Using progress sets on non-deterministic transition systems for multiple UAV motion planning

Paul Rousse, Pierre-Jean Meyer, Dimos Dimarogonas

Using progress sets on non-deterministic transition systems for multiple UAV motion planning ⋆

Paul Rousse * **Pierre-Jean Meyer** * **Dimos V. Dimarogonas** *

* *ACCESS Linnaeus Center and School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden (e-mail:*{rousse,pjmeyer,dimos}@kth.se*).*

Abstract: This paper presents a new approach for control synthesis of non-deterministic transition systems under Linear Temporal Logic (LTL) specifications with applications to multiple Unmanned Aerial Vehicles (UAV) motion planning problems. The consideration of such systems is motivated by the non-determinism possibly introduced while abstracting dynamical systems into finite transition systems. More precisely, we consider transition systems enhanced with a *progress set* describing the fact that the system cannot stay indefinitely in some subset of states. The control synthesis problem is firstly translated into a terminating planning problem. Then, a backward reachability strategy searches for a path from the initial set to the goal set. At each iteration, subsets of states contained in the progress set are added to the path, thus ensuring the reachability to the goal set in finite time. If a solution to the terminating problem is found, the obtained controller is translated back to the initial problem formulation. This approach is validated through an experiment involving two UAVs with a surveillance specification.

Keywords: Guidance, navigation and control of vehicles; Multi-vehicle systems; Formal control synthesis; Temporal logic specifications; Non-deterministic transition system.

1. INTRODUCTION

Robot motion planning with temporal logic specifications has received a lot of attention in the control community. More recently, the expressiveness of temporal specifications coupled with automated control synthesis methods brought powerful tools. Such tools can be used for provably correct control and planning design of robotic systems under high-level specifications in the form of temporal logic formulas (Belta et al., 2007).

The Linear Temporal Logic (LTL) specification language provides an expressive framework where safety, reachability and reactive properties are suitably formulated (Baier and Katoen, 2008). More importantly, they can be translated to finite automata (Clarke et al., 1999; Babiak et al., 2012) which makes relevant the use of computer science tools (e.g., graph search and fix point algorithms). In this work, we use LTL formulas that can be expressed as deterministic automata. Even if the expressiveness power is weaker than the whole LTL language, it still encapsulates a wide variety of specifications and is often used in control synthesis (Alur and La Torre, 2004; Fainekos et al., 2006).

Common approaches for control synthesis under high-level specifications involve the construction of an abstraction of the system. This abstraction is supposed to be used in place of a dynamical system and aims at reducing the overall complexity of the initial synthesis problem. If the system and the abstraction verify some behavioural relationship (see e.g., Tabuada, 2009), then the controller synthesized on the abstraction under the LTL specification can be refined into a controller such that the original system satisfies the same specification. Although some methods result in deterministic abstractions (Kloetzer and Belta, 2008b; Boskos and Dimarogonas, 2015), other abstraction approaches induce some non-determinism (i.e., a control input can induce several successors from the same initial state, see e.g., Moor and Raisch, 2002; Nilsson and Ozay, 2014). Non-deterministic transitions in the abstraction can arise due to the state space partitioning or because of initial disturbances of the continuous system. Methods to lower these effects exist, such as partitioning the state space according to the system flow (see e.g., Lafferriere et al., 2000; Tabuada, 2009) or designing local controllers that confine disturbances in each cell of the partition (see e.g., Kloetzer and Belta, 2008b). However, none of them can guaranty to get ride of the non-determinism (see e.g., Habets et al., 2006), and, in such cases, the non-deterministic nature of the system needs to be taken into account.

While control synthesis problems with deterministic models can easily be solved through a reformulation into a model checking problem (Clarke et al., 1999), the same approach cannot be applied with non-deterministic models (Kloetzer and Belta, 2008a). Fixed points methods have been widely used in correct-by-design control synthesis for non-deterministic models (see e.g., Cimatti et al., 2003; Kloetzer and Belta, 2008a). They determine the set of valid

⋆ This work was supported by the H2020 ERC Starting Grant BUCOPHSYS, the EU H2020 AEROWORKS project, the EU H2020 Co4Robots project, the Swedish Foundation for Strategic Research, the Swedish Research Council and the KAW Foundation.

controllers by trying to maximize some winning region which corresponds to the subspace of states where there exists a control strategy solving the synthesis problem. As it has been highlighted in Cimatti et al. (2003), such planner prevents any cycle (path of state-transition that starts and ends at the same state) from being part of a solution as they might keep the state away from the goal set forever. Other approaches use a *fairness assumption*: for any infinite run, every transition will be taken an infinite number of times (Cimatti et al., 2003; Fu et al., 2011). A direct consequence of this assumption is that every trajectory cannot stay indefinitely in a cyclic path, and thus, these cycles can be part of the solution plan. This fairness assumption is justified in probabilistic models where the probability of any transition is not zero. For this reason, it has been used in action planning where any action can be assigned to a success probability, and an infinite number of attempts will almost surely result in a success. Such probabilistic approaches cannot always hold in the case of control synthesis of a dynamical system: any transition from one state of the abstraction to itself might correspond to an equilibrium point in the original continuous system, and any cycle in the abstraction might correspond to a stable orbit of the system. Therefore the global fairness property is not granted.

Models with local fairness property have been investigated in De Giacomo et al. (2010) where the fairness assumption is modelled as an LTL formula $\square\diamond\varphi \Rightarrow \square\diamond\psi$ which stands for: "by trying φ infinitely often, ψ will happen infinitely often". This formulation can then be integrated in a general reactivity framework. In this approach, and contrary to fixed point techniques where cycles are all eliminated, or all accepted (when fairness property stands), only non-blocking cycles identified by the model can be part of the solution plan. Other common approaches in control synthesis have been using the quotient transition system in order to deal with cycles in motion planning: this abstraction suppresses any self-transition of the model and only considers a fragment of the LTL formulas (Tůmová et al., 2010; Pappas, 2003).

In this paper we propose a new method for control synthesis of non-deterministic transition systems under LTL specifications. We consider a non-deterministic augmented Finite Transition System (FTS) (firstly introduced for switching systems by Nilsson and Ozay, 2014). This model extends the standard FTS structure with the knowledge of a *progress set* that identifies local control strategies that are terminating (trajectories cannot stay indefinitely in the subset of states using the associated control inputs). We use this model as an abstraction for a dynamical system where the size of the progress set was large, and for this reason, the approach of De Giacomo et al. (2010), that efficiently deals with small progress sets, was not appropriate. We first introduce the model and the problem to be solved (Section 2). Then the product automaton of the model and the Büchi Automaton is defined and translated to a terminating planning problem (Section 3.1). A backward reachability algorithm is used (Section 3.3) to find a path through a terminating control strategy (Section 3.2) bringing the initial state deterministically in finite time to the goal set. Correctness and termination of the solutions are investigated. This approach is then validated in an experiment (Section 4) involving multiple Unmanned Aerial Vehicles (UAV).

2. PRELIMINARIES

2.1 Definitions

We denote by $\mathbb{N}$ the set of natural numbers and by $\mathbb{R}$ the set of real numbers. For the sets X, Y, we denote by $|X| \in \mathbb{N}$ the cardinality of X, 2^X its power set and X^ω the set of infinite words with elements chosen in X. For a word w of X^ω, $Inf(w) \subseteq X$ denotes the set of elements appearing infinitely often in w. For $Z \subseteq X \times Y$, $Z|_X \subseteq X$ denotes the projection of the set Z on the set X. Let $X \setminus Y$ be the set of elements of X not in Y. If $\mathbb{K}$ is $\mathbb{R}$ or $\mathbb{N}$, let $a, b \in \mathbb{K}$, $[a, b]_\mathbb{K}$ be the set $\{x \in \mathbb{K} \mid a \leq x \leq b\} \subset \mathbb{K}$.

In this work, we use a Finite Transition System (FTS) enhanced with a *progress set* adapted from Nilsson and Ozay (2014):

Definition 1. An Augmented Finite Transition System (AFTS) is a tuple defined by $\mathcal{T} = \langle S, S_0, U_\mathcal{T}, \delta_\mathcal{T}, \mathcal{P}_\mathcal{T} \rangle$ where:

- S is the set of states;
- $S_0 \subseteq S$ is the set of initial states;
- $U_\mathcal{T}$ is the input alphabet;
- $\delta_\mathcal{T} : S \times U_\mathcal{T} \to 2^S$ is the transition function;
- $\mathcal{P}_\mathcal{T} \subseteq 2^{S \times U_\mathcal{T}}$ is the progress set.

An *execution* of the AFTS $\mathcal{T}$ is an infinite sequence $r \in (S \times U_\mathcal{T})^\omega$ of state-control input pairs $r = \{(s_k, u_k)\}_{k \in \mathbb{N}}$ such that:

- $s_0 \in S_0$;
- $\forall k \in \mathbb{N}, s_{k+1} \in \delta_\mathcal{T}(s_k, u_k)$;
- $\forall m \in \mathcal{P}_\mathcal{T}, Inf(r) \not\subseteq m$.

The progress set $\mathcal{P}_\mathcal{T}$ identifies local control strategies that are terminating, i.e., any execution reaching an element $m \in \mathcal{P}_\mathcal{T}$ is guaranteed to exit m in finite time. Note that the condition $\forall m \in \mathcal{P}_\mathcal{T}, Inf(r) \not\subseteq m$ does not forbid any execution r to visit infinitely often an element $m \in \mathcal{P}_\mathcal{T}$ as long as the execution also leaves m infinitely often (i.e., if $Inf(r) \cap m \neq \emptyset$ and $Inf(r) \setminus m \neq \emptyset$ then $Inf(r) \not\subseteq m$). In the case of switching systems, Nilsson and Ozay (2014) introduced the progress set where each element is chosen in $2^S \times U_\mathcal{T}$. An element identifies a set of states $p_g \subseteq S$ (a progress group) and a control mode $u_g \in U_\mathcal{T}$, and any trajectory starting in the progress group p_g using the control mode u_g leaves p_g in finite time. In this paper, we do not consider control modes that are related to switching systems (control policy valid for a group of states) but a control strategy (state-control action pairs).

We say that $\rho \in S^\omega$ is a *run* of $\mathcal{T}$ if there exists an execution r of $\mathcal{T}$ such that $\rho = r|_{S^\omega}$. In the present work, we assume that the AFTS $\mathcal{T}$ is *well-formed* meaning that we have: $S_0 \neq \emptyset$; for every reachable state $s \in S$ there exists at least one control action leading to another state, i.e., $\exists u \in U_\mathcal{T}, |\delta_\mathcal{T}(s, u)| \geq 1$; and all elements of $\mathcal{P}_\mathcal{T}$ have an outgoing transition, i.e., $\forall m \in \mathcal{P}_\mathcal{T}, \exists (s, u) \in m, \delta_\mathcal{T}(s, u) \not\subseteq m|_S$.

The reader is referred to the dedicated literature about formal methods (such as Baier and Katoen, 2008) for a

formal definition of the LTL language. In summary, an LTL formula over a set S is defined inductively by:

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$$

where the logical operators $\top$ (*true*), $\neg$ (*negation*), $\wedge$ (*conjunction*), the temporal operators $\bigcirc$ (*next*) and U (*until*) are combined with elements $a \in S$ and LTL formulas φ_1 and φ_2. The useful operators $\vee$ (*disjunction*), $\Rightarrow$ (*implication*), $\Box$ (*always*) and $\Diamond$ (*eventually*) can be derived from the previous ones. These LTL formulas provide a user-friendly language to express specifications, for example "avoid area d, visit area c, and avoid area a until you reach area b" can be expressed by the following formula:

$$\varphi = (\Box\neg d) \wedge (\Diamond c) \wedge ((\neg a)\mathsf{U} b).$$

Every LTL formula over an input alphabet S can be translated into a Büchi Automaton (BA) (Clarke et al., 1999):

Definition 2. A Büchi Automaton is a tuple $\mathcal{A}_\varphi = \langle Q_\varphi, Q_{\varphi_0}, S, \delta_\varphi, \mathcal{F}_\varphi \rangle$ where:

- Q_φ is the finite set of states;
- $Q_{\varphi_0} \subseteq Q_\varphi$ is the set of initial states;
- S is the input alphabet;
- $\delta_\varphi : Q_\varphi \times S \to 2^{Q_\varphi}$ is the transition function;
- $\mathcal{F}_\varphi$ is the set of accepting states.

If Q_{φ_0} is a singleton and $|\delta_\varphi(q,a)| \in \{0,1\}$ for all $a \in S$ and all $q \in Q_\varphi$, then $\mathcal{A}_\varphi$ is called Deterministic Büchi Automaton (DBA). It is called a Non-deterministic Büchi Automaton (NBA) otherwise. In this paper, we use LTL specifications that can be represented with DBA. Such DBA can be generated using a NBA of a given LTL formula and applying determinization processes (Alur and La Torre, 2004; Babiak et al., 2012). However, this is not always achievable, and only a fragment of LTL formulas are translatable into DBA (see Theorem 4.50 in Baier and Katoen, 2008, pp. 190). Nonetheless, this fragment allows to express a wide range of specifications (Fainekos et al., 2006). For an infinite input word $w = w_1 w_2 w_3 ... \in S^\omega$, there exists a set of infinite state trajectories in $\mathcal{A}_\varphi$ produced by w that we denote as $\mathcal{R}_\varphi(w) \subseteq {Q_\varphi}^\omega$.

Definition 3. The input word $w \in S^\omega$ is said to be *accepted* by $\mathcal{A}_\varphi$ if there exists $r \in \mathcal{R}_\varphi(w)$ such that $Inf(r) \cap \mathcal{F}_\varphi \neq \emptyset$.

Note that the input alphabet S of $\mathcal{A}_\varphi$ is chosen as the set of states of $\mathcal{T}$ so that a run of $\mathcal{T}$ produces an input word for the BA $\mathcal{A}_\varphi$. A run r of $\mathcal{T}$ can then be accepted or not by $\mathcal{A}_\varphi$.

2.2 Problem statement

The problem considered in this paper is formulated as follows:

Problem 4. Given a non-deterministic and well-formed AFTS $\mathcal{T} = \langle S, S_0, U_\mathcal{T}, \delta_\mathcal{T}, \mathcal{P}_\mathcal{T} \rangle$ and a DBA $\mathcal{A}_\varphi = \langle Q_\varphi, Q_{\varphi_0}, S, \delta_\varphi, \mathcal{F}_\varphi \rangle$, find a control strategy such that all runs of the controlled system are accepted by $\mathcal{A}_\varphi$.

To satisfy the acceptance condition of $\mathcal{A}_\varphi$ (see Definition 3), the control strategy of the system $\mathcal{T}$ should produce trajectories in $\mathcal{A}_\varphi$ that reach the acceptance set $\mathcal{F}_\varphi$ in finite time. Therefore, in the case of an FTS (AFTS with no progress set, i.e., $\mathcal{P}_\mathcal{T} = \emptyset$), every control strategy creating cyclic trajectories in $\mathcal{A}_\varphi$ outside of the acceptance set $\mathcal{F}_\varphi$ cannot be a solution of the problem. For the AFTS $\mathcal{T}$, if these corresponding cyclic executions are in one element of the progress set $\mathcal{P}_\mathcal{T}$, then these cycles are escaped in finite time, and the controller might be a solution of Problem 4.

3. SOLUTION

Problem 4 is solved in several steps. First, we create the product automaton of $\mathcal{T}$ and $\mathcal{A}_\varphi$, and translate it into an equivalent formulation as a terminating planning problem (Section 3.1). Then, we identify control strategies on subsets of the state space which can reach, deterministically and in finite time, given sets of states (Section 3.2). Finally, we use these local terminating controllers to find a non-blocking control strategy starting the search from the goal set until the initial set is found (Section 3.3).

3.1 Terminating formulation

We first define the product automaton of a DBA and an AFTS:

Definition 5. The product automaton $\mathcal{A}_p$ of the DBA $\mathcal{A}_\varphi$ and of the AFTS $\mathcal{T}$ is defined by $\mathcal{A}_p = \mathcal{T} \otimes \mathcal{A}_\varphi = \langle Q_p, Q_{p0}, U_p, \delta_p, \mathcal{F}_p, \mathcal{P}_p \rangle$ where:

- $Q_p = S \times Q_\varphi$ is the set of states;
- $Q_{p0} = S_0 \times Q_{\varphi_0}$ is the set of initial states;
- $U_p = U_\mathcal{T}$ is the input set;
- $\delta_p : Q_p \times U_p \to 2^{Q_p}$ is the transition function defined by $(s', q') \in \delta_p((s,q),u)$ iff $s' \in \delta_\mathcal{T}(s,u)$, $q' \in \delta_\varphi(q,s)$;
- $\mathcal{F}_p = S \times \mathcal{F}_\varphi$ is the acceptance set;
- $\mathcal{P}_p \subseteq 2^{Q_p \times U_p}$ is the progress set defined for $m \in 2^{Q_p \times U_p}$ by $m \in \mathcal{P}_p \Leftrightarrow m|_{S \times U_\mathcal{T}} \in \mathcal{P}_\mathcal{T}$.

Note that the product automaton $\mathcal{A}_p$ is a Büchi Automaton augmented with the progress set of the AFTS $\mathcal{T}$. We say that an execution $r \in (Q_p \times U_p)^\omega$ of the product automaton is *valid* if $\forall m \in \mathcal{P}_p, Inf(r) \not\subseteq m$ (this condition is inherited from Definition 1 of the AFTS). A valid execution r of the product automaton is said to be *accepted* if the corresponding state trajectory $\rho = r|_{{Q_p}^\omega}$ satisfies $Inf(\rho) \cap \mathcal{F}_p \neq \emptyset$. An accepted execution of $\mathcal{A}_p$ thus corresponds to a valid execution of $\mathcal{T}$ that is accepted by $\mathcal{A}_\varphi$.

A controller of $\mathcal{A}_p$ is a map $\pi : Q_p \to U_p$ that associates to each state $p \in Q_p$ an *available control action* $u \in U_p(p)$ where $U_p(p) = \{u \in U_p \mid |\delta_p(p,u)| \geq 1\}$. The controller is said to be *closed* if every state reachable from the initial set Q_{p0} using the controller π is associated with a control action, and it is *terminating* if all the reachable states can reach the acceptance set $\mathcal{F}_p$ in finite time. Finding a controller for Problem 4 consists in finding a closed and terminating controller over $\mathcal{A}_p$. In what follows, this control problem is translated into a terminating formulation.

Proposition 6. (Theorem 4 in Patrizi et al., 2013). Let a non-deterministic product automaton $\mathcal{A}_p = \langle Q_p, Q_{p0}, U_p, \delta_p, \mathcal{F}_p, \mathcal{P}_p \rangle$, where $\mathcal{F}_p = \{g_1, ..., g_n\}$, $|\mathcal{F}_p| = n$, and $Q_p = \{p_1, ..., p_m, g_1, ..., g_n\}$, $|Q_p| = n + m$. Let $G_t^0 = \{g_1^0, .., g_n^0\}$ be a duplicate definition of $\mathcal{F}_p$ (with different names for the corresponding states so that $G_t^0 \cap \mathcal{F}_p = \emptyset$). The *terminating*

Fig. 1. Construction of $\mathcal{A}_t$ from $\mathcal{A}_p$. Similar node shape represents equivalent state sets. Similar line style represents equivalent transition function between sets.

formulation of $\mathcal{A}_p$ is denoted as $\mathcal{A}_t = \langle Q_t, Q_{t0}, U_t, \delta_t, G_t, \mathcal{P}_t \rangle$ such that:

- $Q_t = Q_p \cup G_t^0$ is the set of states;
- $Q_{t0} = Q_{p0} \cup G_t^0$ is the set of initial states;
- $U_t = U_p$ is an input alphabet;
- $\delta_t : Q_t \times U_t \to 2^{Q_t}$ is the transition function defined by $p' \in \delta_p(p,u) \Leftrightarrow p' \in \delta_t(\sigma(p),u)$;
- $G_t = \mathcal{F}_p$ is the goal set;
- $\mathcal{P}_t = \mathcal{P}_p$ is the progress set.

where $\sigma : Q_p \to Q_t$ is a function mapping $Q_p \setminus \mathcal{F}_p$ to $Q_t \setminus (G_t \cup G_t^0)$ and $\mathcal{F}_p$ to G_t^0, namely:

$$\begin{cases} \sigma(p) = p & \text{if } p \in Q_p \setminus \mathcal{F}_p \\ \sigma(g_j) = g_j^0 & \text{for } j \in [1,n]_{\mathbb{N}} \end{cases}$$

$\mathcal{A}_t$ is said to be *well-formed* if $\mathcal{A}_p$ is well-formed. A controller $\pi_t : Q_t \to U_t$ of $\mathcal{A}_t$ is said to be closed (resp. terminating) if the associated controller $\pi_p = \pi_t \circ \sigma$ of $\mathcal{A}_p$ is closed (resp. terminating).

The terminating formulation $\mathcal{A}_t$ of $\mathcal{A}_p$ is built by mapping all transitions from $\mathcal{F}_p$ to transitions from a duplicate set G_t^0 of $\mathcal{F}_p$ and by adding G_t^0 to the initial set Q_{t0}. Figure 1 illustrates the construction of the transition function δ_t and of the set of states Q_t of $\mathcal{A}_t$. $\mathcal{A}_p$ is supposed to be well-formed (see Problem 4), thus $\mathcal{A}_t$ is also well-formed which means that for every state $q \in Q_t \setminus G_t$ there exists an available control action $u \in U_t(q)$ where $U_t(q) = \{u \in U_t \mid |\delta_t(q,u)| \geq 1\}$. Let an *execution* of $\mathcal{A}_t$ be a finite or infinite sequence $r = \{(q_k, u_k)\}_{k<I}$ of length $I \in \mathbb{N} \cup \{\infty\}$ with elements chosen in $Q_t \times U_t$ such that:

- $q_0 \in Q_{t0}$;
- $\forall k < I, q_{k+1} \in \delta_t(q_k, u_k)$;
- $\forall m \in \mathcal{P}_t, Inf(r) \not\subseteq m$.

For a closed controller π_t, a π_t-execution corresponds to a sequence of $\mathcal{A}_t$ such that $r = \{(q_k, \pi_t(q_k))\}_{k \leq I}$ with $I \in \mathbb{N} \cup \{\infty\}$. From Proposition 6, we can now equivalently solve Problem 4 by trying to find a closed and terminating controller π_t for the system $\mathcal{A}_t$ such that all the π_t-executions of $\mathcal{A}_t$ reach the goal set G_t in finite time.

3.2 Terminating modules search

Any control strategy in $\mathcal{A}_t$ creating cyclic execution might produce runs that loop ad infinitum, hence keeping the state away from G_t forever. If, however, each of these potential cycles is included into elements of the progress set $\mathcal{P}_t$, then none of them can block the system indefinitely. Thus, a control strategy π_t in $\mathcal{A}_t$ can ensure the termination property if the following statement is true: π_t bring the state from terminating modules (defined in the sequel) to other terminating modules strictly closer to the goal set G_t. This section details the search of a terminating module that reaches, deterministically and in finite time, a given set of states.

Function PRECEDINGMODULEBFS($\mathcal{A}_t, g, W$)
Data: $\mathcal{A}_t$: the terminating problem formulation,
$g \subseteq Q_t$: a set of states to be reached,
$W \subseteq 2^{Q_t}$: a set of already visited sets of states.
Result: m if a valid module is found, $\emptyset$ otherwise

```
M ← {PrecedingSingletonModules(g)} ;    // modules to visit
M_v ← ∅ ;                               // visited modules
repeat
    m ← arg min_{x∈M}(|x|) ;            // smallest cardinality module
    M ← M \ {m} ;                       // Remove m from M
    M_v ← M_v ∪ {m} ;                   // mark m as visited
    if Post-bar(m) ⊆ g and (m ∈ P_t or Post(m) ⊆ g) and g ∪ m|_Q ∉ W then
        return m ;                      // terminating module found
    else
        M ← M ∪ (ExpandModule(g, m) \ M_v) ;   // Add the new modules to the set of modules to visit
    end
until |M| = 0;
return ∅ ;                              // no module found
```

Algorithm 1. Breadth-First Search (BFS) of the terminating modules

We first introduce some definitions. Let a *module* $m \subseteq Q_t \times U_t$ of $\mathcal{A}_t$ be a set of state-control input pairs such that $\forall (q,u) \in m, u \in U_t(q)$ and every state is associated to a unique control action ($\forall q \in m|_{Q_t}, |m \cap (\{q\} \times U_t)| = 1$). We call $\mathscr{M}$ the set of modules of $\mathcal{A}_t$. Then, let $Post : \mathscr{M} \to 2^{Q_t}$ be the function that associates to a module its successor states, i.e., for $m \in \mathscr{M}$, $Post(m) = \bigcup_{(q,u) \in m} \delta_t(q,u)$, let $\overline{Post} : \mathscr{M} \to 2^{Q_t}$ be defined as the set of the *outgoing* successor states of a given module, i.e., for $m \in \mathscr{M}$, $\overline{Post}(m) = Post(m) \setminus m|_{Q_t}$.

Definition 7. $m \in \mathscr{M}$ is a *terminating module* of $\mathcal{A}_t$ if all executions of $\mathcal{A}'_t = \langle Q_t, m|_{Q_t}, U_t, \delta_t, G_t, \mathcal{P}_t \rangle$ exit m in finite time, i.e., for every execution $r = \{(q_k, u_k)\}_{k<I}$, $I \in \mathbb{N} \cup \{\infty\}$ of $\mathcal{A}'_t$, there exists $i < I$ such that $q_i \notin m|_{Q_t}$, and $\forall k < i, (q_k, u_k) \in m$.

Note that each module $m \in \mathscr{M}$ in the progress set $\mathcal{P}_t$ or such that $Post(m) \cap m|_{Q_t} = \emptyset$ (i.e., without self-transitions) is terminating.

For a goal set $g \subseteq Q_t$, let PRECEDINGSINGLETONMODULES(g) $\subseteq \mathscr{M}$ be defined as the set of singleton modules with states chosen in $Q_t \setminus g$ that have one or more successors in g, namely:

$$\{(q,u)\} \in \text{PRECEDINGSINGLETONMODULES}(g) \Leftrightarrow \begin{cases} q \in Q_t \setminus g \\ \delta_t(q,u) \cap g \neq \emptyset \end{cases}$$

and, for a module $m \in \mathscr{M}$, let EXPANDMODULE(g, m) $\subseteq \mathscr{M}$ be defined by:

$$m' \in \text{EXPANDMODULE}(g,m) \Leftrightarrow \begin{cases} m'|_{Q_t} = m|_{Q_t} \cup (Post(m) \setminus g) \\ m \subset m' \end{cases}$$

EXPANDMODULE(g, m) is the set of possible expansions of m with states chosen in $Q_t \setminus g$.

Function BACKWARDSEARCH($\mathcal{A}_t$)
Data: $\mathcal{A}_t$: the terminating problem formulation
Result: K if a solution is found, $Fail$ otherwise
14 $L \leftarrow \{K_G\}$; // plans to visit
15 $L_v \leftarrow \emptyset$; // visited plans
16 **repeat**
17 $K \leftarrow \arg\max_{x \in L}(|x|)$; // biggest cardinality plan
18 **if** $Q_{t0} \subseteq \tilde{K}|_{Q_t}$ **then** // initial set found
19 **return** K ; // return valid plan
20 $L_v \leftarrow L_v \cup \{K\}$; // Add K to L_v
21 $W \leftarrow \{\tilde{l}|_{Q_t} \mid l \in L \cup L_v\}$; // set of visited sets of states
22 $m \leftarrow$ PRECEDINGMODULEBFS($\mathcal{A}_t, \tilde{K}|_{Q_t}, W$) ;
23 **if** $m \neq \emptyset$ **then**
24 $L \leftarrow L \cup \{K \cup \{m\}\}$; // add the new plan to L
25 **else**
26 $L \leftarrow L \setminus \{K\}$; // remove the plan from L
27 **end**
28 **until** $|L| = 0$;
29 **return** $Fail$; // no valid plan found

Algorithm 2. Backward reachability algorithm (BRA)

Algorithm 1 details the search of terminating modules that go, deterministically and in finite time, to a given subset g of Q_t. The search is implemented as a Breadth-First Search (BFS) over the set of possible terminating modules. For a set of winning regions $W \subseteq 2^{Q_t}$, $w \in W$ corresponds to a set of states where a winning control strategy (i.e., terminating and reaching the goal set G_t) has already been found (the next section will more specifically state the definition of W). The set `M` of modules to visit is initialized (line 1) with the preceding singleton modules of g (PRECEDINGSINGLETONMODULES(g)). At each iteration, one of the smallest modules m of `M` (smallest in terms of cardinality; lines 4, 5 and 6) is removed from `M` and added to the set of visited modules $\mathtt{M}_v$. All possible successor modules of m are obtained with EXPANDMODULE(g, m), and those that have not been visited added to the set `M` of modules to visit (line 10). Finally, the module m is returned if it verifies the following properties (line 7):

- $\overline{Post}(m) \subseteq g$: all the successor states of m are either going to the set of states $m|_{Q_t}$ or to the set g,
- $m \in \mathcal{P}_t$ or $Post(m) \subseteq g$: m is a terminating module,
- $g \cup m|_Q \notin W$: no control strategy has been found yet.

The first property ensures that any trajectory beginning from module m either stays in m or reaches g in finite time. The second one ensures that the trajectory actually leaves $m|_{Q_t}$. The first two properties combined together show that modules returned by Algorithm 1 deterministically reach the set of states g and are terminating. The last property ensures that the new set of reachable states has not been considered yet. If there is no module found, $\emptyset$ is returned (line 13).

3.3 Backward Reachability Algorithm

We are now able to find terminating modules that bring any execution of $\mathcal{A}_t$ deterministically and in finite time to a given set of states through Algorithm 1. The same algorithm can then be used to iteratively expand a winning region where a terminating and closed control strategy exists. Algorithm 2 implements this as a Depth-First Search (DFS) using a backward reachability strategy: preceding terminating modules are searched and added to a terminating controller starting from the goal set G_t until all initial states in Q_{t0} are found.

Let a *plan* $K \subseteq \mathscr{M} \cup \{m_0\}$ be a set of terminating modules of $\mathcal{A}_t$ (see Definition 7) and of:

$$m_0 = \{(g, \emptyset) \mid g \in G_t\} \tag{1}$$

a fake module comprised of each goal state associated to none of the control actions. Let $\tilde{K} = \bigcup_{k \in K} k$ be the corresponding set of state-control action pairs. L corresponds to the set of plans to visit and is initialized (line 14) with $K_G = \{m_0\}$. At each step, the highest cardinality plan K is selected from L (line 17). If the initial set belongs to $\tilde{K}|_{Q_t}$ (the set of states of plan K), then plan K is returned (line 19). Otherwise, a terminating module m preceding $\tilde{K}|_{Q_t}$ (line 22) is returned by Algorithm 1, and the new plan $K \cup \{m\}$ is added to the list L of plans to visit (line 24). Note that the module m is chosen to produce a plan with a set of reachable states that have not been created yet ($m|_{Q_t} \cup \tilde{K}|_{Q_t} \notin W$; line 7 of Algorithm 1). If there is no valid module found (lines 22 and 23), then plan K is removed from L (line 26). When the set L of plans to visit is empty (line 28), that means that among all the possible configurations of reachable states comprised of terminating modules, none of them could find a solution and a $Fail$ flag is returned (line 29).

Theorem 8. Let K be a valid plan returned by Algorithm 2 ($K \neq Fail$) and π_K be the controller associated with K defined by $\pi_K(q) = u$ for all $(q, u) \in \tilde{K} = \bigcup_{k \in K} k$. The controller π_K is closed and terminating.

Proof. Let K be a plan found from Algorithm 2, let $m_0, m_1, ...$ be the modules that compose the plan K indexed from the first one added (m_0, defined in (1)) to the last one ($m_{|K|-1}$), $K = \{m_0, m_1, ...\}$ with $m_0|_{Q_t} = G_t$. By construction of K (see Algorithm 1 line 7, and Algorithm 2 lines 21 and 22), for all $i \in [1, |K| - 1]_{\mathbb{N}}$:

$$\overline{Post}(m_i) \in \bigcup_{0 \leq j < i} m_j|_{Q_t} \tag{2}$$

$$m_i|_{Q_t} \cap \Big(\bigcup_{0 \leq j < i} m_j|_{Q_t} \Big) = \emptyset \tag{3}$$

Using (3) recursively, for all $i, i' \in [0, |K| - 1]_{\mathbb{N}}$ s.t. $i \neq i'$, we have:

$$m_i|_{Q_t} \cap m_{i'}|_{Q_t} = \emptyset. \tag{4}$$

As each module $m_i \in \mathscr{M}$, for $i \in [1, |K| - 1]_{\mathbb{N}}$, associates to each state of $m_i|_{Q_t}$ a unique control action (see the definition of $\mathscr{M}$ in Section 3.2), and thanks to (4), every state in $(\tilde{K} \setminus m_0)|_{Q_t}$ is associated to one and only one control input. Thus, the controller π_K is correctly defined in Theorem 8.

As for all $(q, u) \in \tilde{K} \setminus m_0$, $\delta_t(q, u) \subseteq \tilde{K}|_{Q_t}$ from (2) and as $Q_{t0} \subseteq \tilde{K}|_{Q_t}$ (Algorithm 2, line 18), the controller π_K is closed. As π_K is closed, there exists a π_K-execution $r = \{(q_k, u_k)\}_{k<I}$ with $I \in \mathbb{N} \cup \{\infty\}$ such that $(q_i, u_i) \in \tilde{K}$ for every $i < I$. From (4), for every $q \in \tilde{K}|_{Q_t}$ there exists a unique $j \in [0, |K| - 1]_{\mathbb{N}}$ such that $q \in m_j|_{Q_t}$. We define the

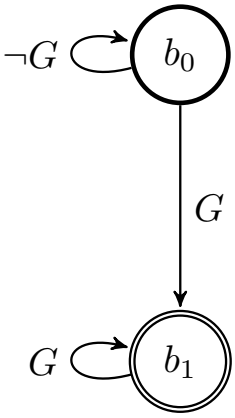

Fig. 2. Deterministic Büchi Automaton that represents the formula $\varphi = (\neg G)\mathsf{U}(\Box G)$. The acceptance set is $\mathcal{F}_\varphi = \{b_1\}$ and the initial set is $Q_{\varphi 0} = \{b_0\}$.

unique sequence of indexes $J = j_0 j_1 ...$ such that $q_i \in m_{j_i}$ for every $i < I$. As $\mathcal{A}_t$ is well-formed (Problem 4 and Property 6), there is always an available transition as long as m_0 is not reached. Therefore, we choose r to be infinite ($I = \infty$) iff m_0 is not reached. By definition of r and as $(q_0, u_0) \in m_{j_0}$, we have $q_1 \in \delta_t(q_0, u_0) \subseteq Post(m_{j_0})$. Then from (2), all the successors of m_{j_0} are in the modules of smaller indexes, consequently $j_1 \leq j_0$. This argument can be used for all $i < I$, meaning that J is a decreasing sequence. Moreover J is lower bounded by 0, therefore it is converging to a value j^* such that $0 \leq j^* < |K|$. Assuming $j^* > 0$ implies that $Inf(r) \subseteq m_{j^*}$. However, this invalidates the fact that m_{j^*} is a terminating module (line 7 of Algorithm 1). The only solution is $j^* = 0$ and $m_{j^*}|_{Q_t} = m_0|_{Q_t} = G_t$, which means that there exists $i \in \mathbb{N}$ such that $q_i \in G_t$, i.e., the goal set is reached in finite time, and therefore, the controller π_K is terminating. □

As the controller π_K is closed and terminating, the controller (after mapping with σ - previously defined by $\forall p \in Q_p \setminus \mathcal{F}_p, \sigma(q) = q$ and $\forall j \in [1, n]_\mathbb{N}, \sigma(g_j) = g_j^0$) is a valid solution of Problem 4.

Corollary 9. Let K be a valid plan returned by Algorithm 2 (i.e., $K \neq Fail$) and π_K be the controller associated to K defined by $\pi_K(q) = u$ for all $(q, u) \in \tilde{K}$. The controller $\pi_p = \pi_K \circ \sigma$ is a solution of Problem 4.

Remark 10. The termination of Algorithm 1 and 2 is ensured by two facts: the browsed state space is finite and the use of a visited set ($\mathtt{M}_v$ and L_v in Algorithm 1 and 2) avoids any element to be visited twice.

3.4 Illustration

Figure 3 illustrates how Algorithm 2 finds a solution plan. An agent starting in Q_0 needs to verify $\varphi = (\neg G)\mathsf{U}(\Box G)$, namely "reach and stay in G", using four given control actions (↖, ↗, ↙ and ↘). Note that φ can be represented by a DBA (e.g., see `ltl3ba`, Babiak et al., 2012, Figure 2 shows a DBA representing φ). The progress set $\mathcal{P}_\mathcal{T}$ is partially described by $\{a, b, c, d\} \subset \mathcal{P}_\mathcal{T}$. The transition system $\mathcal{T}$ is not explicitly defined and the reader is instead referred to Figure 3.

Due to the special form of the LTL formula φ, the terminating formulation $\mathcal{A}_t$ can be defined equivalently

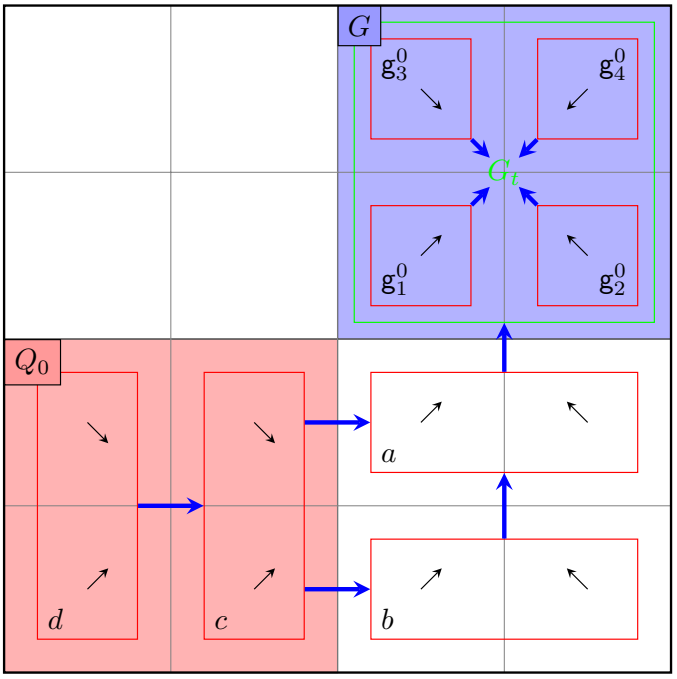

Fig. 3. Illustration of Algorithm 1 and 2. The agent goes from set Q_0 (in red) to reach and stay in set G (in blue) following the control actions (black arrows). Four control actions are available (↖, ↗, ↙ and ↘) and each of them points in the direction of the 3 successors of the considered state (square cell). Each red box corresponds to a terminating module and the green box corresponds to the goal set G_t of $\mathcal{A}_t$.

than in Section 3.1 by: the goal set $G_t = G$, the set $G_t^0 = \{g_1^0, ..., g_4^0\}$ duplicate of $G = \{g_1, ..., g_4\}$, the set of states $Q_t = S \cup G_t^0$, the initial set $Q_{t0} = Q_0 \cup G_t^0$, the progress set $\mathcal{P}_t = \mathcal{P}_\mathcal{T}$, and the transition function defined by: $s' \in \delta_\mathcal{T}(s, u) \Leftrightarrow s' \in \delta_t(\sigma(s), u)$ (where $\sigma : S \to Q_t$ is the mapping function similarly defined as in Proposition 6, i.e., $\sigma(s) = s$ iff $s \in S \setminus G_t$ and $\sigma(g_i) = g_i^0$ for $i \in [1, 4]_\mathbb{N}$). Note that in Figure 3, G_t and G_t^0 are superimposed however, we remind that $G_t \cap G_t^0 = \emptyset$.

Algorithm 2 is initialized with a plan $K_0 = \{m_0\}$ where m_0 is the fake goal module introduced in Section 3.3 (i.e, $m_0|_{Q_t} = G_t$, green box in Figure 3). At the first iteration, the plan K_0 is selected from the set $L = \{K_0\}$ of plans to visit. Algorithm 1 searches for a terminating module preceding the set of states $G_t = \tilde{K}_0|_{Q_t}$: e.g., $m_1 = \mathtt{g}_1^0 = \{(g_1^0, \nearrow)\}$ is returned as all successors $Post(m_1)$ of m_1 are in $\tilde{K}_0|_{Q_t}$. The new plan $K_1 = K_0 \cup \{m_1\}$ is added to L. At the next iteration of Algorithm 2 (and at each iteration in this example), the highest cardinality plan of L is the last one added to L. For the same reasons than for module m_1 with plan K_0, modules $m_2 = \mathtt{g}_2^0$, $m_3 = \mathtt{g}_3^0$ and $m_4 = \mathtt{g}_4^0$ are successively added to plans $K_2 = K_1 \cup \{m_2\}$, $K_3 = K_2 \cup \{m_3\}$ and $K_4 = K_3 \cup \{m_4\}$. At the 5^{th} iteration of Algorithm 2, K_4 is the biggest plan of $L = \{K_0, ..., K_4\}$ and Algorithm 1 searches for a terminating module going to $\tilde{K}_4|_{Q_t} = G_t \cup G_t^0$ deterministically and in finite time; there is no singleton terminating module, the smallest valid ones are composed of at least 2 elements, e.g., a is returned as it belongs to $\mathcal{P}_t$ and goes to $G_t \subset \tilde{K}_4|_{Q_t}$. $m_5 = a$ is added to $K_5 = K_4 \cup \{m_5\}$.

In the next iteration, the terminating module b is returned as it goes to $a|_{Q_t} \subset \tilde{K}_6|_{Q_t}$ and as $b \in \mathcal{P}_t$, then $m_6 = b$ is added to the plan $K_6 = K_5 \cup \{m_6\}$. $m_7 = c$ is added to $K_7 = K_6 \cup \{m_7\}$ as it goes deterministically to $a|_{Q_t} \cup b|_{Q_t} \subset \tilde{K}_6|_{Q_t}$ and as $c \in \mathcal{P}_\mathcal{T}$. Note that the transition between the modules is not deterministic ($\overline{Post}(c) \subseteq (a|_{Q_t} \cup b|_{Q_t})$). Finally, $m_8 = d$ is added to $K_8 = K_7 \cup \{m_8\}$, and as $d|_{Q_t} \cup c|_{Q_t} = Q_{t0} \subset \tilde{K}_8|_{Q_t}$, K_8 is a plan that brings all the trajectories starting in Q_{t0} to the goal set in finite time. Algorithm 2 returns the solution plan $K = K_8$. Every trajectory of the system $\mathcal{T}$ controlled by $\pi_p = \pi_K \circ \sigma$ (Theorem 8 and Corollary 9) verifies the specification φ.

4. EXPERIMENT

The experiment detailed below shows that the approach presented in this paper is appropriate for UAV motion planning scenarios. Multiple quadricopters (Iris Plus, 3DRobotics) are tracked with a motion capture system (Qualisys) and controlled from an offboard computer (using the Robot Operating System -ROS- framework). Algorithms are implemented in python, and the LTL translation to DBA is done using `ltl3ba` (Babiak et al., 2012).

Each quadricopter is modelled as a 2D first integrator system with additive disturbances. The multi-agent model is built as the concatenation of each single agent linear model, namely $\mathscr{S}$ is defined by:

$$\dot{x} = u + w$$

where $x \in \mathbb{R}^{2n}$, $n \in \mathbb{N}$ is the number of agents ($n = 2$ in our case), $u \in \mathcal{U}$ is the control input with $\mathcal{U} = \{-0.2, 0.2\}^{2n}$, and $w \in \mathcal{W}$ is a disturbance with $\mathcal{W} = [-0.1, 0.1]^{2n}_{\mathbb{R}}$. Such a system verifies a monotonicity property (Angeli and Sontag, 2003) which can be described in the following way: let $\phi(t, x, u, w)$ the state reached at time t from x under input u and disturbance w, if $x \leq x'$, $u(t) \leq u'(t)$ and $w(t) \leq w'(t)$ for all $t \geq 0$, then $\phi(t, x, u, w) \leq \phi(t, x', u', w')$ where $\leq$ is the component-wise inequality. For $x, y \in \mathbb{R}^{2n}$, we define the closed interval $[x, y] = \{z \in \mathbb{R}^{2n} \mid x \leq z \wedge z \leq y\} \subset \mathbb{R}^{2n}$. Let the following intervals of $\mathbb{R}^{2n}$ be defined by:

$$X = [\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}, \begin{bmatrix}1\\1\\1\\1\end{bmatrix}],\ X_a = [\begin{bmatrix}-1\\-1\\0.2\\0.2\end{bmatrix}, \begin{bmatrix}-0.2\\-0.2\\1\\1\end{bmatrix}],\ X_b = [\begin{bmatrix}0.2\\0.2\\-1\\-1\end{bmatrix}, \begin{bmatrix}1\\1\\-0.2\\-0.2\end{bmatrix}].$$

A discrete time system $\mathscr{S}_d$ is obtained by sampling the continuous system $\mathscr{S}$ at a sampling time $\Delta_T = 1.0s$. In order to build the FTS structure of $\mathcal{T}$, the state space of $\mathscr{S}_d$ is partitioned, let $\Xi : X \to S$ be the partitioning function which associates to each continuous state of $\mathscr{S}_d$ a symbol of $\mathcal{T}$. The set X is uniformly partitioned (with a step $x_d = 0.4$ for every dimension of $\mathbb{R}^4$) and $\mathbb{R}^4 \setminus X$ is mapped to a single symbol. As in Meyer (2015), a non-deterministic abstraction is created after the computation of reachable set over-approximations of $\mathscr{S}_d$ (using the monotonicity property). The set $\mathcal{P}_\mathcal{T}$ is defined by:

$$m \in \mathcal{P}_\mathcal{T} \Leftrightarrow \begin{cases} X_m \text{ is bounded and} \\ \min_{dx \in \mathscr{C}_m} \|dx\| > 0 \end{cases}$$

where $\mathscr{C}_m$ is the convex hull of the set $m|_{U_\mathcal{T}} + \mathcal{W}$ (with $+$ the Minkowski sum) and $X_m \subseteq X$ is the set of continuous states covering the symbols of $m|_{Q_t}$ (i.e., $X_m = \{x \in X \mid \Xi(x) \in m|_S\}$). For $m \in \mathcal{P}_\mathcal{T}$, as X_m is bounded and as the time average of the state displacement has a strictly positive norm, every trajectory of $\mathscr{S}_d$ starting in X_m will escape the set X_m in finite time (details are omitted due to space limitation). This guaranties the existence of some behavioural relationship (see Nilsson and Ozay, 2014) between the abstraction and the system $\mathscr{S}_d$. Hence, the control synthesis for the abstraction will be valid as well for the original continuous system.

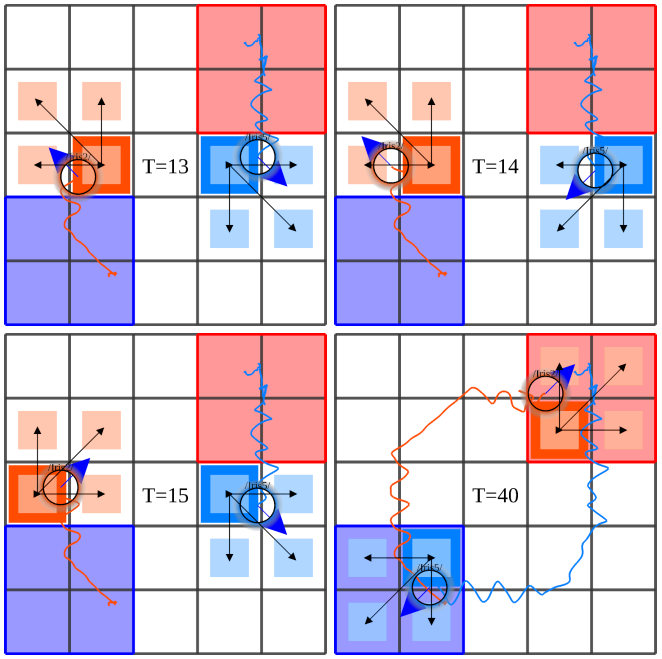

Fig. 4. Agent 1 (in blue) and agent 2 (in red) need to exchange position infinitely often. At timestep T, the current state of the agents (cell in plain color) is associated to a control input (thick blue arrows) and have multiple successors (thin black arrows and cells in light color).

For $x = [x_1, x_2, x_3, x_4]^\top$ the continuous state of the system, where $[x_1, x_2]^\top$ is the position of agent 1, and $[x_3, x_4]^\top$ the position of agent 2, we define the following atomic propositions: *out* $\Leftrightarrow x \notin X$, $a \Leftrightarrow x \in X_a$, $b \Leftrightarrow x \in X_b$, *collide* $\Leftrightarrow \max_{i \in \{1,2\}} |x_i - x_{i+2}| < 0.4$. In other words, the label a (resp. b) corresponds to agent 1 in region blue (resp. in region red) and agent 2 in region red (resp. in region blue; see Figure 4), *collide* corresponds to a situation where the quadricopters are too close and can collide, and the label *out* corresponds to one or more agents leaving the environment $X|_{\mathbb{R}^2}$.

The system must verify the following specification:

$$\varphi = (\square \neg \mathit{out}) \wedge (\square \neg \mathit{collide}) \wedge (\square \diamond a) \wedge (\square \diamond b).$$

Note that there exists a DBA translation of φ (e.g., see `ltl3ba -in` Babiak et al., 2012). The specification φ can be expressed in plain words as: "always stay inside the environment, always avoid collision, infinitely often go to a, infinitely often go to b".

We ran the experiment with real quadricopters using the controller π_p solution of Problem 4 found with Algorithm 2 and Corrolary 9. Figure 4 shows a few steps of the

experiment and a video is available at [1]. All states of $\mathcal{T}$ have non-deterministic transitions with 16 possible successors (for each control input). Modules of the solution plan K, obtained with Algorithm 2, have a cardinality of 4 when the module must be terminating for both agents and of 8 when the module is terminating for only one of the agents (e.g., if one of the agents has already reached its region and must wait for the other one to reach its own region). Agent 1 and agent 2 start in b. The discrete state of the system $p \in \mathcal{A}_p$ is initialized with the measurement of the continuous state x_0: $p = (\Xi(x_0), p_0) \in Q_{p_0}$. At each step, the continuous state x of the system is measured and mapped to a symbol $s = \Xi(x)$ of S, then the current discrete state $p \in Q_p$ of $\mathcal{A}_p$ is updated with the new state $p' = (t, b)$, unique element of $\delta_p(p, \pi_p(p))$ such that $t = s$. Steps at T = 13, 14 and 15 highlight the cyclic transitions of the AFTS in one of the terminating module of the plan K. Step T = 40 shows the end of the experiment where the agents reached a.

5. CONCLUSION

We proposed a solution to the control synthesis problem of a non-deterministic transition system under Linear Temporal Logic specifications that can be represented by Deterministic Büchi Automata. The considered system is modelled as a finite transition system enhanced with a progress set. Elements of the progress set identify local control strategies that are terminating (the state will escape a given set of states in finite time). Experiments with multiple UAVs show that this approach is relevant for world applications where the non-determinism of the system cannot be narrowed after the abstraction of it.

Only a fragment of LTL formulas are translatable into DBA, and if there is no deterministic translation possible, then the product automaton of the FTS and the NBA representing the formula is not fully observable (2 successors of the same state and input control might produce the same observation), and our approach is not valid anymore. How to handle such situations will be the topic of future works.

REFERENCES

Alur, R. and La Torre, S. (2004). Deterministic generators and games for LTL fragments. *ACM Transactions on Computational Logic (TOCL)*, 5(1), 1–25.

Angeli, D. and Sontag, E.D. (2003). Monotone control systems. *IEEE Transactions on automatic control*, 48(10), 1684–1698.

Babiak, T., Křetínskỳ, M., Řehák, V., and Strejček, J. (2012). LTL to büchi automata translation: Fast and more deterministic. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 95–109. Springer.

Baier, C. and Katoen, J.P. (2008). *Principles of Model Checking*. The MIT Press.

Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., and Pappas, G.J. (2007). Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1), 61–70.

Boskos, D. and Dimarogonas, D.V. (2015). Decentralized abstractions for feedback interconnected multi-agent systems. In *2015 54th IEEE Conference on Decision and Control (CDC)*, 282–287. IEEE.

Cimatti, A., Pistore, M., Roveri, M., and Traverso, P. (2003). Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1), 35 – 84.

Clarke, E.M., Grumberg, O., and Peled, D. (1999). *Model checking*. MIT press.

De Giacomo, G., Patrizi, F., and Sardina, S. (2010). Generalized planning with loops under strong fairness constraints. In *12th International Conference on the Principles of Knowledge Representation and Reasoning*.

Fainekos, G.E., Loizou, S.G., and Pappas, G.J. (2006). Translating temporal logic to controller specifications. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 899–904. IEEE.

Fu, J., Ng, V., Bastani, F.B., Yen, I.L., et al. (2011). Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, 1949.

Habets, L., Collins, P.J., and van Schuppen, J.H. (2006). Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6), 938–948.

Kloetzer, M. and Belta, C. (2008a). Dealing with nondeterminism in symbolic control. In *International Workshop on Hybrid Systems: Computation and Control*, 287–300. Springer.

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

Lafferriere, G., Pappas, G.J., and Sastry, S. (2000). O-minimal hybrid systems. *Mathematics of Control, Signals and Systems*, 13(1), 1–21.

Meyer, P.J. (2015). *Invariance and symbolic control of cooperative systems for temperature regulation in intelligent buildings*. Ph.D. thesis, Université Grenoble Alpes.

Moor, T. and Raisch, J. (2002). Abstraction based supervisory controller synthesis for high order monotone continuous systems. In *Modelling, Analysis, and Design of Hybrid Systems*, 247–265. Springer.

Nilsson, P. and Ozay, N. (2014). Incremental synthesis of switching protocols via abstraction refinement. In *53rd IEEE Conference on Decision and Control*, 6246–6253. IEEE.

Pappas, G.J. (2003). Bisimilar linear systems. *Automatica*, 39(12), 2035–2047.

Patrizi, F., Lipovetzky, N., and Geffner, H. (2013). Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *IJCAI*.

Tabuada, P. (2009). *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media.

Tůmová, J., Yordanov, B., Belta, C., Černá, I., and Barnat, J. (2010). A symbolic approach to controlling piecewise affine systems. In *49th IEEE Conference on Decision and Control (CDC)*, 4230–4235. IEEE.

[1] `https://youtu.be/yj0_olU1tYI`