Output-Feedback Symbolic Control

Mahmoud Khaled, Kuize Zhang, and Majid Zamani

Abstract

Symbolic control is an abstraction-based controller synthesis approach that provides, algorithmically, certifiable-by-construction controllers for cyber-physical systems. Symbolic control approaches usually assume that full-state information is available which is not suitable for many real-world applications with partially-observable states or output information. This article introduces a framework for output-feedback symbolic control. We propose relations between original systems and their symbolic models based on outputs. They enable designing symbolic controllers and refining them to enforce complex requirements on original systems. We provide example methodologies to synthesize and refine output-feedback symbolic controllers.

I. INTRODUCTION

Symbolic control [1]–[4] is an approach to automatically synthesize certifiable controllers that handle complex requirements including objectives and constraints given by formulae in linear temporal logic (LTL) or automata on infinite strings [1,5]. In symbolic control, a dynamical system (e.g., a physical process described by a set of differential equations) is related to a symbolic model (i.e., a system with finite state and input sets) via a formal relation. The relation ensures that the symbolic model captures some required features from the original system. Since symbolic models are finite, reactive synthesis techniques [6]–[8] can be applied to algorithmically synthesize controllers enforcing the given specifications. The designed controllers are usually referred to as symbolic controllers.

Symbolic models can be used to abstract several classes of control systems [1,3,4,9,10]. Unfortunately, the majority of current techniques assume control systems with full-state or quantized-state information and, hence, they are not applicable to control systems with outputs or partially-observable states. Moreover, none of state-of-the-art tools of symbolic controller
synthesis [11]–[13] support output-feedback systems since the required theories for them are not yet fully established.

In this article, we consider control systems with partial-state or output information. We refer to these particular types of systems as output-based control systems. We introduce a framework for symbolic control that can handle this class of systems. We first extend the work in [4] to provide mathematical tools for constructing symbolic models of output-based systems. More precisely, output-feedback refinement relations (OFRRs) are introduced as means of relating output-based systems and their symbolic models. They are extensions of feedback refinement relations (FRRs) in [4]. OFRRs allow abstractions to be constructed by quantizing the state and output sets of concrete systems, such that the output quantization respects the state quantization. We prove that OFRRs ensure external (i.e., output-based) behavioral inclusion from original systems to symbolic models. Symbolic controllers synthesized based on the outputs of symbolic models can be refined via simple and practically implementable interfaces.

In Sections VI, VII and VIII, we present example methodologies that realize the introduced framework. The first methodology is based on games of imperfect information. The second one proposes designing observers for output-based systems. The third one proposes detectors designed for symbolic models. Three case studies are presented in Section IX to demonstrate the effectiveness of proposed methodologies.

II. NOTATION

The identity map on a set \( X \) is denoted by \( id_X \). Symbols \( \mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{R}^+, \) and \( \mathbb{R}^+_0 \) denote, respectively, the sets of natural, integer, real, positive real, and nonnegative real numbers.

For a set \( A \), we denote by \(|A|\) the cardinality of the set, and by \( 2^A \) the set of all subsets of \( A \) including the empty set \( \emptyset \). A partition of a set \( A \) is a set of pairwise disjoint nonempty subsets of \( A \) whose union equals \( A \). We denote by \( A^* \) the set of all finite strings (a.k.a. sequences) obtained by concatenating elements in \( A \). For any finite string \( s \), \(|s|\) denotes the length of the string, \( s_i, i \in \{0, 1, \cdots, |s| - 1\} \), denotes the \( i \)-th element of \( s \), and \( s[i, j], j \geq i \), denotes the substring \( s_is_{i+1} \cdots s_j \). Symbol \( e \) denotes the empty string and \(|e| = 0 \). We use the dot symbol \( \cdot \) to concatenate two strings.

Consider a relation \( \mathcal{R} \subseteq A \times B \). \( \mathcal{R} \) is strict when \( \mathcal{R}(a) \neq \emptyset \) for every \( a \in A \). \( \mathcal{R} \) naturally introduces a map \( \mathcal{R} : A \to 2^B \) such that \( \mathcal{R}(a) = \{b \in B \mid (a, b) \in \mathcal{R}\} \). \( \mathcal{R} \) also admits an inverse relation \( \mathcal{R}^{-1} := \{(b, a) \in B \times A \mid (a, b) \in \mathcal{R}\} \). Given an element \( r = (a, b) \in \mathcal{R}, \pi_A(r) \) denotes
the natural projection of \( r \) on the set \( A \), i.e., \( \pi_A(r) = a \). We sometimes abuse the notation and apply the projection map \( \pi_A \) to a string (resp., a set of strings) of elements of \( \mathcal{R} \), which means applying it iteratively to all elements in the string (resp., all strings in the set). When \( \mathcal{R} \) is an equivalence relation on a set \( X \), we denote by \( [x] \) the equivalence class of \( x \in X \) and by \( X/\mathcal{R} \) the set of all equivalence classes (a.k.a. quotient set). We also denote by \( \pi_\mathcal{R} : X \rightarrow X/\mathcal{R} \) the natural projection map taking a point \( x \in X \) to its equivalence class, i.e., \( \pi_\mathcal{R}(x) = [x] \in X/\mathcal{R} \). We say that an equivalence relation is finite when it has finitely many equivalence classes.

Given a vector \( v \in \mathbb{R}^n \), we denote by \( v_i, i \in \{0, 1, \ldots, n-1\} \), the \( i \)-th element of \( v \) and by \( \|v\| \) its infinity norm.

### III. Preliminaries

First, we present the notion of systems. We use a similar definition for systems as in [1].

**Definition III.1 (System).** A system is a tuple

\[ S := (X, X_0, U, \rightarrow, Y, H), \]

where \( X \) is the set of states, \( X_0 \subseteq X \) is a set of initial states, \( U \) is the set of inputs, \( \rightarrow \subseteq X \times U \times X \) is the transition relation, \( Y \) is the set of outputs, and \( H : X \rightarrow Y \) is the output map.

All sets in tuple \( S \) are assumed to be non-empty. For any \( x \in X \) and \( u \in U \), we denote by \( \text{Post}^S_u(x) := \{x' \in X \mid (x, u, x') \in \rightarrow\} \) the set of \( u \)-successors of \( x \) in \( S \). When \( S \) is known from the context, the set of \( u \)-successors of \( x \) is simply denoted by \( \text{Post}_u(x) \). The inputs admissible to a state \( x \) of system \( S \) is denoted by \( U_S(x) := \{u \in U \mid \text{Post}_u(x) \neq \emptyset\} \).

For any output element \( y \in Y \), the map \( H^{-1} : Y \rightarrow 2^X \) recovers the underlying set of states \( X_y \subseteq X \) generating \( y \), and it is defined as follows: \( H^{-1}(y) := \{x \in X \mid H(x) = y\} \).

We sometimes abuse the notation and apply maps \( H \) and \( H^{-1} \) to subsets of \( X \) and \( Y \), respectively, which refers to applying them element-wise and then taking the union.

The definitions of static, autonomous, total, deterministic, and symbolic systems follow the ones in [4]. System \( S \) is state-based when \( X = Y, H = \text{id}_X \), and \( X_0 = X \). For any \( \bar{x} \subseteq X_0 \), we denote by \( S^{(\bar{x})} \) the restricted version of \( S \) with \( X_0 = \bar{x} \). For any output-based system \( S \), one can always construct its state-based version by assuming the availability of state information, i.e., \( Y = X, X_0 = X \) and \( H = \text{id}_X \), and we denote it by \( S_X \).
Let $S$ be an output-based system. Map $\bar{U}_S : Y \to 2^U$ provides all inputs admissible to outputs of $S$. It is defined as follows for any $y \in Y$:

$$\bar{U}_S(y) := \bigcap_{x \in H^{-1}(y)} U_S(x).$$

Additionally, for any $y \in Y$ and $u \in \bar{U}_S(y)$, $\overline{\text{Post}}^S_{u}(y)$ denotes all $u$-successor observations of $y$ and we define it as follows:

$$\overline{\text{Post}}^S_{u}(y) := H( \bigcup_{x \in H^{-1}(y)} \text{Post}^S_{u}(x)).$$

Given a system $S$, for all $x \in X$ and $\alpha \in U^*$ such that $|\alpha| \geq 1$, $x' \in X$ is called an $\alpha$-successor of $x$, if there exist states $x_0, \ldots, x_{|\alpha|} \in X$ such that $x_0 = x$, $x_{|\alpha|} = x'$, and $(x_i, \alpha_i, x_{i+1}) \in \rightarrow$ for all integers $0 \leq i \leq |\alpha| - 1$. The set of $\alpha$-successors of a state $x \in X$ (resp., a subset $X' \subset X$) is denoted by $\text{Post}^\alpha(x)$ (resp., $\text{Post}^\alpha(X') := \cup_{x \in X'} \text{Post}^\alpha(x)$). For all $x \in X$, $\alpha \in U^*$ and $\beta \in Y^*$ such that $|\alpha| + 1 = |\beta|$, $x' \in X$ is called an $(\alpha, \beta)$-successor of $x$, if there exist states $x_0, \ldots, x_{|\alpha|} \in X$ such that $x_0 = x$, $x_{|\alpha|} = x'$, $H(x_{|\alpha|}) = \beta_{|\alpha|}$, and $H(x_i) = \beta_i$ and $(x_i, \alpha_i, x_{i+1}) \in \rightarrow$ for all integers $0 \leq i \leq |\alpha| - 1$. The set of $(\alpha, \beta)$-successors of a state $x \in X$ (resp., a subset $X' \subset X$) is denoted by $\text{Post}^{\beta}_{\alpha}(x)$ (resp., $\text{Post}^{\beta}_{\alpha}(X') := \cup_{x \in X'} \text{Post}^{\beta}_{\alpha}(x)$).

An internal run of system $S$ is an infinite sequence $r_{\text{int}} := x_0u_0x_1u_1 \cdots x_{n-1}u_{n-1}x_n \cdots$ such that $x_0 \in X_0$, and for any $i \geq 0$ we have $(x_i, u_i, x_{i+1}) \in \rightarrow$. An external run is an infinite sequence $r_{\text{ext}} := y_0u_0y_1 \cdots y_{n-1}u_{n-1}y_n \cdots$ such that $y_0 = H(x_0)$ for some $x_0 \in X_0$, and for any $i \geq 0$ there exist $x_i \in X$ and $x_{i+1} \in X$ such that $y_i = H(x_i)$, $y_{i+1} = H(x_{i+1})$, and $(x_i, u_i, x_{i+1}) \in \rightarrow$. The internal (resp., external) prefix up to $x_n$ (resp., $y_n$) of $r_{\text{int}}$ (resp., $r_{\text{ext}}$) is denoted by $r_{\text{int}}(n)$ (resp., $r_{\text{ext}}(n)$) and its last element is $\text{Last}(r_{\text{int}}(n)) := x_n$ (resp., $\text{Last}(r_{\text{ext}}(n)) := y_n$). The set of all internal (resp., external) runs and the set of all internal (resp., external) $n$-length prefixes are denoted by $\text{RUNS}_{\text{int}}(S)$ (resp., $\text{RUNS}_{\text{ext}}(S)$) and $\text{PREFS}_{\text{int}}^n(S)$ (resp., $\text{PREFS}_{\text{ext}}^n(S)$), respectively. A state $x$ is said to be reachable iff there exists at least one internal prefix $r_{\text{int}}(n) \in \text{PREFS}_{\text{int}}^n(S)$ such that $\text{Last}(r_{\text{int}}(n)) = x$ for some $n \in \mathbb{N}$.

For composing systems, we follow the notation in [4]. Given two systems $S_i, i \in \{1, 2\}$, a serial composition of $S_1$ and $S_2$ is denoted by $S_2 \circ S_1$ and their feedback composition is denoted by $S_1 \times S_2$. We also introduce an observation composition.

**Definition III.2** (Observation Composition). Consider two systems $S_i := (X_i, X_{i,0}, U_i, \rightarrow, Y_i, H_i), i \in \{1, 2\}$, such that $U_1 \times Y_1 \subseteq U_2$. The observation composition of $S_1$ and $S_2$, denoted by
Given a specification $S$, a new system $S_{12} := (X_1 \times X_2, X_{1,0} \times X_{2,0}, U_1, \longrightarrow, X_2, H_{12})$, where $((x_1, x_2), u_1, (x_1', x_2')) \in \longrightarrow$ iff there exist two transitions: $(x_1, u_1, x_1') \in \longrightarrow$ and $(x_2, (u_1, H_1(x_1)), x_2') \in \longrightarrow$, and $H_{12} := \pi_{X_2}$.

Let $S$ be a system as defined in Definition III.1. The internal and external behaviors of $S$ are subsets of the set of all (possibly infinite) internal and external prefixes of $S$, i.e., $B_{\text{int}}(S) \subseteq \bigcup_{n \in \mathbb{N} \cup \{\infty\}} \text{PREFS}_{\text{int}}^n(S)$ and $B_{\text{ext}}(S) \subseteq \bigcup_{n \in \mathbb{N} \cup \{\infty\}} \text{PREFS}_{\text{ext}}^n(S)$. Specifications are defined next.

**Definition III.3** (Specification). Let $S$ be a system as defined in Definition III.1. Let $\Gamma_S := \pi_Y(B_{\text{ext}}(S))$ be the set of all output sequences of $S$. A specification $\psi \subseteq \Gamma_S$ is a set of output sequences that must be enforced on $S$. System $S$ satisfies $\psi$ (denoted by $S \models \psi$) iff $\pi_Y(B_{\text{ext}}(S)) \subseteq \psi$.

Now, we introduce the control problem considered in this article. We then introduce controllers and their domains.

**Problem III.4** (Control Problem). Consider a system $S$ as defined in Definition III.1. Let $\psi$ be a given specification on $S$ following Definition III.3. We denote by the tuple $(S, \psi)$ the control problem of finding a system $C$ such that $C \times S \models \psi$.

**Definition III.5** (Controller). Given a control problem $(S, \psi)$ as defined in Problem III.4, a controller solving the control problem is a feedback-composable system $C := (X_C, X_{C,0}, U_C, \longrightarrow_C, Y_C, H_C)$, where $U_C := Y$ and $Y_C := U$. All of $X_C$, $X_{C,0} \longrightarrow_C$, and $H_C$ are constructed such that $C \times S \models \psi$.

**Definition III.6** (Domain of Controller). Consider a controller $C$ solving $(S, \psi)$, as defined in Definition III.5. The domain of $C$ is denoted by $D(C) \subseteq X_0$ and defined as follows:

$$D(C) := \{ x \in X_0 \mid C \times S^i(x) \models \psi \}.$$

### IV. Output-Feedback Refinement Relations

We first revise FRRs [4] and then introduce OFRRs.

**Definition IV.1** (FRR). Consider two state-based systems $S_i := (X_i, X_{i,0}, U_i, \longrightarrow_i, X_i, id_{X_i})$, $i \in \{1, 2\}$, and assume that $U_2 \subseteq U_1$. A strict relation $Q \subseteq X_1 \times X_2$ is an FRR from $S_1$ to $S_2$ if all of the followings hold for all $(x_1, x_2) \in Q$:

1. $U_{S_2}(x_2) \subseteq U_{S_1}(x_1)$,
2. $u \in U_{S_1}(x_2) \implies Q(\text{Post}_{u}^{S_1}(x_1)) \subseteq \text{Post}_{u}^{S_2}(x_2)$, and
3. $x_1 \in X_{1,0} \implies x_2 \in X_{2,0}$.

When $Q$ is an FRR from $S_1$ to $S_2$, this is denoted by $S_1 \preceq_Q S_2$. 
FRRs are only applicable to state-based systems. We introduce OFRRs as extensions of FRRs so that one can construct symbolic models, synthesize symbolic controllers and refine them for output-based systems.

If \( S_1 \) and \( S_2 \) are output-based systems, we use \( S_1 \preceq_Q S_2 \) to denote that \( Q \subseteq X_1 \times X_2 \) is an FRR from \( S_{1,X_1} \) to \( S_{2,X_2} \).

**Definition IV.2 (OFRR).** Consider two output-based systems \( S_i := (X_i, X_{i,0}, U_i, \rightarrow_i, Y_i, H_i) \), \( i \in \{1, 2\} \), such that \( U_2 \subseteq U_1 \). Let \( Q \subseteq X_1 \times X_2 \) be an FRR such that \( S_1 \preceq_Q S_2 \). A relation \( Z \subseteq Y_1 \times Y_2 \) is an OFRR if all of the followings hold:

(i) For any \( (y_1, y_2) \in Z \), \( \bar{U}_{S_2}(y_2) \subseteq \bar{U}_{S_1}(y_1) \),

(ii) For any \( (x_1, x_2) \in Q \), \( \exists (y_1, y_2) \in Z \) s.t. \( y_1 = H_1(x_1) \) \& \( y_2 = H_2(x_2) \), and

(iii) For any \( (y_1, y_2) \in Z \), \( \exists (x_1, x_2) \in Q \) s.t. \( x_1 \in H_1^{-1}(y_1) \) \& \( x_2 \in H_2^{-1}(y_2) \).

Condition (i) ensures the admissibility of inputs of \( S_2 \) for \( S_1 \). This is not restrictive as we show later in Remark [V.2]. Conditions (ii) and (iii) ensure that observed outputs correspond to evolving states that obey a valid FRR between the two systems. For the sake of a simpler presentation, we slightly abuse the notation hereinafter and use \( S_1 \preceq_Z S_2 \) to indicate the existence of OFRR \( Z \) from \( S_1 \) to \( S_2 \).

**Proposition IV.3.** Consider two systems \( S_i := (X_i, X_{i,0}, U_i, \rightarrow_i, Y_i, H_i) \), \( i \in \{1, 2\} \) having \( U_2 \subseteq U_1 \). Let \( Q \subseteq X_1 \times X_2 \) be an FRR such that \( S_1 \preceq_Q S_2 \), \( Y_1 \) partitions \( Y_2 \), and

\[
y \in Y_2 \implies H_1(Q^{-1}(H_2^{-1}(y))) \equiv y.
\]

Then, there exists a unique OFRR \( Z \subseteq Y_1 \times Y_2 \) corresponding to FRR \( Q \) such that \( S_1 \preceq_Z S_2 \).

**Proof.** Let \( Q \subseteq X_1 \times X_2 \) be an FRR. We first prove by construction that \( Z \) exists. Let \( Z \) be as follows:

\[
Z := \{ (y_1, y_2) \mid y_1 = H_1(x_1) \land y_2 = H_2(x_2) \text{ for some } (x_1, x_2) \in Q \},
\]

which satisfies conditions (i)-(iii) in Definition [IV.2].

Now, we prove that \( Z \) is unique. Consider two OFRRs \( Z_1 \) and \( Z_2 \) having the same underlying FRR \( Q \). We show that they are equal. Consider any \( (y_{1,1}, y_{1,2}) \in Z_1 \). We know from condition (iii) in the definition of OFRR \( Z_1 \) that there exists \( (x_1, x_2) \in Q \) such that \( x_1 \in H_1^{-1}(y_{1,1}) \) and \( x_2 \in H_2^{-1}(y_{1,2}) \). We also know from condition (ii) in the definition of OFRR \( Z_2 \) that there exists \( (y_1, y_2) \in Z_2 \) such that \( y_1 = H_1(x_1) \) and \( y_2 = H_2(x_2) \). Clearly, \( y_1 = y_{1,1} \) and \( y_2 = y_{1,2} \) since the output maps are single-valued. This implies that \( (y_{1,1}, y_{1,2}) \in Z_2 \) and, hence, \( Z_1 \subseteq Z_2 \). One can, similarly, show that \( Z_2 \subseteq Z_1 \) which proves that \( Z_1 = Z_2 \).

The following proposition shows that, when two systems are related via an OFRR \( Z \) and as we observe one of the systems, we can always find corresponding outputs of the other system such that the successor outputs of both systems are in \( Z \). Such a feature is used to prove the output-based behavioral inclusion from original systems to symbolic ones in Subsection [V.D].
Proposition IV.4. Consider two systems $S_i := (X_i, X_{i,0}, U_i, \to_i, Y_i, H_i)$, $i \in \{1,2\}$ having $U_2 \subseteq U_1$. Let $Z \subseteq Y_1 \times Y_2$ be an OFRR s.t. $S_1 \preceq Z S_2$. Then, for any $(y_1, y_2) \in Z$ we have:

$$\forall u \in \bar{U}_{S_2}(y_2) \ \forall y' \in \bar{Post}_{u}^{S_1}(y_1) \ \exists y_2' \in \bar{Post}_{u}^{S_2}(y_2) \text{ s.t.} \ ((y_1', y_2') \in Z).$$

Proof. Consider any $(y_1, y_2) \in Z$ and any $u \in \bar{U}_{S_2}(y_2)$. We know by condition (i) in Definition IV.2 that $u \in \bar{U}_{S_1}(y_1)$. We also know from condition (iii) in Definition IV.2 that there exists $(x_1, x_2) \in Q$ such that $y_1 = H_1(x_1)$ and $y_2 = H_2(x_2)$. Now, consider any $x_1' \in \bar{Post}_{u}^{S_1}(x_1)$. Also, consider the output of $x_1'$ which is $y_1' = H_1(x_1') \in H(\bar{Post}_{u}^{S_1}(x_1)) \subseteq \bar{Post}_{u}^{S_1}(y_1)$.

We know from Definition IV.1 for $Q$ that $Q(x_1') \subseteq \bar{Post}_{u}^{S_2}(x_2)$ which implies that there exists $x_2' \in X_2$ such that $(x_1', x_2') \in Q$. From Definition IV.2 for $Z$, there exists $(y_1', y_2') \in Z$ with $y_2' = H_2(x_2')$. What remains is to show that $y_2' \in \bar{Post}_{u}^{S_2}(y_2)$. By definition, we have $\bar{Post}_{u}^{S_2}(y_2) = H_2(\bar{Post}_{u}^{S_2}(H_2^{-1}(y_2)))$. We also know from Definition IV.2 that $x_2 \in H_2^{-1}(y_2)$ which implies that $x_2' \in \bar{Post}_{u}^{S_2}(H_2^{-1}(y_2))$. Note that $y_2' = H_2(x_2')$ implies that $y_2' \in H_2(\bar{Post}_{u}^{S_2}(H_2^{-1}(y_2))) = \bar{Post}_{u}^{S_2}(y_2)$.

\[\square\]

V. OUTPUT-FEEDBACK SYMBOLIC CONTROL

We first introduce control systems.

A. Control systems

Definition V.1 (Control System). A control system is a tuple $\Sigma := (\mathcal{X}, \mathcal{U}, f, \mathcal{Y}, h)$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is the state set; $\mathcal{U} \subseteq \mathbb{R}^m$ is an input set; $\hat{f} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a continuous map satisfying the following Lipschitz assumption: for each compact set $\mathcal{X} \subseteq \mathcal{X}$, there exists a constant $L \in \mathbb{R}^+$ such that

$$\|f(x_1, u) - f(x_2, u)\| \leq L\|x_1 - x_2\|,$$

for all $x_1, x_2 \in \mathcal{X}$ and all $u \in \mathcal{U}$; $\mathcal{Y} = \mathbb{R}_a$ is the output set; and $h : \mathcal{X} \to \mathcal{Y}$ is an output (a.k.a. observation) map.

Let $\mathcal{U}$ be the set of all functions of time from $[a, b] \subseteq \mathbb{R}$ to $\mathcal{U}$ with $a < 0$ and $b > 0$. We define a trajectory of $\Sigma$ by a locally absolutely continuous curve $\xi : ]a, b[ \to \mathcal{X}$ if there exists a $v \in \mathcal{U}$ that satisfies $\dot{\xi}(t) = f(\xi(t), v(t))$ at any $t \in ]a, b[$. We redefine $\xi : [0, t] \to \mathcal{X}$ for trajectories over closed intervals with the understanding that there exists a trajectory $\xi' : ]a, b[ \to \mathcal{X}$ for which $\xi = \xi'|_{[0,t]}$ with $a < 0$ and $b > t$. $\xi_{xv}(t)$ denotes the state reached at time $t$ under input $v$ and with the initial condition $\xi_{xv}(0) = x$. Such a state is uniquely determined since the assumptions on $f$ ensure the existence and uniqueness of its trajectories [14]. System $\Sigma$ is said to be forward complete if every trajectory is defined on an interval of the form $]a, \infty[$. Here, we consider forward complete control systems. We also define $\zeta : [0, t] \to \mathcal{Y}$ as an output trajectory of $\Sigma$ if there exists a trajectory $\xi_{xv}$ over $[0, t]$ such that at any time $\bar{t} \in [0, t]$ we have that $\zeta(\bar{t}) = h(\xi_{xv}(\bar{t}))$. 
B. Control Systems as Systems

Let \( \Sigma \) be a control system as defined in Definition \[ \Box.1 \]. The sampled version of \( \Sigma \) (a.k.a. concrete system) is a system

\[
S_\tau(\Sigma) := (X_\tau, X_\tau, U_\tau, \longrightarrow_\tau, Y_\tau, H_\tau),
\]

that encapsulates the information contained in \( \Sigma \) at sampling times \( k\tau \), for all \( k \in \mathbb{N} \), where \( X_\tau \subseteq X, U_\tau \) is the set of piece-wise constant curves of length \( \tau \) defined as follows:

\[
U_\tau := \{ v_\tau : [0, \tau[ \to U | \forall t \in [0, \tau[ \ (v_\tau(t) = v_\tau(0)) \},
\]

\[
Y_\tau := \{ y_\tau \in Y | \exists x_\tau \in X_\tau (y_\tau = h(x_\tau)) \},
\]

\[
H_\tau := h, \text{ and a transition } (x_\tau, v_\tau, x'_\tau) \in \longrightarrow_\tau \text{ iff there exists a trajectory } \xi : [0, \tau[ \to X \text{ in } \Sigma \text{ such that } \xi_{x_\tau v_\tau}(\tau) = x'_\tau. \]

We sometimes use \( S_\tau \) to refer to the sampled-data system \( S_\tau(\Sigma) \).

**Remark V.2.** System \( S_\tau \) is deterministic since any trajectory of \( \Sigma \) is uniquely determined. Sets \( X_\tau \) and \( U_\tau \) are uncountable, and hence, \( S_\tau \) is not symbolic. Since all trajectories of \( \Sigma \) are defined for all inputs and all states, we have \( U_{S_\tau}(x_\tau) = U_\tau \), for all \( x_\tau \in X_\tau \), and \( \bar{U}_{S_\tau}(y) = U_\tau \), for all \( y \in Y_\tau \).

We also consider a state-based version of \( S_\tau \) (denoted by \( S_{\tau,X}(\Sigma) \)) and defined as follows:

\[
S_{\tau,X}(\Sigma) := (X_\tau, X_\tau, U_\tau, \longrightarrow_\tau, X_\tau, \text{id}_{X_\tau}).
\]

C. Symbolic Models of Control Systems

We utilize OFRRs (and their underlying FRRs) to construct symbolic models that approximate \( S_\tau \). Given a control system \( \Sigma \), let \( S_\tau \) be its sampled-data representation, as defined in \( \Box.2 \). A symbolic model of \( S_\tau \) is a system:

\[
S_q := (X_q, X_q, U_q, \longrightarrow_q, Y_q, H_q),
\]

where \( X_q := X_\tau/\bar{Q} \), \( \bar{Q} \) is a finite equivalence relation on \( X_\tau \), \( U_q \) is a finite subset of \( U_\tau \), \( (x_q, u_q, x'_q) \in \longrightarrow_q \) if there exist \( x \in x_q \) and \( x' \in x'_q \) such that \( (x, u_q, x') \in \longrightarrow_\tau \), \( Y_q := H_\tau(X_\tau)/\bar{Z} \), where \( \bar{Z} \) is a finite equivalence relation on \( Y_\tau \), \( H_q(x_q) := \{ y_q \in Y_q | y_q \cap H_\tau(x_q) \neq \emptyset \} \), and condition \( \Box.1 \) holds for \( S_1 := S_\tau \) and \( S_2 := S_q \).

Starting with a given equivalence relation \( \bar{Z} \) on \( Y_\tau \), one can construct the underlying equivalence relation \( \bar{Q} \) on \( X_\tau \) using the following relation condition for any \( (x_a, x_b) \in \bar{Q} \):

\[
x_a \sim x_b \iff (H_\tau(x_a), H_\tau(x_b)) \in \bar{Z},
\]

(5)
which ensures that condition (1) is satisfied. The following theorem shows that the above introduced construction of \( S_q \) implies the existence of some OFRR \( Z \) such that \( S_\tau \preceq_Z S_q \).

**Theorem V.3.** Let \( S_\tau \) be defined as in (2). Also, let \( S_q \) be defined as in (4) for some equivalence relations \( Q \) on \( X_\tau \) and \( Z \) on \( H_\tau(X_\tau) \). Then,

\[
Z := \{(y, [y]) \in Y_\tau \times Y_q \mid y \in H_\tau(X_\tau)\},
\]

is an OFRR such that \( S_\tau \preceq_Z S_q \) and

\[
Q := \{(x, [x]) \in X_\tau \times X_q \mid x \in X_\tau\},
\]

is its underlying FRR.

**Proof.** First, we show that \( Q \) is an FRR. Clearly, conditions (i) and (iii) in Definition [IV.1] hold since \( S_\tau \) represents a control system. See Remark [V.2] for more details. We show that condition (ii) holds. Consider any \( (x, [x]) \) \( \in Q \) and any input \( u_q \in U_q([x]) \). Also consider any successor state \( x' \in \text{Post}^{S_\tau}_{u_q}(x) \). Remark that \( x \in [x] \) and \( x' \in [x'] \) since \( Q \) is an equivalence relation. Now, from the definition of \( S_q \) in (4), we know that there exists a corresponding transition \( ([x], u_q, [x']) \) in \( \rightarrow \). Since, \( [x'] \in Q(x) \), by the definition of \( Q \), we have that \( [x'] \in \text{Post}^{S_q}_{u_q}([x]) \). Consequently, \( Q \) is an FRR from \( S_{\tau,X_\tau} \) to \( S_{q,X_q} \).

Now, we show that \( Z \) is an OFRR. Again, condition (i) in Definition [IV.2] holds since \( S_\tau \) represents a control system.

We show that condition (ii) in Definition [IV.2] holds. Consider any \((x, [x]) \) \( \in Q \). Since \( x \in X_\tau \), there exists one observation \( y := H_\tau(x) \). Note that \( [x] \in X_q \). Now, by the definition of \( Y_q \) in (4), we know there exists \([y] \in Y_q\) such that \([y] = H_\tau([x])\). Finally, by the definition of \( Z \), which is based on the equivalence relation \( \bar{Z} \), we have that \((y, [y]) \in Z\), and this, consequently, satisfies condition (ii) in Definition [IV.2].

We show that condition (iii) in Definition [IV.2] holds. Consider any \((y, [y]) \) \( \in Z \). Note that \( y \in H_\tau(X_\tau) \) (i.e., inside the the image of \( X_\tau \) using \( H_\tau \)). From the definition of system \( S_\tau \) in (2), we know that there exists \( x \in X_\tau \) such that \( x = H_\tau^{-1}(y) \). Also, we know from condition (1) and the definition of \( X_q \) that there exists \([x] \in X_q\) such that \([x] = H_\tau^{-1}([y])\). Finally, by the definition of \( Q \), which is based on the equivalence relation \( \bar{Q} \), we conclude that \((x, [x]) \in Q\), and this, consequently, satisfies condition (iii) in Definition [IV.2].

Now, recall Proposition [IV.3] and set \( S_1 := S_\tau \) and \( S_2 := S_q \). Hence, we have that \( S_\tau \preceq_Z S_q \).

\[\square\]

**D. Synthesis and Refinement of Symbolic Controllers**

Let \( \psi_q \) be a given output-based specification on \( S_q \) as introduced in (4). \( \psi_\tau \) is the corresponding concrete specification that should be enforced on \( S_\tau \) and it is interpreted as follows:

\[
\psi_\tau := \{\bar{s} \in \Gamma_{S_\tau} \mid \exists s \in \psi_q \forall i \in \{0, 1, \ldots, |s| - 1\} (s_i = Z(\bar{s}_i))\} \quad (6)
\]

Here, \( S_\tau \) and \( \psi_\tau \) represent together a concrete control problem \((S_\tau, \psi_\tau)\), whereas \((S_q, \psi_q)\) represents an abstract control problem. To algorithmically design controllers solving \((S_\tau, \psi_\tau)\), we utilize \((S_q, \psi_q)\) to automatically synthesize a symbolic controller \( C_q \) that can be refined to
solve \((S_r, \psi_r)\). Later in Section III we propose a methodology for synthesizing \(C_q\), which is then refined with a suitable interface to a controller \(C_r\) that solves the concrete control problem \((S_r, \psi_r)\).

Now, we show that OFRRs preserve the behavioral inclusion from concrete systems to symbolic models.

**Theorem V.4.** Consider systems \(S_r\) and \(S_q\) as introduced in (2) and (4), respectively, where \(Z\) is an OFRR and \(S_r \preceq Z S_q\). Let \(C_q\) be a controller that solves \((S_q, \psi_q)\). Then,

(i) \((C_q \circ Z)\) is feedback-composable with \(S_r\);

(ii) \(Z(B_{init}(C_q \circ Z) \times S_r)) \subseteq B_{init}(C_q \times S_r)\); and

(iii) \(Z(B_{ext}(C_q \circ Z) \times S_r)) \subseteq B_{ext}(C_q \times S_r)\).

**Proof.** Proof of (i): Let system \(C_r\) be of the form

\[
C_r := C_q \circ Z := (X_C, X_C, 0, U_C, \overrightarrow{C}, Y_C, H_C),
\]

for some sets \(X_C, X_C, 0, U_C, \overrightarrow{C}, Y_C\), and a map \(H_C\). Now, based on the given assumptions and [4, Definition III.2], we have that \(Y_C \subseteq U_q\) and \(Y_r \subseteq U_C\). Since \(U_q \subseteq U_r\), we know that \(Y_C \subseteq U_r\). From Definition IV.1 and since \(C_q\) is feedback-composable with \(S_r\), we get

\[
y_q = H_q(x_q) \land u_q = H_C(q, x_q) \land \text{Post}^{C_q}_{q}(x_C) = \emptyset
\]

\[
\implies \text{Post}^{C_q}_{q}(x_q) = \emptyset.
\]

From condition (i) in Definition IV.2 and considering \(Z\) as a serially composed static map with \(C_q\), we get

\[
y_r = H_r(x_r) \land u_q = H_C(x_C) \land \text{Post}^{C_r}_{r}(x_C) = \emptyset
\]

\[
\implies \text{Post}^{C_r}_{r}(x_r) = \emptyset,
\]

which completes the proof of (i).

Proof of (ii): The results in [4, Theorem V.4] are directly applicable here since \(S_r, X_q\) and \(S_q, X_q\) are state-based systems that are related via an FRR. This completes the proof of (ii).

To proof (iii), consider any external run \(r_{C_r \times S_r, ext} \in B_{ext}(C_r \times S_r)\) defined as:

\[
r_{C_r \times S_r, ext} := (u_0^0, y_0^0)(u_1^1, y_1^1)(u_i^i, y_i^i)\cdots
\]

where \(i \in \mathbb{N}\). According to Definitions III.1 and [4, Definition III.3], there exist two external runs:

\[
r_{C_r, ext} := u_0^0 y_0^0 u_1^1 y_1^1 \cdots u_i^i y_i^i \cdots
\]

\[
r_{S_r, ext} := u_0^0 y_0^0 u_1^1 \cdots u_i^i u_i^i \cdots
\]

where \(i \in \mathbb{N}\).

Notice how the output sets \(Y_r\) and \(Y_q\) are constructed in (2) and (4), respectively. Both of them use map \(H_r\) to project the state set \(X_r\). Then, one can easily show that \(Z\) is a strict relation. Now, using the given relation \(Z\) and for any \(y_i^i, i \in \mathbb{N}\), we know that there exists a corresponding \(y_q^i \in Y_q\) such that \((y_q^i, y_i^i) \in Z\). This allows us to apply \(Z\) on the concrete output elements of each of the runs in (7).

Now, by applying Proposition IV.4 inductively to (7) starting with \((y_r^0, y_q^0) \in Z\), we conclude that the following external run \(r_{S_q, ext} \in B_{ext}(S_q)\) exits:

\[
r_{S_q, ext} := y_q^0 u_q^0 y_q^0 u_q^1 \cdots y_q^i u_q^i \cdots
\]
Also, since map \( Z \) is strict, and it interfaces the input to \( C_q \), one can assume that run \( r_{C_q,ext} \) is synchronized with an run \( r_{C_q,ext} \) given by:

\[
r_{C_q,ext} := u_0^0 y_0^0 u_1^1 y_1^1 \cdots u_i^i y_i^i \cdots .
\]

Again, according to Definitions III.1 and [4, Definition III.3], the two runs \( r_{C_q,ext} \) and \( r_{S_q,ext} \) imply the existence of the external run of the feedback-composed system \( C_q \times S_q \):

\[
r_{C_q \times S_q,ext} := (u_0^0, y_0^0) \cdots (u_i^i, y_i^i) \cdots ,
\]

where \( i \in \mathbb{N} \), which proves that \( r_{(C_q \circ Z) \times S_r,ext} \) is synchronized with \( r_{C_q \times S_q,ext} \), and completes the proof of (iii).

The following corollary shows that internal behavioral inclusion from a concrete closed-loop to a symbolic closed-loop implies an external behavioral inclusion.

**Corollary V.5.** Let \( S_r \) and \( S_q \) be as introduced in (2) and (4), respectively, where \( Z \) is an OFRR and \( S_r \preceq_Z S_q \). Then,

\[
B_{int}(C_q \circ Z) \times S_r) \subseteq B_{int}(C_q \times S_q) \quad \Longrightarrow \quad B_{ext}(C_q \circ Z) \times S_r) \subseteq B_{ext}(C_q \times S_q).
\]

**Proof.** The proof is similar to that of part (iii) in Theorem V.4 by mapping the internal sequences to external sequences. \( \square \)

**Remark V.6.** Given two systems \( S_r \) and \( S_q \) such that \( S_r \preceq_Z S_q \), for some OFRR \( Z \), a controller \( C_q \) that solves the abstract control problem \((S_q, \psi_q)\) can be refined to solve the concrete control problem \((S_r, \psi_r)\) using \( Z \) as a static map.

**Remark V.7.** Theorem V.4 and Corollary V.5 provide general results for output-feedback symbolic control. They can be applied to any methodology that can synthesize controllers (cf. Definition III.5) for the outputs of symbolic models (cf. the definition in (4)) to enforce output-based specifications (cf. Definition III.3).

The next three sections provide example methodologies that realize the introduced framework.
VI. METHODOLOGY 1: GAMES OF IMPERFECT INFORMATION

Two-player games on graphs arise in many computer science problems [15]. We utilize the results in [16]–[18] and construct perfect-information (a.k.a. knowledge-based) games from output-based symbolic models. Then, we solve the abstract control problem (or the game) as presented in [18]. We then refine the synthesized controller in two steps: 1) the symbolic controller synthesized for the game structure is refined to work with the symbolic model, and 2) Theorem V.4 is used to refine the controller once again for the concrete system. Figure 1 provides a high-level overview of this methodology.

A. Output-based Symbolic Control using Two-player Games

We assume having a symbolic model $S_q$, as defined in (4), related via an OFRR $Z$ to a sampled output-based system $S_r$, as defined in (2). The following assumptions are required [17]:

1) the abstract system $S_q$ is total; and
2) the set $\{H_q^{-1}(y_q)|y_q \in Y_q\}$ partitions $X_q$.

The first assumption is not restrictive since all inputs are admissible to all states in control systems (see Remark V.2). The second assumption is already satisfied as we consider quotient systems, based on the definition of the symbolic model in (4) and the result from Theorem V.3.

The symbolic model $S_q$ is seen as a game structure of two players played in rounds. The symbolic controller $C_q$ is named Player1 and, at each game round, it selects an input $u_q \in U_q$ for the game structure $S_q$. A hypothetical player Player2, or simply the symbolic model itself, responds by resolving the nondeterminism and selects a successor $x_q'$ for the state $x_q$ using the supplied input $u_q$ such that $(x_q, u_q, x_q') \in \rightarrow_q$.

$S_q$ is considered as a game structure of imperfect information since Player1 has no access to the states of the game. During the game play, only observations of the game structure are available to Player1. Given an internal run (a.k.a. a play) $r_{S_q,int}$, we construct a corresponding external run $obs_q$ as the unique sequence of observations:

\[ obs_q := r_{S_q,ext} = H_q(\pi_{X_q}(r_{S_q,int})) = y_q,0y_q,1 \cdots y_{q,n-1}y_{q,n} \cdots \].
The knowledge associated with the prefix \( \text{obs}_q(n) := y_{q,0} y_{q,1} \cdots y_{q,n-1} y_{q,n} \) is given by the set:

\[
\begin{align*}
\mathcal{K}(\text{obs}_q(n)) := & \left\{ \text{Last}(r_{S_q,\text{int}}(n)) \mid \\
r_{S_q,\text{int}}(n) & \in \text{PREFS}_{\text{int}}(S_q) \land H(r_{S_q,\text{int}}(n)) = \text{obs}_q(n) \right\},
\end{align*}
\]

which represents the set of possible underlying states expected at the end of the monitored observation sequence. Having an initial knowledge \( s_0 := X_{q,0} \subseteq X_q \), the knowledge \( s_i := \mathcal{K}(\text{obs}_q(i)) \), at any step \( i \in \mathbb{N}, i \geq 1 \), can be constructed iteratively [17, Lemma 2.1] using the received observation and the input [18]:

\[
s_i := \text{Post}_{u_{q,i}}^S(s_{i-1}) \cap H_q^{-1}(\text{Last}(\text{obs}_q(i))),
\]

where \( u_{q,i} \) is the input at time step \( i \).

**Remark VI.1.** Since Player 1 generates the inputs, it can construct the knowledge at every step by having \( s_0 \) and monitoring the observations of the game structure.

**B. Controller Synthesis and Refinement**

Consider a concrete game \((S_\tau, \psi_\tau)\) and its corresponding abstract game \((S_q, \psi_q)\), where \( S_\tau \preceq_Z S_q \) and the specification \( \psi_q \) is constructed from \( \psi_\tau \) using \( Z \) as a static map as introduced in (6). The first goal is to synthesize a controller \( C_q \) that solves \((C_q \times S_q)\).

A strategy for Player 1 is a map \( C : Y_q^{*} \rightarrow U_q \) that accepts a sequence of observations and produces a control input. \( C \) is said to be memoryless strategy (a.k.a. a static controller) if \( C(r \cdot y_q) = C(r' \cdot y_q) \) for all \( r, r' \in Y_q^{*} \). A memoryless strategy \( C \) induces another strategy \( \bar{C} : Y_q \rightarrow U_q \) that works with the last element of the observation for which \( C(r) = \bar{C}(\text{Last}(r)) \) for all \( r \in Y_q^{*} \).

Having a strategy \( C \), we denote by \( \text{Outcome}_{S_q}(C) \) the set of all possible state sequences resulting from closing the loop between \( S_q \) and \( C \), and we define it as follows:

\[
\text{Outcome}_{S_q}(C) := \{ x_{q,0} x_{q,1} \cdots \mid x_{q,0} \in X_{q,0} \land \\
(\forall i \geq 0, x_{q,i}, u_{q,i}, x_{q,i+1}) \in r_q \land u_{q,i} = C(\text{obs}_q(i)) \}.
\]

We say that game \((S_q, \psi_q)\) is solvable when there exists a strategy \( C \) such that for all \( r \in \text{Outcome}_{S_q}(C) \), we have \( H_q(r) \in \psi_q \). The strategy is then called a winning strategy.

To check the existence of a winning strategy, we construct another game of perfect information [16, 18]. The knowledge-based perfect-information game structure is a system:

\[
S^K_q := (X^K, s_0, U_q, \xrightarrow{\kappa}, X^K, \text{id}_{X^K}),
\]
where $X_K := 2^{X_q} \setminus \emptyset$, and $(s_1, u_q, s_2) \in \overrightarrow{K}$ iff there exists an observation $y_q \in Y_q$ such that:

$$s_2 := \text{Post}^{SK}_{u_q}(s_1) \cap H_q^{-1}(y_q).$$

(8)

**Proposition VI.2.** Player1 has a winning strategy in the game $S_q$ starting at the initial set $X_{q,0}$ iff Player1 has a winning strategy in $S^K_q$ starting at $X_{q,0}$.

**Proof.** The proof is similar to that in [16, Proposition 2.1] and [17, Proposition 2.4].

In [18, Algorithm 1], the game of imperfect information is solved using an antichain-based technique. The technique is implemented in a tool named ALPAGA [19]. Using the tool, one can possibly synthesize a winning memoryless strategy $\bar{C}^K$ for the game $(S^K_q, \psi^K_q)$, where $\psi^K_q$ is an extended version of $\psi_q$ constructed by the same tool. The memoryless strategy is refined to work with $S_q$ by embedding it inside the symbolic controller $C_q$:

$$C_q := (X_{C_q}, X_{C_q,0}, U_{C_q}, \overrightarrow{C_q}, Y_{C_q}, H_{C_q}),$$

where

- $X_{C_q} := U_q \times 2^{X_q}$;
- $X_{C_q,0} := \{u_q,0\} \times X_q$, where $u_q,0 \in U_q$;
- $U_{C_q} := Y_q$;
- $\overrightarrow{C_q} := \{((u_q, x_{C_q}), y_q, (u'_q, x'_{C_q})) | x'_{C_q} = \text{Post}^{S_q}_{u_q}(x_{C_q}) \cap H_q^{-1}(y_q) \land u'_q = \bar{C}^K(x'_{C_q})\}$;
- $Y_{C_q} := U_q$; and
- $H_{C_q} := \pi_{U_q}$.

**Remark VI.3.** The strategy $\bar{C}^K$ synthesized via the knowledge-based game is static. The refined game controller $C_q$ contains the symbolic model $S_q$ as a building block inside it, in order to compute the knowledge and, hence, it is not static anymore.

The following theorem shows how the controller is refined and concludes this section.

**Theorem VI.4.** Let $(S_\tau, \psi_\tau)$ be a concrete game and $(S_q, \psi_q)$ be an abstract game, where $S_\tau \leq Z S_q$ and $\psi_q$ is a specification constructed from $\psi_\tau$ using $Z$ as a static map. If a controller $C_q$, as defined in (9), solves the game $(S_q, \psi_q)$ then $(C_q \circ Z)$ solves the game $(S_\tau, \psi_\tau)$.

**Proof.** The proof follows directly from Proposition VI.2 and Theorem VI.4.

VII. METHODOLOGY 2: OBSERVERS FOR CONCRETE SYSTEMS

Observers estimate state values of control systems by observing their input and output sequences. We consider observers of concrete systems for output-based symbolic control. We give an informal overview of the methodology and then present it, in details, in the following subsections. Figure 2 depicts the abstraction and refinement phases of the proposed methodology.
Consider the concrete system $S_\tau$, as introduced in (2), and its symbolic model $S_q$, as introduced in (4). We first design an observer $O$ that estimates the states of $S_\tau$, with some upper bound $\epsilon \in \mathbb{R}^+$ for the error between actual states and observed ones. A state-based symbolic model $\hat{S}_q, \hat{X}_q$ is then related to the observed system and used for symbolic controller synthesis. We show that $\hat{S}_q, \hat{X}_q$ can be directly constructed from $S_q$ by inflating each of its states (a state of $S_q$ is a set in $X_\tau$) by $\epsilon$. The synthesized symbolic controller $C_q$ is finally refined with an interface that uses the observer.

A. Observer Design

Let $S_\tau$ be an output-based system and $S_q$ be its symbolic model, as introduced in (2) and (4), respectively, such that $S_\tau \preceq Z S_q$, where $Z$ is an OFRR. Let $Q$ be the underlying FRR of $Z$. Given a specification $\psi_\tau$, let $(S_\tau, \psi_\tau)$ be a concrete control problem. We first introduce observers and show how they are composed with $S_\tau$.

**Definition VII.1** (Observers). Given a precision $\epsilon > 0$, an observer for concrete system $S_\tau$ is a system:

$$O := (\hat{X}, \hat{X}, \hat{U}, \rightarrow, \hat{X}, id_{\hat{X}}),$$

where $\hat{X} := X_\tau$, $\hat{U} := U_\tau \times Y_\tau$, and $\rightarrow$ is defined such that the following holds for all $x_0 \in X_\tau$ and all $\hat{x}_0 \in \hat{X}$:

$$\forall r_{int} \in \text{RUNS}_{int}(S_\tau(x_0)) \forall \hat{r}_{int} \in \text{RUNS}_{int}(O(\hat{x}_0))$$

$$(\pi_{U_\tau}(r_{int}) = \pi_{U_\tau}(\hat{r}_{int}) \Rightarrow \forall n \geq 1 (||\text{Last}(\hat{r}_{int}(n)) - \text{Last}(r_{int}(n))|| \leq \epsilon)).$$

Note that, for any linear time-invariant control systems, it is always possible to construct $O$ by embedding a Luenberger observer with a suitable gain inside it [20]. Additionally, for some
classes of nonlinear systems, one can utilize high-gain observers [21]. We define the observed system $\hat{S}_\tau$ as the system resulting from composing the observer $O$ to the sampled-data system $S_\tau$ as follows:

$$\hat{S}_\tau := O \triangleleft S_\tau,$$

where $\triangleleft$ denotes the observation composition introduced in Definition [III.2] and the output set of $\hat{S}_\tau$ is consequently equals to $\hat{X} = X_\tau$. Here, $\hat{S}_\tau$ coincides with its state-based system $\hat{S}_{\tau,X_\tau}$ version and we use them interchangeably.

In Definition [VII.1] the distance between the runs of $S_\tau$ and those of $\hat{S}_\tau$ is always upper bounded by $\epsilon$ after the first sampling period. We synthesize symbolic controllers to solve $(S_\tau, \psi_\tau)$ only after the first sampling period. In Subsection [VII-D] we show how to handle the first sampling period.

B. A symbolic model for $\hat{S}_\tau$

We approximate $O$ with a static perturbation map, denoted by $\tilde{O} : X_\tau \Rightarrow X_\tau$ ($\Rightarrow$ denotes set-valued mapping), such that its perturbation is upper bounded by $\epsilon$. Formally, we define map $\tilde{O}$ as follows for any $x \in X_\tau$:

$$\tilde{O}(x) := \{ \tilde{x} \in X_\tau \mid \| x - \tilde{x} \| \leq \epsilon \}.$$

One can simply show that $B(O \triangleleft S_\tau) \subseteq B(\tilde{O} \circ S_{\tau,X_\tau})$. Now let us recall the symbolic model $S_q$ of $S_\tau$. Note that the elements of $X_q$ are disjoint subsets of $X_\tau$. A symbolic model for $\hat{S}_\tau$ is constructed by inflating each state $x_q \in \hat{X}_q$ of $S_q$ by $\epsilon$. Formally, we denote by $\hat{S}_q$ the symbolic model of the observed system $\hat{S}_\tau$ and we define it as follows:

$$\hat{S}_q := (\hat{X}_q, \hat{X}_q, U_q, \hat{\rightarrow}, id_{\hat{X}_q}),$$

where $\hat{X}_q := \{ \bigcup_{x \in x_q} \tilde{O}(x) \mid x \in X_q \}$, and $(\hat{x}_q, u_q, \hat{x}_q') \in \hat{\rightarrow}$ if there exist $x \in \hat{x}_q$ and $x' \in \hat{x}_q'$ such that $((x, \hat{x}), u_q, (x', \hat{x}')) \in \rightarrow$ for some $\hat{x}, \hat{x}' \in \hat{X}$, and $\hat{\rightarrow}$ is the transition relation of $\hat{S}_\tau$.

Notice that $\hat{S}_q$ also coincides with its state-based version $\hat{S}_{\tau,X_q}$ and we use them interchangeably.

**Remark VII.2.** For any $\epsilon > 0$, the elements of $\hat{X}_q$ form a cover of $X_\tau$ and its elements have one-to-one correspondence with the partition elements of $X_q$.

Now we derive a version of the given specification $\psi_\tau$ to be used later for controller synthesis. First, a state-based abstract specification $\psi_{q,X_q}$ is derived using maps $Z$ and $H_q$ as follows:

$$\psi_{q,X_q} := H_q^{-1}(Z(\psi_\tau)).$$

Here, we abuse the notation and apply $Z$ and $H_q^{-1}$ to elements of state
sequences in $\psi_r$. Then, we define a map $\tilde{O}_q : X_q \rightarrow \hat{X}_q$ that accepts a partition element $x_q \in X_q$ and translates it to its corresponding cover element $\hat{x}_q \in \hat{X}_q$. Using $\tilde{O}_q$, any state-based abstract specification $\psi_{q,X_q}$ can be translated to an abstract specification $\psi_{q,\hat{X}_q}$ as follows: $\psi_{q,\hat{X}_q} := \tilde{O}_q(\psi_{q,X_q})$. Finally, we have $(\hat{S}_q,\hat{X}_q,\psi_{q,\hat{X}_q})$ as an observed-based abstract control problem and its construction is depicted with steps (1) to (4) in Fig. 3.

**Remark VII.3.** Although observer $O$ is designed for $S_\tau$, the choice of $\epsilon$ should be based on states set $X_q$ in $S_q$. Selecting a larger value of $\epsilon$ increases the nondeterminism of transitions of $\hat{S}_q,\hat{X}_q$ making control problem $(\hat{S}_q,\hat{X}_q,\psi_{q,\hat{X}_q})$ unsolvable.

C. Controller Synthesis and Refinement

In the previous subsection, we demonstrated how $(S_\tau,\psi_r)$ is translated to $(\hat{S}_q,\hat{X}_q,\psi_{q,\hat{X}_q})$, as depicted in Fig. 3. We know from Corollary V.5 that a controller designed for control problem in step (2) can be refined to solve control problem in step (1). Here, we rely on two facts: the behavior of $S_{\tau,X_r}$ is the internal behavior of $S_\tau$, and $B(O \triangleleft S_\tau) \subseteq B(\tilde{O} \circ S_{\tau,X_r})$. The control problem in step (3) is a symbolic representation of the control problem in step (2) using the FRR $Q$. The results from [4] apply directly and any controller designed to solve the control problem in step (3) can be refined to solve the control problem in step (2) using $Q$ as a static quantization map. The only missing link is how a controller designed to solve the control problem in step (4) is refined to solve the control problem in step (3). Note that in (11), we designed $\hat{S}_q,\hat{X}_q$ by inflating the states of $S_q,X_q$ using the perturbation map $\tilde{O}$. Hence, we can use the results in [4, Theorem VI.4] to ensure the behavioral inclusion when refining the controller designed for the control problem in step (4). We introduce a version of [4, Theorem VI.4] adapted to our notation.
Theorem VII.4. Let \((S_q, X_q, \psi_q)\) be an abstract control problem. Consider an abstract observer-based control problem \((\hat{S}_q, \hat{X}_q, \psi_q)\) constructed using the map \(\tilde{O}_q\). If a controller \(C_q\) solves \((\hat{S}_q, \hat{X}_q, \psi_q)\), then the controller \(C_q \circ \tilde{O}_q\) solves \((S_q, X_q, \psi_q)\).

Proof. The proof is very similar to that of [4, Theorem VI.4] and is omitted here due to lack of space.

D. The First Sampling Period

The symbolic controller \(C_q\) is only valid after the first sampling period. One solution to ensure that the system is ready for \(C_q\) for times \(t \geq \tau\), is to choose an input \(u_p \in U_\tau\) and an initial state set \(X_p \in X_\tau\) satisfying

\[
\forall x_0 \in X_p \forall x_\tau \in \text{Post}_{u_p}^{S_\tau}(x_0) \exists x_q \in \mathcal{D}(C_q) \text{ s.t.} \quad x_\tau \in x_q,
\]

where \(\mathcal{D}\) extracts the controller’s domain as introduced in Definition III.6. Condition (12) ensures that states at times \(t \geq \tau\) remains in \(\mathcal{D}(C_q)\). We then need to solve a special control problem \((S_\tau^{(X_p)}, \psi_p)\), where \(\psi_p\) is defined as follows:

\[
\psi_p := \begin{cases} 
\text{Safe}_{[0,1]}(H_\tau(X_D)), & \text{if } X_p \subseteq \mathcal{D}(C_q) \\
\text{Reach}_{[0,1]}(H_\tau(X_D)), & \text{if } \mathcal{D}(C_q) \subset X_p 
\end{cases}
\]

and \(X_D := \bigcup_{x_q \in \mathcal{D}(C_q)} x_q\). The selection of \(X_p\) is critical and depends on the dynamics of \(\Sigma\). A good strategy is to start with \(X_p = X_D\) and expand (or shrink) it until condition (12) is met for some input \(u_p\). We discuss this again with an example in Section IX.

VIII. METHODOLOGY 3: CONSTRUCTING DETECTORS FOR SYMBOLIC MODELS

We revise the notion of detectability of non-deterministic finite transition systems (NFTS) [22], and use it to design detectors for \(S_q\). First, we introduce non-deterministic finite automata (NFA).

Definition VIII.1. An NFA \(A\) is a tuple \(A := (\mathcal{Q}, \Delta, \delta, q_0, F)\), where \(\mathcal{Q}\) is a finite set of states, \(\Delta\) is a finite set of labels (which is an alphabet), \(\delta \subset \mathcal{Q} \times \Delta \times \mathcal{Q}\) is the transition relation, \(q_0 \in \mathcal{Q}\) is the initial state, and \(F \subset \mathcal{Q}\) is a set of final states.

The transition relation \(\delta\) of NFA \(A\) is extended to \(\delta^* \subset \mathcal{Q} \times \Delta^* \times \mathcal{Q}\) in the usual way: for all \(q, q' \in \mathcal{Q}\), \((q, e, q') \in \delta^*\) iff \(q = q'\); and for all \(q, q' \in \mathcal{Q}\) and \(\sigma_0 \ldots \sigma_{n-1} \in \Delta^* \setminus \{e\}\), \((q, \sigma_0 \ldots \sigma_{n-1}, q') \in \delta^*\) iff there exists \(q_1, \ldots, q_{n-1} \in \mathcal{Q}\) such that \((q, \sigma_0, q_1), (q_1, \sigma_1, q_2), \ldots, (q_{n-1}, \sigma_{n-1}, q') \in \delta\). Hereinafter, we use \(\delta\) to denote \(\delta^*\), as no confusion shall occur. A state
\[ q \in Q \] is said to be reachable from a state \( q' \in Q \), if there exists \( \sigma \in \Delta^* \) such that \((q', \sigma, q) \in \delta\).

A state \( x \in Q \) is called reachable from a subset \( Q' \) of \( Q \), if \( x \) is reachable from some states of \( Q' \). A sequence \( q_0, \ldots, q_n \in Q \) is called a path, if there exist \( \sigma_0, \ldots, \sigma_n-1 \in \Delta \) such that \((q_0, \sigma_0, q_1), \ldots, (q_{n-1}, \sigma_{n-1}, q_n) \in \delta\). A path \( q_0, \ldots, q_n \in Q \) is called a cycle, if \( q_0 = q_n \).

We borrow the concept of limit points from the theory of cellular automata [23] and use it for NFAs. Limit points are defined as the points that can be visited at each time step. If one regards an NFA \( A \) as a system in which each state is initial, and regard each state of \( A \) as a point, then limit points are exactly the states reachable from some cycles. The limit set of \( A \) consists of limit points and we denote it by \( LP(A) \).

A. Detectability of Symbolic Models

Consider a concrete control problem \((S_\tau, \psi_\tau)\) and its abstract control problem \((S_q, \psi_q)\) such that \( S_q \preceq_Z S_\tau \), for some OFRR \( Z \), and \( \psi_q \) is constructed as introduced in (6). We first introduce the concept of detectability for symbolic models.

**Definition VIII.2** (Detectability of Symbolic Models). A symbolic model \( S_q \), as defined in (4), is said to be detectable if there exists \( N \in \mathbb{R}^+ \) such that for all input sequences \( \alpha \in U^* \), \( |\alpha| \geq N \), and all output sequences \( \beta \in Y^* \), \( |\beta| = |\alpha| + 1 \), we have that \( |\text{Post}_\alpha^\beta(X_q)| \leq 1 \).

We introduce Algorithm VIII.3 that takes \( S_q \) as input, and returns NFA \( A \) which is used to check the detectability of \( S_q \).

**Algorithm VIII.3.** Receive a symbolic model \( S_q := (X_q, X_q, U_q, \rightarrow_q, Y_q, H_q) \), and initiate an NFA \( A := (Q, \Delta, \delta, q_0, F) \), where \( Q := \{ \phi \} \), \( \phi \) is a dummy symbol, \( \Delta := \delta := F := \emptyset \), and \( q_0 := \phi \). \( Q_1 := \emptyset \), \( Q_2 := \emptyset \). Let \( \phi \) be a dummy symbol not in \( Y_q \).

1) For each \( y \in Y_q \), denote \( X_y := \{ x \in X_q | H_q(x) = y \} \).

   a) if \( |X_y| = 1 \), then \( Q_1 := Q_1 \cup \{ X_y \} \), \( \Delta := \Delta \cup \{ (\phi, y) \} \), \( \delta := \delta \cup \{ (\phi, (\phi, y), X_y) \} \),

Fig. 4: Output-feedback symbolic control using detectors.
b) else if \(|X_q| > 1\), then \(Q_1 := Q_1 \cup \{Z \subset X_q | |Z| = 2\}\), \(\Delta := \Delta \cup \{(\phi, y)\}\), for each \(Z \subset X_q\) satisfying that \(|Z| = 2\), \(\delta := \delta \cup \{(\phi, (\phi, y), Z)\}\). \(Q := Q \cup Q_1, Q_2 := Q_2 \cup Q_1, Q_1 := \emptyset\).

2) If \(Q_2 = \emptyset\), stop. Else, for each \(q_2 \in Q_2\), denote \(y_0 := H_q(x)\), where \(x \in q_2\), for each \(u \in U_q\) and each \(y \in Y_q\),

a) if \(|post_u^{\beta}(q_2)| = 1\), then \(\Delta := \Delta \cup \{(u, y)\}\), \(\delta := \delta \cup \{(q_2, (u, y), post_u^{\beta}(q_2))\}\), if \(post_u^{\beta}(q_2) \notin Q\) then \(Q_1 := Q_1 \cup \{post_u^{\beta}(q_2)\}\),

b) else if \(|post_u^{\beta}(q_2)| > 1\), then \(\Delta := \Delta \cup \{(u, y)\}\), for each \(Z \subset post_u^{\beta}(q_2)\) satisfying \(|Z| = 2\), \(\delta := \delta \cup \{(q_2, (u, y), Z)\}\), if \(Z \notin Q\) then \(Q_1 := Q_1 \cup \{Z\}\). \(Q := Q \cup Q_1, Q_2 := Q_2 \cup Q_1, Q_1 := \emptyset\).

3) Go to Step (2). (Since \(X_q, U_q\), and \(Y_q\) are finite, the algorithm will terminate.)

Let \(A\) be the NFA resulting from Algorithm \ref{algorithm:controller-synthesis-refinement}. We denote by \(T_t\) the transient period of \(S_q\) and define it as follows:

\[
T_t := \min \{t \in \mathbb{N} | \forall u_1, \ldots, u_t \in U_q \forall y_0, \ldots, y_t \in Y_q \}
\]

\[
(\delta(\phi, (\phi, y_0)(u_1, y_1) \ldots (u_t, y_t)) \neq \emptyset \implies \delta(\phi, (\phi, y_0)(u_1, y_1) \ldots (u_t, y_t)) \subseteq LP(A))
\]

Next, we show how to check the detectability of \(S_q\).

**Theorem VIII.4.** Let \(S_q\) be a symbolic model as introduced in (4). Let \(A\) be the NFA resulting from running Algorithm \ref{algorithm:controller-synthesis-refinement} with \(S_q\) as input and \(T_t\) be its transient period. Then,

(i) \(S_q\) is detectable iff in \(A\), each state reachable from some cycle is a singleton, and

(ii) \(S_q\) is detectable, then for all input sequences \(\alpha \in U_q^*\), \(|\alpha| \geq T_t\), and all output sequences \(\beta \in Y_q^*\), \(|\beta| = |\alpha| + 1\), we have that \(|Post_{\alpha}^{\beta}(X_q)| \leq 1\).

**Proof.** The proof of (i) is given in [22] Theorem 8.1.

The proof of (ii) is given in [22] Proposition 8.1. \(\square\)

**B. Controller Synthesis and Refinement**

Consider a detectable symbolic model \(S_q\). We show how to design a detector for it. Let \(A := (Q, \Delta, \delta, \{\phi\}, F)\) be the NFA resulting from Algorithm \ref{algorithm:controller-synthesis-refinement} with \(S_q\) as input. We introduce the detector system as follows:

\[
D := (X_D, X_{D,0}\times Y_q, \xrightarrow{D}, Y_D, H_D), \tag{13}
\]

where

- \(X_D := X_q \times Q \times \{0, 1\}\);

- \(X_{D,0} := \{(x_q, \phi, 0) | x_q \in X_q\}\);

- \(\xrightarrow{D} := \{(x_q, q, 0), (u_q, y_q), (x_q', q', 1) | (x_q, u_q), x_q' \in \xrightarrow{q} \land (q, (u_q, y_q), q') \in \delta \land |q'| \leq 1 \} \cup \{(x_q, q, f), (u_q, y_q), (x_q', q', f) | (x_q, u_q, x_q') \in \xrightarrow{q} \land (q, (u_q, y_q), q') \in \delta \land (|q'| > 1 \lor f = 1)\};\)
• \( Y_D := X_q \cup \{ p \} \), where \( p \) is a dummy symbol denoting incomplete detection of the state of \( S_q \); and

• \( H_D \) is defined as follows:

\[
H_D((x_q, q, f)) = \begin{cases} 
  x_q & f = 1 \\
  p & f = 0 
\end{cases}
\]

Remark VIII.5. After \( T_t \) sampling periods of providing inputs and observations of \( S_q \) to \( D \), we have that:

1. \( H_D(x_D) \neq p \), for any \( x_D \in X_D \),
2. \( H_D(x_D) \) provides the detected current state of \( S_q \), and
3. \( B_{\text{ext}}(D \triangle (Z \circ S)) = B_{\text{int}}(S_q) \).

A controller \( C_q \), as defined in Definition III.5, can be synthesized to solve \((S_q, \psi_q)\), as discussed in [V-D]. Then, using Theorems [V.4] and Remark [VIII.5.3], \( C_q \) is refined using the detector system \( D \) and the static map \( Z \) as interface, as shown in Fig. 4. We only need to encapsulate \( C_q \), in the following system \( C_m \), to handle the detection signal \( p \):

\[
C_m := (X_{C_m}, X_{C_m,0}, U_{C_m}, \rightarrow_{C_m}, Y_{C_q} \cup \{ \kappa \}, H_{C_m}),
\]

where

• \( \kappa \) is a dummy symbol for unavailability of control inputs;

• \( X_{C_m} := X_{C_q} \cup \{ 0, 1 \} \);

• \( X_{C_m,0} := \{ (x_{C_q}, 0) \mid x_{C_q} \in X_{C_q,0} \} \);

• \( U_{C_m} := U_{C_q} \cup \{ p \} \), where \( p \) is the symbol from (13);

• \( \rightarrow_{C_m} := \{ ((x_{C_q}, 0), u_{C_m}, (x'_{C_q}, 1)) \mid (x'_{C_q}, u_{C_m}, x'_{C_q}) \in \rightarrow_{C_q} \wedge u_{C_m} \neq p \} \cup \{ ((x_{C_q}, 1), u_{C_m}, (x'_{C_q}, 1)) \mid (x'_{C_q}, u_{C_m}, x'_{C_q}) \in \rightarrow_{C_q} \wedge u_{C_m} \neq p \} \cup \{ ((x_{C_q}, 0), u_{C_m}, (x_{C_q}, 0)) \mid u_{C_m} = p \} \);

and

• \( H_{C_m}((x_{C_q}, f)) := \begin{cases} 
  H_{C_q}(x_{C_q}) & f = 1 \\
  \kappa & f = 0
\end{cases} \)

To handle the time period \([0, T_t]\), we need to find a static open-loop controller \( C_p \) that solves \((S_p^X, \psi_p)\), where

\[
\psi_p := \begin{cases} 
  \text{Safe}_{[0, T]}(H_r(X_D)), & \text{if } X_p \subseteq D(C_q) \\
  \text{Reach}_{[0, T]}(H_r(X_D)), & \text{if } D(C_q) \subset X_p
\end{cases}
\]

and \( X_D := \bigcup_{x_q \in D(C_q)} x_q \). \( C_p \) encapsulates at least one control input sequence \( \tilde{u}_p \in U_{T_q}^q \) that results
in an output sequence $\tilde{y}_p \in Y^T_p$ such that $\tilde{y}_p \in \psi_p$. One direct approach to find $\tilde{u}_p$ is via an exhaustive search in $U^T_q$.

**IX. CASE STUDIES**

We provide different examples to demonstrate the practicality and applicability of the presented methodologies. Implementations of all examples are done using available open-source toolboxes and some customized C++ programs developed for each methodology. All closed-loop simulations of refined controllers are done in MATLAB. We use a PC (Intel Xeon E5-1620 3.5 GHz and 32 GB RAM) for all the examples.

In all of the examples, given a concrete system $S_\tau$, we construct a symbolic model $S_q$. We use tool SCOTS\cite{11} to construct $S_q,X_q$. SCOTS can only construct $S,q,X_q$ with an FRR $Q$ in the form:

$$Q := \{ (x_\tau,x_q) \mid x_\tau \in X_\tau \cap x_q \land x_q \in X_q \},$$

where $X_q$ is a partition on $X_\tau$ constructed by a uniform quantization parameter $\eta \in \mathbb{R}^n$. Declaring $\eta$ is sufficient to define $X_q$ and $Q$. $X_q$ is a set of polytopes of identical shapes forming a partition on $X_\tau$. This is a limited structure in constructing $S_q$ that we must comply with. Another restriction imposed by SCOTS is the need to use easily invertible output maps $h$ such that $H_q^{-1}(y_q), y_q \in Y_q$, complies with the hyper-rectangular structure of $X_q$ needed by SCOTS.

**A. Output-Feedback Symbolic Control using Games of Imperfect Information**

We consider one example to illustrate the methodology presented in Section VI. In this example, after constructing $S_q$, tool ALPAGA\cite{19} is used to construct the knowledge-based game $S^K_q$ and synthesize a winning strategy $\bar{C^K}$. We refine the strategy as previously depicted in Fig. [F].

Consider the following dynamics of a DC motor:

$$\begin{bmatrix} x_1 \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} -R/L & 0 & -K/L \\ 0 & 0 & 1 \\ K/J & 0 & -b/J \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1/L \\ 0 \\ 0 \end{bmatrix} v,$$

where $x_1$ is the armature current, $x_2$ is the rotation angle of the rotor, $x_3$ is the angular velocity of the rotor, $v$ is the input voltage, $L := 5 \times 10^{-2}$ is the electric inductance of the motor coil, $R := 5$ is the resistance of the motor coil, $J := 5 \times 10^{-4}$ is the moment of inertia of the rotor, $b := 1 \times 10^{-2}$ is the viscous friction constant, and $K := 0.1$ is both the torque and the back EMF.
constants. We consider a state set \( X_\tau := [-0.6, 0.6] \times [-0.3, 0.3] \times [-4.8, 4.8] \) and an input set \( U_\tau := [-4.25, 4.25] \). One sensor is attached to the motor’s rotor and it can measure \( x_2 \). Hence, the output is as follows:

\[
y := \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},
\]

and, consequently, \( Y_\tau := [-0.3, 0.3] \). We consider a reachability specification with a target set \( T := [0.18, 0.3] \).

To construct \( S_q \), we consider an abstract output spaces \( Y_q := \{ y_0, y_2, y_3, \cdots, y_{30} \} \) that forces a partition on \( Y_\tau \). Here, each \( y_q \in Y_q \) represents one subset in \( Y_\tau \) from 31 subsets by dividing \( Y_\tau \) equally using a quantization parameter 0.02. More precisely, we use an OFRR:

\[
Z := \{(y_\tau, y_q) \in Y_\tau \times Y_q \mid y_q = \lfloor (y_\tau + 0.3) / 0.02 \rfloor \}.
\]

With such a \( Y_q \), the abstract specification is to synthesize a controller to reach any of the symbolic outputs \( y_{24}, y_{25}, \cdots, y_{30} \). To construct \( S_{q,X_q} \), we use the following parameters in \textsc{scots}: a state quantization vector \((0.3, 0.02, 1.6)\), an input quantization parameter 0.75, and a sampling time \( \tau := 0.05 \) seconds. \textsc{scots} constructs \( S_{q,X_q} \) in 2 seconds with \( X_q \) having 1085 elements (each representing a hyper-rectangle in \( X_\tau \)) and \( S_{q,X_q} \) having 88501 transitions. We then define \( H_q \) as follows:

\[
H_q((x_{q,1}, x_{q,2}, x_{q,3})) := y\lfloor (x_{q,2}+0.3)/0.02 \rfloor,
\]

which satisfies condition \( \square \). We pass \( S_q \) to \textsc{alpaga} which takes around 24 hours to construct \( S^K_q \) and synthesize \( \bar{C}^K \), which is then refined as discussed in Fig. \ref{fig:dc_motor_example}.

The closed-loop behavior is simulated in \textsc{matlab} and the output is depicted in Fig. \ref{fig:output}. The target region is highlighted with a green rectangle. The actual initial state of the system is set

\[
\begin{align*}
\text{Fig. 5: The output of the DC motor example.}
\end{align*}
\]
(0, 0, 0), which is of course unknown to the controller.

**B. Output-Feedback Symbolic Control using Observers**

As an example for the methodology presented in Section VII, consider the double-integrator model:

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} = \begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} + \begin{bmatrix}
0 \\
1
\end{bmatrix} v,
\]

where \((x_1, x_2) \in [-1, 1] \times [-5, 5],\) and \(v \in [-10, 10].\) The output of the system is seen through a single sensor monitoring \(x_1,\) i.e., \(y = x_1.\) We consider the following LTL specification:

\[
\psi = \square \Diamond (\text{Target}1) \land \square \Diamond (\text{Target}2),
\]

where \(\Diamond (T)\) denotes the reachability requirement that the output of \(S_q\) visits, at least once, some elements in \(T,\) \(\text{Target}1 := [0.65, 1.0]\) and \(\text{Target}2 := [-1, -0.65]\) are two subsets of \(Y_\tau := [-1, 1].\)

We first design an observer for the system. We choose a precision value of \(\epsilon := 0.001\) and design a Luenberger observer using pole placement. It is then embedded in an observer system \(O\) that fulfills condition (10). System \(O\) is needed in order to refine the designed controller as depicted in Fig. 2.

To construct \(S_q,\) we set \(Y_q := \{y_0, y_1, \cdots, y_{50}\}\) forcing a partition on \(Y_\tau\) such that each \(y_q \in Y_q\) represents one subset of \(Y_\tau\) from 51 subsets by dividing \(Y_\tau\) equally using a quantization parameter 0.04. More precisely, use an OFRR:

\[
Z := \{(y_\tau, y_q) \in Y_\tau \times Y_q \mid y_q = y_{\lfloor (y_\tau+1)/0.04 \rfloor}\}.
\]

Then, we use SCOTS to construct \(S_{q,x_q}\) with a sampling time \(\tau := 0.05\) seconds, a state quantization vector \((0.04, 0.01)\), and an input quantization parameter 1.0. Error \(\epsilon\) is used as a state error parameter in SCOTS to emulate the inflation discussed in Subsection VII-C. SCOTS constructs \(S_{q,x_q}\) in 39 seconds and it has \(5.59325 \times 10^7\) transitions. We then have an output map defined as follows: \(H_q((x_{q,1}, x_{q,2})) := y_{\lfloor (x_{q,1}+1)/0.04 \rfloor},\) which satisfies condition (I). With the above setup, we can use the results of the observer-based methodology and refine any synthesized controller for \(S_{q,x_q}\) using \(O\) and \(Q.\)

We continue with controller synthesis and refinement. Since SCOTS requires specifications over symbolic states, the corresponding symbolic target state sets are computed by \(Q(H_\tau^{-1}(\text{Target}1))\) and \(Q(H_\tau^{-1}(\text{Target}2)),\) respectively. The controller is synthesized in 24 seconds. The set of
possible control-actions for the first sampling period are identified as discussed in Subsection VII-D The input $v := 0$ is selected for the first sampling period.

We simulate the closed-loop in MATLAB with $(0, 0)$ and $(1, 1)$ as initial states of the system and observer, respectively. At the first sampling period, the controller applies input $v := 0$ to keep the system in the controller’s domain. From the second sampling period, we switch to the symbolic controller. Figure 6 depicts the output $y$ and Fig. 7 depicts the applied inputs.

C. Output-Feedback Symbolic Control using Detectors

Now, consider a pendulum system [24]:

$$\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} = \begin{bmatrix}
0 & x_2 \\
-\frac{g}{l} \sin(x_1) & -\frac{k}{m} x_2
\end{bmatrix} + \begin{bmatrix}
0 \\
1
\end{bmatrix} u, \quad y = \begin{bmatrix}
1 & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix},$$

where $x_1 \in [-1, 1]$ is the angular position, $x_2 \in [-1, 1]$ is the angular velocity, $u \in [-1.5, 1.5]$ is the input torque, $g := 9.8$ is the gravitational acceleration constant, $l := 5$ is the length of the pendulum’s massless rod, $m := 0.5$ is a mass attached to the rod, $k := 3$ is the friction’s coefficient, and $y \in [-1, 1]$ is the measured angular position. We consider designing a symbolic controller to enforce the angle of the rod to infinitely alternate between two regions $\theta_1 := [0.3, 0.4]$ and $\theta_2 := [-0.4, -0.3]$. When it reaches one region, the pendulum should hold for 10 consequent time steps.
To construct $S_q$, we set $Y_q := \{y_0, y_1, \cdots, y_{50}\}$ forcing a partition on $Y_\tau$ such that each $y_q \in Y_q$ represents one subset in $Y_\tau$ from 51 subsets by dividing $Y_\tau$ equally using a quantization parameter 0.04. More precisely, we use an OFRR:

$$Z := \{(y_\tau, y_q) \in Y_\tau \times Y_q \mid y_q = y_{\lfloor (y_\tau + 1)/0.04 \rfloor}\}.$$

$S_{q,x_q}$ is constructed using the following parameters: state quantization vector $(0.4, 0.4)$, input quantization parameter 0.15, and a sampling time 2 seconds. The resulting $S_{q,x_q}$ has 25 states and 525 transitions. We then have an output map defined as follows: $H_q((x_{q,1}, x_{q,2})) := y_{\lfloor (x_{q,1}+1)/0.04 \rfloor}$, which satisfies condition (1). We then use the results from Section VIII and refine any synthesized controller for $S_q$.

We implemented Algorithm VIII.3 in C++ and ran it with $S_q$ as input. NFA $A$ has 60 states and 1485 transitions. System $S_q$ is detectable with $T_i = 1$. A controller is synthesized using SCOTS and map $H^{-1}_\tau$ is used to construct a state-based specification. The controller is refined using $Z$ and the detector. A closed-loop simulation is depicted in Fig. 8.

X. RELATED WORKS

The work in [25] provides a symbolic control approach based on outputs. It is limited to partially observable linear time-invariant systems, as long as the system is detectable and stabilizable. Some extensions are made in [26] for probabilistic safety specifications and in [27] for nonlinear systems. The latter is limited to a class of feedback-linearizable systems and the results are limited to safety.

The work in [28] proposes designing symbolic output-feedback controllers for control systems. It designs observers induced by abstract systems and obtain output-feedback controllers similar to the methodology we presented in Section VIII. The authors, unlike our approach, require the availability of a controller for the abstract system when the state of the control system is fully...
measured. Then, they reduce the controller to work with the original system with the designed observer.

In [29,30], the authors use state-based strong alternating approximate simulation relations to relate concrete systems with their abstractions. They make sure that a partition constructed on the output space imposes a partition on the state space, which allows designing output-based controllers using state-based symbolic models. The work in [30] is different from ours in three main directions: (1) our work introduces OFRRs as general relations between the outputs of symbolic models and original systems, (2) we utilize FRRs which avoid the drawbacks of approximate alternating simulation relations (see [4, Section IV] for a comparison between both types of relations), and (3) we introduce multiple practical methodologies that realize the framework we introduced; in Sections [VI, VII, and VIII] In [31], the authors design observers for original systems. Then, the observed state-based systems are related, via FRRs, to state-based symbolic models that are used for controller synthesis. Unlike our work, the behavioral inclusion from original closed-loop to abstract closed-loop is shown in state-based setting. Also, the specifications are given over the states set. In [32], the authors provide an extension to FRR to ensure that controllers designed for state-based symbolic models can be refined to work for output-based concrete systems. Abstractions are designed using a modified version of the knowledge-based algorithm (a.k.a. KAM). Unfortunately, the authors can not decide whether a correct abstraction is constructed or not unless a controller is synthesized which requires to iteratively run the algorithm. KAM needs to be stopped once an upper bound for the number of iterations is reached. Although Algorithm VIII.3 is more restrictive in the sense that KAM can produce an abstraction for a symbolic model that is not detectable, it is more predictable since it always terminates. Additionally, Algorithm VIII.3 runs in polynomial time, while the KAM algorithm runs in exponential time. Hence, although KAM algorithm can work for undetectable systems, Algorithm VIII.3 is significantly more efficient for detectable systems. Having both algorithms available to the designer of symbolic controllers offers a trade-off between decidability and applicability.

The main contributions of this work are:

1. OFRRs are introduced as extensions to FRRs allowing abstractions to be constructed by quantizing the state and output sets of concrete systems, such that the output quantization respects the state quantization in the sense that every quantized state belongs to one quantized output. Symbolic controllers of output-based symbolic models can be refined to work
for output-based concrete systems.

2. OFRRs and the results following them in Section[V] serve as a general framework to host different methodologies of output-feedback symbolic control.

3. We introduced three example methodologies (Sections[VI][VII] and[VIII] to synthesize and refine output-based symbolic controllers for output-based systems.

XI. CONCLUSION

We have shown that symbolic control can be extended to work with output-based systems. OFRR are introduced as tools to relate systems based on their outputs. They allow symbolic models to be constructed by quantizing the state and output sets of concrete systems, such that the output quantization respects the state quantization. Consequently, this allows refining symbolic controllers designed based on the outputs of symbolic models to work with the outputs of original systems. An example methodology for output-feedback symbolic control based on detectors for symbolic models was also introduced.

REFERENCES

[1] P. Tabuada, Verification and control of hybrid systems, A symbolic approach. USA: Springer, 2009.
[2] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada, “Symbolic models for nonlinear control systems without stability assumptions,” IEEE Transactions on Automatic Control, vol. 57, no. 7, pp. 1804–1809, July 2012.
[3] R. Majumdar and M. Zamani, “Approximately bisimilar symbolic models for digital control systems,” in Computer Aided Verification, P. Madhusudan and S. A. Seshia, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 362–377.
[4] G. Reissig, A. Weber, and M. Rungger, “Feedback refinement relations for the synthesis of symbolic controllers,” IEEE Transactions on Automatic Control, vol. 62, no. 4, pp. 1781–1796, April 2017.
[5] C. Baier and J. P. Katoen, Principles of model checking. The MIT Press, April 2008.
[6] A. Pnueli and R. Rosner, “On the synthesis of an asynchronous reactive module,” in Proceedings of the 16th International Colloquium on Automata, Languages and Programming, ser. ICALP ’89. London, UK: Springer-Verlag, 1989, pp. 652–671.
[7] M. Y. Vardi, An automata-theoretic approach to fair realizability and synthesis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 267–278.
[8] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa’ar, “Synthesis of reactive(1) designs,” Journal of Computer and System Sciences, vol. 78, no. 3, pp. 911 – 938, 2012, in Commemoration of Amir Pnueli.
[9] M. Khaled, M. Rungger, and M. Zamani, “Symbolic models of networked control systems: A feedback refinement relation approach,” in 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Sept 2016, pp. 187–193.
[10] M. Zamani, M. M. Jr, M. Khaled, and A. Abate, “Symbolic abstractions of networked control systems,” IEEE Transactions on Control of Network Systems, accepted, to appear. [Online]. Available: https://arxiv.org/abs/1401.6396
[11] M. Rungger and M. Zamani, “Scots: A tool for the synthesis of symbolic controllers,” in Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, ser. HSCC ’16. New York, NY, USA: ACM, 2016, pp. 99–104.
[12] S. Mouelhi, A. Girard, and G. Gossler, “Cosyma: A tool for controller synthesis using multi-scale abstractions,” in Proceedings of 16th International Conference on Hybrid Systems: Computation and Control, ser. HSCC ’13. New York, NY, USA: ACM, 2013, pp. 83–88.
[13] M. Khaled and M. Zamani, “pFaces: An acceleration ecosystem for symbolic control,” in Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, ser. HSCC ’19. New York, NY, USA: ACM, 2019.
[14] E. D. Sontag. Mathematical control theory: Deterministic finite dimensional systems, 2nd ed., ser. Texts in Applied Mathematics, Springer-Verlag, New York, 1999, vol. 6.
[15] A. E. Roth, “Two-person games on graphs,” Journal of Combinatorial Theory, Series B, vol. 24, no. 2, pp. 238 – 241, 1978.
[16] J. H. Reif, “The complexity of two-player games of incomplete information,” Journal of Computer and System Sciences, vol. 29, no. 2, pp. 274 – 301, 1984.
[17] J. Raskin, K. Chatterjee, L. Doyen, and T. Henzinger, Algorithms for Omega-Regular Games with Imperfect Information. Lars Birkedal, 2007, vol. 3:3, pp. 4 – 23.
[18] D. Berwanger, K. Chatterjee, L. Doyen, T. A. Henzinger, S. Raje, and M. Chechik, *Strategy Construction for Parity Games with Imperfect Information*. Springer Berlin Heidelberg, 2008, pp. 325–339.

[19] D. Berwanger, K. Chatterjee, M. De Wulf, L. Doyen, and T. A. Henzinger, *Alpaga: A Tool for Solving Parity Games with Imperfect Information*. Berlin: Springer Berlin Heidelberg, 2009, pp. 58–61.

[20] G. F. Franklin, M. L. Workman, and D. Powell, *Digital Control of Dynamic Systems*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[21] H. K. Khalil and L. Praly, “High-gain observers in nonlinear feedback control,” *International Journal of Robust and Nonlinear Control*, vol. 24, no. 6, pp. 993–1015, 2014.

[22] K. Zhang, L. Zhang, and L. Xie, *Detectability of Non-deterministic Finite-Transition Systems*. Cham: Springer International Publishing, 2020, pp. 165–175. [Online]. Available: https://doi.org/10.1007/978-3-030-25972-3_8

[23] J. Kari, “Theory of cellular automata: A survey,” *Theoretical Computer Science*, vol. 334, no. 1, pp. 3 – 33, 2005.

[24] G. Pola, A. Girard, and P. Tabuada, “Approximately bisimilar symbolic models for nonlinear control systems,” *Automatica*, vol. 44, no. 10, pp. 2508 – 2516, 2008.

[25] S. Haesaert, A. Abate, and P. M. J. V. den Hof, “Correct-by-design output feedback of lti systems,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 6159–6164.

[26] K. Lesser and A. Abate, “Controller synthesis for probabilistic safety specifications using observers,” *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 329 – 334, 2015, analysis and Design of Hybrid Systems ADHS.

[27] K. Lesser and A. Abate, “Safety verification of output feedback controllers for nonlinear systems,” in *2016 European Control Conference (ECC)*, 2016, pp. 413–418.

[28] M. Mizoguchi and T. Ushio, “Deadlock-free output feedback controller design based on approximately abstracted observers,” *Nonlinear Analysis: Hybrid Systems*, vol. 30, pp. 58 – 71, 2018.

[29] G. Pola and M. D. D. Benedetto, “Approximate supervisory control of nonlinear systems with outputs,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec 2017, pp. 2991–2996.

[30] G. Pola, M. D. Di Benedetto, and A. Borri, “Symbolic control design of nonlinear systems with outputs,” *Automatica*, vol. 109, p. 108511, 2019.

[31] W. A. Apaza-Perez, A. Girard, C. Combastel, and A. Zolghadri, “Symbolic observer-based controller for uncertain nonlinear systems,” *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1297–1302, 2021.

[32] R. Majumdar, N. Ozay, and A.-K. Schmuck, “On abstraction-based controller design with output feedback,” in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, ser. HSCC ’20. New York, NY, USA: Association for Computing Machinery, 2020.