Compositional abstraction refinement for control synthesis under lasso-shaped specifications

Pierre-Jean Meyer and Dimos V. Dimarogonas

Abstract—This paper presents a compositional approach to specification-guided abstraction refinement for control synthesis of a nonlinear system associated with a method to over-approximate its reachable sets. The control specification consists in following a lasso-shaped sequence of regions of the state space. The dynamics are decomposed into subsystems with partial control, partial state observation and possible overlaps between their respective observed state spaces. A finite abstraction is created for each subsystem through a refinement procedure, which starts from a coarse partition of the state space and then proceeds backwards on the lasso sequence to iteratively split the elements of the partition whose coarseness prevents the satisfaction of the specification. The composition of the local controllers obtained for each subsystem is proved to enforce the desired specification on the original system. This approach is illustrated in a nonlinear numerical example.

I. INTRODUCTION

For model checking and control synthesis problems on continuous systems under high-level specifications, a classical approach is to abstract the continuous dynamics into a finite transition system [26]. Although both model checking and abstraction fields have received significant attention, the link between them is not as straightforward as it appears: due to over-approximations involved in the abstraction procedure, the unsatisfaction of the specification on an abstraction cannot be propagated to the original system. This led to the introduction of an interface layer named abstraction refinement aiming at iteratively refining an initial coarse abstraction until the specification is satisfied on the obtained refined abstraction. This topic has been extensively studied in the context of model checking for hardware design, thus primarily focused on verification problems (as opposed to control synthesis) for large but finite systems [16], [23], [17], with the most popular approach being based on CounterExample-Guided Abstraction Refinement (CEGAR) [8], [4], [3], [12]. Later work then also considered control problems [13], [10] and infinite systems [7], [6], [27], [9].

In this paper, we present a method for specification-guided abstraction refinement for control synthesis of continuous systems. We consider a control specification consisting in following a lasso-shaped sequence of regions of the state space, which can be seen as a satisfying trace of a Linear Temporal Logic formula [2]. A coarse abstraction of the system is then initially considered and iteratively refined in its elements preventing the satisfaction of this specification. In most continuous systems, exact computation of the reachable sets as in [13], [27] is not possible. We thus rely on methods to efficiently compute over-approximations of the reachable sets (for a given finite time), using for example polytopes [5], oriented hyper-rectangles [25], ellipsoids [15], zonotopes [11], level sets [20] or the monotonicity property [24], which is considered in the examples of this paper. Other relevant works with similar objectives include: [22] which focuses on reach-avoid-stay control specifications and computes abstractions based on infinite-time reachability of neighbor states; and [21] which uses sets of finite prefixes to describe abstractions of infinite behaviors.

A novelty compared to the mentioned literature is that we combine the abstraction refinement approach with the compositional framework from [18], thus widening the range of applications to systems of larger dimensions. In this work, the global dynamics are decomposed into subsystems with partial control and partial observation of the state (with possible overlaps on their respective state spaces), then the abstraction refinement is applied to each subsystem and the obtained local controllers are combined to control the original system. A journal version of this approach was presented in [19] with the main differences in the current submission being: i) the refinement algorithm considers lasso-shaped sequences as its specification (as opposed to finite sequences in [19]); ii) the numerical application considers a nonlinear system (as opposed to a linear one in [19]).

The structure of this paper is as follows. The problem is formulated in Section II. Section III describes the general method to obtain compositional abstractions. The abstraction refinement algorithm to be applied to each subsystem is presented in Section IV. Then, Section V provides the main result that the local controllers can be composed to control the original system. Finally, a numerical illustration of this method is presented in Section VI.

II. PROBLEM FORMULATION

A. Notations

Let \( \mathbb{N}, \mathbb{Z}^+ \) and \( \mathbb{R} \) be the sets of positive integers, non-negative integers and 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. In this paper, a decomposition of a system into subsystems is considered. As a result, both scalar and set variables are used as subscript of other variables, sets or functions:

- lower case letters and scalars give naming information relating a variable, set or function to the subsystem of
corresponding index (e.g. $x_1$ and $u_1$ are the state and input of the $i$-th subsystem $S_i$);

- index sets denoted by capital letters are used to represent the projection of a variable to the dimensions contained in this set. Alternatively, we also use the operator $\pi_i$ to denote the projection on the dimensions contained in $I$ (e.g. for $x \in \mathbb{R}^n$ and $I \subseteq \{1, \ldots, n\}$, $x_I = \pi_I(x)$).

B. System description

We consider a discrete-time nonlinear control system subject to disturbances described by

$$x^+ = f(x, u, w),$$

with state $x \in X \subseteq \mathbb{R}^n$, bounded control and disturbance inputs $u \in U \subseteq \mathbb{R}^p$ and $w \in W \subseteq \mathbb{R}^r$, respectively. The one step reachable set of (1) from a set of initial states $X \subseteq X$ and for a subset of control inputs $U \subseteq U$ is defined as

$$RS(X, U) = \{f(x, u, w) \mid x \in X, u \in U, w \in W\}. \quad (2)$$

Throughout this paper, we assume that we are able to compute over-approximations $\overline{RS}(X, U)$ of the reachable set defined in (2):

$$RS(X, U) \subseteq \overline{RS}(X, U). \quad (3)$$

Several methods exist for over-approximating reachable sets for fairly large classes of linear and nonlinear systems, see e.g. [5], [25], [15], [11], [20], [24].

System (1) can also be described as a non-deterministic infinite transition system $S = (X, U, \rightarrow)$ where

- $X \subseteq \mathbb{R}^n$ is the set of states,
- $U \subseteq \mathbb{R}^p$ is the set of inputs,
- a transition $x \xrightarrow{u} x'$, equivalently written as $x' \in \text{Post}(x, u)$, exists if $x' \in RS(\{x\}, \{u\})$.

C. Specification

We assume that the state space $X \subseteq \mathbb{R}^n$ is an interval of $\mathbb{R}^n$ and we consider a uniform partition $P$ of $X$ into smaller identical intervals. To ensure that $P$ is a partition, all intervals (including $X$) are assumed to be half-closed. In what follows, the elements of $P$ are called cells of the state space. In this paper, we focus on a control objective consisting in following a lasso-shaped path $\psi = \psi_\text{pref}(\psi_\text{pref})\omega$ composed of two strings of cells in $P$; a finite prefix path $\psi_\text{pref}$, followed by a finite suffix path $\psi_\text{suffix}$ repeated infinitely often.

**Problem 1:** Find a controller $C : X \rightarrow U$ such that the system $S$ follows the infinite path $\psi = \psi(0)\psi(1)\psi(2)\ldots$, i.e. for any trajectory $x : Z^+ \rightarrow X$ of the controlled system, we have $x(k) \in \psi(k)$ for all $k \in \mathbb{Z}^+$.

Although considering this particular type of control objectives may seem restrictive, a wider range of control problems can actually be covered from the observation that, given a Linear Temporal Logic formula, at least one of its satisfying traces takes the form of a lasso-shaped path as above [2]. Solving Problem 1 then also ensures that the controlled system satisfies the corresponding formula from which the lasso path $\psi$ is derived.

III. COMPOSITIONAL ABSTRACTIONS

In this paper, Problem 1 is addressed with a compositional abstraction refinement approach, where the system is decomposed into subsystems before applying an abstraction refinement algorithm to each of them. In this section, we first present the general method adapted from [18] to obtain compositional abstractions.

A. System decomposition

We decompose the dynamics (1) into $m \in \mathbb{N}$ subsystems. Let $(I_1, \ldots, I_m)$ be a partition of the state indices $\{1, \ldots, n\}$ and $(J_1, \ldots, J_m)$ a partition of the control input indices $\{1, \ldots, p\}$. Subsystem $i \in \{1, \ldots, m\}$ can then be described using the following sets of indices:

- $I_i^s$ represents the state components to be controlled;
- $I_i \supseteq I_i^s$ are all the state components whose dynamics are modeled in the subsystem;
- $I_i^o = I_i \setminus I_i^s$ are the state components that are only observed but not controlled;
- $K_i = \{1, \ldots, n\} \setminus I_i$ are the unobserved state components considered as external inputs to subsystem $i$;
- $J_i$ are the input components actually used for control;
- $L_i = \{1, \ldots, p\} \setminus J_i$ are the remaining control components considered as external inputs to subsystem $i$.

The role of all the index sets above can be summarized as follows: for subsystem $i \in \{1, \ldots, m\}$, we model the states $x_I = (x_{I_i^s}, x_{I_i^o})$ where $x_{I_i^s}$ are to be controlled using the inputs $u_{J_i}$ and $x_{I_i^o}$ are only observed to increase the precision of the subsystem while $x_K$ and $u_L$ are considered as external disturbances. It is important to note that the subsystems may share common modeled state components (i.e. the sets $I_i$ may overlap), though the sets of controlled state components $I_i^s$ and modeled control input components $J_i$ are assumed to be disjoints for two subsystems.

B. Subsystem’s abstraction

For each subsystem $i \in \{1, \ldots, m\}$, we want to create a finite abstraction $S_i$ of the original system $S$, which models only the state and input components $x_{I_i}$ and $u_{J_i}$, respectively. $S_i$ will then be used to synthesize a local controller focusing on the satisfaction of the specification for the controlled state components $x_{I_i^s}$ using the modeled control inputs $u_{J_i}$. The general structure of the abstraction $S_i = (X_i, U_i, \rightarrow_i)$ is as follows.

- $X_i$ is a partition of $x_{I_i}(X)$ into a finite set of intervals called symbols. It is initially taken equal to $x_{I_i}(P)$ and will then be refined in Section IV.
- $U_i$ is a finite subset of the projected control set $\pi_{I_i}(U)$.
- A transition $s_i \xrightarrow{u_{i}} s'_i$, equivalently written as $s'_i \in \text{Post}_i(s_i, u_i)$, exists if $s'_i \cap \pi_{I_i}(RS_{i \rightarrow_i}(s_i, u_i)) \neq \emptyset$.

The set $RS_{i \rightarrow_i}(s_i, u_i) \subseteq X$ represents an over-approximation of the reachable set of (1) based on the partial knowledge available to subsystem $i$. The remaining of this section describes how this set is obtained.

The unmodeled inputs $u_{L_i}$ are known to be bounded in $\pi_{L_i}(U)$. We also know that other subsystems will synthesize
controllers satisfying the specification for the unobserved and uncontrolled state components \((x_{K_i}, x_{I_i})\), respectively) of subsystem \(i\). This is formalized by the following assumption:

**A/G Obligation 1**: For all \(x \in X, i \in \{1, \ldots, m\}\) and \(k \in \mathbb{Z}^+,\) if \(x_{I_i} \in \pi_{I_i}(\psi(k))\), then \(x_{K_i} \in \pi_{K_i}(\psi(k))\).

**A/G Obligation 2**: For all \(i \in \{1, \ldots, m\}\), \(s_i \in X_i\) and \(k \in \mathbb{Z}^+,\) if \(s_i \subseteq \pi_{I_i}(\psi(k))\), then \(s_i \subseteq \pi_{I_i}(\psi(k+1))\).

Intuitively, if the state of subsystem \(i\) is in the projection \(\pi_{I_i}(\psi(k))\) of some cell \(\psi(k)\) in \(P\), then the unobserved states \(x_{K_i}\) also start from the projection \(\pi_{K_i}(\psi(k))\) of this cell (A/G Obligation 1) and the uncontrolled states \(x_{I_i}\) will reach the next step \(\pi_{I_i}(\psi(k+1))\) of the specification (A/G Obligation 2).

Given a symbol \(s_i \in X_i\) of \(s_i\) with \(s_i \subseteq \pi_{I_i}(\psi(k))\) and a control value \(u_i \in U_i\), the set \(RS_{AG1}(s_i, u_i) \subseteq X\) is obtained in the following two steps. We first compute an intermediate set \(RS_{AG1}(s_i, u_i) \subseteq X\) using A/G Obligation 1 and the operator \(RS\) in (3) as follows,

\[
RS_{AG1}(s_i, u_i) = RS(\psi(k) \cap \pi_{I_i}^{-1}(s_i), U \cap \pi_{I_i}^{-1}(\{u_i\})).
\]

resulting in a larger over-approximation of the reachable set (2) where the unobserved variables \(x_{K_i}\) and \(u_{I_i}\) are considered as bounded disturbances: given \(s \subseteq X\) such that \(s \subseteq \psi(k)\) and a control input \(u \in U\), (2) and (4) give

\[
RS(s, \{u\}) \subseteq RS(s, \{u\}) \subseteq RS_{AG1}(\pi_{I_i}(s), \pi_{I_i}(u)).
\]

Next, \(RS_{AG1}(s_i, u_i)\) is updated into \(RS_{AG2}(s_i, u_i)\) using A/G Obligation 2:

\[
RS_{AG2}(s_i, u_i) = RS_{AG1}(s_i, u_i) \cap \pi_{I_i}^{-1}(\pi_{I_i}(\psi(k+1))).
\]

The set \(RS_{AG2}\) is thus the same set as the over-approximation \(RS_{AG1}\), but without the states that violate the specification on the uncontrolled state dimensions \(I_{K_i}\), since they are known to be controlled by other subsystems. The particular case where \(RS_{AG2} = \emptyset\) means that despite the best control actions from other subsystems, the state of the system will always be driven out of the targeted cell \(\psi(k+1)\).

### IV. Refinement Algorithm

For each subsystem \(i \in \{1, \ldots, m\}\), starting from the coarsest abstraction corresponding to the initial partition \(X_i = \pi_{I_i}(P)\), the abstraction refinement method presented in this section aims at iteratively identifying elements of this abstraction preventing the satisfaction of the specification \(\psi\) for subsystem \(i\) and refining these elements to obtain a more precise abstraction. The advantages of this specification guided approach are thus to automatically refine the state partition if the specification is not initially satisfied and to avoid the computation of the whole abstraction when only a small part is actually relevant to the specification.

**Assumption 2**: \(\psi = \psi(0) \psi(1) \ldots \psi(r)\) for some \(r \in \mathbb{N}\). For any \(k, l \in \{0, \ldots, r\}\) such that \(k \neq l\) and for any subsystem \(i \in \{1, \ldots, m\}\) we have \(\pi_{I_i}(\psi(k)) \neq \pi_{I_i}(\psi(l))\).

For clarity of notations, this approach is presented in Algorithm 1 in the particular case of Assumption 2 where the desired lasso-shaped path \(\psi = \psi_{pref}(\psi_{suff})\) is finite (i.e., \(\psi_{suff} = \emptyset\)) and for each subsystem it does not visit the same cell twice. The straightforward modifications required to cover the general case without Assumption 2 are provided at the end of this section.

**Input**: Partition \(P\) of \(X\), discrete control set \(U\).

**Input**: Partition projection \(P_{\psi} : P \rightarrow 2^X\), such that \(P_{\psi}(\sigma) = \{s_i \in X_i \mid s_i \subseteq \pi_{I_i}(\sigma)\}\).

**Initialization**: \(X_i = \pi_{I_i}(P), V_{I_i} = \{\pi_{I_i}(\psi(r))\}, V_{I_i}^X = \pi_{I_i}(\psi(r)), Queue = \emptyset\).

**for** \(k = 1\) **to** \(r\) **do**

- \(\{V_{i}^k, V_{iX}^k, C_i\} = \text{ValidSets}(k, V_{iX}^{k+1})\)
- \(Queue = \text{AddToQueue}(\psi(k))\)
- **while** \(\psi(j) = \text{FirstInQueue}(Queue)\) **do**
  - for all \(s_i \in P_{\psi}(\psi(j))(j)\) do \(X_i = \text{Split}(s_i)\);
  - for \(l\) **from** \(j\) **to** \(k\) **do**
    - \(\{V_{i}^l, V_{iX}^l, C_i\} = \text{ValidSets}(l, V_{iX}^{l+1})\)
- return \(X_i, \bigcup_{k=0}^{r-1} V_{iX}^k \subseteq X_i, C_i : \bigcup_{k=0}^{r-1} V_{iX}^k \rightarrow U_i\)

**Algorithm 1**: Refinement algorithm for subsystem \(i\).

**a) Input**: Algorithm 1 is provided with the initial partition \(P\) of the state space \(X\), a finite set \(U_i\) of control values for subsystem \(i\) as in Section III-B, the finite sequence of cells \(\psi(0) \ldots \psi(r) \in P^{r+1}\) defining the specification \(\psi\) from Section II-C as in Assumption 2 and an operator \(P_i : P \rightarrow 2^X_i\), giving the set of all symbols \(s_i \in X_i\) included in the projection \(\pi_{I_i}(\sigma)\) of each cell \(\sigma \in P\). For each cell \(\psi(k)\) in the sequence \(\psi(0) \ldots \psi(r)\) the goal is to compute the subset \(V_{iX}^k \subseteq P_i(\psi(k))\) of symbols that are valid with respect to the specification \(\pi_{I_i}(\psi(k) + 1)\). The set \(V_{iX}^k\) then corresponds to the projection of \(V_{iX}^k\) on the continuous state space \(\pi_{I_i}(X)\).

**b) Initialization**: The set of symbols \(X_i\) is initially taken as the coarsest partition of the state space \(\pi_{I_i}(X)\) (i.e., \(\pi_{I_i}(P)\)) and will be refined during the algorithm when unsatisfaction of \(\psi\) is detected. We proceed backward on the finite sequence \(\psi(0) \ldots \psi(r)\) and thus take the final cell \(\psi(r)\) as fully valid: \(V_{iX}^r = P_i(\psi(r)) = \{\pi_{I_i}(\psi(r))\}\) and \(V_{iX}^r = \pi_{I_i}(\psi(r))\). We also initialize a priority queue \((Queue = \emptyset)\) which will be used to determine which cell of \(P\) is to be refined at the next iteration of the algorithm.

**c) External functions**: Algorithm 1 calls four external functions. The function \text{ValidSets} looks for the valid symbols and their associated control inputs for a particular step of the specification sequence. This function is detailed in Algorithm 2 and explained in the next paragraph. Func-
tions AddToQueue and FirstInQueue deal with the management of the priority queue and Split represents the refinement of the partition. Although these 3 functions offer significant degrees of freedom towards maximizing the efficiency of the algorithm, this optimization problem is beyond the scope of this paper and is left as future research.

Input: \( P, U_i, \psi \) and \( P_i \) from Input to Algorithm 1.

Input: Index \( k \in \{0, \ldots, r-1\} \) of considered cell \( \psi(k) \)

Input: Next cell’s valid set \( V_{k+1}^i \)
\[ V_i^k = \{ s_i \in P_i(\psi(k)) \mid \exists u_i \in U_i \text{ such that} \} \]
\[ \emptyset \neq P_i(\psi(k)) \subseteq V_{k+1}^i \]
\[ V_{k+1}^i = \{ x_i \in \pi_i(X) \mid \exists s_i \in V_k^i \text{ such that} x_i \in s_i \} \]
\[ \forall s_i \in V_k^i, C_i(s_i) \text{ is chosen in} \]
\[ \{ u_i \in U_i \mid \emptyset \neq P_i(\psi(k)) \subseteq V_{k+1}^i \} \]

Algorithm 2: ValidSets. Computes the valid sets and controller for subsystem \( i \) at step \( k \) of the specification \( \psi \).

d) Valid sets: In the main loop of Algorithm 1, assuming we have previously found non-empty valid sets \( \{ V_k^{i-1}, \ldots, V_k^r \} \), we call the function ValidSets for step \( k \) of the specification as in Algorithm 2. This function first computes the valid set \( V_k^i \) for step \( k \) by looking for the symbols in \( P_i(\psi(k)) \) for which the over-approximation \( RS^{AG2} \) of the reachable set is both non-empty and contained in the valid set \( V_{k-1}^i \) of the next cell \( \psi(k+1) \) for at least one value of the discrete control input. Note that in this particular call where the cell \( \psi(k) \) is visited for the first time, \( P_i(\psi(k)) \) contains a single element: \( \pi_i(\psi(k)) \). The set \( V_k^i \) is taken as the projection of \( V_k^i \) on the continuous state space \( \pi_i(X) \). Then, the controller \( C_i \) associates each valid symbol in \( V_k^i \) to the first of such satisfying control values that has been found. Algorithm 2 finally outputs \( V_k^i, V_{k+1}^i \) and \( C_i \) to Algorithm 1. Since the cell \( \psi(k) \) is considered here for the first time, we add it to the priority queue with the function AddToQueue.

e) Refinement and update: If the valid set \( V_k^i \) is empty, we select (with function FirstInQueue) the first cell \( \psi(i) \) of the priority queue and refine it. The refinement is achieved by the function Split and consists in uniformly splitting all the invalid symbols of \( P_i(\psi(i)) \) into a number of identical subsymbols (e.g. 2 in each state dimension of \( I_i \)). After this, we need to update the valid sets \( V_k^i \) and \( V_{k+1}^i \) and controller \( C_i \) for the refined cell \( \psi(i) \) using the ValidSets function. The possibly larger valid set \( V_{k+1}^i \) obtained after this refinement can then induce a larger valid set at step \( j-1 \), which in turns influences the following steps. The refinement and update of the valid set at step \( j \) thus requires an update (using function ValidSets) for all other cells from \( \psi(j-1) \) to \( \psi(k) \). The refined cell \( \psi(j) \) can then be moved to any other position in the priority queue (here assumed to be handled by the function FirstInQueue) and these steps are repeated until \( V_k^i \neq \emptyset \).

f) Outputs: The algorithm provides three outputs. The first one is the refined partition \( X_i \) for subsystem \( i \). The second one gathers the sets \( V_k^{i-1} \subseteq P_i(\psi(k)) \subseteq X_i \) of valid symbols for all \( k \in \{0, \ldots, r-1\} \). Finally, the controller \( C_i \) associates a unique control value (since we stop looking as soon as a satisfying control is found) to each valid symbol.

g) General case: The general case without Assumption 2 can be covered by modifying Algorithm 1 as follows. For duplicated cells \( \pi_i(\psi(k)) = \pi_i(\psi(l)) \) with \( k \neq l \), we need a controller \( C_i: X_i \times \{0, \ldots, r\} \to U_i \) which now also depends on the current position \( k \in \{0, \ldots, r\} \) in \( \psi \) in order to know which next cell \( \psi(k+1) \) should be targeted.

When \( \psi = \psi_{pref}(\psi_{suff})^\omega \) is a lasso path with non-empty suffix \( \psi_{suff} = \psi(r+1) \ldots \psi(f) \), Algorithm 1 is first called on \( \psi_{suff} \). This call is then repeated with the new initialization \( [V_i^f, V_i^{f+1}, C_i] = \text{ValidSets}(f, V_i^{f+1}) \) (i.e. the last suffix cell \( \psi(f) \)) must be driven towards the first suffix cell \( \psi_{suff}(r+1) \) until further calls of the ValidSets function have no more influence on the sets \( V_k^i \) for \( k \in \{r+1, \ldots, f\} \). In this loop, the valid set of a refined suffix cell \( \psi(k) \) needs to be reset to fully valid \( V_k^i = P_i(\psi(k)) \) to avoid propagation of empty valid sets. A final call of Algorithm 1 is then done for \( \psi_{pref} \) with the initialization \( [V_i^f, V_i^{f+1}, C_i] = \text{ValidSets}(r, V_i^{r+1}) \) (i.e. the last prefix cell \( \psi(r) \)) must be driven towards the first suffix cell \( \psi(r+1) \).

V. COMPOSITION

Algorithm 1 in Section IV is applied to each subsystem \( i \in \{1, \ldots, m\} \) separately. In this section, we then show that combining the controllers \( C_i \) of all subsystems results in a global controller solving Problem 1 by ensuring that the original system \( S \) follows the lasso-shaped sequence \( \psi \).

A. Operator for partition composition

Due to the possible overlap of the state space dimensions for two subsystems and the fact that the refined partitions do not necessarily match on these common dimensions, we first need to define an operator for the composition of sets of symbols (either the refined partition \( X_i \) or the valid sets \( V_k^i \) obtained in Algorithm 1).

Given two refined sets \( X_i \) and \( X_j \) as obtained in Section IV, we first introduce an intermediate operator \( \cap \):

\[ X_i \cap X_j = \{ s \in \pi_i \cup \pi_j(2^X) \mid \exists s_i \in X_i, \pi_i(s) \subseteq s_i, \exists s_j \in X_j, \pi_j(s) \subseteq s_j \} \]

followed by the main composition operator \( \otimes \) defined as:

\[ X_i \otimes X_j = X_i \cap X_j \setminus \{ s \in X_i \cap X_j \mid \exists s' \in X_i \cap X_j, s \subseteq s' \} \]

Intuitively, we first ensure that the set \( X_i \cap X_j \) is at least as fine as both partitions \( X_i \) and \( X_j \), thus providing a covering of \( \pi_i \cup \pi_j(X) \):
\[ \bigcup_{s \in X_i \cap X_j} s = \pi_i \cup \pi_j(X) \]

Then, \( X_i \cap X_j \) is converted into a partition \( X_i \otimes X_j \) of \( \pi_i \cup \pi_j(X) \) by removing all subsets strictly contained in another element of \( X_i \cap X_j \).

Proposition 3: If \( X_i \) and \( X_j \) are partitions of \( \pi_i(X) \) and \( \pi_j(X) \), respectively, then \( X_i \otimes X_j \) is a partition of \( \pi_i \cup \pi_j(X) \).

Proof: Let \( x \in \pi_i \cup \pi_j(X) \). Since \( X_i \) and \( X_j \) are partitions, there exists \( s_i \in X_i \) and \( s_j \in X_j \) such that \( \pi_i(x) \in s_i \) and \( \pi_j(x) \in s_j \), which implies that there exists
s ∈ X_i ∩ X_j such that x ∈ s. Then, the set X_i ∩ X_j is also a covering since it only removes elements of X_i ∩ X_j which are strictly contained in other elements of X_i ∩ X_j.

Let now s, s′ ∈ X_i ∩ X_j such that x ∈ s ∩ s′. Since X_i and X_j are partitions, we also know that s_i ∈ X_i and s_j ∈ X_j as defined above are unique. From X_i ∩ X_j ⊆ X_i ∩ X_j, we thus have π_{i,t}(s), π_{i,t}(s′) ⊆ s_i and π_{i,t}(s), π_{i,t}(s′) ⊆ s_j, which implies that s ∪ s′ ∈ X_i ∩ X_j. Therefore, s and s′ can only be in X_i ∩ X_j if s = s′ = s ∪ s′.

B. Composed transition system

We now define the transition system S_c = (X_c, U_c, →) as the composition of the abstractions S_i obtained in Algorithm 1 for each subsystem i ∈ {1, . . . , m}. S_c contains the following elements:

- X_c = X_1 ∩ · · · ∩ X_m is the composition of the refined partitions for each subsystem. From Proposition 3, we know that X_i is a partition of X.

From the definition of the operator ∩, the projection π_{i,t}(s) of s ∈ X_c does not necessarily correspond to a symbol of X_i. However, we know (see proof of Proposition 3) that there exists a unique symbol s_i ∈ X_i containing this projection. Therefore, for each i ∈ {1, . . . , m}, we define the decomposition function d_i : X_c → X_i such that d_i(s) = s_i is the unique symbol s_i ∈ X_i satisfying π_{i,t}(s) ⊆ s_i.

- U_c = U_1 × · · · × U_m is the composition of the discretized control sets (which is a simple Cartesian product since they are defined on disjoint dimensions).

We can then introduce the controller C_c : X_c → U_c as the composition of the controllers C_i : X_i → U_i obtained on the abstraction of each subsystem in Algorithm 1:

∀s ∈ X_c, C_c(s) = (C_1(d_1(s)), . . . , C_m(d_m(s))),

(7)

which is then used to define the transition relation of S_c.

- ∀s, s′ ∈ X_c, u = C_c(s), s →_c u → s′ \iff \forall i ∈ {1, . . . , m}, d_i(s) \xrightarrow{u_i} d_i(s′).

Intuitively, the transition s →_c s′, equivalently written as s′ ∈ Post_c(s, u), exists when the control input u ∈ U_c is allowed by the local controllers C_i for all i ∈ {1, . . . , m} and when the transition in S_i can be decomposed (using the decomposition functions d_i : X_c → X_i) into existing transitions for all subsystems. Finally, we define the set U_c(s) = \{u ∈ U_c \mid Post_c(s, u) \neq \emptyset\}.

C. Main result

To control S with the controller C_c in (7), the systems S = (X, U, →) and S_c = (X_c, U_c, →_c) must satisfy the following alternating simulation relation, adapted from [26].

Definition 4 (Alternating simulation): A map H : X → X_c is an alternating simulation relation from S_c to S if it holds: \forall x ∈ X, s = H(x), u_c ∈ U_c(s), \exists u ∈ U such that \forall x' ∈ Post(x, u), H(x') ∈ Post_c(s, u_c).

This definition means that for any pair (x, s) of matching state and symbol and any control u_c of the abstraction S_c, there exists an equivalent control for the original system S such that any behavior of S is matched by a behavior of S_c. As a consequence, if a controller is synthesized so that S_c satisfies some specification, then we know that there exists a controller ensuring that S also satisfies the same specification. We can show that such a relation can be found when both S_c and S use the same controls u = C_i(s).

Theorem 5: The map H : X → X_c such that x ∈ s ↔ H(x) = s is an alternating simulation relation from S_c to S.

Proof: Let x ∈ X, s = H(x) ∈ X_c and u ∈ U_c(s). By definition of S_c, we have U_c(s) ⊆ \{C_i(s)\} for all s ∈ X_c. If U_c(s) = ∅, the condition in Definition 4 is trivially satisfied. Otherwise, we have u = C_i(s) defined as in (7) which implies that x ∈ ψ(k) for some k ∈ ℤ^+. Let x′ ∈ Post(x, u), s′ = H(x′) and denote the decompositions of s and s′ as s_i = d_i(s) and s_i′ = d_i(s′) for all i ∈ {1, . . . , m}, by definition of the over-approximation operator RS in (3), we have x′ ∈ RS(s, \{u\}). With the inclusion in (5) and the fact that π_{i,t}(s) ⊆ s_i, we obtain x′ ∈ RS^{AGL}_i(s_i, u_j) for all i. If x′ ∈ ψ(k + 1), then x_i′ ∈ s_i ∩ π_{i,t}(RS^{AGL}_i(s_i, u_j)) and this intersection is thus non-empty, which implies that s_i′ ∈ Post_i(s_i, u_j) for all i. Then, s′ ∈ Post_c(s, u) by definition of S_c. On the other hand, if x′ /∈ ψ(k + 1), then there exists l ∈ {1, . . . , n} such that x_i′ \not∈ π_{i,l}(ψ(k + 1)) and there exists a unique subsystem j ∈ {1, . . . , m} such that l ∈ I_j. Therefore, we have x_i′ /∈ π_{i,l}(ψ(k + 1)) and then π_{i,l}'(RS^{AGL}_i(s_j, u_j)) \not⊆ π_{i,l}'(ψ(k + 1)). This implies that u_j /∈ U_j(s_j) which contradicts the fact that u ∈ U_c(s).

Theorem 5 thus confirms that using AG Obligations 1 and 2 is reasonable since it preserves the alternating simulation relation on the composition S_c while reducing the conservatism of the over-approximations in each subsystem.

The next result immediately follows from the definition of S_c (U_c(s) ⊆ \{C_i(s)\}) and the proof of Theorem 5 (if C_c(s) exists, then Post_c(s, C_c(s)) = \emptyset), i.e. C_c(s) ∈ U_c(s).

Corollary 6: U_c(s) = \{C_i(s)\} for all s ∈ X_c.

These two results can then be exploited to solve Problem 1.

Theorem 7: Let x : ℤ^+ → X be any trajectory of S from an initial state x(0) ∈ X such that H(x(0)) ∈ V^0 m \mid · · · \mid V^0 n and subject to the controller C_cX : X → U with C_cX(x) = C_c(H(x)) for all x ∈ X. Then x(k) ∈ ψ(k) for all k ∈ ℤ^+.

Proof: From Theorem 5, it is sufficient to prove that the composed system S_c controlled by C_c in (7) follows ψ if it starts at s^0 = H(x(0)) ∈ V^0 m \mid · · · \mid V^0 n and subject to the controller C_cX : X → U with C_cX(x) = C_c(H(x)) for all x ∈ X. Then Algorithm 2 gives that d_i(s′) ∈ V_i^k for all i and it follows that s′ ∈ V_i^k+1 \mid · · · \mid V_m^k+1, therefore s′ ⊆ ψ(k + 1).

If Algorithm 1 terminates in finite time for all subsystems i, Theorem 7 thus defines a controller C_cX ensuring that the continuous system S follows the desired path ψ. However, if S follows ψ, we cannot guarantee that Algorithm 1 will find partitions X_i for all subsystems i where ψ can be followed.
VI. NUMERICAL ILLUSTRATION

The use of intervals as the elements of the state partition (required by the compositional approach in Section III) particularly suits the computation of over-approximations of the reachable set using the monotonicity property. The reader is referred to [24], [1] for a description of monotone (control) systems and to e.g. [18] for their use to over-approximate the reachable set and create abstractions. In this section, we thus consider the 8D nonlinear monotone system described by:

$$\dot{x} = Ax - \beta x^3 + u,$$  \hspace{1cm} (8)

with state $x \in \mathbb{R}^8$, bounded control input $u \in [-5,5]^8$, constant parameter $\beta = 0.01 \in \mathbb{R}$ and componentwise cubic power $x^3$. The diagonal elements of the matrix $A \in \mathbb{R}^{8 \times 8}$ are equal to $-0.8$ and the remaining elements represent state interactions and are shown in the directed graph of Figure 1. To match the description of (1) in Section II-B, the system (8) is then sampled with a period of 1.2 seconds.

![Fig. 1: State interactions in (8). A directed edge from node $i$ to node $j$ is labeled with the value of $\partial x_j / \partial x_i$.](image)

In view of the state coupling shown in Figure 1, we decompose the system (8) into 5 subsystems defined as follows by their index sets $I_1$, $I_2$ and $J_1$. We first take three pairs $I_1 = I^c_1 = J_1 = \{1,2\}$, $I_2 = I^c_2 = J_2 = \{4,5\}$ and $I_3 = I^c_3 = J_3 = \{6,7\}$ where all the observed states are also controlled ($I^c_1 = I^c_2 = I^c_3 = \emptyset$). The last two subsystems only aim at controlling a single state each, but also observe an additional state: $I_4 = \{2,3\}$, $I^c_4 = J_4 = \{3\}$, $I_5 = \{7,8\}$, $I^c_5 = J_5 = \{8\}$, $I^c_5 = \{7\}$.

The considered state space $X = [-9,9]^8$ is partitioned into 3 elements per dimension, thus resulting in a partition $P$ of 6561 cells. The control interval $U = [-5,5]^8$ is discretized into 5 values per dimension ($[-5,-2.5,0,2.5,5]$).

We consider a control objective initially formulated as the Linear Temporal Logic formula $\square \sigma_2 \land \square \sigma_3$ representing the surveillance task of visiting infinitely often both partition cells $\sigma_2 = [-3,3]^8$ and $\sigma_3 = [3,9]^8$. Assuming that the initial state of the system is in the cell $\sigma_0 = [-9,-3]^8$, this can then be reformulated as a lasso-shaped sequence $\psi = \sigma_0, \sigma_1, (\sigma_2, \sigma_3)^\omega$, where the second cell $\sigma_1$ of the prefix is $[-3,3]$ for the state dimensions 3 and 7 while it remains $[-9,-3]$ (as in $\sigma_0$) on the other dimensions. Note that $\psi$ does not satisfy Assumption 2 due to both its non-empty suffix and duplicated prefix cells (e.g. $\pi_{I_1}(\sigma_0) = \pi_{I_1}(\sigma_1)$).

Algorithm 1 is then applied to each subsystem, where the $\text{Split}$ function uniformly splits a symbol into 2 subsymbols per dimension and the priority queue is handled as follows: we only refine a cell when no coarser candidate exists, and when more than one cell can be refined we prioritize the one whose last refinement is the oldest. In Figure 2, we display the resulting refined partitions and valid symbols for each subsystem. Below, we detail the refinement process in the case of subsystem 5 in Figure 2e. We start with the top right cell $\pi_{I_5}(\sigma_3)$ as fully valid. For $\pi_{I_5}(\sigma_2)$ (center), no satisfying control is found to drive the whole cell into $\pi_{I_5}(\sigma_3)$, so it is split into 4 identical subsymbols, two of which are valid. We loop back on the last cell $\pi_{I_5}(\sigma_3)$ of the suffix and find that the whole cell can be controlled towards the valid symbols of $\pi_{I_5}(\sigma_2)$. The valid set $V_{I_5}^0$ is thus unchanged by the last call of $\text{ValidSets}$ and Algorithm 1 is done with the suffix.

Since no satisfying control is found to bring the last cell $\pi_{I_5}(\sigma_1)$ (bottom center) of the prefix to the valid symbols of $\pi_{I_5}(\sigma_2)$, we then split $\pi_{I_5}(\sigma_1)$ into 4 subsymbols, 3 of which are valid. Similarly, $\pi_{I_5}(\sigma_0)$ (bottom left) is split into 4 subsymbols since it cannot be controlled towards the valid set of $\pi_{I_5}(\sigma_1)$. None of the obtained subsymbols of $\pi_{I_5}(\sigma_0)$ are valid and we thus refine the next cell in the queue: $\pi_{I_5}(\sigma_1)$. This refinement only splits into 4 subsymbols the unique invalid symbol of $\pi_{I_5}(\sigma_1)$ (i.e. its top right symbol). All new subsymbols are valid (they can be controlled towards the valid set of $\pi_{I_5}(\sigma_2)$), and an update of $\pi_{I_5}(\sigma_0)$ gives that all 4 of its symbols are valid ($V_{I_5}^0 = \pi_{I_5}(\sigma_0)$), thus ending Algorithm 1.

Using Matlab on a laptop with a 2.6 GHz CPU and 8 GB of RAM, these results after applying Algorithm 1 to all subsystems were obtained in 11.1 seconds. As a comparison, the same abstraction refinement algorithm applied in a centralized way (no decomposition and a single abstraction representing the whole system) was still in its first suffix call of Algorithm 1 after more than 48 hours of computation.

VII. CONCLUSION

In this paper, we presented a novel approach to abstraction creation and control synthesis in the form of a compositional specification-guided abstraction refinement procedure. This approach applies to nonlinear systems associated with a method to over-approximate its reachable sets, and to lasso-shaped specifications. The dynamics are decomposed into subsystems representing partial descriptions of the system and a finite abstraction is then created for each subsystem through a refinement procedure starting from a coarse partition of the state space. Each refined abstraction is associated with a local controller and the composition of these local controllers enforces the specification on the original system.

Current efforts aim at maximizing the algorithm efficiency using its degrees of freedom in the splitting strategy and the management of the priority queue. We also work towards combining this approach into a common framework with plan revision methods.

REFERENCES

Fig. 2: Refined partitions and valid symbols (in yellow) for all 5 subsystems.