider $s_1$. From here, the robot should reach room 6 within $a_2$ time units. Similarly to the first steps, we define $s_3 = \Diamond_{\leq a_2} r_6$ and we define an edge from $s_1$ to $s_2$ for $x_2 > a_2$ as well as an edge from $s_1$ to itself for $\neg r_6$ and $x_2 \leq a_2$. Now, we mark $s_0$ as initial and $s_3$ as accepting. Edges from $s_3$ to itself as well as from $s_2$ to itself are defined for all atomic propositions and all time. The result is the TBA:

$$\begin{aligned} A &= (S, S_{init}, X, I, E, F, AP, \mathcal{L}) && \\ S &= \{s_0, s_1, s_2, s_3\} && \\ S_{init} &= \{s_0\} & X &= \{x_1, x_2\} \\ I : I(s_0) &: x_1 \leq a_1, & I(s_1) &: x_2 \leq a_2 \\ F &= \{s_3\} & AP &= \{r_1, r_2, r_3, r_4, r_5, r_6, c\} \\ \mathcal{L}(s_0) &= AP \setminus \{r_2\} & \mathcal{L}(s_1) &= AP \setminus \{r_6\} \\ \mathcal{L}(s_2) = \mathcal{L}(s_3) &= AP && \end{aligned} \tag{51}$$

where the edges are:

$$\begin{aligned} E = \quad & \{(s_0, x_1 \leq a_1, s_0), (s_0, x_1 \leq a_1, s_1), (s_0, x_1 > a_1, s_2), (s_1, x_2 \leq a_2, s_1), \\ & (s_1, x_2 \leq a_2, s_3), (s_1, x_2 > a_2, s_2), (s_2, \top, s_2), (s_3, \top, s_3)\} \end{aligned}$$

The resulting TBA is illustrated in figure 20.

## 5.3 Constructing the BWTS

The resulting BWTS constructed as the automata product of the WTS and the TBA, has 34 states $q = (r, s)$. Each state is a pair of a transition state $r$ from the WTS and a location $s$ from the TBA. The maximal number of states which the given systems could have resulted in is $9 \times 4 = 36$ which is the number of possible combinations. In this case however, there are fewer due to the labelling functions. The combination which are of interest are those which share the same label. Therefore,

$(r_2, s_0)$ and $(r_6, s_1)$ are invalid combinations. The states of the BWTS are:

$$\begin{aligned} Q = \quad & \{(c_1, s_0), (c_2, s_0), (c_3, s_0), (r_1, s_0), (r_3, s_0), \\ & (r_4, s_0), (r_5, s_0), (r_6, s_0), (c_1, s_1), (c_2, s_1), \\ & (c_3, s_1), (r_1, s_1), (r_2, s_1), (r_3, s_1), (r_4, s_1), \\ & (r_5, s_1), (c_1, s_2), (c_2, s_2), (c_3, s_2), (r_1, s_2), \\ & (r_2, s_2), (r_3, s_2), (r_4, s_2), (r_5, s_2), (r_6, s_2), \\ & (c_1, s_3), (c_2, s_3), (c_3, s_3), (r_1, s_3), (r_2, s_3), \\ & (r_3, s_3), (r_4, s_3), (r_5, s_3), (r_6, s_3) \end{aligned}$$

Among these, the initial states are:

$$Q_{init} = \{(c_1, s_0, 0, 0), (c_2, s_0, 0, 0), (c_3, s_0, 0, 0)\}$$

The transition map $\rightsquigarrow$ consists of transitions between all states such that $r$ (or $c$) follows the transition map in the WTS and $s$ follows the transition map of the TBA. The result is: Transitions from states $(r_i, s_0)$ and $(c_i, s_0)$:

$$\begin{array}{lll} (c_1, s_0) \rightsquigarrow (r_1, s_0) & (c_1, s_0) \rightsquigarrow (r_4, s_0) & (c_1, s_0) \rightsquigarrow (c_2, s_0) \\ (c_1, s_0) \rightsquigarrow (r_1, s_2) & (c_1, s_0) \rightsquigarrow (r_4, s_2) & (c_1, s_0) \rightsquigarrow (c_2, s_2) \\ (c_1, s_0) \rightsquigarrow (c_1, s_2) & & \end{array}$$

$$\begin{array}{lll} (c_2, s_0) \rightsquigarrow (r_2, s_1) & (c_2, s_0) \rightsquigarrow (r_5, s_0) & (c_2, s_0) \rightsquigarrow (c_3, s_0) \\ (c_2, s_0) \rightsquigarrow (c_1, s_0) & (c_2, s_0) \rightsquigarrow (c_2, s_2) & (c_2, s_0) \rightsquigarrow (r_2, s_2) \\ (c_2, s_0) \rightsquigarrow (r_5, s_2) & (c_2, s_0) \rightsquigarrow (c_3, s_2) & (c_2, s_0) \rightsquigarrow (c_1, s_2) \end{array}$$

$$\begin{array}{lll} (c_3, s_0) \rightsquigarrow (r_3, s_0) & (c_3, s_0) \rightsquigarrow (r_6, s_0) & (c_3, s_0) \rightsquigarrow (c_2, s_0) \\ (c_3, s_0) \rightsquigarrow (r_3, s_2) & (c_3, s_0) \rightsquigarrow (r_6, s_2) & (c_3, s_0) \rightsquigarrow (c_2, s_2) \\ (c_3, s_0) \rightsquigarrow (c_3, s_2) & & \end{array}$$

$$\begin{array}{lll} (r_1, s_0) \rightsquigarrow (c_1, s_0) & (r_1, s_0) \rightsquigarrow (c_1, s_2) & (r_1, s_0) \rightsquigarrow (r_1, s_2) \end{array}$$

$$\begin{array}{lll} (r_3, s_0) \rightsquigarrow (c_3, s_0) & (r_3, s_0) \rightsquigarrow (c_3, s_2) & (r_3, s_0) \rightsquigarrow (r_3, s_2) \end{array}$$

$$\begin{array}{lll} (r_4, s_0) \rightsquigarrow (c_1, s_0) & (r_4, s_0) \rightsquigarrow (c_1, s_2) & (r_4, s_0) \rightsquigarrow (r_4, s_2) \end{array}$$

$$\begin{array}{lll} (r_5, s_0) \rightsquigarrow (c_2, s_0) & (r_5, s_0) \rightsquigarrow (c_2, s_2) & (r_5, s_0) \rightsquigarrow (r_5, s_2) \end{array}$$

$$\begin{array}{lll} (r_6, s_0) \rightsquigarrow (c_3, s_0) & (r_6, s_0) \rightsquigarrow (c_3, s_2) & (r_6, s_0) \rightsquigarrow (r_6, s_2) \end{array}$$

Transitions from states $(r_i, s_1)$ and $(c_i, s_1)$:

$$\begin{array}{lll} (c_1,s_1) \rightsquigarrow (r_1,s_1) & (c_1,s_1) \rightsquigarrow (r_4,s_1) & (c_1,s_1) \rightsquigarrow (c_2,s_1) \\ (c_1,s_1) \rightsquigarrow (r_1,s_2) & (c_1,s_1) \rightsquigarrow (r_4,s_2) & (c_1,s_1) \rightsquigarrow (c_2,s_2) \\ (c_1,s_1) \rightsquigarrow (c_1,s_2) & & \end{array}$$

$$\begin{array}{lll} (c_2,s_1) \rightsquigarrow (r_2,s_1) & (c_2,s_1) \rightsquigarrow (r_5,s_1) & (c_2,s_1) \rightsquigarrow (c_1,s_1) \\ (c_2,s_1) \rightsquigarrow (c_3,s_1) & (c_2,s_1) \rightsquigarrow (r_2,s_2) & (c_2,s_1) \rightsquigarrow (r_5,s_2) \\ (c_2,s_1) \rightsquigarrow (c_1,s_2) & (c_2,s_1) \rightsquigarrow (c_3,s_2) & (c_2,s_1) \rightsquigarrow (c_2,s_2) \end{array}$$

$$\begin{array}{lll} (c_3,s_1) \rightsquigarrow (r_3,s_1) & (c_3,s_1) \rightsquigarrow (r_6,s_3) & (c_3,s_1) \rightsquigarrow (c_2,s_1) \\ (c_3,s_1) \rightsquigarrow (r_3,s_2) & (c_3,s_1) \rightsquigarrow (c_2,s_2) & (c_3,s_1) \rightsquigarrow (c_3,s_2) \\ (c_3,s_1) \rightsquigarrow (r_6,s_2) & & \end{array}$$

$$\begin{array}{lll} (r_1,s_1) \rightsquigarrow (c_1,s_1) & (r_1,s_1) \rightsquigarrow (r_1,s_2) & (r_1,s_1) \rightsquigarrow (c_1,s_2) \end{array}$$

$$\begin{array}{lll} (r_2,s_1) \rightsquigarrow (c_2,s_1) & (r_2,s_1) \rightsquigarrow (c_2,s_2) & (r_2,s_1) \rightsquigarrow (r_2,s_2) \end{array}$$

$$\begin{array}{lll} (r_3,s_1) \rightsquigarrow (c_3,s_1) & (r_3,s_1) \rightsquigarrow (c_3,s_2) & (r_1,s_1) \rightsquigarrow (r_1,s_2) \end{array}$$

$$\begin{array}{lll} (r_4,s_1) \rightsquigarrow (c_1,s_1) & (r_4,s_1) \rightsquigarrow (c_1,s_2) & (r_4,s_1) \rightsquigarrow (r_4,s_2) \end{array}$$

$$\begin{array}{lll} (r_5,s_1) \rightsquigarrow (c_2,s_1) & (r_5,s_1) \rightsquigarrow (c_2,s_2) & (r_5,s_1) \rightsquigarrow (r_5,s_2) \end{array}$$

Transitions from states $(r_i, s_2)$ and $(c_i, s_2)$:

$$\begin{array}{ll} (c_1,s_2) \rightsquigarrow (r_1,s_2) & (c_1,s_2) \rightsquigarrow (r_4,s_2) \end{array}$$

$$\begin{array}{ll} (c_2,s_2) \rightsquigarrow (r_2,s_2) & (c_2,s_2) \rightsquigarrow (r_5,s_2) \end{array}$$

$$\begin{array}{ll} (c_3,s_2) \rightsquigarrow (r_3,s_2) & (c_3,s_2) \rightsquigarrow (r_6,s_2) \end{array}$$

$$\begin{array}{lll} (r_1,s_2) \rightsquigarrow (c_1,s_2) & (r_2,s_2) \rightsquigarrow (c_2,s_2) & (r_3,s_2) \rightsquigarrow (c_3,s_2) \\ (r_4,s_2) \rightsquigarrow (c_1,s_2) & (r_5,s_2) \rightsquigarrow (c_2,s_2) & (r_6,s_2) \rightsquigarrow (c_3,s_2) \end{array}$$

Transitions from states $(r_i, s_3)$ and $(c_i, s_3)$

$$\begin{array}{ll} (c_1,s_3) \rightsquigarrow (r_1,s_3) & (c_1,s_3) \rightsquigarrow (r_4,s_3) \end{array}$$

$$\begin{array}{ll} (c_2,s_3) \rightsquigarrow (r_2,s_3) & (c_2,s_3) \rightsquigarrow (r_5,s_3) \end{array}$$

$$\begin{array}{ll} (c_3,s_3) \rightsquigarrow (r_3,s_3) & (c_3,s_3) \rightsquigarrow (r_6,s_3) \end{array}$$

$$\begin{array}{lll} (r_1,s_3) \rightsquigarrow (c_1,s_3) & (r_2,s_3) \rightsquigarrow (c_2,s_3) & (r_3,s_3) \rightsquigarrow (c_3,s_3) \\ (r_4,s_3) \rightsquigarrow (c_1,s_3) & (r_5,s_3) \rightsquigarrow (c_2,s_3) & (r_6,s_3) \rightsquigarrow (c_3,s_3) \end{array}$$

The weights of the BWTS are the labelled to the transitions in accordance with the transitions of the WTS, i.e. all transitions which are from a state including $r_1$ to a state including $c_1$ is labelled with the weight $T_1^F$, and so on. The transitions which only change location, and for which $r_i$ (or $c_i$) remains unchanged, are assigned the weight 0. Hence, the weight assignment follows:

$$d_B((x,s),(x',s')) = \begin{cases} 0, & \text{if } x = x', \forall s, s' \in S, x = x' \in \Pi \\ d(x,x'), & \text{if } x \neq x', \forall s, s' \in S, x = x' \in \Pi \end{cases}$$

The accepting states of the BWTS are all states which include $s_3$, i.e.:

$$\begin{aligned} F_B = \quad & \{(c_1,s_3),(c_2,s_3),(c_3,s_3),(r_1,s_3),(r_2,s_3), \\ & (r_3,s_3),(r_4,s_3),(r_5,s_3),(r_6,s_3)\} \end{aligned}$$

The set of atomic propositions $AP$ is the same set as before, i.e, $AP = \{r_i, c_j\}$ where $i = 1,..,6$ and $j = 1,2,3$. Finally, the labelling function $L_B = L$:

$$L_B(x,s) = L(x)$$

The resulting BWTS is illustrated in figure 21. The transition system consists of many transitions, resulting in a messy illustration. It is however clear that one part of the BWTS, namely the accepting states, have less transitions and appear clearer. The reason for the messy result is that $AP$ isn't fully utilized. Since only $r_2$ and $r_6$ are considered in the MITL formula, it would have been possible to consider $AP = \{r_2, r_6\}$ which would have resulted in a smaller, less messy BWTS. However, this would have caused problems when constructing the WTS. The reason for this, is the walls. If the partition was made differently, we would have been forced to include the walls in the MITL formula, putting restraints on the robot which would forbid it to enter the areas $w_i$, i.e. the areas covered by wall. In the end, this problem would have been much harder to solve.

## 5.4 Designing the Control Signal

From here, we manually apply DFS to find an accepting run. That is, we explore each path from the initial state and determine the word-prefixes which will lead to accepting states.

An example of a possible accepting run is:

$$q_2 \rightarrow q_{13} \rightarrow q_{10} \rightarrow q_{11} \rightarrow q_{34}... \tag{52}$$

which can be determined by the *MATLAB* script, and can be seen in figure 21. The run corresponds to applying the following control:

$$\overline{u_9}, \overline{u_2}, \overline{u_{14}}, \overline{u_{12}}... \tag{53}$$

The maximum time required to satisfy the formula using the above control input is:

$$T_9^F + T_2^F + T_{14}^F + T_{12}^F = 0.0433 + 0.0903 + 0.0591 + 0.0882 = 0.2809 t.u. \tag{54}$$

Note, that the run is only accepting if:

$$T_9^F = 0.0433 \leq a_1 \quad \text{and} \quad T_2^F + T_{14}^F + T_{12}^F = 0.2376 \leq a_2$$

If this is not the case, the transitions won't be possible. This is not illustrated in the figure due to some simplifications where guards, invariants and constraints have been removed to improve readability of the graph.

In MATLAB, Dijkstra's algorithm was used to find the shortest path. The result was identical to the manual DFS and can be viewed in it is entirety in the appendix, in section A.1.1. Applying the resulting controller to the system results in the system-evolution illustrated by the quiver-plot in figure 22

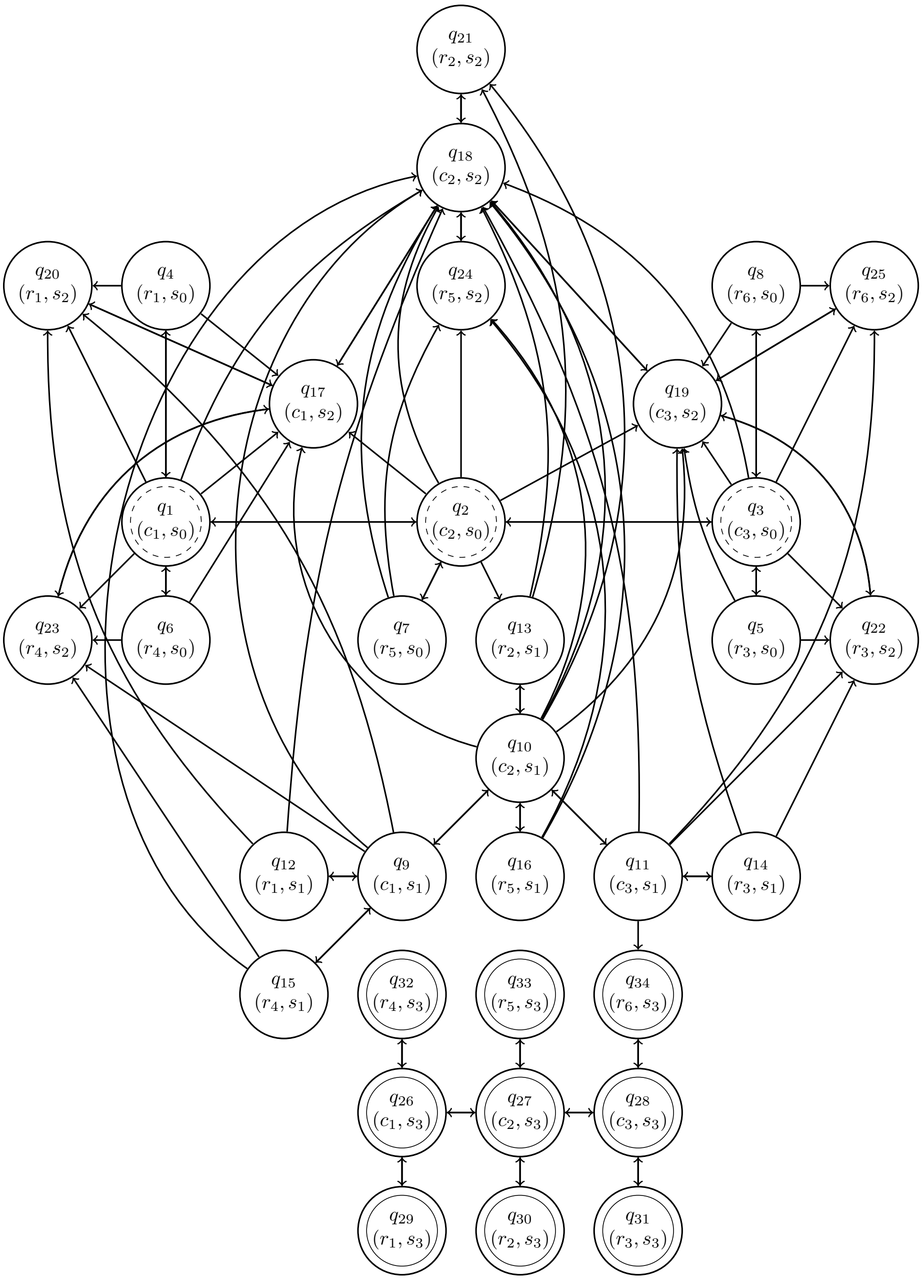
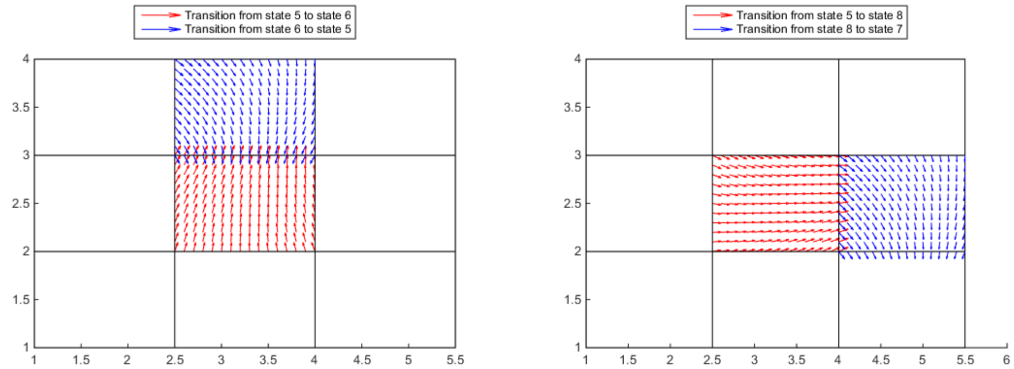



Figure 21: Sketch of the Büchi weighted transition system constructed as an automata product of the abstracted WTS and TBA. The states marked with dashed circles are the initial states and the states marked with whole-drawn circles are accepting states. Guards, invariants and constraints have been removed to improve readability.




(a) Part 1

(b) Part 2

Figure 22: Quiver plots of the system evolution for the closed-loop system of the example in section 5, when the designed controller is applied.

# 6 Problem Definition 2

In the following sections, we consider the multi-agent case, where tasks includes both individual specifications and global assignments which multiple agents are to perform together.

Given $N$ agents governed by the dynamics in (55), synthesize a control input sequence such that the closed-loop systems satisfies the global task specification MITL formula, $\phi_G$, and $N$ local task specifications, $\phi_k$. Where $\phi_G$ is over the atomic proposition set $AP_G$ and consists of tasks which the agents should achieve cooperatively, and $\phi_k$ is over the atomic proposition set $AP_k$, $k \in I$, and consists of individual tasks. Following the theory presented in the previous sections, the problem becomes; synthesize a sequence of individual timed runs $r_1, ..., r_N$ such that (56) holds.

$$\dot{x}_i = A_i x_i + B_i u_i, \;\; i \in I \tag{55}$$

$$(r_G \vDash \phi_G) \wedge (r_1 \vDash \phi_1 \wedge ... \wedge r_N \vDash \phi_N) \tag{56}$$

# 7 Solution Approach 2

The suggested solution follows the idea of [13]. Here, the methods regarding the abstraction of the environment and the translation of the MITL formula have been added. These steps were assumed to have been completed in the previous work. The solution approach is:

1. Abstract the dynamics of each agent into a WTS, by following the method presented in section 4.1.
2. Construct a TBA for each local MITL formula, following the method presented in section 4.2.
3. Construct a BWTS, i.e. a product automata, for all corresponding local WTSs and TBAs, following the method presented in section 4.3. That is, one BWTS for each agent.
4. Construct a product BWTS from the individual BWTSs. The method is presented in section 7.1.
5. Construct a global TBA corresponding to the global MITL formula. The method is presented in section 7.2.
6. Construct the global automata product from the product BWTS and the global TBA. The method is presented in section 7.3.
7. Search for an accepting timed run $r_G$ in the global automata product, following the method presented in section 4.4.

8. Project the accepting timed run $r_G$ onto the individual BWTSs to find the individual accepting timed runs $r_1, .., r_N$. The method is presented in section 7.4.
9. Project the accepting runs $r_1, .., r_N$ onto the corresponding WTS, in accordance with the method presented in section 4.
10. Define the desired control input as the control sequence which yields the runs determined in the previous step, in accordance with the method presented in section 4.

The idea behind the solution is as follows. It follows directly from the theory presented in section 4, that all accepting timed runs of the BWTSs constructed in step 3, corresponds to timed runs which satisfies the local MITL formulas. Furthermore, it is straightforward that an accepting run of the product BWTS will correspond to a run which satisfies all local MITL formulas. This is due to the fact that the accepting runs of the product BWTS will be the set of combinations of the accepting runs of each local BWTS. Finally, the global automata product will simply add the constraints of the global TBA, leaving a BWTS which accepting runs corresponds to the runs which satisfies both the local and the global MITL formulas. It then follows, that the desired control input for each agent is the input that corresponds to the accepting run projected on the individual WTS. This is clear, since the mentioned projection will correspond to the transitions which each individual agent must follow in order to achieve the global accepting run. An example of the method in its entirety is presented in section 8.

## 7.1 Product Büchi Weighted Transition System

The product Büchi weighted transition system (product BWTS), is defined as Definition 7.1.1 in [13]. We suggest a slightly different definition, given in Definition 7.1.2. The difference between the definitions is the use of the variable $b$, which is not used in the latter definition. In the former definition $b$ is a measurement of how much time has passed since each agent performed its previous transition. Hence, each state will hold information on the minimum time required for any transition to occur. However, a consequence is that the number of states will grow significantly which yields a higher computational cost. In the latter definition, $b$ has therefore been removed. Instead of defining the distance between states as the minimum time required for one agent to transition, and creating new states accordingly, the number of states are set and the distance between the states is defined as the maximum time required for all agents to transition in accordance with the given states. The cost of simplifying the definition is the risk of an increased calculated time for some word-sequences. This could in worst case scenario lead to false negative result, i.e. results indicating that there is no solution even though there is. However, it can never yields false positive result, that is if a solution is found it is always guaranteed to be correct. The issue is illustrated by Example 7.1.

**Definition 7.1.1.** Given $N$ local BWTSs $T_1^p, ..., T_N^p$, defined as in Definition 4.3.1, their product BWTS $T_G = T_1^p \otimes ... \otimes T_N^p = (Q_G, Q_G^{init}, \to_G, d_G, F_G, AP_G, L_G)$ is defined as:

- $Q_G \subseteq Q_1 \times ... \times Q_N \times \mathbb{T}^N \times \{1, ..., N\}$,
- $Q_G^{init} = Q_1^{init} \times ... \times Q_N^{init} \times \{0\} \times ... \times \{0\} \times \{1\}$, where $\{0\} \times ... \times \{0\}$ consists of $N$ factors,
- $q_G \to_G q_G'$ iff
  - $q_G = (q_1, ..., q_N, b_1, ..., b_N, l) \in Q_G$,
  - $q_G' = (q_1', ..., q_N', b_1', ..., b_N', l') \in Q_G$,
  - $\exists q_k'' \in Q_k$ s.t. $q_k \rightsquigarrow_k q_k''$ for some $k \in I$,
  -
$$b_k' = \begin{cases} 0, & \text{if } b_k + d_{min} = d_k^p(q_k, q_k'') \\ & \text{and } q_k' = q_k'' \\ b_k + d_min, & \text{if } b_k + d_{min} < d_k^p(q_k, q_k'') \\ & \text{and } q_k' = q_k \end{cases}$$
    where $d_{min} = \min_{k \in \{1,..,N\}} (d_k^p(q_k, q_k'') - b_k)$, i.e. the shortest time-distance required for one agent to perform a transition.
  -
$$l' = \begin{cases} l, & \text{if } q_l \notin F_l \\ ((l+1) mod N), & \text{otherwise} \end{cases}$$

- $d_G(q_G, q'_G) = d_{min}$, if $q_G \rightarrow_G q'_G$,
- $F_G = \{(q_1, ..., q_N, b_1, ..., b_N, N) \in Q_G$ s.t. $q_N \in F_N$,
- $AP_G = \bigcup_{k=1}^{N} AP_k$ and
- $L_G(q_1, .., q_N, b_1, ..., b_N, l) = \bigcup_{k=1}^{N} L_k^p(q_k)$

**Definition 7.1.2.** Given $N$ local BWTSs $T_1^p, ..., T_N^p$, defined as in Definition 4.3.1, and $M_G = \sum_{k=1,..,N} |X_k|$ and $C_G^{max}$ equal to the largest constant in all the BWTSs, the product BWTS $T_G = T_1^p \otimes ... \otimes T_N^p = (Q_G, Q_G^{init}, \rightarrow_G, d_G, F_G, AP_G, L_G)$ is defined as:

- $Q_G \subseteq Q_1 \times ... \times Q_N$
- $Q_G^{init} = Q_1^{init} \times ... \times Q_N^{init}$
- $q_G \rightarrow_G q'_G$ iff
  - $q_G = (q_1, ..., q_N, v_1, ..., v_{M_G}) \in Q_G$,
  - $q'_G = (q'_1, ..., q'_N, v'_1, ..., v'_{M_G}) \in Q_G$,
  - $\exists q'_k \in Q_k$ s.t. $q_k \rightsquigarrow_k q'_k$ for some $k \in I$,
  - For all $i \in \{1, ..., M_G\}$

$$v'_i = \begin{cases} 0, & \text{if } x_i \in R \\ v_i + d_G(r, r'), & \text{if } x_i \notin R \text{ and} \\ & v_i + d_G(r, r') \leq C_G^{max} \\ \infty & \text{otherwise} \end{cases}$$

- $d_G(q_G, q'_G) = d_{max}$, if $q_G \rightarrow_G q'_G$, where $d_{max} = \max_{i=1,..,N}(d_i)$
- $F_G = \{(q_1, ..., q_N, N) \in Q_G$ s.t. $q_N \in F_N\}$,
- $AP_G = \bigcup_{k=1}^{N} AP_k$ and
- $L_G(q_1, .., q_N) = \bigcup_{k=1}^{N} L_k^p(q_k)$

**Example 7.1.** Consider a transition $(q \rightarrow q')$ in a product BWTS constructed from two agents. Let the transition be $(1, 1) \rightarrow (2, 2)$.

If the product BWTS was constructed according to Definition 7.1.1, the transition will be made in several steps, assuming that the cost for each agents transition is not equal. Let's assume that the transition costs 1t.u. for agent 1 and 2t.u. for agent 2. The transition will then be defined as: $(1, 1, 0, 0) \rightarrow (2, 1, 0, 1) \rightarrow (2, 2, 1, 0)$. That is, after 1t.u. agent 1 has transitioned, while agent 2 is only half way towards the transition, after 2t.u. agent 2 manage the transition as well while agent 1 is waiting. The total transition time is hence 2t.u.

Now, let's assume that Definition 7.1.2 was used when constructing the product BWTS. The transition will be defined as one step and the cost will be determined as the maximum of the times each agent requires to transition. That is the transition will be: $(1, 1) \rightarrow (2, 2)$ and the transition time will be: $max(1, 2) = 2$t.u.

Hence, considering only one transition the definitions give the same output. It is therefore clearly advantageously to implement the second definition which yields less states. However, let's now consider a sequence of 2 transitions: $(1, 1) \rightarrow (2, 2) \rightarrow (3, 3)$.

The result of the first transition is given above. Now, let's assume that the individual cost for the second transition is 2t.u. for agent 1 and 1t.u. for agent 2. The entire transition according to Definition 7.1.1 is then: $(1, 1, 0, 0) \rightarrow (2, 1, 0, 1) \rightarrow (2, 2, 1, 0) \rightarrow (3, 3, 0, 0)$ yielding the total time of 3t.u. On the other hand according to Definition 7.1.2 the transition is $(1, 1) \rightarrow (2, 2) \rightarrow (3, 3)$ with the total time of $max(1, 2) + max(2, 1) = 4$t.u.

The reason is that the former definition allows for agent 1 to start moving towards the next state directly while the latter waits for agent 2 to finish the first transition before moving on.

A simple example, describing the construction of product BWTS, is given in Example 7.2.

**Example 7.2.** Consider two agents with continuous linear dynamics, placed in an environment divided into two parts; A and B, by the set of atomic proposition. Furthermore, consider that each agent must satisfy the MITL formula (57), where $X$ is equal to A for agent 1 and B for agent 2.

$$\Box_{\leq 1} X \tag{57}$$

Following the previous presented theory this yields two automata products with 6 states each. Namely,

$$\begin{array}{ccc} (1, A) & (2, A) & (3, A) \\ (1, B) & (2, B) & (3, B) \end{array}$$

where 1 is the initial state, 2 is the accepted state and 3 is the non-accepting state of the TBA.

From here a BWTS product can be constructed. The BWTS product will consist of $6 \times 6 = 36$ states, if Definition 7.1.2 is applied. Namely, all possible combinations of the local states. (If Definition 7.1.1 is applied, the number of states will be a minimum of $2 \times 6 \times 6 = 72$states, i.e. the combinations of local states and $l = 1, 2$. The number of states in this case would then increase when the possibilities of $b$ is considered.) Assuming that agent 1 starts in area A and agent 2 starts in area B (the only starting positions which could yield an accepting run), the initial state will be $((1, A), (1, B))$. Furthermore, the accepting states will be all combinations which corresponds to each agent being in the TBA state 2, i.e. $((2, X_1), (2, X_2))$.

A transition $((s_1, X_1), (s_2, X_2)) \rightarrow ((s'_1, X'_1), (s'_2, X'_2))$ exists if and only if, the local transitions $(s_i, X_i) \rightarrow (s'_i, X'_i)$ are defined for each agent. Furthermore, if the transition exists, the time cost for the transition is given as

$$d(((s_1, X_1), (s_2, X_2)), ((s'_1, X'_1), (s'_2, X'_2))) = max(d_1((s_1, X_1), (s'_1, X'_1)), d_2((s_2, X_2), (s'_2, X'_2))).$$

The global set of atomic proposition is $AP = \{A_1, A_2, B_1, B_2\}$ and the labelling is the combinations of the local labelling, i.e. $L((s_1, A), (s_2, B))) = \{A_1, B_2\}$ and so on.

Notable is that the number of states can be reduced in the case when Definition 7.1.1 was applied, when considering the evolution of $l$. Since $l = 1$ for all states corresponding to the first agent not being in a local accepting state, adding the fact that the local TBAs are constructed such that an agent won't leave a local accepting state when it has reached it, all states which fulfils this and has $l = 2$ will be unreachable, and hence could be removed. This state reduction would result in 24 states being removed, yielding a total of 48 states. This reduction will have a great impact on the computational demand, especially when the size of each local automata product- and the number of agents are increased.

## 7.2 Translation of a Global MITL Formula into a Global Timed Büchi Automaton

In this section, the translation of a global MITL formula into a global TBA is considered.

Firstly, let us consider what a global MITL formula is. The global MITL formula, should impose tasks which requires multiple agents to participate. A simple example is $\phi_G^1$ in (58), where $\phi_1$ and $\phi_2$ are local MITL formulas which should be satisfied by agents 1 and 2. $\phi_G^1$ states that the local MITL formulas should both be satisfied at the same time. A concrete example of this is *'Agent 1 must be in room 1 at the same time as agent 2 is in room 4.'*. The specification must be handled as a global formula, since the local formulas only have the capacity to state them individually. That is, the local formulas can state *'Agent 1 must be in room 1 at some time.'* and *'Agent 2 must be in room 4 at some time.'*, while the condition of the two statements occurring simultaneously can only be expressed globally.

$$\phi_G^1 = \Diamond(\phi_1 \wedge \phi_2) \tag{58}$$

The advantage of allowing global specifications is the possibility for cooperative tasks. An example of this is two robots picking up and carrying an object which is too heavy for one of them to carry alone. The example is further discussed in Example 7.3. The construction of a global TBA from a global MITL formula follows the same steps as the construction of the local TBA, presented in section 4.2.

**Example 7.3.** The specification considering two robots which are to cooperate with carrying an object from area A to area B, can be specified using global formula as *'Agent 1 and agent 2 must reach area A before time T. They must then pick up the object and go to area B. At area B they must put the object down.'* or

$$\begin{aligned}\phi_G^2 = \quad & \Diamond_{<T}(A_1 \wedge A_2) \wedge ((A_1 \wedge A_2) \rightarrow (PickUp_1 \wedge PickUp_2)) \quad \wedge \\ & ((PickUp_1 \wedge PickUp_2) \rightarrow (HoldOn_1 \wedge HoldOn_2)\mathcal{U}(B_1 \wedge B_2)) \quad \wedge \\ & (((B_1 \wedge B_2) \wedge (HoldOn_1 \wedge HoldOn_2)) \rightarrow (PutDown_1 \wedge PutDown_2))\end{aligned}$$

which directly translates to

- Agent 1 and agent 2 must both be in area 1 within time T, and
- agent 1 and agent 2 being in area 1 at the same time, implies that they must both pick up the object at the same time, and
- both agents having picked up the object at the same time, implies that they must hold on to the object until they have reached area B, and
- both agents being in area B and holding on to the object, implies that they must both put the object down at the same time.

The global TBA corresponding to the formula is illustrated in figure 23.

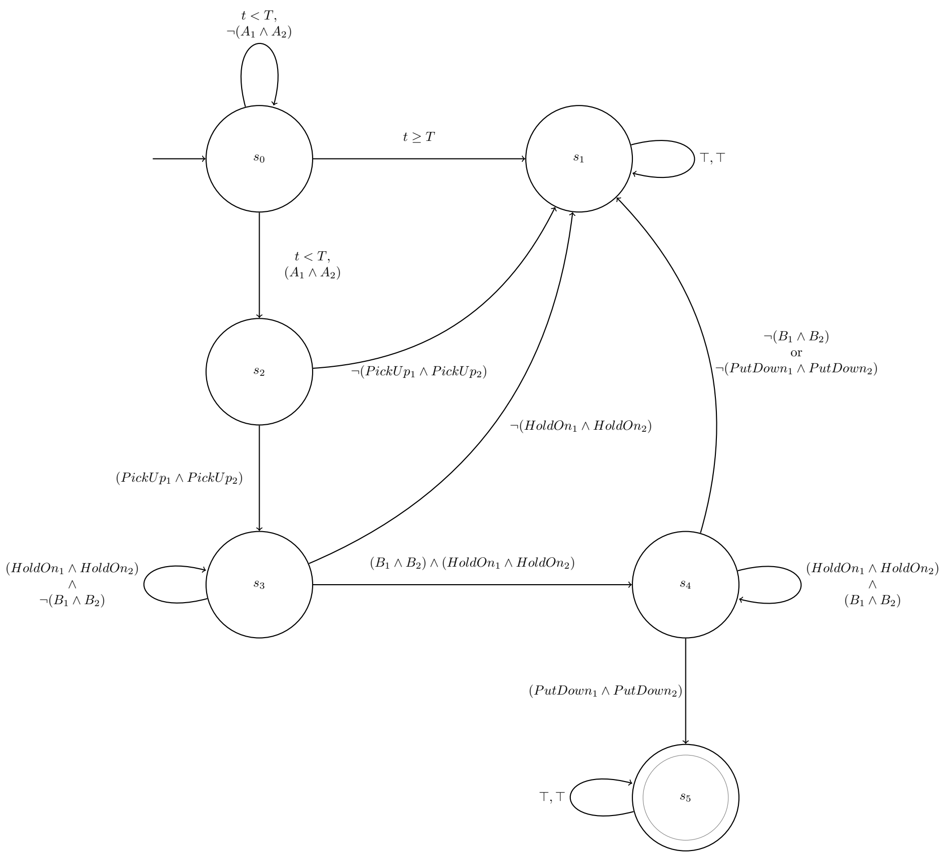



Figure 23: The global TBA corresponding to the global MITL formula $\phi_G^2$.

## 7.3 Global Automata Product

The global automata product is defined as Definition 7.3.1. The definition follows [13] directly.

**Definition 7.3.1.** Given a product BWTS $T_G = (Q_G, Q_G^{init}, \to_G, d_G, F_G, AP_G, L_G)$ and a global TBA $A_G = (S_G, S_G^{init}, X_G, I_G, E_G, \mathcal{F}_G, \mathcal{L}_G)$, with $M_G = |X_G|$ and $C_G^{max}$ equal to the largest constant in $A_G$, their product $\hat{T}_G = T_G \otimes A_G = (\hat{Q}_G, \hat{Q}_G^{init}, \rightsquigarrow_G, \hat{d}_G, \hat{F}_G, AP_G, \hat{L}_G)$ is defined as:

- $\hat{Q}_G \subseteq \{(q,s) \in Q_G \times S_G \text{ s.t. } L_G(q) = \mathcal{L}_G(s)\} \times \mathbb{T}_\infty^{M_G}$,
- $\hat{Q}_G^{init} = Q_G^{init} \times S_G^{init} \times \{0\} \times ... \times \{0\} \times \{1,2\}$, where $\{0\} \times ... \times \{0\}$ consists of $M_G$ factors,
- $q \rightsquigarrow_G q'$ iff
  - $q = (r, s, v_1, ..., v_{M_G}, l) \in \hat{Q}_G$,
  - $q' = (r', s', v'_1, ..., v'_{M_G}, l') \in \hat{Q}_G$,
  - $r \to_G r'$,
  - $\exists \gamma, R$ s.t. $(s, \gamma, R, s') \in E_G$, $v_1, ..., v_{M_G} \models gamma$, $v'_1, ..., v'_{M_G} \models I_G(s')$,
  - For all $i \in \{1, ..., M_G\}$

$$v'_i = \begin{cases} 0, & \text{if } x_i \in R \\ v_i + d_G(r, r'), & \text{if } x_i \notin R \text{ and} \\ & v_i + d_G(r,r') \leq C_G^{max} \\ \infty & \text{otherwise} \end{cases}$$

  - 

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

- $\hat{d}_G(q, q') = d_G(r, r')$ if $q \rightsquigarrow_G q'$,
- $\hat{F}_G = \{(r, s, v_1, ..., v_{M_G}, 1) \in \hat{Q}_G \text{ s.t. } r \in F_G\}$ and
- $\hat{L}_G(r, s, v_1, ..., v_{M_G}) = L_G(r)$.

**Example 7.4.** Consider the BWTS product constructed in Example 7.2 and some global TBA consisting of three states; initial, accepting and non-accepting, and one clock. Assuming that the state-reduced BWTS product is used, the global product will consist of $36 \times 3 \times 2 = 216$states. The product is defined the same way as the automata product with the added constraint on $l$. In this case however, no state reduction can be made. This is due to the structure of $l$, namely that $l$ can be both 1 and 2 for the same state combination.

When the global product has been constructed, an accepting run is found using a search algorithm such as DFS or Dijkstra, in the same manner as for the local product described in section 4.4.

## 7.4 Projection of a Global Accepting Timed Run onto Local Büchi Weighted Transition Systems

When the accepting run has been found for the global product, projection is used to determine which local paths this corresponds to for each agent. Based on the local paths, the controller can be determined in the same manner as for problem 1 (see section 4.4).

The projection is performed in three steps; projection onto the BWTS product, projection onto BWTSs and finally, projection onto WTSs. In each step one simply check which lower-level[2] state that corresponds to the higher-level[3] state, and repeats this for the entire path. It follows from the construction of the product, that only one lower-level state will correspond to each higher-level state.

[2]Here, lower-level refers to the BWTS product in the first step, the BWTSs in the second step and the WTSs in the third step.

[3]Here higher-level refers to the global product in the first step, the BWTS product in the second step and the BWTSs in the third step.

# 8 Implementation 2

In this section we present the result from some performed simulations. All simulations have been performed following Definition 7.1.2 in section 7.1, rather than Definition 7.1.1. The reason for this is the computational demand.

Let us first consider a simplified sub-problem of Problem 2, namely the case where there is no global MITL specification. The problem then becomes to construct one automata product for each agent and finding an accepting run in each product. Consider Example 8.1.

**Example 8.1.** Given 3 agents which follows the dynamics:

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

$$x(0)_1 = (2, 3.5) \quad x(0)_2 = (4, 3.5) \quad x(0)_3 = (6, 3.5)$$

Find the controllers which satisfies the local MITL formulas:

$$\begin{aligned} \phi_1 &= (c\mathcal{U}_{<2.4} r_1) \wedge (\Diamond_{<10} \Box_{<1} r_5) \\ \phi_2 &= (\Diamond_{<2.1} r_2) \wedge (r_2 \rightarrow \Diamond_{<3.35} r_6) \\ \phi_3 &= (\Diamond_{2.4} (r_{12} \vee r_{10})) \wedge (\Diamond_{<1} r_3) \end{aligned}$$

where the state space is bounded by $1 \leq x_1 \leq 11$ and $1 \leq x_2 \leq 6$, the control signal is bounded by $-30 \leq u_i \leq 30$ for $i = 1, 2$, and the atomic proposition set is defined as:

$$\begin{array}{lll} r_1: & 1 \leq x_1 \leq 5 & 1 \leq x_2 \leq 2 \\ r_2: & 7 \leq x_1 \leq 11 & 1 \leq x_2 \leq 2 \\ r_3: & 1 \leq x_1 \leq 3 & 2 \leq x_2 \leq 3 \\ r_4: & 3 \leq x_1 \leq 5 & 2 \leq x_2 \leq 3 \\ r_5: & 7 \leq x_1 \leq 9 & 2 \leq x_2 \leq 3 \\ r_6: & 9 \leq x_1 \leq 11 & 2 \leq x_2 \leq 3 \\ r_7: & 1 \leq x_1 \leq 3 & 4 \leq x_2 \leq 5 \\ r_8: & 3 \leq x_1 \leq 5 & 4 \leq x_2 \leq 5 \\ r_9: & 7 \leq x_1 \leq 9 & 4 \leq x_2 \leq 5 \\ r_{10}: & 9 \leq x_1 \leq 11 & 4 \leq x_2 \leq 5 \\ r_{11}: & 1 \leq x_1 \leq 5 & 5 \leq x_2 \leq 6 \\ r_{12}: & 7 \leq x_1 \leq 11 & 5 \leq x_2 \leq 6 \end{array}$$

and

$$c: \quad X \setminus (r_1 \cup r_2 \cup r_3 \cup r_4 \cup r_5 \cup r_6 \cup r_7 \cup r_8 \cup r_9 \cup r_{10} \cup r_{11} \cup r_{12}).$$

Simulating the problem in MATLAB yields three accepting runs (one per agent) which satisfies the formulas. The runs are [3813121112131817...], [813121116111213182322] and [138323813141520]. The partitioned environment, complete with state numbering is illustrated by figure 24.

The results in its entirety can be viewed in section A.2.2 in the appendix.

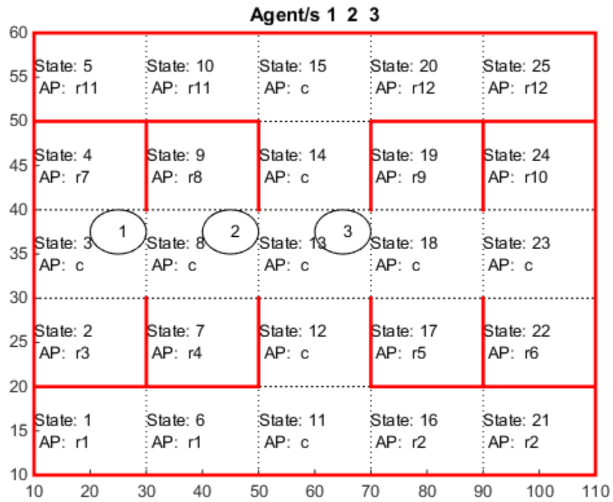


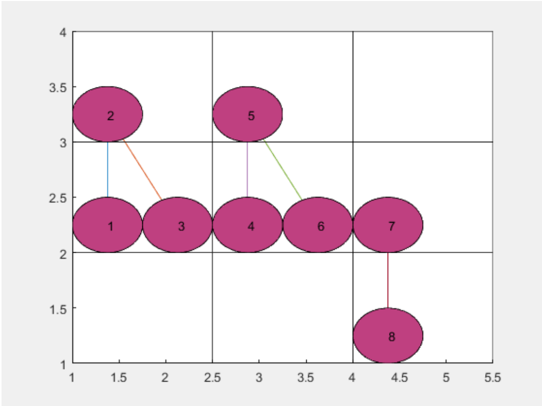

Figure 24: Partition of each agents state space in accordance with the settings used in this section. The circles with numbers 1 to 3 represents the initial states of each agent.

Now, when the construction of multiple BWTSs is done, let's move on to the full problem and consider Example 8.2.

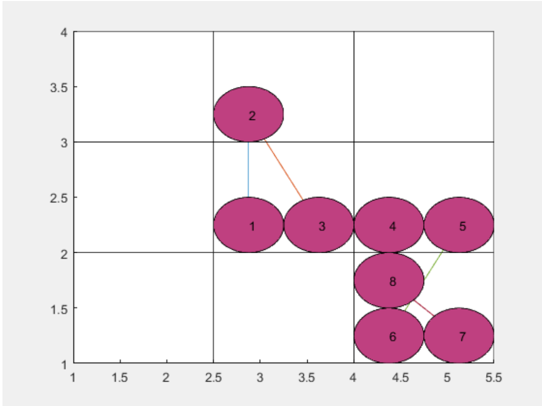
**Example 8.2.** Consider two agents placed in an environment consisting of 6 rooms and a hallway (see figure 18). Each agent is tasked with the local MITL formula $\phi_L$. Furthermore, the global MITL formula $\phi_G$ must be satisfied.

| | |
|---|---|
| $\phi_L = \Diamond_{0.06} r_2 \wedge r_2 \rightarrow \Diamond_{0.3} r_6$ | *'Eventually, within* 0.06 *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.'* |
| $\phi_G = \Diamond_{10}(a_1 = r_1 \wedge a_2 = r_2)$ | *'Eventually, within* 10 *time units, agent 1 must be in room 1 and agent 2 must be in room 2, at the same time.'* |

Implementing the problem in MATLAB gives the following result. First, the environment of each agent is abstracted to a WTS, the local MITL formulas are manually translated into TBAs and the automata product is constructed (see section 5 for details). Next, the product BWTS must be constructed. Implementing Definition 7.1.2 in MATLAB, the result is a product BWTS with $(Q_1 \cdot Q_2) = 1296$ states. Due to the size of the system, it will not be presented in detail. Secondly, a global TBA is constructed. The construction is performed manually and added to the MATLAB scripts as input to the global product. The implementation of the global product follows Definition 7.3.1 and yields a total of $2 \cdot (Q_{pBTWS} \times Q_{gTBA}) = 7776$ states. Finally, implementing the DFS-algorithm yields the accepting run: [321 3860 3563 4452 4811 4520 5383 5240] in the global product. Which then can be projected to the global TBA and the product BWTS, resulting in [1 2 2 2 2 2 2 2] and [161 310 162 606 786 640 1072 1000] respectively. Next, the path in the product BTWS can be projected on the individual local products and finally onto each agents WTS, yielding [2 3 2 5 6 5 8 7] and [5 6 5 8 8 7 7 7], for each agent. The result is visualized in figures 25, 26 and 27. The distances between each transition are [0.04 0.077 0.059 0.04 0.077 0.053 0.067]. From this, it is clear that the given path will satisfy the MITL formulas.

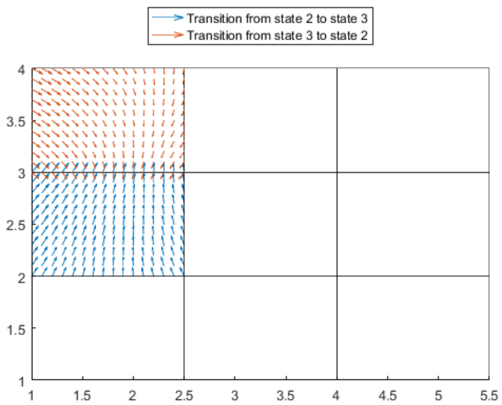



(a) Agent 1

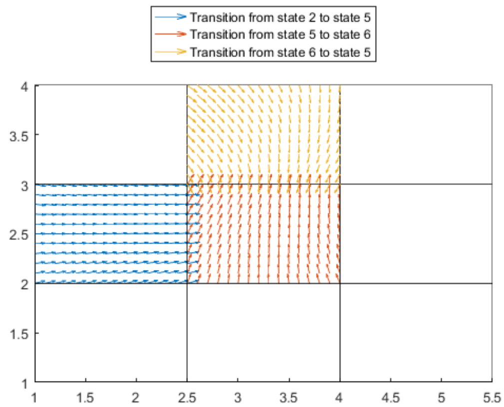



(b) Agent 2

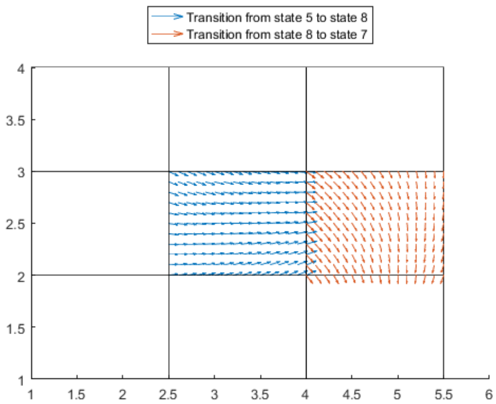
Figure 25: Illustration of the path of the agents.




(a) Part 1: $2 \rightarrow 3 \rightarrow 2$

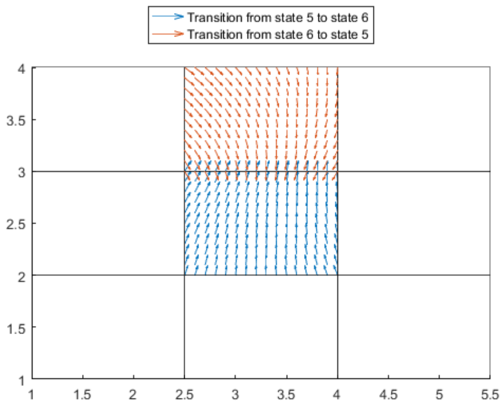



(b) Part 2: $2 \rightarrow 5 \rightarrow 6 \rightarrow 5$

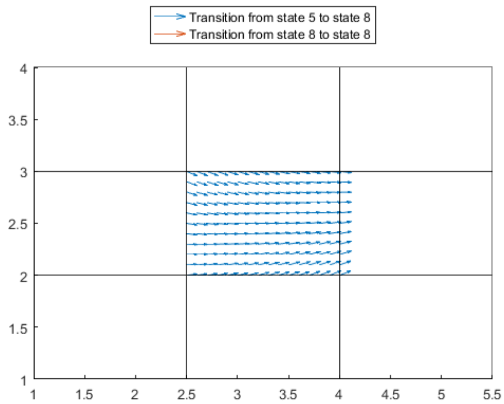



(c) Part 3. $5 \rightarrow 8 \rightarrow 7$

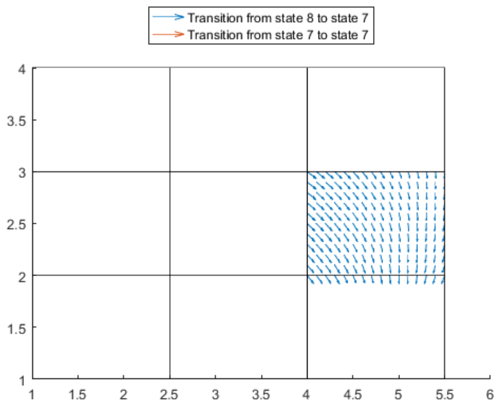
Figure 26: Illustration of the evolution of system 1 when the computed controllers are applied.




(a) Part 1: $5 \to 6 \to 5$




(b) Part 2: $5 \to 8$, stay in 8.




(c) Part 3: $8 \to 7$. Stay in 7 is not illustrated since this would be a blank background.

Figure 27: Illustration of the evolution of system 2 when the computed controllers are applied.

# 9 Discussion and Conclusion

The work presented in this thesis consists of two methods considering control synthesis for continuous linear systems under MITL specifications. The first method considers single-agent systems while the second, which is based on the former, considers multi-agent systems. The methods are supported by theory as well as extended simulations performed in the MATLAB environment. It is clear from the simulations that the methods yield the desired result. Each method guarantees that the returned control design will be such that the closed-loop system will satisfy the MITL specifications. However, due to the transition cost being calculated as the maximum time required and hence only including a maximum limit, it is possible that there are solutions which the methods cannot find. This depends on the position the agent has within a rectangle of the partition and is a result of the discretization of the environment. In the multi-agent case, it can also be caused by assuming that the agents make transitions together, i.e. when Definition 7.1.2 is implemented instead of Definition 7.1.1 in the step of constructing the product BWTS. A suggested overall method is therefore to begin with the simple methods - the methods presented here with the implementation of Definition 7.1.2, and then continue with more advanced methods if the former does not give a result. The more advanced methods would include implementing Definition 7.1.1 for the BWTS product as well as considering a more complicated abstraction. For instance, the partition of the environment could be more refined, allowing the transition costs to be more precise. Another option is to let a rectangle yield more than one state. The transitions cost could then be set to depend on which facet the agent start at, i.e from which direction the previous transition was made from. Each decision which yields a greater number of states will also yield the range of times required to perform a transition (depending on the position within a rectangle) to decrease and hence result in a decreased overestimate. This will lower the risk for false negative result, but also cost more computationally.

This concerns the issue of state explosion. The computational time is fast increasing for the enhancement of both environment and demand (number of states in the weighted transitions system and timed Büchi automata), as well as for added number of agents. This might be fine when considering deterministic systems, i.e. environment and specifications which does not change. It does however, become a problem when considering systems which do change, demanding the control synthesis to be updated during the run. For instance, if the agents should adapt to each other, i.e. re-plan the route based on the position of the other agents. To implement the methods under such circumstances, there would be a need to first develop a method to update the systems without redoing all steps.

Another issue is that the method does not necessarily give the answer to whether a solution exists or not. While the method guarantees that it will eventually find a solution, if there is one within the range of the abstraction, it does not guarantee that it will be able to determine if there does not exist a solution. If all runs of the final Büchi weighted transition system are finite, the search for a path will end either when a solution is found or when there is no more path to investigate. However, if the runs are infinite, the method will keep searching for a path for an infinite time if no solution exists, unless some conditions are added to the path finder demanding it to stop when it is obvious that no path exists. An example of such a condition is if the remaining states to search are all non-accepting, i.e. states which corresponds to some MITL specification being violated.

# 10 Future Work

There is much room for further study of how control synthesis under MITL specifications can be performed. Some obvious continuations are to study how the problems presented in section 9 can be solved. That is, developing methods which can be implemented in real-time and refining the abstraction to achieve more precise transition costs.

One study of interest would be to design controllers for multi-agent systems under the dynamics presented in (59), i.e. systems where each individual agent takes the dynamics of the other agents under consideration.

$$\dot{x}_i = f(x_i, x_j) \tag{59}$$

Another possibility is to study how more complicated specifications can be implemented by incorporating scheduling. For instance creating the possibility of having a task pool which some agents share, allowing any of them to perform a certain tasks. Also, the possibility of realising some tasks upon the achievement of others, implementing collaborative tasks on a form such that one agent performs the first part and another the second.

# References


[1] Wongpinomsarn, Topcu, and Murray, "Receding horizon control for temporal logic specifications," in *2010 International Conference on Hybrid Systems: Computational and Control (HSCC)*.

[2] Kress-Gazit, Fainekos, and Pappas, "Where is waldo? sensor based temporal logic motion planning," *mag*, 2007.

[3] Kress-Gazit, Fainekos, and Pappas, "Translating structured english to robot controllers," *Advanced Robotics*, 2008.

[4] Meng, *Hybrid Control of Multi-robot Systems under Complex Temporal Tasks.* PhD thesis, EES, KTH Royal Institute of Technology, 2016.

[5] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *Automatic Control, IEEE Transactions on*, vol. 53, no. 1, pp. 287–297, 2008.

[6] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Synthesis of control protocols for autonomous systems," *Unmanned Systems*, vol. 1, no. 01, pp. 21–39, 2013.

[7] P. Bouyer, "From qualitative to quantitative analysis of timed systems," *Mémoire d'habilitation, Université Paris*, vol. 7, pp. 135–175, 2009.

[8] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 5025–5032, IEEE, 2013.

[9] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Motion and action planning under ltl specifications using navigation functions and action description language," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 240–245, IEEE, 2013.

[10] *Principles of model checking.* MIT press, 2007.

[11] 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.

[12] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166, Springer, 2004.

[13] A. Nikou, J. Tumova, and D. V. Dimarogonas, "Cooperative task planning of multi-agent systems under timed temporal specifications," *CoRR*, vol. abs/1509.09137, 2015.

[14] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.

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

[16] 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.

[17] J. Fu and U. Topcu, "Computational methods for stochastic control with metric interval temporal logic specifications," *CoRR*, vol. abs/1503.07193, 2015.

[18] R. Alur, "Timed automata," in *Computer Aided Verification*, pp. 8–22, Springer, 1999.

[19] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pp. 239–248, ACM, 2015.

[20] X. Jin, "Verification and validation of hybrid systems," 2013.

[21] A. Donzé and O. Maler, *Robust satisfaction of temporal logic over real-valued signals*. Springer, 2010.

[22] A. Donzé, T. Ferrere, and O. Maler, "Efficient robust monitoring for stl," in *Computer Aided Verification*, pp. 264–279, Springer, 2013.

[23] Donzé, "On signal temporal logic," 2014. Presentation slides.

[24] Tumova, "Control strategy synthesis under infeasible goals: A maximally-satisfying approach," 2015. Presentation slides.

[25] O. Maler, D. Nickovic, and A. Pnueli, "From mitl to timed automata," in *Formal Modeling and Analysis of Timed Systems*, pp. 274–289, Springer, 2006.

[26] J. Worrell, "On the decidability and complexity of metric temporal logic over finite words," in *Logical Methods in Computer Science*, Citeseer, 2007.

[27] D. Ničković and N. Piterman, *From MTL to deterministic timed automata*. Springer, 2010.

[28] T. Brihaye, M. Estiévenart, and G. Geeraerts, "On mitl and alternating timed automata," in *Formal Modeling and Analysis of Timed Systems*, pp. 47–61, Springer, 2013.

[29] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[30] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

# A MATLAB Result

The operators of the MITL formulas includes:

- *Always*, $\Box$, A
- *Eventually*, $\Diamond$, Ev
- *Until*, $\mathcal{U}$, U
- *Implies*, $\rightarrow$, Imp
- *And*, $\wedge$, and
- *Or*, $\vee$, or

where all clock-constraints are illustrated within parentheses in the MATLAB print.

## A.1 Problem 1

Single agent environment which follows the dynamics:

$$\dot{x} = Ax + Bu$$

The result below consists of figures illustrating the partition and text-files with prints summarizing the result of the run. For each setup there are 4 results. These results represent the combinations of the two parameters *lin_ass* and *u_joint*. The former parameter indicates whether the necessary assumptions to calculate the simplified maximum time has been used, i.e. if it is assumed that $\dot{x}_i$ is not dependent on $x_j$ in the closed-loop system. The latter parameter determines which type of control limit has been used; $\sqrt{u_1^2 + u_2^2} \leq u_{max}$ or $u_{min} \leq u_i \leq u_{max}$. Which setting that are used is documented in each text-file.

### A.1.1 Final Result 1

Note: this is from the example used in section 5.

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
________________________________________________________________

The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
(u1^2+u2^2)^(1/2) <= 20

 The simplified time calculation was used, assuming that dxj/dt is not
    ↪ dependent on xi (i~=j)in the closed-loop system.

 MITL formula:
 Phi=Ev(<0.06) r2 and (r2 Imp Ev(<0.3) r6)
 ________________________________________________________________
 The shortest path in T which satisfies the MITL formula is:
```

```
 [ 5 6 5 8 7  ]
The corresponding time is: 3.199671e-01

 The run is achieved if the control input is given as:

 u1 = -8.560020e+00 x1 -1 x2+ 2.574008e+01
u2 = 0 x1 -6.589765e-01 x2+ 1.834000e+01
Until area 6 is reached

 u1 = -1.089434e+01 x1 -1 x2+ 3.507737e+01
u2 = 0 x1 7.505733e-01 x2+ -1.861479e+01
Until area 5 is reached

 u1 = 3.932173e-13 x1 -1 x2+ 2.191428e+01
u2 = 0 x1 -6.938211e+00 x2+ 1.431463e+01
Until area 8 is reached

 u1 = -9.425780e+00 x1 -1 x2+ 4.034179e+01
u2 = 0 x1 9.813774e-01 x2+ -1.671911e+01
Until area 7 is reached

 Which is an accepting state.
```

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
------------------------------------------------------------------------

The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
(u1^2+u2^2)^(1/2) <= 20

 The original time calculation from Belta was used.

 MITL formula:
 Phi=Ev(<0.06) r2 and (r2 Imp Ev(<0.3) r6)
 ------------------------------------------------------------------------
 The shortest path in T which satisfies the MITL formula is:
 [ 5 6 5 8 7  ]
The corresponding time is: 3.270362e-01

 The run is achieved if the control input is given as:

 u1 = -7.229555e+00 x1 -6.556814e-01 x2+ 1.938526e+01
u2 = -2.282441e+00 x1 -2.132734e-01 x2+ 2.613265e+01
```

```
Until area 6 is reached

 u1 = -7.901101e+00 x1 -6.483665e-01 x2+ 2.169787e+01
u2 = 2.768587e+00 x1 2.346271e-01 x2+ -2.762535e+01
Until area 5 is reached

 u1 = -3.768535e-11 x1 -1.085726e+00 x2+ 2.217145e+01
u2 = -7.242839e-11 x1 -6.500053e+00 x2+ 1.300014e+01
Until area 8 is reached

 u1 = -9.247252e+00 x1 -6.291166e-01 x2+ 3.824724e+01
u2 = 3.920116e+00 x1 3.448485e-01 x2+ -3.637016e+01
Until area 7 is reached

 Which is an accepting state.
```

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
------------------------------------------------------------------
The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
-20 <= u1 <= 20
 -20 <= u2 <= 20

 The simplified time calculation was used, assuming that dxj/dt is not
     ↪ dependent on xi (i~=j)in the closed-loop system.

 MITL formula:
 Phi=Ev(<0.06) r2 and (r2 Imp Ev(<0.3) r6)
 ------------------------------------------------------------------
 The shortest path in T which satisfies the MITL formula is:
 [ 5 6 5 8 7  ]
The corresponding time is: 2.365427e-01

 The run is achieved if the control input is given as:

 u1 = -1.751874e+01 x1 -1 x2+ 5.497313e+01
u2 = 0 x1 -1.306949e-08 x2+ 2.000000e+01
Until area 6 is reached

 u1 = -1.699706e+01 x1 -1 x2+ 5.367559e+01
u2 = 0 x1 -7.151945e-09 x2+ -2.000000e+01
Until area 5 is reached
```

```
 u1 = −1.075126e−09 x1 −1 x2+ 2.200000e+01
u2 = 0 x1 −1.722593e+01 x2+ 3.794565e+01
Until area 8 is reached

 u1 = −1.327690e+01 x1 −1 x2+ 5.822073e+01
u2 = 0 x1 −1.451835e−08 x2+ −2.000000e+01
Until area 7 is reached

 Which is an accepting state.
```

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
------------------------------------------------------------
The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
−20 <= u1 <= 20
 −20 <= u2 <= 20

 The original time calculation from Belta was used.

 MITL formula:
 Phi=Ev(<0.06) r2 and (r2 Imp Ev(<0.3) r6)
 ------------------------------------------------------------
 The shortest path in T which satisfies the MITL formula is:
 [ 5 6 5 8 7 ]
The corresponding time is: 2.365427e−01

 The run is achieved if the control input is given as:

 u1 = −1.260795e+01 x1 6.492620e+00 x2+ 1.847703e+01
u2 = −1.898477e−10 x1 −1.979031e−10 x2+ 2.000000e+01
Until area 6 is reached

 u1 = −1.184939e+01 x1 5.406586e+00 x2+ 1.241583e+01
u2 = −7.852220e−11 x1 −7.127313e−11 x2+ −2.000000e+01
Until area 5 is reached

 u1 = −2.174903e−10 x1 −2.876976e−10 x2+ 2.000000e+01
u2 = 6.308873e+00 x1 −1.963221e+01 x2+ 2.558871e+01
Until area 8 is reached

 u1 = −8.135684e+00 x1 4.841103e+00 x2+ 1.540933e+01
```

```
u2 = 3.868424e-07 x1 7.822381e-07 x2+ -2.000000e+01
Until area 7 is reached

 Which is an accepting state.
```

### A.1.2 Final Result 2

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
------------------------------------------------------------
The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
(u1^2+u2^2)^(1/2) <= 20

 The simplified time calculation was used, assuming that dxj/dt is not
    ↪ dependent on xi (i~=j)in the closed-loop system.

 MITL formula:
 Phi=(c U(<2.4) r1 and Ev(<10) A(<1) r5)
 ------------------------------------------------------------
 The shortest path in T which satisfies the MITL formula is:
 [ 5 2 3 2 5 4 ... ]
The corresponding time is: 1.389159e+00

 The run is achieved if the control input is given as:

 u1 = 2.023490e-13 x1 -1 x2+ -1.891428e+01
u2 = 0 x1 -7.649799e+00 x2+ 1.644940e+01
Until area 2 is reached

 u1 = -5.470065e+00 x1 -1 x2+ 8.175159e+00
u2 = 0 x1 1.482530e-10 x2+ 1.922888e+01
Until area 3 is reached

 u1 = -5.469934e+00 x1 -1 x2+ 8.174825e+00
u2 = 0 x1 -1.016143e-11 x2+ -1.922888e+01
Until area 2 is reached

 u1 = -1.342923e-11 x1 -1 x2+ 1.891428e+01
u2 = 0 x1 -9.989368e+00 x2+ 2.346810e+01
Until area 5 is reached
```

```
 u1 = −6.232853e+00 x1 −1 x2+ 1.643141e+01
u2 = 0 x1 −2.515901e−09 x2+ −1.810387e+01
Until area 4 is reached

 u1 = −2 x1 −1 x2+ 0
u2 = 0 x1 −2 x2+ 0
For 1.001000e+00 s, to stay in area 4
 Which is an accepting state.
```

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
————————————————————————————————————————————————————————————
The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
(u1^2+u2^2)^(1/2) <= 20

 The original time calculation from Belta was used.

 MITL formula:
 Phi=(c U(<2.4) r1 and Ev(<10) A(<1) r5)
 ————————————————————————————————————————————————————————————
 The shortest path in T which satisfies the MITL formula is:
 [ 5 2 3 2 5 4 ... ]
The corresponding time is: 1.445580e+00

 The run is achieved if the control input is given as:

 u1 = −6.423305e−12 x1 1.085721e+00 x2+ −2.217144e+01
u2 = 2.494063e−11 x1 −6.500011e+00 x2+ 1.300003e+01
Until area 2 is reached

 u1 = −5.245005e+00 x1 −6.325703e−01 x2+ 6.510216e+00
u2 = −1.166076e+00 x1 −1.470215e−01 x2+ 2.146012e+01
Until area 3 is reached

 u1 = −5.996781e+00 x1 −5.050259e−01 x2+ 7.512051e+00
u2 = 1.511505e+00 x1 1.330303e−01 x2+ −2.191059e+01
Until area 2 is reached

 u1 = 2.176370e−11 x1 −1.085726e+00 x2+ 2.217145e+01
u2 = 2.053960e−10 x1 −6.500052e+00 x2+ 1.300014e+01
Until area 5 is reached
```

```
 u1 = −7.280561e+00 x1 −5.791493e−01 x2+ 1.935969e+01
u2 = 2.299706e+00 x1 1.873735e−01 x2+ −2.612401e+01
Until area 4 is reached

 u1 = −2 x1 −1 x2+ 0
u2 = 0 x1 −2 x2+ 0
For 1.001000e+00 s, to stay in area 4
 Which is an accepting state.
```

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
──────────────────────────────────────────────────────────────
The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
−20 <= u1 <= 20
 −20 <= u2 <= 20

 The simplified time calculation was used, assuming that dxj/dt is not
    ↪ dependent on xi (i~=j)in the closed−loop system.

 MITL formula:
 Phi=(c U(<2.4) r1 and Ev(<10) A(<1) r5)
 ──────────────────────────────────────────────────────────────
 The shortest path in T which satisfies the MITL formula is:
 [ 5 2 3 2 5 4 ... ]
The corresponding time is: 1.387595e+00

 The run is achieved if the control input is given as:

 u1 = −7.089858e−09 x1 −1 x2+ −1.700000e+01
u2 = 0 x1 −1.055826e+01 x2+ 2.009523e+01
Until area 2 is reached

 u1 = −1.654162e+01 x1 −1 x2+ 2.728151e+01
u2 = 0 x1 −5.165307e−09 x2+ 2.000000e+01
Until area 3 is reached

 u1 = −1.614615e+01 x1 −1 x2+ 2.541595e+01
u2 = 0 x1 2.873038e−08 x2+ −2.000000e+01
Until area 2 is reached

 u1 = 4.294845e−12 x1 −1 x2+ 2.200000e+01
u2 = 0 x1 −1.649390e+01 x2+ 3.585565e+01
```

```
Until area 5 is reached

 u1 = −1.810497e+01 x1 −1 x2+ 5.712905e+01
u2 = 0 x1 1.345392e−08 x2+ −2.000000e+01
Until area 4 is reached

 u1 = −2 x1 −1 x2+ 0
u2 = 0 x1 −2 x2+ 0
For 1.001000e+00 s, to stay in area 4
 Which is an accepting state.
```

```
Result of control synthesis for single agent motion planning with
    ↪ continous linear dynamics and MITL specifications.
------------------------------------------------------------------------
The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 5.500000e+00
 1 <= x2 <= 4

 The limits on u:
−20 <= u1 <= 20
 −20 <= u2 <= 20

 The original time calculation from Belta was used.

 MITL formula:
 Phi=(c U(<2.4) r1 and Ev(<10) A(<1) r5)
 ------------------------------------------------------------------------
 The shortest path in T which satisfies the MITL formula is:
 [ 5 2 3 2 5 4 ... ]
The corresponding time is: 1.387595e+00

 The run is achieved if the control input is given as:

 u1 = 1.313801e−08 x1 −9.151899e−10 x2+ −2.000000e+01
u2 = 5.333733e+00 x1 −2.191814e+01 x2+ 3.361528e+01
Until area 2 is reached

 u1 = −1.623397e+01 x1 6.820327e+00 x2+ 7.885062e+00
u2 = −8.180731e−08 x1 −1.171047e−07 x2+ 2.000000e+01
Until area 3 is reached

 u1 = −1.896748e+01 x1 6.687401e+00 x2+ 9.057378e+00
u2 = 3.249737e−10 x1 9.126561e−10 x2+ −2.000000e+01
Until area 2 is reached

 u1 = −1.210176e−08 x1 2.311528e−09 x2+ 2.000000e+01
```

```
u2 = 1.235554e+00 x1 −1.053272e+01 x2+ 1.905264e+01
Until area 5 is reached

 u1 = −1.147894e+01 x1 4.529017e+00 x2+ 2.017605e+01
u2 = 1.750129e−08 x1 7.616488e−09 x2+ −2.000000e+01
Until area 4 is reached

 u1 = −2 x1 −1 x2+ 0
u2 = 0 x1 −2 x2+ 0
For 1.001000e+00 s, to stay in area 4
 Which is an accepting state.
```

## A.2 Problem 2

Multi agent environment, where each agent follows the dynamics:

$$\dot{x} = Ax + Bu$$

Only local MITL formulas are considered in this section.

The results below consists of figures illustrating the partition and text-files with prints summarizing the result of the runs.

In this section, only the settings *u_joint=0* and *lin_ass=0* are considered, i.e. all simulations are performed using the separate control limit. $u_{min} \leq u_i \leq u_{max}, i = 1, 2$ and the full time calculation, allowing $u$ s.t. there is a cross-dependence in the closed-loop system.

### A.2.1 Final Result 1 - Sub-problem

```
Result of control synthesis for multi agent motion planning with
    ↪ continous linear dynamics and local MITL specifications.
————————————————————————————————————————————————————————————

The dynamics:
 A=
 2 1
 0 2

 B=
 1 0
 0 1

 The state space:
 1 <= x1 <= 11
 1 <= x2 <= 6

 The limits on u:
−30 <= u1 <= 30
 −30 <= u2 <= 30

 The original time calculation from Belta was used.

 MITL formula agent 1:
 Phi=(c U(<2.4) r1 and Ev(<10) A(<1) r5)
 ————————————————————————————————————————————————————————————


 MITL formula agent 2:
 Phi=Ev(<2.1) r2 and (r2 Imp Ev(<3.35) r6)
```

```
________________________________________________________________
 The shortest path in T which satisfies the MITL formula is:
 [ 3 8 13 12 11 6 11 12 13 18 17 ... ]
The corresponding time is: 1.511308e+000

 The run is achieved if the control input is given as:

 u1 = 6.640591e-016 x1 -2.664535e-015 x2+ 3.000000e+001
u2 = -1.706810e-001 x1 -3.341362e+000 x2+ 5.036129e+000
Until area 8 is reached

 u1 = -4.905764e-016 x1 -1.776357e-015 x2+ 3.000000e+001
u2 = -3.519371e-002 x1 -3.070387e+000 x2+ 3.887131e+000
Until area 13 is reached

 u1 = -3.368460e+000 x1 6.483049e-001 x2+ 2.485997e+000
u2 = -1.655697e-015 x1 -6.217249e-015 x2+ -3.000000e+001
Until area 12 is reached

 u1 = -2.563724e+000 x1 -9.993112e-001 x2+ 3.317241e+000
u2 = 3.552714e-015 x1 9.629329e-016 x2+ -3.000000e+001
Until area 11 is reached

 u1 = -6.661338e-016 x1 -3.836790e-015 x2+ -3.000000e+001
u2 = 9.680720e-002 x1 -3.466333e+000 x2+ 1.588743e+000
Until area 6 is reached

 u1 = -8.881784e-016 x1 5.275748e-015 x2+ 3.000000e+001
u2 = 1.696705e-001 x1 -3.693998e+000 x2+ 1.684986e+000
Until area 11 is reached

 u1 = -3.005742e+000 x1 -6.730883e-001 x2+ 5.886371e+000
u2 = 4.440892e-016 x1 6.579239e-016 x2+ 3.000000e+001
Until area 12 is reached

 u1 = -3.132946e+000 x1 -1.000000e+000 x2+ 6.164730e+000
u2 = 5.773160e-015 x1 -6.177135e-016 x2+ 3.000000e+001
Until area 13 is reached

 u1 = 2.220446e-016 x1 8.881784e-015 x2+ 3.000000e+001
u2 = 6.253635e-001 x1 -4.479410e+000 x2+ 4.811413e+000
Until area 18 is reached

 u1 = -2.536892e+000 x1 -1.013296e+000 x2+ 4.368068e+000
u2 = -1.332268e-015 x1 1.352729e-016 x2+ -3.000000e+001
Until area 17 is reached

 u1 = -2 x1 -1 x2+ 0
u2 = 0 x1 -2 x2+ 0
For 1.001000e+000 s, to stay in area 17
 Which is an accepting state.
________________________________________________________________________

 The shortest path in T which satisfies the MITL formula is:
 [ 8 13 12 11 16 11 12 13 18 23 22 ]
```

```
The corresponding time is: 5.348376e-001

 The run is achieved if the control input is given as:

 u1 = -4.905764e-016 x1 -1.776357e-015 x2+ 3.000000e+001
u2 = -3.519371e-002 x1 -3.070387e+000 x2+ 3.887131e+000
Until area 13 is reached

 u1 = -3.368460e+000 x1 6.483049e-001 x2+ 2.485997e+000
u2 = -1.655697e-015 x1 -6.217249e-015 x2+ -3.000000e+001
Until area 12 is reached

 u1 = -2.563724e+000 x1 -9.993112e-001 x2+ 3.317241e+000
u2 = 3.552714e-015 x1 9.629329e-016 x2+ -3.000000e+001
Until area 11 is reached

 u1 = 1.776357e-015 x1 5.077374e-015 x2+ 3.000000e+001
u2 = 2.374236e-001 x1 -4.358283e+000 x2+ 1.671165e+000
Until area 16 is reached

 u1 = -8.881784e-016 x1 -4.133854e-015 x2+ -3.000000e+001
u2 = 8.044358e-002 x1 -3.538918e+000 x2+ 1.588321e+000
Until area 11 is reached

 u1 = -3.005742e+000 x1 -6.730883e-001 x2+ 5.886371e+000
u2 = 4.440892e-016 x1 6.579239e-016 x2+ 3.000000e+001
Until area 12 is reached

 u1 = -3.132946e+000 x1 -1.000000e+000 x2+ 6.164730e+000
u2 = 5.773160e-015 x1 -6.177135e-016 x2+ 3.000000e+001
Until area 13 is reached

 u1 = 2.220446e-016 x1 8.881784e-015 x2+ 3.000000e+001
u2 = 6.253635e-001 x1 -4.479410e+000 x2+ 4.811413e+000
Until area 18 is reached

 u1 = -4.440892e-016 x1 1.483869e-014 x2+ 3.000000e+001
u2 = 8.143873e-001 x1 -5.541133e+000 x2+ 5.422689e+000
Until area 23 is reached

 u1 = -2.558031e+000 x1 -1.086697e+000 x2+ 5.869065e+000
u2 = -2.220446e-015 x1 -4.397248e-015 x2+ -3.000000e+001
Until area 22 is reached

 Which is an accepting state.
 ____________________________________________________________
```

### A.2.2 Final Result 2 - Sub-problem

```
Result of control synthesis for multi agent motion planning with
    continous linear dynamics and local MITL specifications.
____________________________________________________________

The dynamics:
 A=
```

```
 2  1
 0  2

 B=
 1  0
 0  1

 The state space:
 1 <= x1 <= 11
 1 <= x2 <= 6

 The limits on u:
-30 <= u1 <= 30
 -30 <= u2 <= 30

 The original time calculation from Belta was used.

 MITL formula agent 1:
 Phi=(c U(<2.4) r1 and Ev(<10) A(<1) r5)
 ----------------------------------------------------------------------

 MITL formula agent 2:
 Phi=Ev(<2.1) r2 and (r2 Imp Ev(<3.35) r6)
 ----------------------------------------------------------------------

 MITL formula agent 3:
 Phi=(Ev(<2.4) (r12 or r10)) and (Ev(<1)  r3)
 ----------------------------------------------------------------------
 The shortest path in T which satisfies the MITL formula is:
 [ 3 8 13 12 11 6 11 12 13 18 17 ... ]
The corresponding time is: 1.511308e+00

 The run is achieved if the control input is given as:

 u1 = -1.552440e-08 x1 -7.811712e-08 x2+ 3.000000e+01
u2 = 1.441441e+00 x1 -4.939352e+01 x2+ 1.705973e+02
Until area 8 is reached

 u1 = -6.621163e-09 x1 -7.075095e-08 x2+ 3.000000e+01
u2 = 8.003788e+00 x1 -2.617927e+01 x2+ 5.217158e+01
Until area 13 is reached

 u1 = -1.293190e+01 x1 1.050930e+01 x2+ 2.948725e+01
u2 = -3.803326e-12 x1 -4.572753e-12 x2+ -3.000000e+01
Until area 12 is reached

 u1 = -1.214231e+01 x1 5.589080e+00 x2+ 4.574371e+01
u2 = -3.339402e-08 x1 -2.764427e-07 x2+ -3.000000e+01
Until area 11 is reached

 u1 = 6.416204e-08 x1 -3.765347e-08 x2+ -3.000000e+01
u2 = 6.400864e+00 x1 -4.510022e+01 x2+ 2.901603e+01
Until area 6 is reached

 u1 = -1.013888e-10 x1 -4.641073e-10 x2+ 3.000000e+01
```

```
u2 = 9.534638e+00 x1 -3.707867e+01 x2+ 1.752058e+01
Until area 11 is reached

 u1 = -1.628175e+01 x1 3.510940e+00 x2+ 8.205440e+01
u2 = 6.886928e-09 x1 -3.950589e-09 x2+ 3.000000e+01
Until area 12 is reached

 u1 = -8.503144e+00 x1 5.444868e+00 x2+ 2.337304e+01
u2 = 3.604479e-09 x1 2.998415e-09 x2+ 3.000000e+01
Until area 13 is reached

 u1 = -9.469764e-08 x1 -1.283220e-07 x2+ 3.000000e+01
u2 = 9.978030e+00 x1 -2.904599e+01 x2+ 3.679800e+01
Until area 18 is reached

 u1 = -3.183032e+00 x1 -4.727966e-01 x2+ 7.592006e+00
u2 = 1.286954e-10 x1 -1.340543e-10 x2+ -3.000000e+01
Until area 17 is reached

 u1 = -2 x1 -1 x2+ 0
u2 = 0 x1 -2 x2+ 0
For 1.001000e+00 s, to stay in area 17
 Which is an accepting state.
 --------------------------------------------------------------------------------

 The shortest path in T which satisfies the MITL formula is:
 [ 8 13 12 11 16 11 12 13 18 23 22 ]
The corresponding time is: 5.348376e-01

 The run is achieved if the control input is given as:

 u1 = -6.621163e-09 x1 -7.075095e-08 x2+ 3.000000e+01
u2 = 8.003788e+00 x1 -2.617927e+01 x2+ 5.217158e+01
Until area 13 is reached

 u1 = -1.293190e+01 x1 1.050930e+01 x2+ 2.948725e+01
u2 = -3.803326e-12 x1 -4.572753e-12 x2+ -3.000000e+01
Until area 12 is reached

 u1 = -1.214231e+01 x1 5.589080e+00 x2+ 4.574371e+01
u2 = -3.339402e-08 x1 -2.764427e-07 x2+ -3.000000e+01
Until area 11 is reached

 u1 = -1.505153e-07 x1 -2.457128e-07 x2+ 3.000000e+01
u2 = 7.331081e+00 x1 -4.420762e+01 x2+ 2.218106e+01
Until area 16 is reached

 u1 = 2.206800e-08 x1 1.357898e-08 x2+ -3.000000e+01
u2 = 6.160808e+00 x1 -4.448092e+01 x2+ 1.723151e+01
Until area 11 is reached

 u1 = -1.628175e+01 x1 3.510940e+00 x2+ 8.205440e+01
u2 = 6.886928e-09 x1 -3.950589e-09 x2+ 3.000000e+01
Until area 12 is reached
```

```
 u1 = −8.503144e+00 x1 5.444868e+00 x2+ 2.337304e+01
u2 = 3.604479e−09 x1 2.998415e−09 x2+ 3.000000e+01
Until area 13 is reached

 u1 = −9.469764e−08 x1 −1.283220e−07 x2+ 3.000000e+01
u2 = 9.978030e+00 x1 −2.904599e+01 x2+ 3.679800e+01
Until area 18 is reached

 u1 = −1.572827e−10 x1 −1.897981e−10 x2+ 2.999999e+01
u2 = 6.582995e+00 x1 −2.893825e+01 x2+ 4.599290e+01
Until area 23 is reached

 u1 = −4.743308e+00 x1 3.453228e+00 x2+ 1.184958e+01
u2 = 2.013922e−08 x1 7.151107e−10 x2+ −3.000000e+01
Until area 22 is reached

 Which is an accepting state.
 ______________________________________________________________________

 The shortest path in T which satisfies the MITL formula is:
 [ 13 8 3 2 3 8 13 14 15 20 ]
The corresponding time is: 5.256829e−01

 The run is achieved if the control input is given as:

 u1 = 4.423518e−08 x1 4.312273e−08 x2+ −3.000000e+01
u2 = 9.898713e+00 x1 −2.561697e+01 x2+ 2.338056e+01
Until area 8 is reached

 u1 = −8.963128e−11 x1 2.185480e−10 x2+ −3.000000e+01
u2 = 5.992553e+00 x1 −1.600432e+01 x2+ 2.498311e+01
Until area 3 is reached

 u1 = −1.887466e+01 x1 9.248873e+00 x2+ 6.894093e+00
u2 = −2.976941e−08 x1 −1.171149e−08 x2+ −3.000000e+01
Until area 2 is reached

 u1 = −1.727069e+01 x1 8.390582e+00 x2+ 9.026890e+00
u2 = −6.595942e−10 x1 9.008105e−10 x2+ 3.000000e+01
Until area 3 is reached

 u1 = −1.552440e−08 x1 −7.811712e−08 x2+ 3.000000e+01
u2 = 1.441441e+00 x1 −4.939352e+01 x2+ 1.705973e+02
Until area 8 is reached

 u1 = −6.621163e−09 x1 −7.075095e−08 x2+ 3.000000e+01
u2 = 8.003788e+00 x1 −2.617927e+01 x2+ 5.217158e+01
Until area 13 is reached

 u1 = −1.136391e+01 x1 9.328376e+00 x2+ 2.288467e+01
u2 = 1.318471e−11 x1 1.685881e−11 x2+ 2.999999e+01
Until area 14 is reached

 u1 = −1.077632e+01 x1 9.664062e+00 x2+ 7.210303e+00
u2 = 5.600617e−11 x1 1.505471e−12 x2+ 2.999999e+01
```

```
Until area 15 is reached

 u1 = −2.250643e−11 x1  −1.220964e−10 x2+ 3.000000e+01
u2 = 5.444230e+00 x1  −2.187355e+01 x2+ 7.693138e+01
Until area 20 is reached

 Which is an accepting state.
```

### A.2.3 Final Result 3 - Full Problem

When simulating the full multi-agent problem, the smaller environment illustrated in figure 18 was used.

```
The path in the global product is : [ 321 3860 3563 4452 4811 4520 5383
    ↪ 5240 ]
The corresponding path in the global TBA is : [ 1 2 2 2 2 2 2 2 ]
The corresponding path in the BWTS product is : [ 161 310 162 606 786
    ↪ 640 1072 1000 ]
The corresponding path in each local product is : [ (5 17) (9 22) (5 18)
    ↪  (17 30) (22 30) (18 28) (30 28) (28 28) ]
The corresponding path in each local environment is : [ (2 5) (3 6) (2
    ↪ 5) (5 8) (6 8) (5 7) (8 7) (7 7) ]
The total time needed is 4.125309e−01
The time needed for respective transition is: [ 4.002136e−02 7.707535e
    ↪ −02 5.889152e−02 4.002135e−02 7.707534e−02 5.268026e−02 6.676570e
    ↪ −02 ]
```

TRITA TRITA-EE 2016:075
ISSN 1653-5146