Submitted, 2014 Conference on Decision and Control (CDC)
http://www.cds.caltech.edu/~murray/papers/fm14-cdc.html

Strategy-Driven Partitioning into Switching Modes for Piecewise-Affine Systems with Continuous Environments

Ioannis Filippidis and Richard M. Murray

***Abstract*— We propose a methodology for abstracting discrete-time piecewise-affine systems influenced by additive continuous environment variables, to synthesize correct-by-construction controllers from linear temporal logic specifications. The proposed algorithm partitions the environment domain into polytopes, considered as modes controlled by the environment. In each mode the environment variable is treated as a bounded disturbance for the system. Mode polytopes are iteratively enlarged while a strategy isomorphism between successive system partitions can be constructed. Isomorphisms are obtained by solving the stable marriage problem after proving that our case admits a unique solution. This leads to mapping high-level symbolic variables to equivalence classes of polytopes, instead of single polytopes as in other works. We thus avoid the need to merge partitions, which can create sliver polytopes, causing numerical problems during reachability computations. The approach enables using the same strategy over partitions that have an isomorphic subgraph relevant to the strategy. Reachability checks between neighboring partitions are used to reduce non-determinism introduced by switching and allow continuous restoration of discrete state. If bifurcations occur that prevent strategy reuse, then switched system game synthesis is performed, and a logic modeling formalism proposed that avoids trivial cyclic counterexamples.**

I. Introduction

As system complexity increases, modular compositional approaches that decompose a large system into components that can be designed and validated independently become more attractive. Though these divide and conquer frameworks reduce the computational complexity of the problem, the challenge is shifted to interfacing these components while preserving the required correctness properties. These interface rules are known as contracts [1] and several related methods have been proposed, including assume-guarantee contracts [2], [3]. Satisfaction of an assume-guarantee contract by a component ensures its correct behavior, under the condition that its environment satisfies the contract's assumption. So a contract defines a set of admissible environments, and enlarging this set makes the components implementing it more versatile and robust to their environments. Such a relaxation of assumptions is analogous to the concept of weakest preconditions for algorithms [4].

A difference with the discrete case is that systems with continuous dynamics are in continuous interaction with their environment. Therefore exploring the set of admissible continuous environments would require a continuous representation of them, which is challenging and renders use of formal methods from computer science difficult. This has led to a large amount of recent work on temporal logic control of hybrid systems [5], [6], [7], [8], where the environment is either discrete, or continuous. Finding a set of assumptions on a discrete environment has been approached using counterexample-guided refinement, and in the case of games, strategy-guided refinement [9]. For continuous environments the authors of [10], [8] lump the uncertainty into a disturbance that is accounted for in reachability computations performed during abstraction of dynamics. Dynamics that are controlled only by mode switching have also been considered [11] and are a particular instance of continuous environment, though its influence is already abstracted there assuming an observation map that discretizes its domain is known a priori.

Here we consider the effect that continuous environment variables can have on the system. Lumping them into a large disturbance can render the problem unrealizable, whereas a refined environment abstraction can reduce the uncertainty over each discrete time step, allowing the system to react by observing continuous environment behavior. For the simple schematic example of Fig. 1 the uncertainty in solar power input can affect the functions of a smart building. The specification can require that security-related devices never be turned off, but appliances can be scheduled so that tasks be eventually completed and power consumption never exceed that available from solar collectors. Designing a controller that achieves this for a wide range of power uncertainty can be impossible, whereas splitting the uncertainty into smaller intervals can be handled by synthesizing a reactive controller that observes the currently available power.

From a contracts viewpoint, it relaxes the environment assumptions by removing the requirement that other system components (its environment) are accompanied by the same discrete representation that was used when computing the assume-guarantee contract for that particular component. A dual viewpoint is specification relaxation [12], [13], differing in that it concerns unrealizable specifications.

We propose computing a polytopic abstraction for the environment, and for each polytope in it a partition for the system. The partitioning into modes is incrementally guided by checking whether the partitions are isomorphic with respect to a strategy found for the first partition in that mode. The resulting environment abstraction is closely related to constructing the observation map assumed as given in [11]. Forms of specification-guided synthesis have been considered in [14], [15], but not as it is used here, where we attempt to map the system strategy as we partition the environment. Robust synthesis has been studied in [16], [17], but from a discrete automata-theoretic perspective, whereas

The authors are with Control and Dynamical Systems, California Institute of Technology, Pasadena, CA, 91125. E-mail: {ifilippi@,murray@cds}.caltech.edu

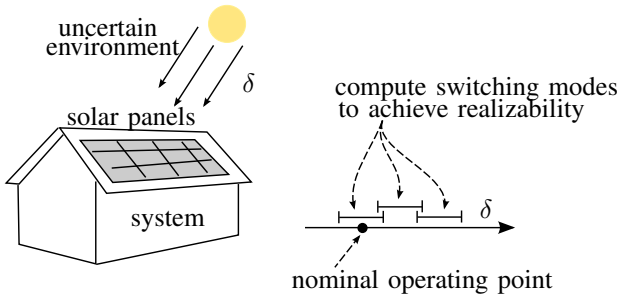


Fig. 1: The system dynamics are abstracted for different environment conditions, such that the given specification remains satisfiable, as the environment state δ changes.

here robustness is considered primarily at the continuous level. Compared to hybridization of nonlinear to switched piecewise-affine (PWA) systems [18], we start from a PWA system that is open, i.e., the environment continuously influences the system and is abstracted by the derived modes.

The rest of this paper is organized as following: Logic and game synthesis are reviewed in section II, the environment partitioning described in section III, computation of overlaps in section IV, isomorphic matching in section V, and partition bifurcations in section VI. A simulation example is presented in section VII and future work considered in section VIII.

II. Preliminaries

Let $\mathbb{N}$ denote non-negative integers, $\mathbb{R}$ real numbers, E^n the n-dimensional Euclidean space, and $\mathbb{B} \triangleq \{0,1\}, Q^* \triangleq Q \setminus \{0\}$. Index sets are denoted by subscripted $I \subset \mathbb{N}$, absolute value and volume by $|x|$.

A. Polytopic Predicates and Propositions

1) Polytopic Predicates: A *predicate* is any function with codomain $\mathbb{B}$. We name predicates differently depending on their preimage $\pi^{-1}(\{1\})$, hereafter abbreviated to $\pi^{-1}(1)$. A *linear* predicate is defined on E^n as $\pi(x) \triangleq (a^{\mathrm{T}}x + a_0 \leq 0)$, where $a \in E^n$ and $a_0 \in \mathbb{R}$ [19] and $\pi^{-1}(1)$ is a half-space. A *convex polytopic* predicate $\pi(x) \triangleq \bigwedge_{i\in I} \pi_i(x)$ where π_i are linear predicates and $i \in I \subset \mathbb{N}, |I| < \infty$, implying that set $\pi^{-1}(1)$ is a convex polytope $\{x \in E^n | \; Ax \leq b\}, A \in \mathbb{R}^{r\times n}, b \in \mathbb{R}^r$, where inequality applies componentwise. A *polytopic* predicate $\pi(x) \triangleq \bigvee_{j\in J} \pi_j(x)$ where π_j are convex polytopic predicates indexed by a finite set $J \subset \mathbb{N}$. So set $\pi^{-1}(1)$ is a (possibly non-convex) polytope. The set of all polytopic predicates over E^n is denoted by $\mathcal{P}^n$. Any Boolean formula φ over a finite set of polytopic predicates $\Pi \subset \mathcal{P}^n$ defines a subset of E^n which is a polytope. Hereafter "predicate" refers to polytopic predicate. Given predicate π, a (convex) *partition* $\Pi \subseteq \mathcal{P}^n$ is a cover $\pi = \bigcup_{q\in\Pi} q, \emptyset \notin \Pi$ comprised of disjoint (convex) predicates.

2) Atomic Propositions: Define a set AP whose elements will be called *atomic propositions* (symbols), intuitively understood as symbolic variables. A *valuation* s of set AP is a predicate $s : AP \to \mathbb{B}$ assigning truth values to the symbols $a \in AP$. "Atomic" signifies that each $a \in AP$ is free to be mapped to any truth value in $\mathbb{B}$ independently of the other symbols. In contrast, a *proposition* defined as a formula over AP must be evaluated based on AP. Unlike predicates, which are functions, symbols are set elements.

3) Polytopic (or Continuous) Atomic Propositions: The symbols AP can be partitioned into primary AP^φ, which are part of the specification formula φ (section II-C), and derived AP^ψ, which are auxiliary, resulting from abstraction (e.g.transitions expressed in logic, or liveness annotations for augmented finite transition systems [20]). As part of the problem specification, we are given a bijection $f : AP^\varphi_c \subseteq AP^\varphi \to \Pi^\varphi$ to a set of predicates Π^φ. Then for any continuous state $x \in E^n$ we can construct a valuation function s_x by defining $s_x : AP^\varphi_c \to \mathbb{B} : a \mapsto s_x(a) \triangleq f(a)(x) = \pi_i(x)$. In this way each specification predicate π_i is represented at the discrete level. A partition Π' is *predicate-preserving* (symbolic) with respect to Π^φ, denoted $\Pi' \models \Pi^\varphi$, if for each $p \in \Pi'$, for all $\pi \in \Pi^\varphi$ either $p \cap \pi = \emptyset$ or $p \subseteq \pi$. The term proposition-preserving [10] is not used here because later symbols in AP^ψ_c are mapped to equivalence classes of predicates. Partition Π_1 refines Π', denoted as $\Pi_1 \preceq \Pi'$, if and only if (iff) $\forall \pi_1 \in \Pi_1 . \exists \pi_2 \in \Pi' . \pi_1 \subseteq \pi_2$ [21]. Thus if $\Pi' \models \Pi^\varphi$ (is symbolic), then so is $\Pi_1 \models \Pi^\varphi$.

B. Piecewise-Affine Linear Systems

A discrete-time time-invariant polytopic *piecewise-affine* (PWA) system with additive uncertainty is a finite set of tuples $\mathcal{W} \triangleq (\pi_i, A_i, B_i, E_i, K_i), i \in I_{pwa} \subset \mathbb{N}$ where $\pi_i \in \Pi \subset \mathcal{P}^n$, with semantics $x[k] \in \pi_i \implies x[k+1] = A_i x[k] + B_i u[k] + E_i \delta[k] + k_i$, where $x : \mathbb{N} \to \mathscr{X} \subseteq \mathbb{R}^n$ is the continuous system state, $k \in \mathbb{N}$ is discrete-time, $u : \mathbb{N} \to \mathbb{R}^p$ is the control input, $\delta : \mathbb{N} \to \mathbb{R}^m$ is the disturbance, $A_i \in \mathbb{R}^{n\times n}$ the system matrix, $k_i \in \mathbb{R}^{n\times 1}$ a drift term, and $B_i \in \mathbb{R}^{n\times p}, E_i \in \mathbb{R}^{n\times m}$. The control input is constrained in some polytope $\forall k \in \mathbb{N} \; . u[k] \in \mathscr{U}$ and the disturbance is also assumed to be in some polytope $\forall k \in \mathbb{N}, \; \delta[k] \in \mathscr{D}$. A *switched* PWA system $\mathcal{S} \triangleq \bigcup_{i\in I_{sw}} \{\mathcal{W}_i\}$. Each $\mathcal{W}_i(\mathscr{D})$ is a *mode* of $\mathcal{S}$, with $\mathscr{D}$ determined later.

C. Linear Temporal Logic

Linear temporal logic (LTL) [22] is used to reason about individual sequences in time. It extends the propositional connectives negation $\neg$, conjunction $\wedge$, and disjunction $\vee$ by the temporal operators next $\bigcirc$ and until $\mathcal{U}$, from which eventually $\Diamond$ and always $\Box$ are derived as $\Diamond p \triangleq (\mathrm{true})\,\mathcal{U} p$ and $\Box p \triangleq \neg\Diamond\neg p, p \in AP$. Atomic propositions are the terminal symbols of the logic. The set Φ of well-formed LTL formulæ can be defined recursively by $AP \subseteq \Phi$ and if $\varphi, \psi \in \Phi$, then $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \bigcirc\varphi, \varphi\,\mathcal{U}\psi \in \Phi$ [23].

D. GR(1) Fragment and Game Synthesis

A model comprises of symbolic variables in AP. By system we refer to the set $\mathcal{Y} \subseteq AP$ of *controlled* variables, whereas the set $\mathcal{X} \subseteq AP$ of *uncontrolled* variables is referred to as environment. A system which reacts to inputs from its environment by producing outputs is called *open*. Given an LTL formula φ and a model describing what behavior is feasible for the environment and system, reactive synthesis

produces a winning strategy represented as a Mealy machine $\mathcal{M}$ [9] for manipulating the controlled variables in order to satisfy φ however the environment plays. Game synthesis for LTL is 2ExpTime-Complete [24] in the size of the specification, whereas for the GR(1) fragment of LTL it is in PTime in the number of states [25]. We assume the specification φ is in GR(1) and use the `gr1c` solver [26]. Note that several LTL specifications can be converted to GR(1) form by introducing auxiliary variables.

E. Transition Systems

Definition 1 (Open Finite Transition System): A labeled graph $\mathcal{T} \triangleq (S, S_0, AP, L, Act_e, Act_s, T)$ where

- S is a finite set of states
- $S_0 \subseteq S$ is a subset of states labeled as initial
- AP is a finite set of atomic propositions
- $L : S \to 2^{AP}$ labels states with symbol words
- Act_e, Act_s are finite sets of environment and system actions, respectively, assuming that $Act_e \cap Act_s = \emptyset$.
- $T \subseteq S \times Act_e \times Act_s \times S$ a transition relation, possibly non-deterministic with respect to actions.

Let $\mathcal{T}_s, \mathcal{T}_e$ model the system and its environment, respectively. Each transition in Def. 1 can be assigned semantics

1) $\square\,(s_i \to \bigcirc\,(s_j \wedge act_e \wedge act_s))$ for $\mathcal{T}_s$, and
2) $\square\,(e_i \to act_s \wedge \bigcirc\,(e_j \wedge act_e))$ for $\mathcal{T}_e$,

where $(s_i, act_e, act_s, s_j) \in T_s$ and $(e_i, act_e, act_s, e_j) \in T_e$. Note that $\mathcal{T}_e$ knows only the previous system action [25].

III. Environment Partitioning

A. Problem Statement

The environment's continuous domain in our case is additive disturbance in $\mathscr{E} \subset \mathbb{R}^m$, though the proposed framework can be extended to systems with parametric uncertainty in A, using appropriate reachability and control algorithms. We assume a GR(1) specification $\varphi \in \Phi$ and a PWA system $\mathcal{W}_0(\mathscr{E})$ are given, with the disturbance δ regarded as controlled by the environment with bounded rate $\|\delta[k+1] - \delta[k]\|_{\ell_2} < K \in (0, +\infty)$.

Problem 2 (Environment partitioning): Compute a partition $\Pi_{\mathscr{E}} \subset \mathcal{P}^m$ of $\mathscr{E}$, a switched system $\mathcal{S}$ and a map $g : \Pi_{\mathscr{E}} \to I_{sw}$ from environment predicates to switching modes, an abstraction $\mathcal{T}_{\mathcal{S}}$ of $\mathcal{S}$, and a control strategy $\mathcal{M}$ on $\mathcal{T}_{\mathcal{S}}$ such that φ be satisfied.

Transitions in $\mathcal{M}$ can then be implemented using receding horizon control as in [8], because their feasibility is ensured during abstraction.

B. Neigboring Partitions differ by Perturbation

The control design process comprises of two stages: firstly the dynamics are abstracted using backward reachability analysis as in [10], [8] to obtain a transition system between polytopes of partitions Π_i of $\mathscr{X}$, then game synthesis [25] is used. In our case different system partitions Π_i are associated to different subsets of the environment domain $\mathscr{E}$.

Between bifurcations, the phase portrait of sufficiently smooth (Lipschitz) systems with parameters varies smoothly [27]. So we expect that polytopes from neighboring (defined later) partitions Π_i, Π_j can be paired based on their Hausdorff distance (intersection volume used later). This observation motivates mapping Π_i to Π_j to track the symbolic abstraction, utilizing metric information from the common underlying state space $\mathscr{X}$. This mapping can then be used to isomorphically map strategies.

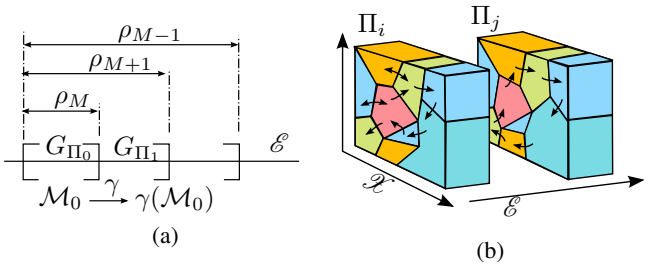


Fig. 2: (a) Mapping an initial strategy $\mathcal{M}_0$ to abstraction G_{Π_1} corresponding to an enlarged disturbance set $\rho_{M+1} \supseteq \rho_M$. (b) Joint environment and system state space partitioning.

Initially a symbolic partition $\Pi' \models \Pi^{\varphi}$ is derived from the specification predicates $\Pi^{\varphi} \leftrightarrow AP_c^{\varphi}$. All partitions hereafter are refinements of Π', therefore predicate-preserving with respect to Π^{φ}. The *adjacency* graph $\mathcal{A}_{\Pi_i}$ of a partition Π_i has edges $E \triangleq \{(\pi, \eta) \big| \overline{\pi} \cap \overline{\eta}\}$. A *reachability* digraph G_{Π_i} is defined with edges the feasible transitions between predicates.

The main steps of the proposed algorithm are the following. An initial predicate $\rho_M \in \mathcal{P}^m$ is found by bisection after M iterations $\rho_0, \rho_1, \ldots, \rho_M$, such that a strategy $\mathcal{M}_0$ exists on an abstraction G_{Π_0} of $\mathcal{W}_0(\rho_M)$. Then supersets $\rho_M \subseteq \rho_{M+1} \subseteq \cdots \subseteq \rho_{M-1}$ are computed by bisection between ρ_M and ρ_{M-1} (for which no strategy existed). For each ρ_i the system dynamics are abstracted into G_{Π_i} based on reachability computations using the algorithm of [10].

For each tuple (ρ_i, ρ_{i+1}) a partial bijection $r_{i,i+1} : \Omega_i \subseteq \Pi_i \to \Pi_{i+1}$ is computed, inducing a mapping γ from subgraphs over Ω_i to subgraphs over $r_{i,i+1}(\Pi_i)$. Note that the Mealy machine $\mathcal{M}_0$ has states that correspond to $\mathcal{T}_{\mathcal{S}}$ *unfolded* [23] with respect to the discrete variables of the model. Therefore the ("spatial") transitions of G_{Π_i} used by $\mathcal{M}_0$ are those of its projection $G_{\mathcal{M}_0,i} \triangleq \mathrm{proj}_{G_{\Pi_i}}(\mathcal{M}_0)$.

Definition 3: A *strategy isomorphism* ξ_{ij} is an isomorphism of $G_{\mathcal{M},i}$ to a subgraph of G_{Π_j}, contained in the relation r_{ij}. If a ξ_{ij} exists for (Π_i, Π_j), then we call G_{Π_i}, G_{Π_j} *strategically isomorphic*, denoted by $G_{\Pi_i} \simeq_{\mathcal{M}, r_{ij}} G_{\Pi_j}$.

If $\gamma(G_{\mathcal{M}_{0,i}}) \subseteq G_{\Pi_{i+1}}$, then strategy $\mathcal{M}_0$ can be used in ρ_{i+1}, so ρ_i, G_{Π_i} can be replaced by the enlarged mode $\rho_{i+1}, G_{\Pi_{i+1}}$. When γ fails to copy the strategy, a new predicate ρ_0' is initialized by shifting the interval in the scalar case, and the above procedure is repeated. By *neighboring partitions* we refer to Π_i, Π_j such that $\overline{\rho}_i \cap \overline{\rho}_j \neq \emptyset$.

The above procedure is repeated until $\mathscr{E}$ has been covered, yielding a partition $\Pi_{\mathscr{E}\times\mathscr{X}}$ comprised of predicates $\pi_i \times \rho_j$ for $\pi_i \in \Pi_j, \rho_j \in \Pi_{\mathscr{E}}$, as schematically shown in Fig. 2b.

Besides enlarging a single predicate ρ_i, mappings between partitions can be utilized also for relating partitions Π_i, Π_j associated to neighboring $\rho_i, \rho_j, i \neq j$. In that case inter-partition reachability must be taken into account as in

section VI-A, to find the deterministic restriction $\hat{r}_{ij}$ of r_{ij}. We can then distinguish three cases of synthesis. The first case is when all partitions Π_i are strategically isomorphic $G_{\Pi_i} \simeq_{\mathcal{M},r_{ij}} G_{\Pi_j}$ with respect to a common strategy $\mathcal{M}$, so we can treat them as a single mode at the discrete synthesis level, applying the required "restoration" control actions (section VI-A) at the continuous level to remain in identified predicates of different Π_i, Π_j as δ changes.

The second case concerns bifurcations between Π_i, Π_j, preventing use of a common intra-mode strategy. This requires resynthesis for each Π_i, to guide partitioning. In absence of liveness assumptions on environment switching, i.e., if the environment can remain infinitely long in any ρ_i, then existence of an intra-mode strategy is a necessary condition for the existence of a switched strategy. This avoids deriving useless modes that would definitely render the switched synthesis unrealizable. After partitioning is complete, a switched system synthesis is performed.

The third case is a variant where assumptions are known on how the environment evolves in $\mathscr{E}$ (e.g., we might know that an uncertain parameter representing power consumption reduces near zero levels during night). These translate to mode switching constraints, which imply that switching is essential to realizability, so intra-mode strategies might not exist. This can be addressed by performing strategy synthesis for the switched system as new modes are created. Then mode enlargement will use the projection of the *switched* strategy on that particular mode's partition. More efficient switched synthesis may utilize iterative extension, related to patching [28], but is outside the scope of this work.

The 1-dimensional case is outlined in Algorithm 1, which can be extended to more dimensions by using polytope bisection. The function TRYISO attempts to find a mapping γ (section V) with which to map the existing Mealy strategy $\mathcal{M}_i$ to another partition for an enlarged interval. In general as environment uncertainty δ is reduced, it becomes more likely that a strategy exists. Therefore Algorithm 1 searches the initial strategy by shrinking the environment interval around the origin. For systems with A_i parameterized by environment variables (instead of an additive disturbance δ) this is not true and an initial feasible point will need to either be known or determined by sampling the domain. Nonetheless in practical applications we are usually interested to explore the limits of a prototype or an existing system, which provides us with an initial feasible instance of the problem. This enlargement of the set of admissible environments is reminiscent of weakest preconditions [4], but on *continuous* environment variables.

Algorithm 1 Partition environment disturbance domain $[a, b]$

procedure ENVPARTITION($[a, b]$, ϵ, λ) ▷ $\lambda \in (0, 1)$
  $c \leftarrow a,\ d \leftarrow \frac{a+b}{2}$
  while ($|d - c| > \epsilon$) and $[a, b]$ not covered **do**
    ENLARGEINTERVAL($[c, d]$)
    $c \leftarrow c + (1 - \lambda)(d - c)$ ▷ shift with λ-overlap
    $d \leftarrow d + (1 - \lambda)(d - c)$
    if $(|b - d| < \epsilon) \vee (b < d)$ **then** $d \leftarrow b$
    end if
  end while
end procedure
procedure ENLARGEINTERVAL($[c, d], \epsilon$)
  SHRINK($[c, d]$) ▷ fix c, halve $[c, d]$ until realizable
  $\Pi_i, G_i \leftarrow$DISCRETIZE($\mathcal{W}_0(\mathscr{D})$, $\mathscr{D} \triangleq [c, d]$)
  $\mathcal{M} \leftarrow$GR(1)-SYNTHESIS(Π_i, G_i, φ)
  $l \leftarrow c, a \leftarrow d,\ b \leftarrow 2d,\ c \leftarrow \frac{a+b}{2},\ c' \leftarrow b$
  while $|c - c'| > \epsilon$ **do**
    $\Pi'_i, G'_i \leftarrow$DISCRETIZE($\mathcal{W}_0(\mathscr{D})$, $\mathscr{D} \triangleq [l, c]$)
    $\mathcal{M}' \leftarrow$TRYISO(Mealy, $(\Pi_i, G_i), (\Pi'_i, G'_i)$)
    if $\mathcal{M}' \neq \emptyset$ **then**
      $a \leftarrow c,\ \mathcal{M} \leftarrow \mathcal{M}',\ \Pi_i \leftarrow \Pi'_i,\ G_i \leftarrow G'_i$
    else
      $b \leftarrow c$ ▷ Subdivide $[a, b] \rightarrow [a, c]$
    end if
    $c' \leftarrow c,\ c \leftarrow \frac{a+b}{2}$
  end while
end procedure

C. Assumptions on environment about switching delays

To derive the partial bijections r_{ij} (sections IV and V) we compute a relation between overlapping polytopes from neighboring partitions Π_i, Π_j. Suppose polytope $\pi_p \in \Pi_i$ intersects multiple polytopes $c_j(\pi_p) \triangleq \{\pi_q \in \Pi_j | \pi_q \cap \pi_p \neq \emptyset\}$. If $x \in \pi_p$ and the environment switches from ρ_i to ρ_j, then next $\bigvee_{\pi \in c(\pi_p)}(x \in \pi)$. This introduces uncontrolled nondeterminism. To alleviate this effect, we employ inter-partition reachability checks (section VI) that determine whether the symbolic state $s \triangleq f_i^{-1}(\pi_p) \in AP_c^{\psi}$ before a switch can be *restored* by a continuous move from $\pi_q \in c_j(\pi_p)$ to $f_j(s)$ after the switch. Such a restoration action is typically of short duration, because polytopes of neighboring partitions are perturbed versions of each other, and requires the assumption of sufficient *settling time* T_m for transients (motivated by reasons similar to average dwell-time [29]), before the next mode switch occurs, i.e., hysteresis.

In order to ensure a sufficiently long T_m, $\Pi_{\mathscr{E}}$ is constructed as a cover with overlapping predicates ρ_i. The amount of overlap is inversely proportional to the maximal environment speed $\|\delta[k+1] - \delta[k]\|_{\ell_2} < K$, taking into account also the control authority on the system. For the 1-d case this reduces to sufficient interval overlaps, as determined by λ in Algorithm 1. It also avoids Zeno switching.

IV. BIPARTITE GRAPH OF INTERSECTION VOLUMES

Metric information can be encoded in a graph to be utilized for constructing a mapping between two neighboring partitions Π_A, Π_B. If initially $\delta[k] \in \rho_A$ and $x[k] \in a_i \in \Pi_A$, then after a mode switch to $\delta[k+1] \in \rho_B$, the system state $x[k+1] = x[k]$. So a switch can cause a transition from polytope $a_i \in \Pi_A$ only to polytopes $b_j \in c_B(a_i) \subseteq \Pi_B$, its minimal cover from Π_B, which we need to compute.

We use this information to construct a bipartite graph encoding volume overlaps, which is used in section V for matching. A greedy algorithm which exhaustively checks all possible intersections is used in [10] for merging partitions, with time complexity $O(N_A N_B)$ where $N_i \triangleq |\Pi_i|$. If

$N_A \approx N_B$ this implies complexity $O(N_A^2)$. The algorithm described next has complexity $O(N_A + N_B)$, whose constant is bounded by the size of the largest polytope cover, which typically depends exponentially on the dimension n, as the number of neighbors scales. The proposed algorithm is close to optimal for computing all possible intersections, because it processes all the non-empty intersections and not more, except for the boundary of them. Let $c_{ij} \triangleq a_i \cap b_j$.

Proposition 4: If $i \neq j$. $p_i, p_j \in \Pi_A$ and $w_k, w_r \in \Pi_B$, then $c_{ik} \cap c_{jr} = \emptyset$.

Proof: By hypothesis Π_A is a partition, so $p_i \cap p_j = \emptyset$, so $(p_i \cap w_k) \cap (p_j \cap w_r) = (p_i \cap p_j) \cap (w_k \cap w_r) = \emptyset$. ■

The proposed algorithm proceeds by local breadth-first search (BFS) over the spatial neighbors in order to cover each polytope and its proof uses induction. The adjacencies $\mathcal{A}_{\Pi_A}, \mathcal{A}_{\Pi_B}$ are needed and computed in linear time during discretization, because partitioning iteratively subdivides polytopes, tracking neighbor information. The algorithm must be restarted for each connected component of $\mathscr{X}$.

Given an initial polytope $a_i \in \Pi_A$, a polytope $b_j \in \Pi_B$ such that $a_i \cap b_j \neq \emptyset$ can always be found in time $O(N_B)$, because both Π_A, Π_B cover $\mathscr{X}$. The pair (a_i, b_j) seeds the algorithm. Searching for such pairs is avoided in subsequent iterations. This search can be avoided by tracking an overlapping pair throughout partitioning, possible because both Π_A, Π_B refine Π', so $\Pi_A \models \Pi^{\varphi}$ and $\Pi_B \models \Pi^{\varphi}$.

Next the algorithm proceeds locally. If $a_i \setminus b_j = \emptyset$ then $a_i \subseteq b_j$, so $c_B(a_i) = \{b_j\}$ and we can proceed with another polytope in Π_A that remains uncovered. Otherwise $a_i \setminus b_j \neq \emptyset$ and next we cover a_i without searching through all of Π_B.

This can be achieved by local breadth-first search over the $\mathcal{A}_{\Pi_B}$-neighbors of b_j. The search is localized by ignoring any $\pi \in \Pi_B$ such that $\overline{\pi} \cap \overline{a_i} = \emptyset$. The search results are then grouped as in Algorithm 2 into interior $\mathring{C}_i$ and boundary $\overline{C}_i$ polytopes of the cover $c_B(a_i)$. Note that computationally interior polytopes may touch ∂a_i, for which reason $\mathring{C}_i \setminus \partial C_i$ is used when updating cover $\hat{\Pi}_B$.

The bounded search always finds a cover because $\mathcal{A}_{\Pi_B}$ is connected and both Π_A, Π_B cover $\mathscr{X}$. In other words, if during the depth-bounded BFS a_i remains uncovered, then there exists some neighbor $\pi \in \Pi_B$ with $(\pi, \eta) \in \mathcal{A}_{\Pi_B}$ for some $\eta \in \hat{c}_B(a_i)$ (cover under construction) such that $\pi \cap a_i \neq \emptyset$, so $\pi \in c_B(a_i)$. Hence $c_B(a_i)$ will be obtained computing $a_i \cap \theta$ only for polytopes θ with distance from $c_B(a_i)$ at most 2 in $\mathcal{A}_{\Pi_B}$, so each cover $c_B(a_i)$ can be found in time at most $|c_B(a_i)|\,\Delta(\mathcal{A}_{\Pi_B})$ (Δ the maximal degree).

Pairing a_i with yet uncovered boundary polytopes $\pi \in \partial c_B(a_i)$ yields new seeds (a_i, π) fed to Algorithm 2, with π as target (Π_A, Π_B are swapped). The reduction in complexity is achieved by alternation between $\mathcal{A}_{\Pi_A}, \mathcal{A}_{\Pi_B}$ and local confinement of the search, proceeding via 1-hop neighbors.

The covers are encoded in a weighted bipartite digraph G_{Π_A,Π_B} over $\Pi_A \cup \Pi_B$, called the *intersection graph*, where an edge (π, ζ) exists iff $\pi \cap \zeta \neq \emptyset$ and is annotated with the volume $|\pi \cap \zeta|$ and rank $r(\pi, \zeta) \in \mathbb{N}^*$ of ζ in $\tilde{c}_i(\pi)$, which is $c_i(\pi)$ sorted in decreasing $|\pi \cap \zeta|$ order. We make the genericity assumption that in each $\tilde{c}_i(\pi)$ no ties occur.

Algorithm 2 Cover Polytope a_i, seeded by $b_j \cap a_i \neq \emptyset$

procedure COVERPOLY($(a_i, b_j), \{\hat{\Pi}_k, G_{\Pi_k}\}_{k\in\{A,B\}}$)
  $C_i \leftarrow$ BOUNDEDBFS((a_i, b_j), G_{Π_B}, Π_A, Π_B)
  $\mathring{C}_i \leftarrow$ FINDINTERIOR(C_i, a_i) ▷ $\{\pi \in C_i | \pi \subseteq a_i\}$
  $\partial C_i \leftarrow$ FINDBOUNDARY(C_i, a_i, ϵ)
  ▷ $\{\pi \in C_i | \pi_\epsilon \subsetneq a_i\}$, where π_ϵ is π shrunk by $\epsilon \ll 1$
  $\hat{\Pi}_A \leftarrow \hat{\Pi}_A \cup (a_i, C_i)$ ▷ update A covers
  $\hat{\Pi}_B \leftarrow \hat{\Pi}_B \cup \bigcup_{\pi \in \mathring{C}_i \setminus \partial C_i} (\pi, \{a_i\})$ ▷ Update B covers
  for $\pi_k \in \partial C_i$ **do**
    $\partial C_i \leftarrow \partial C_i \cup \{q \in G_{\Pi_B}(\pi_k, \cdot) | \, \overline{q} \cap \overline{a}_i \neq \emptyset\}$
  end for
  $seeds \leftarrow \bigcup_{\pi \in \partial C_i \setminus \mathrm{proj}_1(\hat{\Pi}_B)} (a_i, \pi)$
  return $\hat{\Pi}_A, \hat{\Pi}_B, seeds$
end procedure

V. MATCHING NEIGHBORING PARTITIONS

The graph of intersections derived in section IV is used next to match regions from a partition Π_i to those of its neighbor Π_j. The result is going to be used in section VI-A where each match is characterized as deterministic or not based on reachability and additional edges added if needed.

The *circumference* $\mathrm{circ}(G)$ of a directed graph G is the length of the longest simple directed cycle in G. Consider the *first-choice* subgraph $G^1_{\Pi_i,\Pi_j}$ of G_{Π_i,Π_j} obtained by removing all edges $r(p, q) > 1$ (i.e., keep only first choices). Our instance of matching differs from the general case in that $\mathrm{circ}(G^1_{\Pi_i,\Pi_j}) = 2$, as we now prove. Notice that this is the shortest possible circumference in a bipartite digraph, analogous to self-loops in non-bipartite graphs.

Proposition 5: Graph $G^1_{\Pi_i,\Pi_j}$ has $\mathrm{circ}(G^1_{\Pi_i,\Pi_j}) = 2$.

Proof: Suppose $G^1_{\Pi_i,\Pi_j}$ contains a cycle $A_1 \to B_1 \to A_2 \to B_2 \to \cdots A_M \to B_M \to A_1$ with $M > 1$, where $A_k \in \Pi_i, B_k \in \Pi_j$. For any triple $A_k \to B_k \to A_{k+1}$ by construction A_{k+1} is the first choice of B_k, meaning that $A_{k+1} = \arg\max_{A\in\Pi_i}\{|A \cap B_k|\}$, so $|A_k \cap B_k| < |A_{k+1} \cap B_k|$ (ties non-generic). So $|A_1 \cap B_1| < |A_2 \cap B_1| < |A_2 \cap B_2| < \cdots < |A_M \cap B_M| < |A_1 \cap B_M| < |A_1 \cap B_1|$ which is a contradiction. Therefore $G^1_{\Pi_i,\Pi_j}$ contains no directed (simple) cycles of length larger than 2. ■

The above observation leads to adapting the classic Gale-Shapley [30] algorithm and proving that our matching problem has a *unique* stable matching (not true in general).

Theorem 6: There exists a unique stable matching for the intersection graph G_{Π_i,Π_j}.

Proof: Uniqueness is proved by induction as follows. A *blocking edge* $(\pi, \zeta) \in G^1_{\Pi_i,\Pi_j}$ is one for which also $(\zeta, \pi) \in G^1_{\Pi_i,\Pi_j}$, but (π, ζ) is not in the current matching. By hypothesis Π_i, Π_j are partitions of $\mathcal{X}$, so $|c_j(\pi)| \geq 1 \implies \deg(\pi) \geq 1$ for each $\pi \in \Pi_i$ and vice versa. Then by assumption of no ties for all π in $G^1_{\Pi_i,\Pi_j}$ the outdegree $\deg^+(\pi) = 1$. Then by Prop. 5 graph $G^1_{\Pi_i,\Pi_j}$ comprises of linear directed paths each ending in a directed 2-cycle.

In the first iteration, the pairs of these 2-cycles are assigned to each other, because they would be blocking in *any*

matching. So they are part of *all* stable matchings, allowing us to remove them from $G^1_{\Pi_i,\Pi_j}$, denote by $Q^1_{\Pi_i,\Pi_j}$ the result.

This implies that any stable matching must be stable when restricted to the remaining vertices. So any pair that is blocking in this subset will be stable in any stable matching, so it can be assigned to each other and removed.

If a $\pi \in Q^1_{\Pi_i,\Pi_j}$ was in a removed edge (π,ζ), then now $\deg^+(\pi) = 0$. We add edges from each such $\pi \in Q^1_{\Pi_i,\Pi_j}$ to its second preference $r(\pi,\eta) = 2$ (lower for later iterations).

If a new edge introduces a 2-cycle, then we have a pair that is blocking for the current subset of vertices, so it is stable (as noted above) and will be removed at the end of this iteration. Otherwise we have to prove that the new edge does not introduce a cycle of larger length, so that after addition of edges $Q^1_{\Pi_i,\Pi_j}$ will comprise of directed paths ending at 2-cycles. By Prop. 5 it is sufficient to prove that the new edge (π,η) will have $|\pi \cap \eta| > |\pi \cap z|$ for each z with an edge (z,π). Suppose that $|\pi \cap \eta| < |\pi \cap z|$ for some z. Then $r(\pi,\eta) > r(\pi,z)$, so π must have already connected to z in a previous iteration, which would have introduced a 2-cycle, leading to removal of π, z as a stable pair. This contradicts that π, z are still in $Q^1_{\Pi_i,\Pi_j}$, proving that $\mathrm{circ}(Q^k_{\Pi_i,\Pi_j}) \leq 2$ is an inductive invariant (minimal possible bipartite girth). Because edge addition also restored $\deg^+(\pi) = 1$, either $Q_{\Pi_i,\Pi_j} = \emptyset$, or there exists at least one stable pair to remove. By induction the graph's size reduces in each iteration, while the removed pairs are part of any stable matching. So eventually the iterations terminate and *all* removed pairs belong to all stable matchings, which implies that there exists a unique stable matching. ■

Moreover, the above provides us with an adapted algorithm for our case. The time complexity is linear in the longest path created during these iterations, which in our case is typically small, because for many predicates $(\pi,\zeta) \in G^1_{\Pi_i,\Pi_j} \implies (\zeta,\pi) \in G^1_{\Pi_i,\Pi_j}$, because they are perturbed versions of each other. Thus most pairs are removed in the first iteration, requiring overall near-linear time.

For $|\Pi_i| \neq |\Pi_j|$ (unbalanced G_{Π_i,Π_j}) the above yields a unique stable sub-matching. We assign each unmatched predicate (associated to a vertex bifurcation) to its first preference. Denote by M_{ij} the resulting bipartite digraph, which has an edge (π,ζ) iff π has been assigned ζ (if π was unmatched, then $(\zeta,\pi) \notin M_{ij}$).

VI. Modeling Bifurcations

A. Inter-Partition Reachability

The matching graph M_{ij} (section V) encodes significant overlaps between polytopes and by construction $\deg^+(\pi) = 1$ for all $\pi \in M_{ij}$. Note that φ is interpreted over a subsequence k_l of discrete time k determined by transition completion events, let k_1 be the time at the start of a transition and k_2 after its completion. Let $(\pi,\zeta) \in M_{ij}$. If $x[k_1] \in \pi \in \Pi_i, \delta[k_1] \in \rho_i$ and the environment switches to $\delta[k_2] \in \rho_j$, then we would like to ensure that $x[k_2] \in \zeta$. It is true that $x[k_2] \in \eta \cap \pi$ for some $\eta \in c_j(\pi)$, but not necessarily $\eta = \zeta$. To ensure this we have to allow for control actions that restore the state $x[k_2]$ to ζ, when it happens that $\eta \neq \zeta$.

Algorithm 3 Inter-partition reachability analysis

```
procedure RestoreSwitch(M_ij, c̃_j, Π_𝒫, 𝒮)
    for (π, ζ) ∈ M_ij do
        S ← ζ ∪ Pre_{W_0(ρ_j),H}(ζ), R ← c̃_j(π) \ {ζ}
        while π ⊈ S do
            η ← Pop(c̃_j(π))
            S ← S ∪ η ∪ Pre_{W_0(ρ_j),H}(η), R ← c̃_j(π) \ {η}
            M_ij ← M_ij + (π, η)        ▷ Create edge (π, η)
        end while
    end for
end procedure
```

This is achieved by reachability analysis checking that $\pi \subseteq \zeta \cup \mathrm{Pre}_{\mathcal{W}_0(\rho_j),H}(\zeta)$, where Pre the controlled backward reachable set in mode ρ_j (after the switch). If for any $\pi \in \Pi_i \cup \Pi_j$ this does not hold, then we add to M_{ij} edges (π,η) until covering π with predicates η from $\tilde{c}_j(\pi)$ or their controlled backward reachable sets, using Algorithm 3. Denote by Ψ_{ij} the resulting graph, by construction $\deg^+(\pi) \geq 1$ for all $\pi \in \Pi_i \cup \Pi_j$. For those π that $\deg^+(\pi) > 1$ we know from Algorithm 3 that these transitions are nondeterministic and controlled by the environment (determined by where $x[k]$ happens to be in π when the environment switches). Transitions with $\deg^+(\pi) = 1$ are deterministic, because there exists a control action that restores the state to ζ.

Note that both $\Pi_i \preceq \Pi' \models \Pi^\varphi$ and $\Pi_j \preceq \Pi' \models \Pi^\varphi$, so if predicates $\pi \in \Pi_i, \eta \in \Pi_j$ intersect, $\pi \cap \eta \neq \emptyset$, then they are subsets of the same predicate $\xi \in \Pi'$ which preserves specification predicates in Π^φ. So we can use the common predicate $\xi \in \Pi'$ for which both $\pi \subseteq \xi$ and $\eta \subseteq \xi$ as the constraint set within which the transition trajectory must remain, to be predicate-preserving itself. Using ξ instead of π as a trajectory constraint was introduced in [10]. Returning to Ψ_{ij}, define its deterministic symmetric subgraph F_{ij}, containing (undirected) edge $\{\pi,\eta\}$ iff $(\pi,\eta) \in E_{\Psi_{ij}} \wedge (\eta,\pi) \in E_{\Psi_{ij}} \wedge \deg^+(\pi) = 1 = \deg^+(\eta)$.

It follows that any switch $\rho_i \to \rho_j$ leads to $\pi \to \eta$ and $\rho_j \to \rho_i$ to $\eta \to \pi$ (after the restoration actions have been applied). So we can identify predicates π, η as belonging to the same symbolic equivalence class. (Note that there might exist some $\zeta \in \Pi_i \cup \Pi_j$ such that $(\zeta,\pi) \in \Psi_{ij}$, but by definition of F_{ij} it is $(\pi,\zeta) \notin \Psi_{ij}$.

Collect the predicates from all partitions into the auxiliary predicate set $\Pi^\psi \triangleq \bigsqcup_{i \in I_{sw}} \Pi_i$ and define an equivalence relation $\sim\colon \Pi^\psi \to \mathcal{C} \triangleq \mathrm{comp}(\bigcup_{i,j} F_{ij}) \subset \mathbb{N}_{<|\Pi^\psi|} : \pi \mapsto [\pi]_\sim$ that maps each predicate to the index of its connected component in F_{ij}. Select a bijection $\mu : \mathcal{C} \to AP^\psi_c$ from connected components to auxiliary symbols. We can now define an *abstraction map* as $h \triangleq \mu \circ \sim$, i.e., $\Pi^\psi \xrightarrow{\sim} (\Pi^\psi / \sim) = \mathcal{C} \xrightarrow{h} AP^\psi_c, \pi \xmapsto{\sim} [\pi]_\sim \xmapsto{\mu} \mu([\pi]_\sim)$. Therefore the inter-partition reachability analysis identifies the predicates for which we can remain in the same symbolic state $a \in AP^\psi_c$, despite the switching. In the special case that all resulting partitions are isomorphic, $G_{\Pi_i} \simeq G_{\Pi_{i+1}}$, and the graphs Ψ_{ij} fully deterministic, switching can be completely "masked" at the symbolic, logic synthesis level and handled completely

at the continuous level. We thus avoid merging partitions [10], that would create high fragmentation, sliver polytopes (numerically ill-conditioned) and neglect the structural similarities of the phase portrait between neighboring partitions.

B. Nondeterministic Transitions in Logic

It remains to model non-deterministic transitions (typically associated with vertex bifurcations of G_{Π_i,Π_j}) realistically in logic. Consider the non-deterministic transition caused by switching from mode $\delta[k_1] \in \rho_i$ to $\delta[k_2] \in \rho_j$ when $x[k_1] \in \pi \in \Pi_i$, leading to $x[k_2] \in \eta_s$ for some η_s in $P \triangleq \mathrm{Post}^1_{\Psi_{ij}}(\pi) \subseteq \Pi_j$. The switch is assumed (near) instantaneous and followed by a settling time sufficient to ensure (at the continuous level) that $x[k_2] \in \eta_s$.

A naive approach to model this mode switch in logic is

$$x[k_1] \in \pi \wedge \delta[k_1] \in \rho_i \implies \bigcirc((\delta[k_2] \in \rho_j) \wedge q) \quad (1)$$

where $q \triangleq \bigvee_{\eta_s \in P}((x[k_2] \in \eta_s) \wedge (e_l = s))$. where e_l is an auxiliary arbitrage environment variable that models the (uncontrolled) non-determinism and takes values indexing P. Note that the non-determinism is absent at the continuous level (modulo disturbances), and is an artifact of abstraction. To avoid it would require increasing the fidelity of partition Π_i by separating predicate π into polytopes $\pi \cap \eta_s$. The downsides of this are introducing more logic states that increase the complexity of game synthesis (observe how e_l is a "reusable" variable that locally resolves the non-determinism, thus doesn't substantially increase the discrete state space), and having to propagate the separation to partitions adjacent to Π_i and not adjacent to Π_j. Direct transitions between modes with $\overline{\rho}_i \cap \overline{\rho}_j \neq \emptyset$ is impossible. This motivates avoiding separation, confining each bifurcation between the modes where it appears.

The issue with eq. (1) is that it allows the environment counter-strategy forcing the system response $(\pi, \eta_0), (\eta_0, \pi))^\omega$ by repeatedly switching back and forth between ρ_i and ρ_j, and resolving the non-determinism via e_l (ω denotes infinite-time repetition analogous to Kleene star * [23]). To avoid this possibility, the switching is instead modeled by

$$q \triangleq \bigvee_{\eta_s \in P}\Big(e_l = s \wedge x[k_2] \in \eta_s \cup \mathrm{Post}^1_{G_{\Pi_j}}(\eta_s)\Big) \quad (2)$$

In words, this models that a switch does *not* move the system state, instead it switches the partition. It introduces all the transitions available from η_s to other predicates in Π_j, accounting for the fact that if after the switch $x[k_2] \in \eta_s$, then the system can react by transitioning from η_s to another state, because the switch already placed it inside η_s. Although longer counterexamples may still exist, they would imply relatively low controllability for horizon H.

VII. Simulation Example

The algorithms proposed in this paper have been implemented on top of the Temporal Logic Planning Toolbox (TuLiP) [31]. The algorithm is demonstrated on a simple example involving two water tanks, as shown in Fig. 3. The state $x \triangleq \begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathrm{T}}$ comprises of the tank water levels, the inputs are the input valve $u_1 \in [0, 3]$ and pump $u_2 \in [-3, 3]$, whereas the output $\delta \geq 0$ represents the uncertain consumption and is modeled as a continuous variable controlled by the environment. The system's dynamics are

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1[k] \\ u_2[k] \end{bmatrix} + \begin{bmatrix} 0 \\ -\delta \end{bmatrix}$$

The specification objectives are to initially start from some state $x[0] \in I \triangleq [3, 8] \times [5, 11]$, remain within the safe region $\forall k. x[k] \in S \triangleq [1, 9] \times [2.5, 19]$ and achieve satisfactory hydrostatic pressure infinitely often by raising the second tank's level to $x_2 \in T \triangleq [15, 18]$ ("top" in Fig. 3). So the specification is $\varphi \triangleq I \wedge \Box S \wedge \Box\Diamond T$.

The proposed algorithm was initialized over the δ domain $\mathscr{E} \triangleq [0, 1.6]$, with tolerance $\epsilon = 0.3$ and overlap ratio $\lambda = 0.1$ for interval convergence. It partitioned it into two modes, by first halving $[0, 1.6]$ to $[0, 0.8]$, finding a strategy for that disturbance range, then enlarging that to $[0, 1.2]$, computing a matching between $[0, 0.8]$ and $[0, 1.2]$ which did not suffice to map the strategy, then bisecting to $[0, 1.0]$ and converging to the first mode $\rho_0 \triangleq [0, 0.8]$ because $|1.0 - 1.2| < 0.3 = \epsilon$. A horizon $N = 7$ was used for reachability, up to spatial neighbors 2-hops away, with region area lower bounded by 1.5. Next it shifted ρ_0 to the interval $[0.72, 1.52]$ that overlaps ρ_0 by λ, extended it to $[0.72, 1.6]$ because $|1.52 - 1.6| < 0.3 = \epsilon$, found a strategy for it, and terminated with $\rho_1 = [0.72, 1.60]$, having covered $\mathscr{E}$. The overall iterative abstraction and synthesis lasted 683 seconds on a 2.5 GHz Intel Core i5 processor with 4GB RAM.

Fig. 4 shows the matching between the system partitions corresponding to each mode. Regions with solid colors have been perfectly matched by the adapted Gale-Shapley algorithm, whereas regions $\pi_{14}, \pi_{27}, \pi_{49}$ in Π_0 and $\pi_9, \pi_{12}, \pi_{18}, \pi_{19}, \pi_{26}, \pi_{32}, \pi_{37}, \pi_{45}, \pi_{49}, \pi_{53}, \pi_{58}, \pi_{60}$ are shown semi-transparent because they have been matched only weakly (i.e., they are excessive ones that remained unmatched and were assigned to the region they maximally overlap). Next reachability analysis determines which regions can reach their matching region in the other partition after a switch. For example region $\pi_{21} \in \Pi_1$ cannot reach region $\pi_1 \in \Pi_0$ to which it has been matched, so that transition becomes non-deterministic and the transition $\pi_{21} \to \pi_{15} \in \Pi_0$ is added to cover π_{21}. In contrast $\pi_1 \to \pi_{21}$ is deterministic, because $\pi_1 \subseteq \pi_{21}$.

A switched Mealy strategy with 76 states is successfully synthesized for φ using `gr1c` [26]. This should be contrasted with attempting to synthesize a non-switched strategy after abstracting the dynamics (with same discretization parameters) for a disturbance ranging over the initial interval $[0, 1.6]$. Such a strategy does not exist, whereas the proposed algorithm found a strategy because the controller is allowed to observe the continuous environment behavior with finer granularity. It is assumed that the system makes a continuous transition when the environment is outside the overlap of two neighboring δ-intervals, to allow the system sufficient time for completing its transition before the environment switches

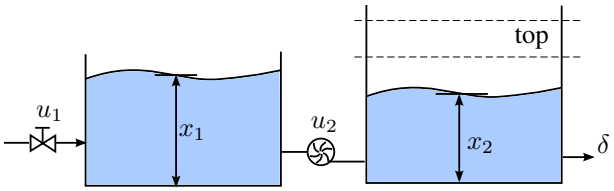
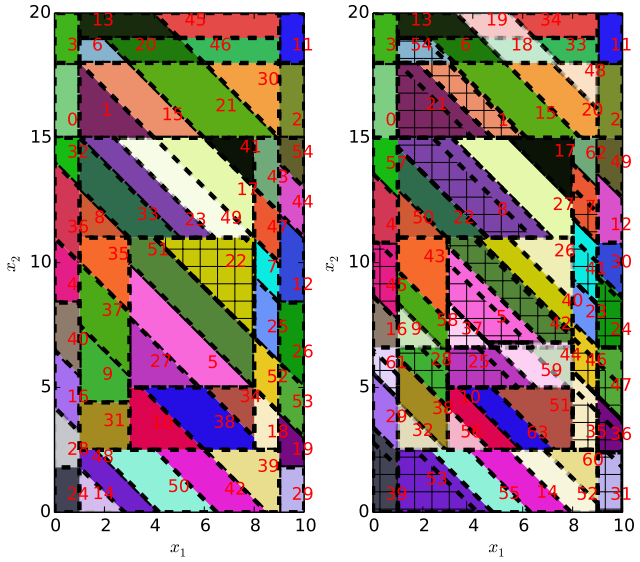


Fig. 3: Two water tanks with input valve u_1, pump u_2 and uncertain consumption δ.



(a) $\Pi_0, \delta \in [0.0, 0.8]$ (b) $\Pi_1, \delta \in [0.72, 1.6]$

Fig. 4: Unique stable marriage matching, where solid colors denote perfect matches and semi-transparent colors weak matches (section V). For non-deterministic regions, the hatched areas are $\zeta \cup \mathrm{Pre}_{\mathcal{W}_j,H}(\zeta)$ for the first choice ζ in $\tilde{c}_j(\pi)$ in Algorithm 3.

mode again.

VIII. Conclusions and Future Work

Future work aims at increasing the dimension of the environment state space, and considering continuous environment dynamics and their abstraction based on the switching modes derived as described here. Another direction concerns considering continuous parameterization of the system matrices A_i by the environment.

IX. Acknowledgment

This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. Larsen, "Contracts for System Design," INRIA, Tech. Rep. RR-8147, 2012.

[2] A. Benveniste, D. Nickovic, and T. Henzinger, "Compositional Contract Abstraction for System Design," INRIA, Tech. Rep. RR-8460, 2014.

[3] A. Cimatti, M. Dorigatti, and S. Tonetta, "Ocra: A tool for checking the refinement of temporal contracts," in *Aut. Soft. Eng. (ASE)*, 2013, pp. 702–705.

[4] E. W. Dijkstra, "Guarded commands, nondeterminacy and formal derivation of programs," *Commun. ACM*, vol. 18, no. 8, pp. 453–457, 1975.

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

[6] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[7] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-uav mission planning," *Int. J.Robust Nonlinear Contr.*, vol. 21, no. 12, pp. 1372–1395, 2011.

[8] T. Wongpiromsarn, U. Topcu, and R. Murray, "Receding horizon temporal logic planning," *IEEE Trans. on Aut. Contr.*, vol. 57, no. 11, pp. 2817–2830, 2012.

[9] R. Alur, S. Moarref, and U. Topcu, "Counter-strategy guided refinement of GR(1) temporal logic specifications," in *FMCAD*, 2013, pp. 26–33.

[10] P. Nilsson, N. Ozay, U. Topcu, and R. Murray, "Temporal logic control of switched affine systems with an application in fuel balancing," in *ACC*, June 2012, pp. 5302–5309.

[11] J. Liu, N. Ozay, U. Topcu, and R. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Trans. Aut. Contr.*, vol. 58, no. 7, pp. 1771–1785, 2013.

[12] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proc. HSCC*, 2013, pp. 1–10.

[13] K. Kim and G. Fainekos, "Approximate solutions for the minimal revision problem of specification automata," in *IROS*, Oct 2012, pp. 265–271.

[14] E. Aydin Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for discrete-time linear systems," in *HSCC*, 2012, pp. 95–104.

[15] M. Rungger, M. Mazo, Jr., and P. Tabuada, "Specification-guided controller synthesis for linear systems and safe linear-time temporal logic," in *HSCC*, 2013, pp. 333–342.

[16] U. Topcu, N. Ozay, J. Liu, and R. M. Murray, "On synthesizing robust discrete controllers under modeling uncertainty," in *HSCC*, 2012, pp. 85–94.

[17] R. Majumdar, E. Render, and P. Tabuada, "Robust discrete synthesis against unspecified disturbances," in *HSCC*, 2011, pp. 211–220.

[18] A. Girard and S. Martin, "Synthesis for constrained nonlinear systems using hybridization and robust controllers on simplices," *IEEE Trans. on Automatic Control*, vol. 57, no. 4, pp. 1046–1051, 2012.

[19] R. Alur, T. Dang, and F. Ivančic, "Counter-example guided predicate abstraction of hybrid systems," in *TACAS*, 2003, pp. 208–223.

[20] N. Ozay, J. Liu, P. Prabhakar, and R. Murray, "Computing augmented finite transition systems to synthesize switching protocols for polynomial switched systems," in *American Control Conference (ACC), 2013*, June 2013, pp. 6237–6244.

[21] J. R. Munkres, *Topology*. Prentice-Hall, 2000.

[22] A. Pnueli, "The temporal logic of programs," in *18^{th} Annual Symposium on Foundations of Computer Science*, 1977, pp. 46–57.

[23] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT, 2008.

[24] R. Alur and S. L. Torre, "Deterministic Generators and Games for LTL Fragments," in *Ann. IEEE Symp. Logic in Comp. Sc.*, ser. LICS '01, 2001, pp. 291–300.

[25] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Int. Conf. Verif., Model Checking, and Abstract Interpr. (VMCAI)*, 2006, pp. 364–380.

[26] S. Livingston and R. Murray, "Just-in-time synthesis for reactive motion planning with temporal logic," in *IEEE Int. Conf. Rob. Aut.*, May 2013, pp. 5048–5053.

[27] L. Perko, *Differential equations and dynamical systems*. Springer, 2001.

[28] S. Livingston, R. Murray, and J. Burdick, "Backtracking temporal logic synthesis for uncertain environments," in *ICRA*, 2012, pp. 5163–5170.

[29] J. Hespanha and A. Morse, "Stability of switched systems with average dwell-time," in *CDC*, vol. 3, 1999, pp. 2655–2660 vol.3.

[30] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, vol. 69, no. 1, pp. pp. 9–15, 1962.

[31] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "Tulip: a software toolbox for receding horizon temporal logic planning," in *HSCC*, 2011, pp. 313–314. [Online]. Available: http://www.cds.caltech.edu/tulip