Abstraction refinement and plan revision for control synthesis under high level specifications ⋆

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

* *ACCESS Linnaeus Center, School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden*
{pjmeyer ; dimos}@kth.se

Abstract: This paper presents a novel framework combining *abstraction refinement* and *plan revision* for control synthesis problems under temporal logic specifications. The control problem is first solved on a simpler *nominal* model in order to obtain a satisfying plan to be followed by the real system. A controller synthesis is then attempted for an abstraction of the real system to follow this plan. Upon failure of this synthesis, cost functions are defined to guide towards either refining the initially coarse partition to obtain a finer abstraction, or looking for an alternative plan using the nominal model as above. This tentative synthesis is then repeated until a plan and an abstraction of the real system able to follow this plan are found. The obtained controller also ensures that the real system satisfies the initial specification.

Keywords: Reachability analysis, verification and abstraction of hybrid systems; Abstraction refinement; Plan revision; Hybrid systems.

1. INTRODUCTION

In model checking and control synthesis problems under temporal logic specifications, when the desired specification is unsatisfiable by the considered system, classical methods would stop and announce that the problem is not feasible (Baier et al., 2008). To overcome this limitation, we can try to create an automated framework which iteratively reformulates or relaxes the problem until satisfaction is reached. Two main approaches can be considered.

The first option is to keep the desired specification while considering a new model which should satisfy it and remain as close as possible to the initial model (Ding and Zhang, 2005). A subset of these methods is based on the notion of *abstraction refinement*. When checking the satisfaction of the specification on the original model is too complicated, we can rely on creating an abstraction of this model which over-approximates its behavior while being simpler to deal with (Tabuada, 2009). As a result of this over-approximation, a specification satisfied on the abstraction will also be satisfied on the initial model, but its unsatisfaction may be due to the choice of a too coarse abstraction. Abstraction refinement thus aims at iteratively improving the accuracy of the abstraction until it satisfies the specification, see e.g. Clarke et al. (2003); Lee et al. (1997) for model checking and Henzinger et al. (2003); Nilsson and Ozay (2014) for control synthesis.

The dual approach consists in keeping the initial model of interest while tuning down the verification or control objective. This can be achieved by a *specification revision* problem, where one looks for a more permissive specification which is satisfied by the model (Kim et al., 2015; Finger and Wassermann, 2008). *Specification relaxation* is an alternative method where one designs some metric to measure the level of satisfaction of the initial unsatisfied specification and thus looks for a path with minimal violation (Tumova et al., 2013) or equivalently, maximal satisfaction (Guo and Dimarogonas, 2013) of the specification.

This paper addresses control synthesis under temporal logic specifications by combining *abstraction refinement* and *specification revision* approaches in a single framework composed of 3 main elements. We first apply a *specification conversion*, where a simpler *nominal* model is used to extract a plan ψ satisfying the main specification θ expressed as a Linear Temporal Logic (LTL) formula. When we fail to synthesize a satisfying controller for an abstraction of the real system to follow this plan ψ, we adapt the problem through either *abstraction refinement*, creating a more accurate abstraction by splitting elements of its partition; or *plan revision*, reusing the nominal model to synthesize an alternative plan ψ' satisfying the formula θ.

To the knowledge of the authors, the proposed approach combining abstraction refinement and plan revision has not been explored yet. In addition, the plan revision approach introduced in this paper is significantly different from the mentioned literature on specification revision and relaxation in that we do not assume the main specification to be unfeasible on the real system. A compositional approach of the abstraction refinement was presented in Meyer and Dimarogonas (2017).

The structure of this paper follows the above decomposition: Section 2 formulates the problem and details the

⋆ 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.

specification conversion, Section 3 presents the *abstraction refinement* and Section 4 introduces the *plan revision*. The overall algorithm is given in Section 5. Section 6 provides a numerical illustration of this approach.

2. PROBLEM FORMULATION

Let $\mathbb{N}$, $\mathbb{Z}^+$, $\mathbb{R}$ and $\mathbb{R}_0^+$ be the sets of positive integers, non-negative integers, reals and non-negative reals, respectively. For $a, b \in \mathbb{R}^n$, the interval $[a, b] \subseteq \mathbb{R}^n$ is defined as $[a, b] = \{x \in \mathbb{R}^n \mid a \leq x \leq b\}$ using componentwise inequalities. For a set S, $|S|$ denotes its cardinality.

2.1 System description

We consider a nonlinear control system described by

$$\dot{x} = f(x, w) + u, \tag{1}$$

with state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, bounded additive control input $u \in \mathcal{U} \subseteq \mathbb{R}^n$ and bounded disturbance input $w \in \mathcal{W} \subseteq \mathbb{R}^q$. $\Phi(t, x, u, \mathbf{w})$ denotes the state (assumed to exist and be unique) reached by (1) at time $t \in \mathbb{R}_0^+$ from initial state $x \in \mathcal{X}$, under the constant control input $u \in \mathcal{U}$ and the piecewise continuous disturbance input $\mathbf{w} : \mathbb{R}_0^+ \to \mathcal{W}$. The reachable set of (1) at time $t \in \mathbb{R}_0^+$, from a set of initial states $\mathcal{X}' \subseteq \mathcal{X}$ and for a subset of constant control inputs $\mathcal{U}' \subseteq \mathcal{U}$ is defined as

$$RS(t, \mathcal{X}', \mathcal{U}') = \left\{ \Phi(t, x, u, \mathbf{w}) \,\middle|\, \begin{array}{c} x \in \mathcal{X}',\ u \in \mathcal{U}', \\ \mathbf{w} : [0, t] \to \mathcal{W} \end{array} \right\}. \tag{2}$$

We assume that we are able to compute over-approximations $\overline{RS}(t, \mathcal{X}', \mathcal{U}')$ of the reachable set defined in (2):

$$RS(t, \mathcal{X}', \mathcal{U}') \subseteq \overline{RS}(t, \mathcal{X}', \mathcal{U}'). \tag{3}$$

Several methods exist for over-approximating reachable sets for fairly large classes of linear (Kurzhanskiy and Varaiya, 2007; Girard, 2005) and nonlinear systems (Reissig et al., 2016; Coogan and Arcak, 2015).

For a sampling period $\tau \in \mathbb{R}_0^+$ whose value is defined in the next section, a sampled version of system (1) can be described as a non-deterministic infinite transition system $S_\tau = (X_\tau, U_\tau, \underset{\tau}{\longrightarrow})$ where $X_\tau = \mathcal{X}$ is the set of states, $U_\tau = \mathcal{U}$ is the set of inputs and $x \overset{u}{\underset{\tau}{\longrightarrow}} x'$ (equivalently written as $x' \in Post_\tau(x, u)$) if $x' \in RS(\tau, \{x\}, \{u\})$.

2.2 Specification conversion

In this section, we define the initial problem and detail the *specification conversion* step mentioned in Section 1. We first assume that the state space $\mathcal{X} \subseteq \mathbb{R}^n$ is an interval of $\mathbb{R}^n$ and we consider a uniform partition P of $\mathcal{X}$ into smaller identical intervals. In what follows, the elements of P are called *cells* of the state space. Next, we consider a control specification θ written as a Linear Temporal Logic (LTL) formula over the set of atomic propositions corresponding to the elements of the partition P. The reader is referred to Baier et al. (2008) for an introduction on the LTL framework. In this paper, we focus on subclasses of LTL formulas which can be satisfied by finite traces $\psi = \psi(0)\psi(1)\dots\psi(r) \in P^{r+1}$ for some $r \in \mathbb{N}$, e.g. if θ is a syntactically co-safe formula (Kupferman and Vardi, 2001) or if it is defined over finite traces (De Giacomo and Vardi, 2013). In what follows, we denote as $\mathcal{F}(\theta)$ the set of all finite plans in P satisfying θ.

Problem 1. Find a controller $C : \mathcal{X} \to \mathcal{U}$ such that the closed loop of the system S_τ (with $x^{k+1} \in Post_\tau(x^k, C(x^k))$ for all $k \in \mathbb{Z}^+$) satisfies the specification θ.

Although our initial objective is to synthesize a controller such that the system S_τ satisfies the specification θ as in Problem 1, we rely on solving this control problem on a simplified model to define a secondary control problem for S_τ consisting in following one of the satisfying finite plans $\psi \in \mathcal{F}(\theta)$ in P. We thus temporarily consider the single integrator model $\Sigma_n : \dot{x} = u$ (in the same state space $\mathbb{R}^n$), typically used for motion of fully-actuated kinematic robotic agents (Mesbahi and Egerstedt, 2010) and such that (1) can be seen as a disturbed version (by disturbance w and state interactions) of the nominal system Σ_n.

Let $size(P, i) \in \mathbb{R}_0^+$ be the width in the i^{th} dimension of $\mathbb{R}^n$ of any cell in the uniform partition P and denote as $u_i \in \mathbb{R}$ the i^{th} component of $u \in \mathbb{R}^n$. Then the time

$$\tau = \max_{i \in \{1,\dots,n\}} \min_{u \in \mathcal{U}, u_i \neq 0} \frac{size(P, i)}{|u_i|} \tag{4}$$

corresponds to the minimal time such that steering any continuous state of Σ_n between any two neighbor cells of P (i.e. whose boundaries have a common facet) exactly in time τ can be done with a constant control u satisfying the constraints $u \in \mathcal{U}$. Using τ as a sampling period, we can then abstract the behavior of the *nominal* system Σ_n by a deterministic transition system. Since throughout this paper, the input used in a transition of this nominal abstraction is irrelevant to the control synthesis on the disturbed system (1), we rather consider the following non-deterministic finite transition system $S_n = (X_n, \underset{n}{\longrightarrow}, \sigma^0)$ defined without input, where $X_n = P$ is the set of states (cells of the partition P), $\sigma^0 \in P$ is the initial cell and $\sigma \underset{n}{\longrightarrow} \sigma' \Leftrightarrow \exists u \in \mathcal{U} \mid \forall x \in \sigma,\ x + \tau u \in \sigma'$.

The above definitions of τ and S_n thus ensure that for any two neighbor cells σ and σ' of P, the transition $\sigma \underset{n}{\longrightarrow} \sigma'$ exists in S_n. Similarly to $\mathcal{F}(\theta)$, we define $\mathcal{F}(S_n)$ as the set of all finite runs that can be generated by S_n, i.e. if $\psi \in P^{r+1} \cap \mathcal{F}(S_n)$, then $\psi(0) = \sigma^0$ and $\psi(k) \underset{n}{\longrightarrow} \psi(k+1)$ for all $k \in \{0, \dots, r-1\}$.

This *nominal* abstraction S_n will be used for both the initial problem conversion described below and the plan revision method in Section 4. Both these steps can be done using classical tools of model checking (Baier et al., 2008) to find a finite plan $\psi \in \mathcal{F}(\theta) \cap \mathcal{F}(S_n)$ satisfying the specification θ on S_n. Problem 1 can then be replaced by a new problem where the plan ψ is to be followed by S_τ.

Problem 2. Find a plan $\psi = \psi(0)\psi(1)\dots\psi(r) \in \mathcal{F}(\theta)$ satisfying the specification θ and a controller $C : \mathcal{X} \to \mathcal{U}$ such that the sampled system S_τ follows ψ, i.e. for any trajectory $x^0 \overset{C(x^0)}{\underset{\tau}{\longrightarrow}} x^1 \overset{C(x^1)}{\underset{\tau}{\longrightarrow}} \dots \overset{C(x^{r-1})}{\underset{\tau}{\longrightarrow}} x^r$ of the controlled system, we have $x^k \in \psi(k)$ for all $k \in \{0, \dots, r\}$.

Assumption 3. For any $k, l \in \{0, \dots, r\}$ such that $k \neq l$, we have $\psi(k) \neq \psi(l)$.

For clarity of notation, we assume that no finite plan $\psi = \psi(0)\psi(1)\dots\psi(r)$ as considered in Problem 2 visits the same cell twice, as provided by Assumption 3. The case when Assumption 3 is relaxed can be covered by designing controllers depending on both the current state of the system and the current position in ψ in order to know which cell is to be targeted next.

3. ABSTRACTION REFINEMENT

3.1 Abstraction

Since in most cases Problems 1 and 2 cannot be solved directly on the infinite transition system S_τ, we rely on creating an abstraction S_a of S_τ which can be described as a finite transition system $S_a = (X_a, U_a, \underset{a}{\longrightarrow})$ where X_a is a partition of the continuous state space $\mathcal{X}$ into a finite set of intervals called *symbols* (in the abstraction refinement procedure, X_a is initially taken equal to P and is then iteratively refined in Algorithm 2), $U_a \subseteq \mathcal{U}$ is a finite subset of the control set $\mathcal{U}$ and a transition $s \underset{a}{\xrightarrow{u}} s'$ (equivalently written as $s' \in Post_a(s, u)$) exists if $s' \cap \overline{RS}(\tau, s, \{u\}) \neq \emptyset$. The use of these over-approximations (3) guarantees the existence of a behavioral relationship between S_τ and S_a (defined formally and proven in Section 5), which ensures that a controller solving Problem 1 for S_a can be converted into a controller solving Problem 1 for S_τ.

Instead of creating the whole abstraction S_a as defined above followed by a controller synthesis (which may fail if the chosen partition is too coarse), the abstraction refinement proposed in this section is guided by the specification and iteratively synthesizes a controller alongside the creation of the abstraction. If no satisfying controller is found, an element of the initial coarse partition P is *refined* by splitting it into smaller elements and the synthesis is tried again. This approach thus aims at creating the abstraction S_a from a refined partition X_a which is as coarse as possible, but still fine enough to satisfy the specification.

3.2 Valid sets

We first introduce the function $P_a : P \to 2^{X_a}$ such that $P_a(\sigma) = \{s \in X_a \mid s \subseteq \sigma\}$ corresponds to the projection of a cell $\sigma \in P$ onto a given finer partition X_a. Next, we define the notion of *valid sets*.

Definition 4. Given a finite plan $\psi = \psi(0) \dots \psi(r)$ as in Problem 2, we define the function $V : P \to 2^{X_a}$ such that $V(\psi(r)) = \{\psi(r)\}$ and for all $k \in \{0, \dots, r-1\}$: $V(\psi(k)) = \{s \in P_a(\psi(k)) | \exists u \in U_a, Post_a(s, u) \subseteq V(\psi(k+1))\}$. The set $V(\psi(k))$ is called the *valid set* of cell $\psi(k)$. A cell $\sigma \in P$ and a symbol $s \in X_a$ such that $s \in P_a(\sigma)$ are said to be *valid* if $V(\sigma) \neq \emptyset$ and $s \in V(\sigma)$, respectively. Conversely, a symbol $s \in P_a(\sigma)$ is *invalid* if $s \notin V(\sigma)$.

Since $\psi(r)$ is the final cell of the plan ψ to be reached in Problem 2, it is considered as valid and the function $V : P \to 2^{X_a}$ is initialized with $V(\psi(r)) = \{\psi(r)\}$. We then proceed backwards on the plan ψ to iteratively define the other valid sets $V(\psi(k))$ as the subset of symbols in $\psi(k)$ which can be driven towards the valid set $V(\psi(k+1))$ of the next cell for at least one control input in U_a. The function $\texttt{ValidSet}(\psi(k), V(\psi(k+1)))$ in Algorithm 1 first computes the valid set $V(\psi(k))$ with respect to a plan ψ as in Definition 4. Then, the controller $C_a : X_a \to U_a$ associates to each valid symbol $s \in V(\psi(k))$ the *first* control value ensuring that s is valid (i.e. stopping the search of such inputs as soon as one is found).

Data: P, X_a, U_a, $P_a : P \to 2^{X_a}$.
Input: Considered cell $\psi(k) \in P$.
Input: Targeted valid set $V(\psi(k+1)) \subseteq P_a(\psi(k+1))$.
$V(\psi(k)) = \{s \in P_a(\psi(k)) \mid \exists u \in U_a$ such that $Post_a(s, u) \subseteq V(\psi(k+1))\}$
forall $s \in V(\psi(k))$ **do**
  $C_a(s)$ taken in $\{u \in U_a \mid Post_a(s, u) \subseteq V(\psi(k+1))\}$
return $\{V(\psi(k)),\ C_a : X_a \to U_a\}$

Algorithm 1: $\texttt{ValidSet}(\psi(k), V(\psi(k+1)))$. Computes the valid set $V(\psi(k))$ and associated controller C_a at step $k \in \{0, \dots, r-1\}$ of the plan $\psi = \psi(0)\psi(1) \dots \psi(r)$.

3.3 Refinement

As stated in Section 3.1, the abstraction S_a is initialized with respect to the initial coarse partition $X_a = P$. We then iteratively compute the valid sets $V(\psi(k))$ from $k = r$ back to $k = 0$ as in Definition 4. If an empty valid set $V(\psi(k)) = \emptyset$ is found for some step k (i.e. the associated controller synthesis in Algorithm 1 fails), the overall algorithm in Section 5 may choose to overcome this problem through abstraction refinement by calling the function $\texttt{Refine}(\psi, j)$ in Algorithm 2 in order to *refine* one of the previously visited cells $\psi(j)$ with $j \in \{k, \dots, r-1\}$. The rule guiding the choice of j is detailed in Section 5.

This refinement is achieved in the following two steps. Firstly, the cell $\psi(j)$ is *refined* by splitting each of its invalid symbols s into a set of subsymbols ($\forall s' \in \texttt{Split}(s)$, $s' \subseteq s$) and updating the partition X_a accordingly. The definition of $\texttt{Split}$ can be arbitrary. The second step consists in calling Algorithm 1 for all the cells of ψ whose valid sets may be expanded as a result of this refinement, i.e. from the refined cell $\psi(j)$ back to the cell $\psi(k)$ (with $k \leq j$) whose valid set was empty.

Data: P, X_a, $P_a : P \to 2^{X_a}$, $V : P \to 2^{X_a}$, $V(\psi(k)) = \emptyset$.
Input: Plan $\psi = \psi(0) \dots \psi(r)$.
Input: Step $j \in \{k, \dots, r-1\}$ of the refinement.
forall $s \in P_a(\psi(j)) \backslash V(\psi(j))$ **do**
  $X_a = (X_a \backslash \{s\}) \cup \texttt{Split}(s)$
for *l from j to k* **do**
  $\{V(\psi(l)),\ C_a\} = \texttt{ValidSet}\ (\psi(l), V(\psi(l+1)))$
return $\{X_a,\ V : P \to 2^{X_a},\ C_a : X_a \to U_a\}$
Algorithm 2: $\texttt{Refine}(\psi, j)$. Refinement of the cell $\psi(l)$ and update of the affected valid sets.

4. PLAN REVISION

4.1 Büchi and product automata

We first define a *Büchi automaton*, where the considered set of atomic propositions is the partition P.

Definition 5. A Büchi automaton $\mathcal{A} = (Q, P, \delta, q^0, F)$ is described by: a finite set of states Q, an input alphabet P, a transition relation $\delta : Q \times P \to 2^Q$, an initial state $q^0 \in Q$ and a set of accepting states $F \subseteq Q$. For an infinite word $\sigma^0\sigma^1\sigma^2 \dots$ over P, the associated run $q^0q^1q^2 \dots$ of $\mathcal{A}$ (such that $q^{i+1} \in \delta(q^i, \sigma^i)$ for all $i \in \mathbb{Z}^+$) is said to be *accepting* if it visits the accepting set F infinitely often.

Büchi automata are used as an alternative structure capturing the set of words that satisfy an LTL formula (Baier et al., 2008). Let $\mathcal{A}_\theta$ denote the Büchi automaton associated to the LTL formula θ in Problem 1. We can then consider the product of the transition system S_n and $\mathcal{A}_\theta$.

Definition 6. The product of $S_n = (P, \underset{n}{\longrightarrow}, \sigma^0)$ and $\mathcal{A}_\theta = (Q, P, \delta, q^0, F)$ is described by the automaton $\Pi = (Q_\Pi, \emptyset, \delta_\Pi, q_\Pi^0, F_\Pi)$ where: $Q_\Pi = P \times Q$; there is no input set (as in S_n); $\delta_\Pi : Q_\Pi \to 2^{Q_\Pi}$ and $(\sigma', q') \in \delta_\Pi((\sigma, q))$ if $\sigma \underset{n}{\longrightarrow} \sigma'$ and $q' \in \delta(q, \sigma)$; $q_\Pi^0 = (\sigma^0, q^0)$; and $F_\Pi = P \times F$.

Given a run $\omega = (\sigma^0, q^0)(\sigma^1, q^1)\dots$ of Π, we denote as $\omega|_{S_n} = \sigma^0\sigma^1\dots$ the projection of ω onto a run of S_n. From Definition 6, an accepting run ω of Π can thus be projected onto a run $\omega|_{S_n}$ of S_n satisfying the formula θ. Due to our focus on LTL formulas satisfiable in finite time, we assume that such accepting run is such that $\omega|_{S_n} \in \mathcal{F}(\theta) \cap \mathcal{F}(S_n)$ with the notations introduced in Section 2.2.

4.2 Iterative Deepening Search

We are interested in a search of Π allowing to be repeatedly called, each time returning a satisfying plan $\psi \in \mathcal{F}(\theta) \cap \mathcal{F}(S_n)$ which was not previously returned. The first call is the *specification conversion* as in Section 2.2 and follow-up calls are iterations of the *plan revision*. More precisely, a call of `Revise`(ψ, j) as in Algorithm 3 aims at finding an admissible revision of ψ up to $\psi(j)$, i.e. a new satisfying plan ψ' ending with the sequence $\psi(j+1)\dots\psi(r)$. Such a revision thus needs to satisfy the following three conditions. Since the search is done on the product automaton Π from its initial state q_Π^0, the first condition $\psi' \in \mathcal{F}(\theta) \cap \mathcal{F}(S_n)$ can be reduced to checking whether the explored path in Π ends with an accepting state. The second condition is $\psi' \in AdmRev(\psi, j)$ where $AdmRev(\psi, j) = \{\sigma^0 \dots \sigma^p \psi(j+1) \dots \psi(r) \mid \sigma^p \neq \psi(j)\}$ ensures that the end sequence $\psi(j+1)\dots\psi(r)$ is kept in ψ' while forcing the revision to start in $\psi(j)$. Denoting as $UsedPlans \subseteq \mathcal{F}(\theta) \cap \mathcal{F}(S_n)$ the set of plans previously considered and discarded in the main algorithm of Section 5, the third condition preventing reusing these discarded plans is combined with the above second condition by considering $\psi' \in NewRev(\psi, j)$ where $NewRev(\psi, j) = AdmRev(\psi, j) \backslash UsedPlans$.

Algorithm 3 implements the search algorithm on Π as an Iterative Deepening Depth-First Search (Korf, 1985), consisting in calling a Limited-Depth Depth-First Search (function `LDDFS` defined recursively in Algorithm 4) with iteratively increasing depth limit until a satisfying plan is found. Intuitively, the function `Revise` initially searches for admissible revisions as above of length 1 (search depth limited to 0). If none is found, this limited-depth search is repeated with an allowed search depth increased by 1.

Data: Initial state q_Π^0 of Π.
Input: Current plan $\psi = \psi(0)\dots\psi(r) \in \mathcal{F}(\theta) \cap \mathcal{F}(S_n)$.
Input: Revision step $j \in \{0, \dots, r\}$.
for *depth from* 0 *to* ∞ **do**
  $\psi' =$ `LDDFS`$(q_\Pi^0, depth)$
  if $\psi' \neq \emptyset$ **then return** $\{\psi',\ depth - r + j\}$;

Algorithm 3: `Revise`(ψ, j). Generates a possible revision ψ' of ψ keeping its end sequence from $j+1$ up to r, and provides the index of the cell in ψ' replacing $\psi(j)$.

The Limited-Depth Depth-First Search is initialized in Algorithm 3 to start from the initial state q_Π^0 of Π with a depth limit denoted as $depth \in \mathbb{Z}^+$. It then proceeds in Algorithm 4, where a successor q_Π^1 of q_Π^0 is chosen and the function `LDDFS` is called again with the new explored path $q_\Pi^0 q_\Pi^1$ and an allowed depth reduced to $depth - 1$. This recursive call is repeated until the allowed depth reaches 0, where we check if the explored path generates an admissible revision (accepting path in Π whose projection onto a plan of S_n is in $NewRev(\psi, j)$). This plan is returned to Algorithm 3 if it is an admissible revision. Otherwise, the search backtracks and explores other paths of Π. Since this search has a limited depth and is applied to the finite graph Π, it will explore all paths of length $depth+1$ in Π in finite time if no admissible revision is found. In such cases, an empty set is returned to Algorithm 3 which will repeat the search with an increased depth limit.

Data: Π, S_n, ψ, j, $NewRev(\psi, j)$.
Input: Explored path $q_\Pi^0 \dots q_\Pi^l$ in Q_Π.
Input: Remaining allowed search depth: $depth \in \mathbb{Z}^+$.
if $depth > 0$ **then**
  forall $q_\Pi^{l+1} \in \delta_\Pi(q_\Pi^l)$ **do**
    $candidate =$ `LDDFS`$(q_\Pi^0 \dots q_\Pi^l q_\Pi^{l+1}, depth - 1)$
    if $candidate \neq \emptyset$ **then return** $candidate$;
else if $q_\Pi^l \in F_\Pi$ *and* $(q_\Pi^0 \dots q_\Pi^l)|_{S_n} \in NewRev(\psi, j)$ **then**
  return $(q_\Pi^0 \dots q_\Pi^l)|_{S_n}$
return $\emptyset$

Algorithm 4: `LDDFS`$(q_\Pi^0 \dots q_\Pi^l, depth)$. Recursive implementation of a limited-depth search.

Once Algorithm 3 receives a plan $\psi' \neq \emptyset$ from Algorithm 4, it returns this plan as well as the index corresponding to the cell of ψ' that replaces $\psi(j)$. For the *specification conversion* in Section 2.2, the initial plan is obtained from a first call of Algorithm 3 denoted as `Revise`$(\emptyset, 0)$ (since no previous plan is to be revised) and where the condition $\psi \in NewRev(\emptyset, 0)$ in Algorithm 4 is always true.

5. OVERALL APPROACH

5.1 Algorithm

Algorithm 5 is an implementation of the overall approach relying on the functions `ValidSet`, `Refine` and `Revise` in Algorithms 1 to 3. The refined partition X_a of the abstraction S_a is initialized with the partition P, an initial plan $\psi = \psi(0)\dots\psi(r)$ is obtained from Algorithm 3 for the *specification conversion* and the last cell $\psi(r)$ of the plan ψ is valid as in Definition 4. The main loop then aims at computing the valid sets and associated controllers as in Algorithm 1 for all cells $\psi(k)$ from $k = r-1$ back to $k = 0$.

If a non-empty valid set $V(\psi(k))$ is obtained, the while loop proceeds with step $k-1$. Otherwise ($V(\psi(k)) = \emptyset$ for some k), Algorithm 5 needs to pick a method between the *abstraction refinement* and the *plan revision*, as well as one of the previously explored cells $\psi(j)$ (with $j \in \{k, \dots, r\}$) where this method should be applied. This choice is made by introducing two cost functions $J_{AR}, J_{PR} : P \to \mathbb{R}^+$ associating the cost of applying either method to each cell of the plan ψ respectively. These functions can be chosen arbitrarily in order to prioritize one method over the other.

We first compute the indices j_{AR} and j_{PR} corresponding to the cells of ψ minimizing the costs $J_{AR}(\psi(j_{AR}))$ and $J_{PR}(\psi(j_{PR}))$, respectively. If the abstraction refinement offers the smallest cost ($J_{AR}(\psi(j_{AR})) < J_{PR}(\psi(j_{PR}))$), Algorithm 2 is called on cell $\psi(j_{AR})$ and the next step of the while loop proceeds without updating k, to check if $V(\psi(k))$ is still empty. Otherwise, the plan revision is applied on cell $\psi(j_{PR})$, where we first reset the valid sets for all cells that will be discarded by the revision, add ψ to the set $UsedPlans$ from Section 4.2 and call Algorithm 3 to obtain the new plan (overwriting ψ) and the associated index k such that $V(\psi(k))$ is to be computed next. The main loop is repeated until $V(\psi(0)) \neq \emptyset$ for some plan ψ. Algorithm 5 outputs the final plan ψ, the refined partition X_a, the valid symbols (in X_a) and the controller C_a.

Data: P, $J_{AR} : P \to \mathbb{R}^+$, $J_{PR} : P \to \mathbb{R}^+$.
Initialization: $X_a = P$, $\psi(0) \dots \psi(r) = \texttt{Revise}(\emptyset, 0)$
Initialization: $V(\psi(r)) = \{\psi(r)\}$, $k = r - 1$
while $k \geq 0$ **do**
  $\{V(\psi(k)),\ C_a\} = \texttt{ValidSet}\ (\psi(k), V(\psi(k+1)))$
  if $V(\psi(k)) \neq \emptyset$ **then** $k = k - 1$;
  else
    $j_{AR} = \arg\min_{j \in \{k,\dots,r-1\}} J_{AR}(\psi(j))$
    $j_{PR} = \arg\min_{j \in \{k,\dots,r\}} J_{PR}(\psi(j))$
    if $J_{AR}(\psi(j_{AR})) < J_{PR}(\psi(j_{PR}))$ **then**
      $\{X_a,\ V,\ C_a\} = \texttt{Refine}(\psi, j_{AR})$
    else
      forall $l \in \{k, \dots, j_{PR}\}$ **do** $V(\psi(l)) = \emptyset$;
      $UsedPlans = UsedPlans \cup \{\psi\}$
      $\{\psi,\ k\} = \texttt{Revise}(\psi, j_{PR})$

Output: $\{\psi,\ X_a,\ V : P \to 2^{X_a},\ C_a : X_a \to U_a\}$

Algorithm 5: Global algorithm.

5.2 Solution to Problem 1

To control the sampled system S_τ with the controller C_a obtained in Algorithm 5, systems $S_\tau = (X_\tau, U_\tau, \underset{\tau}{\longrightarrow})$ and $S_a = (X_a, U_a, \underset{a}{\longrightarrow})$ must satisfy the feedback refinement relation defined below, adapted from Reissig et al. (2016).

Definition 7. A map $H : X_\tau \to X_a$ is a feedback refinement relation from S_τ to S_a if: $\forall x \in X_\tau, s = H(x)$, $\forall u \in U_a \subseteq U_\tau, \forall x' \in Post_\tau(x, u), H(x') \in Post_a(s, u)$.

By finding such a relation, we obtain that Problem 1 can be solved if Algorithm 5 terminates in finite time [1].

Theorem 8. Let $H : X_\tau \to X_a$ such that $H(x) = s \Leftrightarrow x \in s$. Then the controller $C : X_\tau \to U_\tau$ defined by $C(x) = C_a(H(x))$ for all $x \in X_\tau$ solves Problem 1.

6. NUMERICAL ILLUSTRATION

The use of intervals as the elements of the state partition (required by the extraction of a plan ψ satisfying the main specification θ in Section 2.2) particularly suits the computation of over-approximations of the reachable set using the *monotonicity* property. The reader is referred to Angeli and Sontag (2003) for a description of monotone control systems and to e.g. Meyer (2015) for their use to over-approximate the reachable set and create abstractions. We consider a 2D system described by the nonlinear monotone dynamics: $\dot{x} = \begin{pmatrix} -1 & 0.3 \\ 0.3 & -1 \end{pmatrix} x - 0.01x^3 + u$, with state $x \in \mathbb{R}^2$, control $u \in [-5, 5]^2$ and componentwise cubic power x^3.

The considered state space $\mathcal{X} = [-9, 9]^2$ is partitioned into 3 elements per dimension, thus resulting in a partition P of 9 cells. The control interval $\mathcal{U} = [-5, 5]^2$ is discretized uniformly into 5 values per dimension: $U_a = \{-5, -2.5, 0, 2.5, 5\}^2$. Following the guidelines in (4), we take the sampling period $\tau = 6/5 = 1.2$. Below, the cells of the partition P are denoted as $\sigma_{x,y}$ with $x, y \in \{1, 2, 3\}$ such that for example, $\sigma_{1,3}$ represents the top-left cell in Figure 1. The *nominal* abstraction S_n is created such that each cell of P has a transition towards its immediate neighbors (but not in diagonal), and the initial cell is $\sigma_{1,1}$.

The main control specification is taken as the syntactically co-safe LTL formula $\theta = \diamond\sigma_{1,3}$, meaning that we want to eventually reach the top-left cell of P. The corresponding Büchi automaton $\mathcal{A}_\theta$ and the product Π of S_n and $\mathcal{A}_\theta$ are computed with the software P-MAS-TG described in Guo and Dimarogonas (2015). The remaining implementation of Algorithm 5 is done on Matlab.

The cost functions $J_{AR}, J_{PR} : P \to \mathbb{R}^+$ are an estimate of the complexity of the future computations after applying either abstraction refinement or plan revision on a cell of P. This complexity is measured in the number of symbols $s \in X_a$ whose set of successors $Post_a(s, u)$ needs to be computed for some $u \in U_a$. Assuming that $V(\psi(k)) = \emptyset$ for some plan ψ in Algorithm 5, then a call $\texttt{Refine}(\psi, j)$ for $j \in \{k, \dots, r-1\}$ is associated with the cost

$$J_{AR}(\psi(j)) = 2^n * |P_a(\psi(j)) \backslash V(\psi(j))| \qquad (5)$$
$$+ \sum_{l=k}^{j-1} |P_a(\psi(l)) \backslash V(\psi(l))| + (k+1) * (2^n)^2,$$

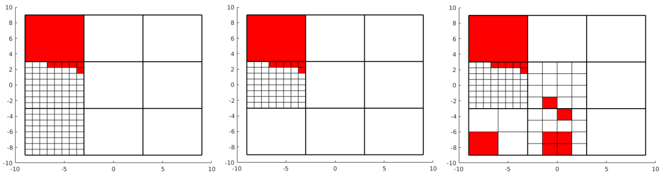
where the first term is the number of subsymbols obtained after splitting the invalid symbols of $\psi(j)$ (assuming that the function $\texttt{Split}(s)$ returns $2^n = 4$ subsymbols of s), the second term is all the invalid symbols of $\psi(j-1)$ back to $\psi(k)$ to be updated as in Algorithm 2 and the last term is a forecast that all invalid cells $\psi(0)$ to $\psi(k)$ that remains to be explored will be refined twice (assuming no plan revision is called in the future). The cost for calling $\texttt{Revise}(\psi, j)$ is defined similarly to the third term of (5):

$$J_{PR}(\psi(j)) = (|\texttt{Revise}(\psi, j)| - |\psi| + j + 1) * (2^n)^2 / 0.6, \qquad (6)$$

where the number of cells that remains to be explored now depends on the size difference between the current plan ψ and the candidate revision obtained in $\texttt{Revise}(\psi, j)$. The weight $1/0.6$ is added in (6) to prioritize the use of abstraction refinement over plan revision.

Figure 1 provides 3 snapshots of the refined partition X_a and the valid symbols (filled in red) during the execution of Algorithm 5. The initialization with $\texttt{Revise}(\emptyset, 0)$ provides a first plan $\psi = \sigma_{1,1}\sigma_{1,2}\sigma_{1,3}$ and the whole cell $\sigma_{1,3}$ (top-left in Figure 1a) is a valid symbol since it has no successor in ψ. Algorithm 5 then computes the valid set of $\psi(1) = \sigma_{1,2}$ which is found to be empty until 3 successive calls of $\texttt{Refine}(\psi, 1)$, where $|V(\sigma_{1,2})| = 6$. Then, $\psi(0) = \sigma_{1,1}$ is considered and its valid set remains empty despite 3

[1] The proof can be found in the extended version of this paper: https://hal.archives-ouvertes.fr/hal-01491845



(a) Refinement on the initial plan. (b) Revision of the plan. (c) Refinement on the revised plan.

Fig. 1. Refined partition and valid symbols (in red) for an execution of Algorithm 5.

calls of Refine$(\psi, 0)$. At this point, displayed in Figure 1a, Algorithm 5 considers it less costly to revise the plan ψ rather than refining a fourth time $\sigma_{1,1}$ or $\sigma_{1,2}$. The function Revise$(\psi, 0)$ is called to change the beginning sequence of ψ while keeping the end $\sigma_{1,2}\sigma_{1,3}$ (Figure 1b), resulting in a revision $\psi' = \sigma_{1,1}\sigma_{2,1}\sigma_{2,2}\sigma_{1,2}\sigma_{1,3}$. Algorithm 5 then proceeds on $\psi'(2) = \sigma_{2,2}$ whose valid set contains one symbol after two calls of Refine$(\psi', 2)$, then on $\psi'(1) = \sigma_{2,1}$ where $|V(\psi'(1))| = 5$ after two calls of Refine$(\psi', 1)$ and finally on $\psi'(0) = \sigma_{1,1}$ where $|V(\psi'(0))| = 1$ after one call of Refine$(\psi', 0)$, thus ending the algorithm (after 25.8s on a laptop with a 2.6 GHz CPU and 8 GB of RAM) with the final partition and valid symbols as in Figure 1c.

7. CONCLUSION

In this paper, we presented a novel framework combining *abstraction refinement* and *plan revision* for control synthesis problems under temporal logic specifications. The control problem is first solved on a simpler *nominal* model in order to obtain a satisfying plan to be followed by the real system. We then try to synthesize a controller for an abstraction of the real system to follow this plan. When this synthesis fails, some cost functions guide us towards either refining the initially coarse partition to obtain a finer abstraction, or looking for an alternative plan using the nominal model as above. This tentative synthesis is then repeated until we find a plan and an abstraction of the real system able to follow this plan. The controller obtained on this abstraction can then be used to define a controller for the real system to satisfy the initial specification.

While this paper mainly focuses on providing the general framework combining abstraction refinement with plan revision, current efforts aim at optimizing the obtained results based on the definition of the cost functions.

REFERENCES

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

Baier, C., Katoen, J.P., et al. (2008). *Principles of model checking*, volume 26202649. MIT press Cambridge.

Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2003). Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM (JACM)*, 50(5), 752–794.

Coogan, S. and Arcak, M. (2015). Efficient finite abstraction of mixed monotone systems. In *Hybrid Systems: Computation and Control*, 58–67.

De Giacomo, G. and Vardi, M.Y. (2013). Linear temporal logic and linear dynamic logic on finite traces. In *International Joint Conference on Artificial Intelligence*, 854–860.

Ding, Y. and Zhang, Y. (2005). A logic approach for LTL system modification. In *International Symposium on Methodologies for Intelligent Systems*, 435–444.

Finger, M. and Wassermann, R. (2008). Revising specifications with CTL properties using bounded model checking. In *Brazilian Symposium on Artificial Intelligence*, 157–166.

Girard, A. (2005). Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, 291–305.

Guo, M. and Dimarogonas, D.V. (2013). Reconfiguration in motion planning of single-and multi-agent systems under infeasible local LTL specifications. In *IEEE Conference on Decision and Control*, 2758–2763.

Guo, M. and Dimarogonas, D.V. (2015). Multi-agent plan reconfiguration under local LTL specifications. *International Journal of Robotics Research*, 34(2), 218–235.

Henzinger, T.A., Jhala, R., and Majumdar, R. (2003). Counterexample-guided control. In *International Colloquium on Automata, Languages, and Programming*, 886–902.

Kim, K., Fainekos, G., and Sankaranarayanan, S. (2015). On the minimal revision problem of specification automata. *The International Journal of Robotics Research*, 1–21.

Korf, R.E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1), 97–109.

Kupferman, O. and Vardi, M.Y. (2001). Model checking of safety properties. *Formal Methods in System Design*, 19(3), 291–314.

Kurzhanskiy, A.A. and Varaiya, P. (2007). Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 52(1), 26–38.

Lee, W., Pardo, A., Jang, J.Y., Hachtel, G., and Somenzi, F. (1997). Tearing based automatic abstraction for ctl model checking. In *IEEE/ACM international conference on Computer-aided design*, 76–81.

Mesbahi, M. and Egerstedt, M. (2010). *Graph theoretic methods in multiagent networks*. Princeton University Press.

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

Meyer, P.J. and Dimarogonas, D.V. (2017). Compositional abstraction refinement for control synthesis under lasso-shaped specifications. In *American Control Conference*.

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

Reissig, G., Weber, A., and Rungger, M. (2016). Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*.

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

Tumova, J., Castro, L.I.R., Karaman, S., Frazzoli, E., and Rus, D. (2013). Minimum-violation ltl planning with conflicting specifications. In *American Control Conference*, 200–205.