An Iterative Abstraction Algorithm for Reactive Correct-by-Construction Controller Synthesis

Robert Mattila
Department of Automatic Control
Royal Institute of Technology (KTH)
Stockholm, Sweden
rmattila@kth.se

Yilin Mo
Department of Control and Dynamical Systems
California Institute of Technology
Pasadena, CA, USA
yilinmo@caltech.edu

Richard M. Murray
Department of Control and Dynamical Systems
California Institute of Technology
Pasadena, CA, USA
murray@cds.caltech.edu

ABSTRACT

In this paper, we consider the problem of synthesizing correct-by-construction controllers for discrete-time dynamical systems. A commonly adopted approach in the literature is to abstract the dynamical system into a finite transition system (FTS) and thus convert the problem into a two player game between the environment and the system on the FTS. The controller design problem can then be solved using synthesis tools for general linear temporal logic or generalized reactivity(1) (GR1) specifications. In this article, we propose a new abstraction algorithm. Instead of generating a single FTS to represent the system, we generate two FTSs, which are under- and over-approximations of the original dynamical system. We further develop an iterative abstraction scheme by exploiting the concept of winning sets, i.e., the sets of states for which there exists a winning strategy for the system. Finally, the efficiency of the new abstraction algorithm is illustrated by numerical examples.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods and Search**]: Control Theory; I.2.2 [**Automatic Programming**]: Program Synthesis

1. INTRODUCTION

The systems that are considered for control purposes have changed fundamentally over the last few decades. Driven by the advancements in computation and communication technologies, the systems of today have become highly complicated with large amounts of components and interactions, which proposes great challenges on controller design. This is exemplified in [17] where the controller for an autonomous vehicle grew so entangled that it was impossible to foresee the failure of it, resulting in a crash.

In order to tame the complexity of the modern control systems, synthesis of correct-by-construction control logic based on temporal logic specifications has gained a considerable amount of attention in the past few years. A commonly adopted approach is to construct an FTS which serves as a symbolic model of the original control system, which typically has infinitely many states. The controller, which is represented by a finite state machine, can then be synthesized to guarantee certain specifications on the system by leveraging formal synthesis tools [8]. Such a design procedure has been applied to various fields including robotics [4, 5, 2, 6, 3], autonomous vehicle control [16], smart-buildings [11] and aircraft power system design [7].

One of the main challenges of this approach is in the abstraction of the control system into finite state models. Zamani et al. [19] propose an abstraction algorithm based on approximate simulation relations and alternating approximate simulation relations. They prove that if certain continuity assumptions on the system trajectory hold, then an FTS can be generated by partitioning the state space into small hypercubes. Furthermore, the original system is approximately simulated by the FTS. Similar ideas are also presented in [12] and [13].

Wongpiromsarn et al. [16][15] propose an iterative approach to first generates a coarse model of the original system and then refine the model based on reachability computations, which has been implemented in a Python software package, namely `TuLiP` [18].

Most of the algorithms proposed in the literature generate the finite state model independently of the system specifications. As such, the abstracted model can be used for any possible specification. However, they typically tend to partition the state space down to equally fine regions everywhere. As a consequence, the time complexity of such general abstraction procedures is quite high and it increases dramatically with the dimension of the system.

In this article, in hope to reduce the computational complexity, we create the finite state models of the system by exploiting the structure of the specifications. To be specific, we will create two finite transition system models for the control system, where one is an over-approximation of the control system and the other is an under-approximation. By solving the synthesis problem on both FTSs, we can categorize the points in the state space into *winning, losing* and *maybe* sets. Conceptually, the *winning* set contains those points for which a "correct" controller is known, i.e., roughly, a controller that can fulfill the given specifications. On the other hand, the *losing* set contains those points for which we know that no "correct" controller exists. Lastly, the *maybe* set represents the points for which the existence of a "correct" controller is *not yet known* since the current model is not fine enough to represent the original system. One can view the *winning* and *losing* set as the "solved" region and the *maybe* set as the "unsolved" region. We can thus focus our computational power on refining the *maybe* set, while leaving the current *winning* and *losing* set intact.

The main merits of our proposed algorithm are twofold:

1. Instead of partitioning the state space into equally fine regions, we can concentrate the computational power on the "unsolved" region, while leaving the "solved" region intact.
2. Comparing to the abstract algorithm proposed in [16, 15, 18], for the case that the specifications are unrealizable, our algorithm can provide a proof that no "correct" controller exists.

The rest of the paper is organized as follows: in Section 2, we provide an introduction to transition systems and linear temporal logic. The problem of abstracting a discrete-time control system into FTSs is proposed in Section 3. The abstraction algorithm is then discussed in Section 4. Two numerical examples are provided in Section 5 to illustrate the effectiveness of the proposed algorithm. Finally, Section 6 concludes the paper.

2. PRELIMINARIES

This section introduces the notation and concepts that will be used for the rest of the article. We adopt the notation of [16] for consistency with previous work in the field. Thus, most of the definitions can be found in [16]. However, they are included in this section for the sake of completeness. For a more thorough presentation of the concepts, see [1].

2.1 Transition Systems and Linear Temporal Logic

A central concept in this paper is that of a system:

Definition 1. A *system* consists of a set V of variables. The *domain* of V, denoted by $dom(V)$, is the set of valuations of V. A *state* of the system is an element $v \in dom(V)$.

In this paper, we consider a system with a set $V = S \cup \mathcal{E}$ of variables. The domain of V is given by $dom(V) = dom(S) \times dom(\mathcal{E})$, where a state $\varsigma \in dom(S)$ is called the *controlled state* and a state $e \in dom(\mathcal{E})$ the uncontrolled *environmental state*. As a result, the state v can be written as (ς, e). We further assume that the set $dom(\mathcal{E})$ is finite.

Definition 2. A *transition system* (TS) is a tuple $\mathbb{T} := (\mathcal{V}, \mathcal{V}_{init}, \rightarrow)$ where $\mathcal{V} \subseteq dom(V)$ is a set of states, $\mathcal{V}_{init} \subseteq \mathcal{V}$ is a set of initial states and $\rightarrow \subseteq \mathcal{V} \times \mathcal{V}$ is a transition relation. Given states $\nu_i, \nu_j \in \mathcal{V}$, we write $\nu_i \rightarrow \nu_j$ if there is a transition from ν_i to ν_j in $\mathbb{T}$. We say that $\mathbb{T}$ is a *finite transition system* (FTS) if $\mathcal{V}$ is finite.

Definition 3. An *atomic proposition* is a statement on system variables ν that has a unique truth value for a given value of ν. Letting $\nu \in dom(V)$ and p be an atomic proposition, we write $\nu \models p$ if p is true at the state ν.

To formulate specifications on a system, we will use linear temporal logic (from here on referred to as LTL), which is an extension of normal logic that introduces additional temporal operators. Apart from the standard logical operators negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and implication ($\Rightarrow$), it includes the temporal operators next ($\bigcirc$), always ($\Box$), eventually ($\Diamond$) and until ($\mathcal{U}$). LTL formulas are defined inductively as

1. Any atomic proposition p is an LTL formula.
2. Given the LTL formulas φ and ψ; $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi \,\mathcal{U}\, \psi$ are LTL formulas as well.

Definition 4. The *satisfaction relation* $\models$ between an execution $\sigma = \nu_0\nu_1\ldots$ and an LTL formula is defined inductively as

- $\sigma \models p$ if $\nu_0 \models p$.
- $\sigma \models \neg\varphi$ if σ does not satisfy φ.
- $\sigma \models \varphi \vee \psi$ if $\sigma \models \varphi$ or $\sigma \models \psi$.
- $\sigma \models \bigcirc\varphi$ if $\nu_1\nu_2\ldots \models \varphi$.
- $\sigma \models \varphi \,\mathcal{U}\, \psi$ if there exists an $i \geq 0$, such that $\nu_i\nu_{i+1}\ldots \models \psi$ and for any $0 \leq k < i$, $\nu_k\nu_{k+1}\ldots \models \varphi$.

For a more in depth explanation of LTL, see [1].

It is well known that the complexity of synthesizing a controller for a general LTL formula is double exponential in the length of the given specification [9]. However, for a specific class of LTL formulas, namely those known as *Generalized Reactivity(1)* (GR1) formulas, an efficient polynomial time algorithm [8] exists. As a result, in this article, we will restrict the specification φ to be a GR1 formula, which takes the following form:

$$\varphi = \bigwedge_{i=1}^{M} \Box\Diamond p_i \rightarrow \bigwedge_{j=1}^{N} \Box\Diamond q_j, \tag{1}$$

where each p_i, q_j is a boolean combination of atomic propositions.

2.2 Winning Controllers and Winning Sets

Definition 5. A *controller* for a transition system $(\mathcal{V}, \mathcal{V}_{\text{init}}, \to)$ and environment $\mathcal{E}$ is an ordered set of mappings $\gamma_t : \mathcal{S} \times \mathcal{E}^t \to \mathcal{S}$,

$$\gamma \triangleq (\gamma_1, \gamma_2, \dots, \gamma_t, \dots),$$

each taking the initial controlled state $\varsigma[0]$ and all the environmental actions up to time $t-1$, $e[0]\dots e[t-1]$, giving another state in $\mathcal{S}$ as output. Furthermore, a controller γ is called *consistent* if for all t and $\varsigma[0]$, $e[0]$, $\dots$, $e[t+1]$, the following transition relation is satisfied:

$$(\gamma_t(\varsigma[0], e[0], \dots, e[t-1]), e[t]) \\ \to (\gamma_{t+1}(\varsigma[0], e[0], \dots, e[t]), e[t+1]).$$

Definition 6. Given an infinite sequence of environmental states $e[0]e[1]\dots$, a *controlled execution* σ using the controller γ and starting at $\varsigma[0]$ is an infinite sequence

$$\sigma = \nu_0\nu_1 \dots = (\varsigma[0], e[0])(\varsigma[1], e[1])\dots,$$

such that $\varsigma[t+1] = \gamma_t(\varsigma[0], e[0], \dots, e[t+1])$.

Definition 7. A set of controlled states $\mathcal{W}$ is *winning* if there exists a consistent controller γ, such that for any infinite sequence of $e[0]e[1]\dots$ and any initial controlled state $\varsigma[0] \in \mathcal{W}$, the controlled execution σ using controller γ starting at $\varsigma[0]$ satisfies φ. The corresponding controller γ is called a *winning controller* for $\mathcal{W}$.

The following observations are important for the rest of the paper:

PROPOSITION 1. *Let $\{\mathcal{W}_i\}_{i\in\mathcal{I}}$ be a collection of winning sets, then the set $\bigcup_{i\in\mathcal{I}} \mathcal{W}_i$ is also winning.*

PROOF. Let us define an index function $h : \bigcup_{i\in\mathcal{I}} \mathcal{W}_i \to \mathcal{I}$, such that for any $\varsigma \in \bigcup_{i\in\mathcal{I}} \mathcal{W}_i$, the following set inclusion holds:

$$\varsigma \in \mathcal{W}_{h(\varsigma)}.$$

Now assume that the winning controller for the set $\mathcal{W}_i$ is $\gamma^{(i)} = (\gamma_1^{(i)}, \gamma_2^{(i)}, \dots, \gamma_t^{(i)}, \dots)$. We can define the new controller $\gamma = (\gamma_1, \gamma_2, \dots)$ as

$$\gamma_t(\varsigma[0], e[0], \dots, e[t-1]) = \gamma_t^{(h(\varsigma[0]))}(\varsigma[0], e[0], \dots, e[t-1]).$$

It is easily verified that γ is a winning controller for $\bigcup_{i\in\mathcal{I}} \mathcal{W}_i$. $\square$

As a result, there exists a largest winning set, which leads to the following definition:

Definition 8. The *largest winning set*, W, of a transition system $\mathbb{T}$, for the specification φ, is defined as the union of all winning sets, i.e.,

$$W(\mathbb{T}, \varphi) = \bigcup_{\mathcal{W} \text{ is winning}} \mathcal{W}. \tag{2}$$

The *losing set*, L, is defined as

$$L(\mathbb{T}, \varphi) = dom(S) \setminus W(\mathbb{T}, \varphi). \tag{3}$$

A state ς is called a *losing state* if $\varsigma \in L(\mathbb{T}, \varphi)$.

Remark 1. Notice that the controllers defined in Definition 5 have infinite memory (since they require all environmental actions $e[0]e[1]\dots$). However, from [8], we know that for a *finite* transition system, if a winning controller exists, there will also exist a winning controller with finite memory.

3. PROBLEM FORMULATION

We consider the following discrete-time control system:

$$\begin{aligned} s[t+1] &= f(s[t], u[t]), \\ u[t] &\in U,\ s[t] \in dom(S), \\ s[0] &\in S_{\text{init}}, \end{aligned} \tag{4}$$

where $dom(S) \subseteq \mathbb{R}^n$, $S_{\text{init}} \subseteq dom(S)$ is the set of possible initial states, $U \subseteq \mathbb{R}^m$ is the admissible control set and f the system dynamics (possibly non-linear). It is evident that the discrete-time control system is completely characterized by f, U, $dom(S)$ and S_{init}, which leads to the following formal definition:

Definition 9. A *discrete-time control system* Σ is a quadruple $\Sigma \triangleq (f, U, dom(S), S_{\text{init}})$.

A discrete-time control system Σ can be converted into a transition system in the following manner:

Definition 10. Let $\Sigma \triangleq (f, U, dom(S), S_{\text{init}})$ be a discrete-time control system. The transition system $TS(\Sigma) = (\mathcal{V}, \mathcal{V}_{init}, \to)$ associated with Σ is defined as:

- $\mathcal{V} = dom(S) \times dom(\mathcal{E})$.
- $\mathcal{V}_{init} = S_{\text{init}} \times dom(\mathcal{E})$.
- For any $(s_1, e_1), (s_2, e_2) \in V$, $(s_1, e_1) \to (s_2, e_2)$ if and only if there exists $u \in U$, such that $s_2 = f(s_1, u)$.

The problem of controller synthesis for the discrete-time control system Σ can be written as a controller synthesis problem for $TS(\Sigma)$ as follows:

PROBLEM 1. Realizability: *Given $TS(\Sigma)$ and a specification φ, decide whether S_{init} is a winning set.*

PROBLEM 2. Synthesis: *Given $TS(\Sigma)$ and a specification φ, if S_{init} is winning, construct the winning controller γ.*

In general, Problem 1 and 2 are very challenging, even for a very simple formula φ [14, 10]. As a result, we will attack this problem by leveraging the tools developed for controller synthesis for FTSs. The main difficulty in directly applying these techniques is that $TS(\Sigma)$ has infinitely (uncountably) many states. In the next section, we develop abstraction techniques to convert $TS(\Sigma)$ into FTSs.

4. ABSTRACTION ALGORITHM

In this section, we abstract $TS(\Sigma)$ into two FTSs with the same set of states by partitioning the state space into equivalence classes. We will refer to $s \in dom(S)$ as a *continuous state* for $TS(\Sigma)$ and any state ς of the FTSs as a *discrete state*.

4.1 Constructing the Initial Transition Systems

Our proposed method builds upon the idea of creating an over-approximation and an under-approximation of the reachability relations of the system. To this end, we (iteratively) construct two FTSs. One that we will refer to as the *pessimistic FTS* and one that we will refer to as the *optimistic FTS*. We introduce the notation $\mathbb{D}_o^{(i)} = (\mathcal{V}^{(i)}, \mathcal{V}_{\text{init}}^{(i)}, \rightarrow_o^{(i)})$ and $\mathbb{D}_p^{(i)} = (\mathcal{V}^{(i)}, \mathcal{V}_{\text{init}}^{(i)}, \rightarrow_p^{(i)})$, respectively, for the ith iteration of these systems.

To simplify the notation, we define two reachability relations as:

Definition 11. The relation $\mathcal{R}_p : 2^{dom(S)} \times 2^{dom(S)} \rightarrow \{0,1\}$ is defined such that $\mathcal{R}_p(X,Y) = 1$ if and only if for all $x \in X$, there exists an $y \in Y$ and $u \in U$, such that $f(x,u) = y$.

Definition 12. The relation $\mathcal{R}_o : 2^{dom(S)} \times 2^{dom(S)} \rightarrow \{0,1\}$ is defined such that $\mathcal{R}_o(X,Y) = 1$ if and only if there exist $x \in X$, $y \in Y$ and $u \in U$, such that $f(x,u) = y$.

Remark 2. Informally, $\mathcal{R}_p$ indicates whether there is some control action for every continuous state in a region X that takes that state to some state in the region Y in one time step. $\mathcal{R}_o$ indicates whether there is some point in X that can be controlled to Y in one time step. The results can be generalized to longer horizon lengths, but for simplicity we only consider reachability in one time step.

We further define a partition function of the continuous state space $dom(S)$:

Definition 13. A *partition function* of $dom(S)$ is a mapping $T_{\mathcal{S}} : dom(S) \rightarrow \mathcal{S}$. The inverse of $T_{\mathcal{S}}$ is defined as $T_{\mathcal{S}}^{-1} : \mathcal{S} \rightarrow 2^{dom(S)}$, such that

$$T_{\mathcal{S}}^{-1}(\varsigma) = \{s \in dom(S) : T_{\mathcal{S}}(s) = \varsigma\}.$$

Definition 14. The partition function $T_{\mathcal{S}}$ on $dom(S)$ is called *proposition preserving* if for any atomic proposition p and any pair of continuous states $s_a, s_b \in dom(S)$, which satisfy $T_{\mathcal{S}}(s_a) = T_{\mathcal{S}}(s_b)$, we have that $s_a \models p$ implies that $s_b \models p$.

If $T_{\mathcal{S}}$ is proposition preserving, then we can label the discrete states with atomic propositions. To be specific, we say $\varsigma \models p$ if and only if for every $s \in T_{\mathcal{S}}^{-1}(\varsigma)$, we have that $s \models p$.

To initialize the abstraction algorithm, we assume that we are given the atomic propositions on the continuous state space $dom(S)$. We can create a proposition preserving partition function $T_{\mathcal{S}^{(0)}}$, a set of discrete states $\mathcal{S}^{(0)} = \{\varsigma_0, \varsigma_1, \ldots, \varsigma_n\}$, and a set of initial states $\mathcal{S}_{\text{init}}^{(0)} \subseteq \mathcal{S}^{(0)}$. The state space $\mathcal{V}^{(0)}$ and the initial state $\mathcal{V}_{init}^{(0)}$ are defined as

$$\mathcal{V}^{(0)} = \mathcal{S}^{(0)} \times dom(\mathcal{E}),$$
$$\mathcal{V}_{init}^{(0)} = \mathcal{S}_{init}^{(0)} \times dom(\mathcal{E}).$$

Next, we perform a reachability analysis to establish the transition relations in $\mathbb{D}_o^{(0)}$ and $\mathbb{D}_p^{(0)}$: for every pair of states, $\nu_a = (\varsigma_a, e_a)$, $\nu_b = (\varsigma_b, e_a)$, we add a transition in $\mathbb{D}_p^{(0)}$ from ν_a to ν_b if and only if $\mathcal{R}_p(T_{\mathcal{S}^{(0)}}^{-1}(\varsigma_a), T_{\mathcal{S}^{(0)}}^{-1}(\varsigma_b)) = 1$ and a transition in $\mathbb{D}_o^{(0)}$ if and only if $\mathcal{R}_o(T_{\mathcal{S}^{(0)}}^{-1}(\varsigma_a), T_{\mathcal{S}^{(0)}}^{-1}(\varsigma_b)) = 1$.

Remark 3. $\mathbb{D}_o^{(0)}$ is optimistic in the sense that even if only some part of a region corresponding to a discrete state can reach another, we consider there to be a transition between these two discrete states. In $\mathbb{D}_p^{(0)}$ we require every point in a region corresponding to a discrete state to be able to reach to some point in the other for there to be a transition.

The idea is illustrated in Figure 1. We assume that we are given an initial proposition preserving partition of the continuous state space, corresponding to the four colored quadrants. The dashed and solid lines in the continuous state space separate regions with different reachability properties, which the arrows are meant to illustrate. An arrow from a region to another means that there are control actions that can take the system state from any point in the region it originates from to some points in the region it terminates in. For simplicity, we assume that the environment is trivial, i.e., without any variables.

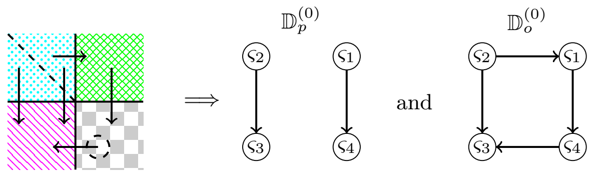


Figure 1: Construction of $\mathbb{D}_p^{(0)}$ and $\mathbb{D}_o^{(0)}$ given an initial proposition preserving partition of the state space (assumed to correspond to the four colored quadrants) and a reachability analysis (illustrated with arrows in the state space). For simplicity, the environment is assumed to have no variables.

We now provide two theorems regarding the (largest) winning sets of $\mathbb{D}_p^{(0)}$, $\mathbb{D}_o^{(0)}$ and $TS(\Sigma)$, the proofs of which are reported later in this subsection for the sake of legibility.

THEOREM 1. *For any discrete state $\varsigma[0] \in W(\mathbb{D}_p^{(0)}, \varphi)$ that is winning for the pessimistic FTS $\mathbb{D}_p^{(0)}$, the corresponding continuous state is also winning in $TS(\Sigma)$, i.e., $T_{\mathcal{S}^{(0)}}^{-1}(\varsigma[0]) \subseteq W(TS(\Sigma), \varphi)$.*

THEOREM 2. *For any continuous state $s[0] \in W(TS(\Sigma), \varphi)$ that is winning for $TS(\Sigma)$, the corresponding discrete state is also winning in $\mathbb{D}_o^{(0)}$, i.e., $T_{\mathcal{S}^{(0)}}(s[0]) \in W(\mathbb{D}_o^{(0)}, \varphi)$.*

PROOF OF THEOREM 1. Suppose the winning controller for $W(\mathbb{D}_p^{(0)}, \varphi)$ is $\gamma_p = (\gamma_{p,1}, \gamma_{p,2}, \ldots, \gamma_{p,t}, \ldots)$. Consider a discrete state $\varsigma[0] = T_{\mathcal{S}^{(0)}}(s[0]) \in W(\mathbb{D}_p^{(0)}, \varphi)$. For all possible environmental actions $e[0]e[1]\ldots$, we can create the controlled execution using γ_p. This gives a sequence of states $(\varsigma[0], e[0])(\varsigma[1], e[1])\ldots$, which satisfies the specification φ.

Consider now a continuous state $s[0] \in T^{-1}_{\mathcal{S}^{(0)}}(\varsigma[0])$. From the construction of $\mathbb{D}^{(0)}_p$, we know that

$$\mathcal{R}_p(T^{-1}_{\mathcal{S}^{(0)}}(\varsigma[t]), T^{-1}_{\mathcal{S}^{(0)}}(\varsigma[t+1])) = 1.$$

Thus, we can recursively define the continuous consistent controller $\gamma = (\gamma_1, \gamma_2, \dots)$ to be

1. $\gamma_1(s[0], e[0])$ returns an $s[1] \in T^{-1}_{\mathcal{S}^{(0)}}(\varsigma[1])$ such that there exists an $u[0] \in U$ and $f(s[0], u[0]) = s[1]$.
2. $\gamma_{t+1}(s[0], e[0], \dots, e[t])$ returns an $s[t+1] \in T^{-1}_{\mathcal{S}^{(0)}}(\varsigma[t+1])$ such that there exists an $u[t] \in U$ and
$$f(\gamma_t(s[0], e[0], \dots, e[t-1]), u[t]) \\ = \gamma_{t+1}(s[0], e[0], \dots, e[t]).$$

As a result, we have a sequence $(s[0], e[0])(s[1], e[1])\dots$, where $T_{\mathcal{S}^{(0)}}(s[t]) = \varsigma[t]$. Hence, the controller γ is also winning at $s[0]$, which completes the proof. $\square$

Proof of Theorem 2. Suppose $\gamma = (\gamma_1, \gamma_2, \dots)$ is winning for $W(TS(\Sigma), \varphi)$ and $s[0] \in W(TS(\Sigma), \varphi)$. For all possible environmental actions $e[0]e[1]\dots$, we create a controlled execution using γ: $(s[0], e[0])(s[1], e[1])\dots$, which is winning.

Now consider the discrete state $\varsigma[t] = T_{\mathcal{S}^{(0)}}(s[t])$. By the definition of $\mathcal{R}_o$, we know that

$$(\varsigma[t], e[t]) \to^{(0)}_o (\varsigma[t+1], e[t+1]).$$

As a result, we can construct a consistent controller $\gamma_o = (\gamma_{o,1}, \dots)$ for $\varsigma[0] = T_{\mathcal{S}^{(0)}}(s[0])$ as

$$\gamma_{o,t}(\varsigma[0], e[0], \dots, e[t-1]) = T_{\mathcal{S}^{(0)}}(\gamma_t(s[0], e[0], \dots, e[t-1])).$$

Thus, we get a sequence $(\varsigma[0], e[0])(\varsigma[1], e[1])\dots$, where $\varsigma[t] = T_{\mathcal{S}^{(0)}}(s[t])$. Hence, the controller γ_o is winning at $\varsigma[0]$, which completes the proof. $\square$

We now define the following sets:

$$\mathcal{W}^{(i)} = W(\mathbb{D}^{(i)}_p, \varphi), \tag{5}$$

referred to as *the winning set*,

$$\mathcal{L}^{(i)} = L(\mathbb{D}^{(i)}_o, \varphi) \tag{6}$$

as *the losing set* and

$$\mathcal{M}^{(i)} = \mathcal{S}^{(i)} \backslash \left(\mathcal{W}^{(i)} \cup \mathcal{L}^{(i)}\right). \tag{7}$$

which we will call *the maybe set*. It will be clear from the context to which iteration i of these sets we refer to. We can further define the inverse image of the sets $\mathcal{W}^{(i)}, \mathcal{L}^{(i)}, \mathcal{M}^{(i)}$ on $dom(S)$ as

$$\mathcal{W}^{(i)}_c = T^{-1}_{\mathcal{S}^{(i)}}(\mathcal{W}^{(i)}),$$
$$\mathcal{L}^{(i)}_c = T^{-1}_{\mathcal{S}^{(i)}}(\mathcal{L}^{(i)}),$$
$$\mathcal{M}^{(i)}_c = T^{-1}_{\mathcal{S}^{(i)}}(\mathcal{M}^{(i)}).$$

By Theorem 1 and 2, it is clear that

1. If $S_{init} \subseteq \mathcal{W}^{(0)}_c$, then S_{init} is a winning set for $TS(\Sigma)$. Furthermore, the winning controller can be constructed in a similar fashion as is discussed in the proof of Theorem 1.
2. If $S_{init} \bigcap \mathcal{L}^{(0)}_c \neq \emptyset$, then S_{init} is not a winning set for $TS(\Sigma)$.
3. If neither 1) nor 2) is true, then we cannot give a definitive answer on whether S_{init} is winning or not. As a result, a finer partition function is needed to answer the Realizability Problem.

For case 3), one may naively create a finer partition function and the corresponding pessimistic and optimistic FTS. In the next subsection, we show how to iteratively generate the pessimistic and optimistic FTS in order to reduce the computational complexity of the abstraction algorithm by exploiting the properties of the winning set.

4.2 Refinement Procedure

We define a refinement operation as

$$\mathbf{split}_m : 2^{dom(S)} \times \{1, \dots, m\} \to 2^{dom(S)} \tag{8}$$

such that $\forall X \subseteq dom(S)$ and $i, j \in \{1, \dots, m\}, i \neq j$ and it has the following properties:

$$\begin{aligned} \mathbf{split}_m(X, i) &\subset X, \\ \mathbf{split}_m(X, i) \cap \mathbf{split}_m(X, j) &= \emptyset \end{aligned}$$

and

$$\bigcup_{k=1}^{m} \mathbf{split}_m(X, k) = X.$$

Remark 4. The index m on $\mathbf{split}_m$ is the number of children that a region should be split into upon refinement.

We will focus our computational resources on the states in the maybe set $\mathcal{M}^{(i)}$. Intuitively, these states have the potential to become winning when we create finer partitions. Assuming that $\mathcal{S}^{(i)}$ and $T_{\mathcal{S}^{(i)}}$ are the set of discrete state and the partition function of the ith iteration, we define $\mathcal{S}^{(i+1)}$ and $T_{\mathcal{S}^{(i+1)}}$ in the following way:

1. If $\varsigma \in \mathcal{W}^{(i)} \bigcup \mathcal{L}^{(i)}$, then $(\varsigma, 1) \in \mathcal{S}^{(i+1)}$ and
$$T^{-1}_{\mathcal{S}^{(i+1)}}((\varsigma, 1)) = T^{-1}_{\mathcal{S}^{(i)}}(\varsigma).$$
2. If $\varsigma \in \mathcal{M}^{(i)}$, then $(\varsigma, j) \in \mathcal{S}^{(i+1)}$ for all $j = 1, \dots, m$ and
$$T^{-1}_{\mathcal{S}^{(i+1)}}((\varsigma, j)) = \mathbf{split}_m(T^{-1}_{\mathcal{S}^{(i)}}(\varsigma), j).$$

Remark 5. One can consider the discrete state spaces $\mathcal{S}^{(0)}$, $\mathcal{S}^{(1)}$, ... to form a forest (a disjoint union of trees), where the states in $\mathcal{S}^{(0)}$ are the roots and $(\varsigma, j) \in \mathcal{S}^{(i+1)}$ is the jth child of $\varsigma \in \mathcal{S}^{(i)}$.

An example of the refinement procedure is provided in Figure 2. An initial preposition preserving partition is constructed from the continuous state space $dom(S)$, which in this case, results in three discrete states (and corresponding regions in the continuous state space). The discrete states are marked as to belonging to either the winning (crosshatched green), maybe (solid yellow) or losing (dotted red) set. To refine the partition, the **split**$_3$-operator is applied to the state in the maybe set, ς_2. The refined partition can be seen in the rightmost figure, where a new reachability analysis has been performed. The next step of the procedure would further refine $(\varsigma_2, 3)$.

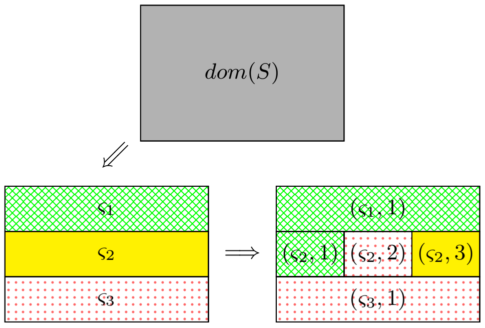


Figure 2: An example of the proposed refinement procedure. An initial preposition preserving partition is constructed in the first step. The regions are labeled with their corresponding discrete state. The states are colored differently depending on if they belong to the winning (crosshatched green), maybe (solid yellow) or losing (dotted red) set. The split$_3$-operator is used to further refine the states in the maybe set (only one iteration is illustrated).

Given the discrete states, the state space $\mathcal{V}^{(i+1)}$ can be defined as

$$\mathcal{V}^{(i+1)} = \mathcal{S}^{(i+1)} \times dom(\mathcal{E}),$$

and the initial states $\mathcal{V}_{init}^{(i+1)}$ can be defined in a similar fashion.

We now define the transition relations of the two FTSs. We begin with the relations in the pessimistic FTS. For any two states $(\varsigma_a, j), (\varsigma_b, k) \in \mathcal{S}^{(i+1)}$ and environmental states e_a, e_b, we have that $((\varsigma_a, j), e_a) \rightarrow_p^{(i+1)} ((\varsigma_b, k), e_b)$ if and only if one of the following statements holds:

1. *WW-transition*: $\varsigma_a, \varsigma_b \in \mathcal{W}^{(i)}$, $j = k = 1$ and
$$(\varsigma_a, e_a) \rightarrow_p^{(i)} (\varsigma_b, e_b).$$
2. *MW-transition*: $\varsigma_a \in \mathcal{M}^{(i)}$, $\varsigma_b \in \mathcal{W}^{(i)}$, $k = 1$ and
$$\mathcal{R}_p(\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_a, j)\,),\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_b, 1)\,)\,) = 1.$$
3. *MM-transition*: $\varsigma_a, \varsigma_b \in \mathcal{M}^{(i)}$ and
$$\mathcal{R}_p(\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_a, j)\,),\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_b, k)\,)\,) = 1.$$

Remark 6. Here, WW stands for a transition between two winning states, and analogously for MW and MM.

Remark 7. Notice that we omit many possible transitions. For example, if $\varsigma_a \in \mathcal{W}^{(i)}$ and $\varsigma_b \in \mathcal{M}^{(i)}$, then no transitions will be added even if (ς_b, k) is reachable from $(\varsigma_a, 1)$. This allows us to focus our computational resources on the critical transitions that affects the computation of the winning set.

The update rule for the optimistic FTS is similar. We have that $((\varsigma_a, j), e_a) \rightarrow_o^{(i+1)} ((\varsigma_b, k), e_b)$ if and only if one of the following three statements holds:

1. *WW-transition*: $\varsigma_a, \varsigma_b \in \mathcal{W}^{(i)}$, $j = k = 1$ and
$$(\varsigma_a, e_a) \rightarrow_p^{(i)} (\varsigma_b, e_b).$$
Notice that we are using the transition relation $\rightarrow_p^{(i)}$ instead of $\rightarrow_o^{(i)}$ for this case.
2. *MW-transition*: $\varsigma_a \in \mathcal{M}^{(i)}$, $\varsigma_b \in \mathcal{W}^{(i)}$, $k = 1$ and
$$\mathcal{R}_o(\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_a, j)\,),\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_b, 1)\,)\,) = 1.$$
3. *MM-transition*: $\varsigma_a, \varsigma_b \in \mathcal{M}^{(i)}$ and
$$\mathcal{R}_o(\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_a, j)\,),\, T_{\mathcal{S}^{(i+1)}}^{-1}(\,(\varsigma_b, k)\,)\,) = 1.$$

We will now expand upon Theorem 1 and 2 to provide a characterization of the winning sets $W(\mathbb{D}_p^{(i)}, \varphi)$ and $W(\mathbb{D}_o^{(i)}, \varphi)$. The proofs of the following theorems are deferred to the appendix for the sake of legibility.

THEOREM 3. *For any discrete state $\varsigma[0] \in W(\mathbb{D}_p^{(i)}, \varphi)$ that is winning for the pessimistic FTS $\mathbb{D}_p^{(i)}$, the corresponding continuous state is also winning in $TS(\Sigma)$, i.e.,*

$$T_{\mathcal{S}^{(i)}}^{-1}(\varsigma[0]) \subseteq W(TS(\Sigma), \varphi). \tag{9}$$

Furthermore, its child $(\varsigma[0], 1)$ is also winning for $\mathbb{D}_p^{(i+1)}$, i.e.,

$$(\varsigma[0], 1) \in W(\mathbb{D}_p^{(i+1)}, \varphi). \tag{10}$$

THEOREM 4. *For any continuous state $s[0] \in W(TS(\Sigma), \varphi)$ that is winning for $TS(\Sigma)$, the corresponding discrete state is also winning in $\mathbb{D}_o^{(i)}$, i.e.,*

$$T_{\mathcal{S}^{(i)}}(s[0]) \in W(\mathbb{D}_o^{(i)}, \varphi). \tag{11}$$

Furthermore, if the discrete state $\varsigma[0] \in L(\mathbb{D}_o^{(i)}, \varphi)$ is losing for $\mathbb{D}_o^{(i)}$, then its child is also losing in $\mathbb{D}_o^{(i+1)}$, i.e.,

$$(\varsigma[0], 1) \in L(\mathbb{D}_o^{(i+1)}, \varphi). \tag{12}$$

Combining Theorem 3 and 4, we have the following corollary:

COROLLARY 1. $\mathcal{W}_c^{(0)} \subseteq \mathcal{W}_c^{(1)} \subseteq \cdots \subseteq W(TS(\Sigma), \varphi) \subseteq \cdots \subseteq dom(S) \setminus \mathcal{L}_c^{(1)} \subseteq dom(S) \setminus \mathcal{L}_c^{(0)}$.

With these results, we can concisely write the algorithm as:

- If $S_{init} \subseteq \mathcal{W}_c^{(i)}$, then S_{init} is a winning set for $TS(\Sigma)$. A winning controller can be constructed in a similar fashion as is discussed in the proof of Theorem 1.
- If $S_{init} \bigcap \mathcal{L}_c^{(i)} \neq \emptyset$, then S_{init} is not a winning set for $TS(\Sigma)$. Thus, we can stop the refinement procedure because there is no winning controller.
- If neither of the above statements is fulfilled, then we cannot give a definitive answer on whether S_{init} is winning or not at the ith iteration. As a result, we create the FTSs $\mathbb{D}_p^{(i+1)}$ and $\mathbb{D}_o^{(i+1)}$ and try to solve the winning sets for them.

Remark 8. It is worth noticing that we do not use any properties of the f function or the sets U, $dom(S)$ and S_{init}. As a result, the algorithm presented in this article can be used to handle any transition system.

5. SIMULATION RESULTS

In this section, we do a comparison between the current algorithm in `TuLiP` and our proposed algorithm. All the simulations are performed on a MacBook Air (1.3 GHz, 4 GB RAM).

We first consider an example included with the `TuLiP` package (namely, `robot_continuous.py`). The dynamical equations describing the system are

$$\begin{aligned} s[t+1] &= \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} s[t] + \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} u[t],\\ u[t] &\in U = \{v \in \mathbb{R}^2 : |v|_\infty \leq 1\},\\ s[t] &\in dom(S) = [0,3]\times[0,2],\\ s[0] &\in S_{\text{init}} = [0,3]\times[0,2]. \end{aligned} \tag{13}$$

Two regions in the state space are marked with propositions:

$$\text{propositions} = \begin{cases}[0,1]\times[0,1] & \text{as } \mathit{home},\\ [2,3]\times[1,2] & \text{as } \mathit{lot}\end{cases} \tag{14}$$

and the environment is equipped with a boolean variable, *park*. The specification of system is the following:

$$\varphi = \Box\Diamond \mathit{home} \wedge \Box(\mathit{park} \rightarrow \Diamond \mathit{lot}),$$

which can be converted into GR1 form (1). Roughly speaking the specification implies that the system should visit the parking *lot* whenever the environment sets *park* true, and always returns back *home*.

In the algorithm employed in `TuLiP`, described in [16], the whole state space is discretized according to a reachability analysis until no region corresponding to a discrete state can be refined further without going below a pre-specified threshold volume. This leads to a problem when the threshold volume is set too high, since not enough transitions can be established on a discrete level. As illustrated by the red crosses in Figure 3, `TuLiP` fails to find a controller realizing the specification when the threshold volume is taken larger than 0.2. When the threshold is chosen below 0.2, it succeeds in finding a controller and announces that the specifications are realizable (green dots). Note that the cumulative sum of the times testing to see if the specifications are realizable using a smaller and smaller threshold is large. Our implementation will iteratively refine the partition of the state space until a controller can be synthesized. Furthermore, our algorithm only refines the "interesting" areas of the state space, which results in less computational time comparing to the time that `TuLiP` used to give a correct answer. This is illustrated by the dashed blue line.

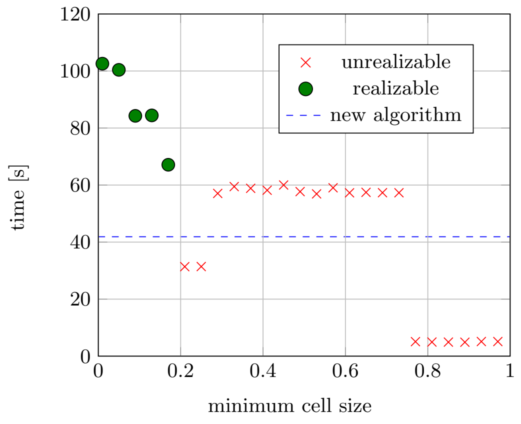


Figure 3: Timing data for the current algorithm in `TuLiP` and our proposed algorithm. The specifications that we are considering for the continuous system are realizable, but `TuLiP` cannot synthesize a controller until the threshold volume is below 0.2. The dots and crosses indicate the time for `TuLiP` to discretize the state space and then try to synthesize a controller, giving a positive or a negative answer, respectively, on whether the specifications are realizable. Our algorithm concludes that the specifications are realizable without taking any threshold volume as input, illustrated by the dashed blue line.

The next example shows the actual partition that results from the two methods. Consider the system

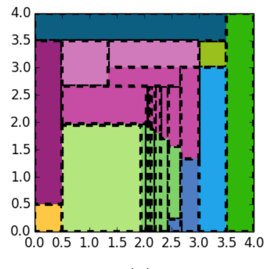
$$\begin{aligned} s[t+1] &= \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} s[t] + \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} u[t],\\ u[t] &\in U = \{v \in \mathbb{R}^2 : |v|_\infty \leq 1\},\\ s[t] &\in dom(S) = [0,4]\times[0,4],\\ s[0] &\in S_{\text{init}} = [3,3.5]\times[3,3.5]. \end{aligned} \tag{15}$$

Let us introduce the concept of an *invariant set*: a set Ω is *invariant* if $s(t_0) \in \Omega \implies s(t) \in \Omega,\ \forall t \geq t_0$ and for all possible controls $u(t)$. A simple application of the triangle inequality will show that the region $\mathbb{R}^2 \setminus [0,2]^2$ is invariant for the system under consideration. We consider the following set of propostions

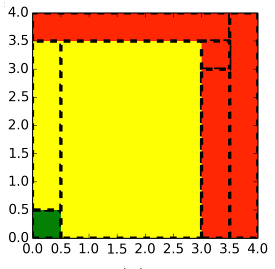
$$\text{propositions} = \begin{cases}[0,0.5]\times[0,0.5] & \text{as } \mathit{goal},\\ [3,3.5]\times[3,3.5] & \text{as } \mathit{start}.\end{cases} \tag{16}$$

In this example, we assume that the environment has no variables. The initial assumption on the system is

$$\mathit{start} \tag{17}$$



(a)



(b)

Figure 4: (a) shows the discretization by `TuLiP` on the second example when the threshold volume is chosen to be 1.0. Regions of the same color are considered as one discrete state. (b) shows the discretization resulting from our algorithm, with the winning (green), maybe (yellow) and losing (red) sets marked. Here, every region is its own discrete state.

and the progress specification of the system is

$$\Box\Diamond goal. \tag{18}$$

This means that the systems starts in *start* and should always eventually reach *goal*. Since *start* lies in an invariant region, that does not contain *goal*, we know that there does not exist a winning controller.

The resulting discretizations can be seen in Figure 4. (a) shows the discretization that `TuLiP` provides when the threshold volume is set to 1.0. Note that the invariant region is finely partitioned. The runtime of the algorithm is 620 s. No controller that fulfills the specifications can be synthesized using this discretization. Note that from the output of `TuLiP`, it is not possible to say whether no winning controller exists, or if a winning controller of the original system exists but `TuLiP` cannot find it because of the partition being too coarse.

The output of our algorithm can be seen in Figure 4.(b). The coloring illustrates the winning (green), maybe (yellow) and losing (red) states. The states in the maybe set are marked as such since some of the continuous states in them lie within the invariant region, and some lie within the region that can reach *goal*. Since *start* lies in the losing set, the algorithm terminates and concludes with a definitive answer that there is no winning controller. This takes 25 s.

6. CONCLUSION

We have in this paper presented an iterative method for abstracting a discrete-time control system into two FTSs, representing an under- and over-approximation of the reachability properties of the original dynamical system. We have provided theorems regarding the existence of controllers fulfilling specifications given in LTL for the continuous systems, based on the existence of such controllers for the two FTSs. Our proposed algorithm provides a way of focusing the computational resources on refining only certain areas of the state space, leading to a decrease in the time complexity of the abstraction procedure compared to previous methods. We have made a comparison between the proposed algorithm and the one currently used in Python package `TuLiP` on numerical examples with positive results.

7. ACKNOWLEDGMENTS

The authors would like to thank Vasumathi Raman, Scott Livingston and Mattias Fält for valuable discussions during the development of this work.

This work is supported in part by IBM and UTC through the Industrial Cyber-Physical Systems Center (iCyPhy) consortium.

8. REFERENCES

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2007.

[2] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In *ICRA*, pages 2689–2696, 2010.

[3] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343 – 352, 2009.

[4] S. Karaman and E. Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *CDC*, 2009.

[5] M. Kloetzer. and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics*, 26(1):48–61, 2010.

[6] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[7] P. Nuzzo, H. Xu, N. Ozay, J. Finn, A. Sangiovanni-Vincentelli, R. Murray, A. Donze, and S. Seshia. A contract-based methodology for aircraft electric power system design. *Access, IEEE*, PP(99):1–1, 2013.

[8] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. In *Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

[9] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, pages 179–190, New York, NY, USA, 1989. ACM.

[10] S. V. Rakovic, E. Kerrigan, D. Mayne, and J. Lygeros. Reachability analysis of discrete-time systems with disturbances. *IEEE Transactions on Automatic Control*, 51:546–561, Apr. 2006.

[11] V. Raman, M. Maasoumy, A. Donzé, R. M. Murray, A. Sangiovanni-Vincentelli, and S. Seshia. Model predictive control with signal temporal logic specifications. In *IEEE Conference on Decision and Control (CDC)*, 2014.

[12] P. Tabuada. An approximate simulation approach to symbolic control. *IEEE Trans. Automat. Contr.*, 53(6):1406–1418, 2008.

[13] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[14] R. Vidal, S. Schaffert, J. Lygeros, and S. Sastry. Controlled Invariance of Discrete Time Systems. *Lecture Notes in Computer Science LNCS*, (1790):437–450, 2000.

[15] T. Wongpiromsarn, U. Topcu, and R. Murray. Receding horizon temporal logic planning for dynamical systems. *Proc. IEEE Conf. Decision Control*, pages 5997–6004, 2009.

[16] T. Wongpiromsarn, U. Topcu, and R. Murray. Receding horizon temporal logic planning. *Automatic Control, IEEE Transactions on*, 11(57):2817–2830, 2012.

[17] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Synthesis of control protocols for autonomous systems. *Unmanned Systems*, 01(01):21–39, 2013.

[18] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray. Tulip: a software toolbox for receding horizon temporal logic planning. In *HSCC*, pages 313–314, 2011.

[19] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *Automatic Control, IEEE Transactions on*, 57(7):1804–1809, 2012.

APPENDIX

A. PROOFS OF THEOREM 3 AND 4

Before proving Theorem 3 and 4, we need the following lemmas:

LEMMA 1. *For any two sequences $\sigma = \nu_0\nu_1\dots$, $\sigma' = \nu_0'\nu_1'\dots$, such that $\sigma \models \varphi$ and $\sigma' \models \varphi$, where φ is a GR1 formula defined in* (1)*, the following properties hold:*

1. *Define a time-shifted sequence $\sigma_t = \nu_t\nu_{t+1}\dots$, then $\sigma_t \models \varphi$.*
2. *Suppose that there exists $\tau \geq 0$, such that $\nu_\tau = \nu_0'$, then the following sequence $\nu_0\dots\nu_\tau\nu_1'\nu_2'\dots \models \varphi$.*

PROOF. By definition, $\sigma \models \varphi$ if and only if

$$\sigma \models \left(\neg \bigwedge_{i=1}^{M} \Box\Diamond p_i\right) \bigvee \left(\bigwedge_{j=1}^{N} \Box\Diamond q_j\right). \tag{19}$$

The lemma follows directly from the fact that the right hand side of (19) is a liveness formula. □

LEMMA 2. *Consider an FTS $\mathbb{T}$ and a GR1 formula φ. If the controller γ is winning for some non-empty set $\mathcal{W}$, then for any initial condition $\varsigma[0] \in \mathcal{W}$ and environmental actions $e[0]e[1]\dots$, the controlled execution $(\varsigma[0], e[0])(\varsigma[1], e[1])\dots$ satisfies*

$$\varsigma[t] \in W(\mathbb{T}, \varphi),\ \forall t = 0, 1, \dots\ .$$

PROOF. This result follows directly from Lemma 1. □

PROOF OF THEOREM 3. By the recursive definition of $\mathbb{D}_p^{(i)}$ and $\mathbb{D}_o^{(i)}$, we know that for any $\varsigma_a, \varsigma_b \in \mathcal{S}^{(i)}$,

$$(\varsigma_a, e_a) \rightarrow_p^{(i)} (\varsigma_b, e_b)$$

implies that

$$\mathcal{R}_p(T^{-1}_{\mathcal{S}^{(i)}}(\varsigma_a)), T^{-1}_{\mathcal{S}^{(i)}}(\varsigma_b))) = 1.$$

Hence, (9) can be proved in a similar way as Theorem 1.

We now prove (10). For the FTS $\mathbb{D}_p^{(i)}$, suppose the winning controller for $\mathcal{W}^{(i)} = W(\mathbb{D}_p^{(i)}, \varphi)$ is $\gamma_p^{(i)} = (\gamma_{p,1}^{(i)}, \gamma_{p,2}^{(i)}, \dots)$. We can define the controller $\gamma_p^{(i+1)} = (\gamma_{p,1}^{(i+1)}, \gamma_{p,2}^{(i+1)}, \dots)$ for the FTS $\mathbb{D}_p^{(i+1)}$ as

$$\begin{aligned}\gamma_{p,t}^{(i+1)}((\varsigma[0], 1), &e[0], \dots, e[t-1]) \\ &= (\gamma_{p,t}^{(i)}(\varsigma[0], e[0], \dots, e[t-1]), 1).\end{aligned}$$

Thus, the controlled execution of the FTS $\mathbb{D}_p^{(i+1)}$ is given by

$$((\varsigma[0], 1), e[0])((\varsigma[1], 1), e[1])((\varsigma[2], 1), e[2])\dots,$$

which satisfies the specification φ. Therefore, we only need to prove that the controller $\gamma_p^{(i+1)}$ is consistent.

By Lemma 2, we know that for any $\varsigma[0] \in \mathcal{W}^{(i)}$, the controlled execution $(\varsigma[0], e[0])\dots$ satisfies

$$\varsigma[t] \in \mathcal{W}^{(i)},$$

which implies that the transition from $((\varsigma[t], 1), e[t])$ to $((\varsigma[t+1], 1), e[t+1])$ in $\mathbb{D}_p^{(i+1)}$ is a *WW-transition* and hence it exists. Hence, $\gamma_p^{(i+1)}$ is consistent, which completes the proof. □

PROOF OF THEOREM 4. We first prove (12). Notice that by the construction of $\mathbb{D}_o^{(i+1)}$, if $\varsigma[0] \in \mathcal{L}^{(i)} = L(\mathbb{D}_o^{(i)}, \varphi)$, then $((\varsigma[0], 1), e[0])$ has no successors in $\mathbb{D}_o^{(i+1)}$. Thus, $(\varsigma[0], 1) \in L(\mathbb{D}_o^{(i+1)}, \varphi)$ since no consistent controller exists for $(\varsigma[0], 1)$.

We now prove (11) by induction. Notice that we *cannot* use the same argument as Theorem 2 since $s_a \to s_b$ does not necessarily imply $T_{\mathcal{S}^{(i+1)}}(s_a) \rightarrow_o^{(i+1)} T_{\mathcal{S}^{(i+1)}}(s_b)$.

By Theorem 2, we know that (11) holds when $i = 1$. For the transition system $TS(\Sigma)$, suppose that the controller $\gamma = (\gamma_1, \gamma_2, \dots)$ is winning for $W(TS(\Sigma), \varphi)$. For any $s[0] \in W(TS(\Sigma), \varphi)$ and environmental actions $e[0]e[1]\dots$, we create a controlled execution using γ: $\sigma = (s[0], e[0])(s[1], e[1])\dots$, which is winning.

Let us define a hitting time τ as

$$\tau = \inf\{t \in \mathbb{N}_0 : T_{\mathcal{S}^{(i-1)}}(s[t]) \in \mathcal{W}^{(i-1)}\}.$$

In other words, τ is the first time that $T_{\mathcal{S}^{(i-1)}}(s[t])$ enters the winning set $\mathcal{W}^{(i-1)}$. We further assume that the infimum over an empty set is ∞.

For the FTS $\mathbb{D}_p^{(i-1)}$, suppose that the controller $\gamma_p = (\gamma_{p,1}, \dots)$ is winning for $W(\mathbb{D}_p^{(i-1)}, \varphi) = \mathcal{W}^{(i-1)}$. If $\tau < \infty$, we define $\varsigma_p[0] = T_{\mathcal{S}^{(i-1)}}(s[\tau])$ and $e_p[t] = e[t+\tau]$. Now we create a controlled execution using γ_p with environmental actions $e_p[0]e_p[+1]\dots$: $\sigma_p = (\varsigma_p[0], e_p[0])(\varsigma_p[1], e_p[1])\dots$, which is also winning.

We now construct a controller $\gamma_o = (\gamma_{o,1}, \dots)$ of the FTS

$\mathbb{D}_o^{(i)}$, such that it is winning at $\varsigma[0] = T_{\mathcal{S}^{(i)}}(s[0])$. The construction can by divided into two steps:

1. If $t \leq \tau$, then γ_o follows the winning controller γ of the FTS $TS(\Sigma)$, i.e.,
$$\begin{aligned}\gamma_{o,t}(\varsigma[0], &e[0], \ldots, e[t-1]) \\ &= T_{\mathcal{S}^{(i)}}(\gamma_t(s[0], e[0], \ldots, e[t-1])).\end{aligned}$$
2. If $t > \tau$, we switch to the winning controller γ_p of the FTS $\mathbb{D}_p^{(i-1)}$, i.e.,
$$\begin{aligned}&\gamma_{o,t}(\varsigma[0], e[0], \ldots, e[t-1]) \\ &\quad = (\gamma_{p,t-\tau}(\varsigma_p[0], e_p[0], \ldots, e_p[t-\tau-1]), 1).\end{aligned}$$

Now we prove that γ_o is winning at $\varsigma[0]$. Define the controlled execution using γ_o on the FTS $\mathbb{D}_o^{(i)}$ to be
$$\sigma_o = (\varsigma_o[0], e[0])(\varsigma_o[1], e[1]) \ldots .$$
We need to prove that σ_o satisfies the specification and γ_o is consistent. The proof is divided into two cases depending on whether $\tau = \infty$ or $\tau < \infty$.

Case 1: $\tau = \infty$

By the definition of γ_o, we know that
$$\varsigma_o[t] = T_{\mathcal{S}^{(i)}}(s[t]).$$
Since σ is winning, we only need to check the consistency of γ_o, i.e., whether the transition from $(\varsigma_o[t], e[t])$ to $(\varsigma_o[t+1], e[t+1])$ exists in $\mathbb{D}_o^{(i)}$. By Lemma 2, we know that
$$s[t] \in W(TS(\Sigma), \varphi).$$
And hence, by the induction assumption,
$$T_{\mathcal{S}^{(i-1)}}(s[t]) \in \mathcal{M}^{(i-1)} \bigcup \mathcal{W}^{(i-1)}.$$
By the fact that $\tau = \infty$,
$$T_{\mathcal{S}^{(i-1)}}(s[t]) \in \mathcal{M}^{(i-1)}.$$
As a result, there exists an $j_t \in \{1, \ldots, m\}$, such that $\varsigma_o[t]$ is the j_tth child of $T_{\mathcal{S}^{(i-1)}}(s[t])$, i.e.,
$$\varsigma_o[t] = (T_{\mathcal{S}^{(i-1)}}(s[t]), j_t).$$
Furthermore, since there exists an $u[t]$, such that $f(s[t], u[t]) = s[t+1]$, we know that
$$\mathcal{R}_o(T^{-1}_{\mathcal{S}^{(i)}}(\varsigma_o[t]), T^{-1}_{\mathcal{S}^{(i)}}(\varsigma_o[t+1])) = 1,$$
Hence, the transition from $(\varsigma_o[t], e[t])$ to $(\varsigma_o[t+1], e[t+1])$ is an *MM-transition* and it exists in $\mathbb{D}_o^{(i)}$. And thus, γ_o is consistent.

Case 2: $\tau < \infty$

By the construction of γ_o, σ_o satisfies
$$\varsigma_o[t] = \begin{cases} T_{\mathcal{S}^{(i)}}(s[t]) & \text{if } t \leq \tau, \\ (\varsigma_p[t-\tau], 1) & \text{if } t > \tau. \end{cases}$$
By Lemma 1 and the fact that both σ and σ_p satisfy φ, we only need to check the consistency of γ_o, i.e., whether the transition from $(\varsigma_o[t], e[t])$ to $(\varsigma_o[t+1], e[t+1])$ exists in $\mathbb{D}_o^{(i)}$. This can be done in three steps:

1. $t < \tau - 1$:
By the same argument as for the case where $\tau = \infty$, we know that the transition from $(\varsigma_o[t], e[t])$ to $(\varsigma_o[t+1], e[t+1])$ is an *MM-transition* and it exists in $\mathbb{D}_o^{(i)}$.
2. $t = \tau - 1$:
By the definition of τ, we know that
$$T_{\mathcal{S}^{(i-1)}}(s[\tau-1]) \in \mathcal{M}^{(i-1)},\ T_{\mathcal{S}^{(i-1)}}(s[\tau]) \in \mathcal{W}^{(i-1)}.$$
Hence, the transition from $(\varsigma_o[\tau-1], e[\tau-1])$ to $(\varsigma_o[\tau], e[\tau])$ is an *MW-transition* and it exists in $\mathbb{D}_o^{(i)}$.
3. $t > \tau - 1$:
By Lemma 2, we know that
$$\varsigma_p[t] \in W(\mathbb{D}_p^{(i-1)}, \varphi) = \mathcal{W}^{(i-1)}.$$
Hence, the transition from $(\varsigma_o[t], e[t])$ to $(\varsigma_o[t+1], e[t+1])$ is a *WW-transition* and it exists in $\mathbb{D}_o^{(i)}$.

Therefore, γ_o is consistent and we can conclude the proof. $\square$