A Framework for Output-Feedback Symbolic Control

Mahmoud Khaled, Kuize Zhang, Senior Member, IEEE, and Majid Zamani, Senior Member, IEEE

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 an example methodology to synthesize and refine output-feedback symbolic controllers.

Index Terms—Control systems, control design.

I. INTRODUCTION

Symbolic control [1], [2], [3], [4] is an approach to automatically synthesize certifiable controllers that handle complex requirements including objectives and constraints given by formulae in linear temporal logic 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], [7], [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 the state-of-the-art tools of symbolic controller synthesis [11], [12], [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 (OFFRs) are introduced as means of relating output-based systems and their symbolic models. They are extensions of feedback refinement relations (FRRs) in [4]. OFFRs 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 OFFRs 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.

We then present an example methodology that realizes the introduced framework. We introduce a notion of detectability of output-based symbolic models and an algorithm to check it. Detectors are designed for symbolic models to infer their current and subsequent states. We show how symbolic controllers are designed and refined using the designed detectors. To demonstrate the applicability of the introduced methodology, we present a case study where an output-feedback symbolic controller is designed for a pendulum system. The controller is designed to force the angle of the pendulum’s rod to infinitely-often alternate between two subsets of the set of angles.

II. NOTATION

The identity map on a set $X$ is denoted by $id_X$. Symbols $N, Z, R, R^+$, and $R^+_0$ 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, $s_i, i \in \{0, 1, \ldots, |s| - 1\}$, denotes the $i$th element of $s$, and $s[i, j], j \geq i$, denotes the substring $s_i \cdots s_j$. Symbol $e$ denotes the empty string and $|e| = 0$. We use the dot symbol to concatenate two strings.

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

Manuscript received 27 June 2022; accepted 26 October 2022. Date of publication 2 November 2022; date of current version 30 August 2023. This work was supported in part by the H2020 ERC Starting Grant AutoCPS under Grant 804639 and in part by the National Science Foundation under Grant CNS-2145184. Recommended by Associate Editor U. Topcu. (Corresponding author: Mahmoud Khaled.)

Mahmoud Khaled is with the Faculty of Engineering, Minia University, Minia 61519, Egypt (e-mail: mkhaled@mu.edu.eg).

Kuize Zhang is with the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: kuize.zhang@surrey.ac.uk).

Majid Zamani is with the Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309 USA, and also with the Department of Computer Science, Ludwig Maximilian University of Munich, 80539 Munich, Germany (e-mail: majid.zamani@colorado.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2022.3218932.

Digital Object Identifier 10.1109/TAC.2022.3218932

0018-9286 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
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 nonempty. For any $x \in X$ and $u \in U$, we denote by $\text{Post}^S_u(x) := \{ x' \in X \mid (x, u, x') \rightarrow \}$ 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 $S$ are denoted by $U_S(x) := \{ u \in U \mid \text{Post}^S_u(x) \neq \emptyset \}$.

For any output element $y \in Y$, the map $H^{-1} : Y \rightarrow 2^X$ recovers the underlying set of states $S_X := \{ x \in X \mid H(x) = y \}$, 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 $X_0 = \emptyset$. For any $x \in X$, we denote by $S(x)$ the restricted version of $S$ with $X_0 = \emptyset$. 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 $\bar{U}_S : Y \rightarrow 2^U$ 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 $u \in \bar{U}_S(y)$, $\text{Post}^S_u(y)$ denotes all $u$-successor observations of $y$ and we define it as follows:

$$\text{Post}^S_u(y) := H \left( \bigcup_{x \in H^{-1}(y)} \text{Post}^S_u(x) \right).$$

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

An internal run of system $S$ is an infinite sequence $r_{\text{int}} := x_0, x_1, u_1, x_2, \ldots$ such that $x_0 \in X_0$, and for any $i \geq 0$, we have $(x_i, u_i, x_{i+1}) \rightarrow$. An external run is an infinite sequence $r_{\text{ext}} := y_0, y_1, \ldots, y_{n-1}, u_n, y_n, \ldots$ such that $y_0 = H(x_0)$ for some $x_0 \in X_0$, and for any $i > 0$, 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 $(x_i, u_i, x_{i+1}) \rightarrow$. The internal (respectively, external) prefix up to $x_n$ (respectively, $y_n$) of $r_{\text{int}}$ (respectively, $r_{\text{ext}}$) is denoted by $r_{\text{int}}(n)$ (respectively, $r_{\text{ext}}(n)$) and its last element is $\text{Last}(r_{\text{int}}(n)) := x_n$ (respectively, $\text{Last}(r_{\text{ext}}(n)) := y_n$).

The set of all internal (respectively, external) runs and the set of all internal (respectively, external) $n$-length prefixes are denoted by $\text{RUNS}_{\text{int}}(S)$ (respectively, $\text{RUNS}_{\text{ext}}(S)$) and $\text{PREFIX}_{\text{int}}(n)(S)$ (respectively, $\text{PREFIX}_{\text{ext}}(n)(S)$), respectively. A state $x$ is said to be reachable if there exists at least one internal prefix $r_{\text{int}}(n) \in \text{PREFIX}_{\text{int}}(n)(S)$ 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 $S_1 \circ S_2$ and their feedback composition is denoted by $S_1 \circ S_2$. 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 $S_1 \odot S_2$, is a new system $S_{12} := (X_1 \times X_2, X_{1,0} \times X_{2,0}, U_1 \rightarrow_{12} X_1 \times X_2, Y_1 \rightarrow_{12} Y_1 \times Y_2, H_1 \rightarrow_{12} H_1 \times H_2)$, where $(((x_1, x_2), u_1, (x_1', x_2')) \rightarrow_{12}$ iff there exist two transitions: $(x_1, u_1, x_1') \rightarrow_1$ and $(x_2, (u_1, H_1(x_1)), x_2') \rightarrow_2$, and $H_{12} := \pi_{X_2} H_1$.

Let $S$ be a system as defined in Definition 3.1. The internal and external behaviors of $S$ are subsets of the set of all (possibly infinite) internal and external prefixes of $S$, i.e., $\text{inan}(S) \subseteq \bigcup_{n \in \mathbb{N}} \text{PREFIX}_{\text{int}}(S)$ and $\text{exan}(S) \subseteq \bigcup_{n \in \mathbb{N}} \text{PREFIX}_{\text{ext}}(S)$. Specifications are defined next.

**Definition III.3 (Specification):** Let $S$ be a system as defined in Definition 3.1. Let $\Sigma_S := \pi_X (\text{inan}(S))$ be the set of all output sequences of $S$. A specification $\psi \subseteq \pi_X (\Sigma_S)$ is a set of output sequences that must be enforced on $S$. System $S$ satisfies $\psi$ (denoted by $S \models \psi$) iff $\pi_X (\text{inan}(S)) \subseteq \psi$.

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 3.1. Let $\psi$ be a given specification on $S$ following Definition 3.3. 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 3.4, a controller solving the control problem is a feedback-composable system

$$C := (X_C, X_{C,0}, U_C, \rightarrow_C, Y_C, H_C)$$

where $U_C := Y$ and $Y_C := U$. All of $X_C, X_{C,0}, \rightarrow_C$, 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 3.5. The domain of $C$ is denoted by $\text{D}(C) \subseteq X_0$ and defined as follows:

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

## IV. OUTPUT-FEEDBACK REFINEMENT RELATIONS

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

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

i) $U_2(x_2) \subseteq U_1(x_1)$;

ii) $u \in U_2(x_2) \Rightarrow Q(\text{Post}^{S_1}_{u}(x_1)) \subseteq \text{Post}^{S_2}_{u}(x_2)$;

iii) $x_1 \in X_{1,0} \Rightarrow x_2 \in X_{2,0}$.

When $Q$ is an FRR from $S_1$ to $S_2$, this is denoted by $S_1 \preceq Q S_2$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
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 $S_1 \lessdot S_2$ to denote that $Q \subseteq Q_1 \times Q_2$ is an FRR from $S_1$ to $S_2$.

Definition IV.2 (OFRR): Consider two output-based systems $S_i := (X_i, X_{i,0}, U_i, Y_i, H_i), i \in \{1, 2\}$, such that $U_0 \subseteq U_1$. Let $Q \subseteq X_1 \times X_2$ be an FRR such that $S_1 \lessdot S_2$. A relation $Z \subseteq Y_1 \times Y_2$ is an OFRR if all of the following hold:

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

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

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

Condition (i) ensures the admissibility of inputs of $S_2$ for $S_1$. This is not restrictive as we show later in Remark 5.2. 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 $S_1 \lessdot Z$ to indicate the existence of OFRR $Z$ from $S_1$ to $S_2$.

Proposition IV.3: Consider two systems $S_i := (X_i, X_{i,0}, U_i, Y_i, H_i), i \in \{1, 2\}$, having $U_0 \subseteq U_1$. Let $Q \subseteq X_1 \times X_2$ be an FRR such that $S_1 \lessdot Z, S_2$ partitions $Y_1$, and

$$ y \in \bar{y} = H_1(Q^{-1}(H_2^{-1}(y))) \equiv y. $$

Then, there exists a unique OFRR $Z \subseteq Y_1 \times Y_2$ corresponding to FRR $Q$ such that $S_1 \lessdot Z$.

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) of Definition 4.2.

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 $(y_1, y_2) \in Z_1$. We know from condition (iii) in the definition of OFRR $Z_1$ that there exists $(x_1, x_2) \in Q$ such that $x_1 \in H_1^{-1}(y_1)$ and $x_2 \in H_2^{-1}(y_2)$. We also know from condition (ii) of 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, $y_1 = y_1$ and $y_2 = y_2$ since the output maps are single-valued. This implies that $(y_1, y_2) \in Z_1$ 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 useful to prove the output-based behavioral inclusion from original systems to symbolic ones in Section V-D.

Proposition IV.4: Consider two systems $S_i := (X_i, X_{i,0}, U_i, Y_i, H_i), i \in \{1, 2\}$, having $U_0 \subseteq U_1$. Let $Z \subseteq Y_1 \times Y_2$ be an OFRR s.t. $S_1 \lessdot Z$. Then, for any $(y_1, y_2) \in Z$, we have

$$ \forall u \in \bar{U}_{S_1}(y_1) \forall y'_1 \in \Post_{S_1}^Z(y_1) \exists y_2' \in \Post_{S_2}^Z(y_2) \text{ s.t. } ((y_1, y_2) \in Z). $$

Proof: Consider any $(y_1, y_2) \in Z$ and any $u \in \bar{U}_{S_1}(y_1)$. We know by condition (i) in Definition 4.2 that $u \in \bar{U}_{S_1}(y_1)$. We also know from condition (iii) in Definition 4.2 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 $x'_1 \in \Post_{S_1}^Z(x_1)$. Also, consider the output of $x'_1$, which is $y'_1 = H_1(x'_1) \in H(\Post_{S_1}^Z(x_1)) \subseteq \Post_{S_1}^Z(y_1)$.

We know from Definition 4.1 for $Q$ that $Q(x'_1) \subseteq \Post_{S_2}^Z(x_2)$, which implies that there exists $x'_2 \in X_2$ such that $(x'_1, x'_2) \in Q$. From Definition 4.2 for $Z$, there exists $(y'_2, y) \in Z$ with $y' = H_2(x'_2)$. What remains is to show that $y' \in \Post_{S_2}^Z(y_2)$. By definition, we have $\Post_{S_2}^Z(y_2) = H_2(\Post_{S_2}^Z(H_2^{-1}(y_2)))$. We also know from Definition 4.2 that $x_2 \in H_2^{-1}(y_2)$, which implies that $x'_2 \in \Post_{S_2}^Z(H_2^{-1}(y_2))$. Note that $y' = H_2(x'_2)$ implies that $y' \in H_2(\Post_{S_2}^Z(H_2^{-1}(y_2))) = \Post_{S_2}^Z(y_2)$.

V. OUTPUT-FEEDBACK SYMBOLIC CONTROL

We first introduce control systems.

A. Control Systems

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

$$ \|f(x, u) - f(x', u')\| \leq L\|x - x'\| $$

for all $x, x' \in X$ and all $u \in U$; $\mathcal{Y} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is the output set; and $h : X \rightarrow \mathcal{Y}$ is an output (a.k.a. observation) map.

Let $U$ be the set of all functions of time from $[a, b] \subseteq \mathbb{R}$ to $U$ with $a < 0$ and $b > 0$. We define a trajectory of $\Sigma$ by a locally absolutely continuous curve $\xi : [a, b] \rightarrow \mathcal{X}$ if there exists a $\varepsilon \in U$ that satisfies $\xi(t) = f(\xi(t), v(t))$ for all $t \in [a, b]$. We redefine $\xi : [0, t] \rightarrow \mathcal{X}$ for trajectories over closed intervals with the understanding that there exists a trajectory $\xi' : [a, b] \rightarrow \mathcal{X}$ for which $\xi(t) = \xi'(t)$ with $a < 0$ and $b > t$, $\xi_0(t)$ denotes the state reached at time $t$ under input $v$ and with the initial condition $\xi_0(0) = x_0$. 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 $\zeta : [0, t] \rightarrow \mathcal{Y}$ as an output trajectory of $\Sigma$ if there exists a trajectory $\xi_0(t)$ such that at any time $t \in [0, t]$, we have $\zeta(t) = h(\xi_0(t))$.

B. Control Systems as Systems

Let $\Sigma$ be a control system as defined in Definition 5.1. The sampled version of $\Sigma$ is a discrete system $S_{\tau}(\Sigma) := (X, \Sigma, X_{\tau}, U_{\tau}, \rightarrow_{\tau}, Y_{\tau}, H_{\tau})$, 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:

$$ U_{\tau} := \{u_{\tau} : [0, \tau] \rightarrow U \mid \forall t \in [0, \tau] \{v(t) = u_{\tau}(0)\} \} $$

where $Y_{\tau} := \{y_{\tau} : [0, \tau] \rightarrow \mathcal{Y} \mid \exists x_t \in X \{y_{\tau} = h(x_t)\} \}, H_{\tau} := h$, and a transition $(x, v, x') \rightarrow_{\tau} i$ iff there exists a trajectory $\xi : [0, \tau] \rightarrow \mathcal{X}$ in $\Sigma$ such that $\xi_0 + \tau = x'$. 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 $\Sigma$ 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 $U_{S_r}(x_r) = U_r$, for all $x_r \in X_r$, and $\tilde{U}_{S_r}(y) = U_r$, for all $y \in Y_r$.

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

$$S_{r, X(\Sigma)} := \{ (X_r, X_r, U_r, \rightarrow_r, X_r, \text{id}_{X_r}) \}. \quad (3)$$

C. Symbolic Models of Control Systems

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

$$S_q := \{ X_q, X_q, U_q, \rightarrow_q, Y_q, H_q \} \quad (4)$$

where $X_q := X_r / Q$ is a finite equivalence relation on $X_r$, $U_q$ is a finite subset of $U_r(x, u, x') \in \rightarrow$ if there exist $x \in x_r$ and $x' \in x'_r$ such that $(x, u, x') \rightarrow_y = H_r(x_r) / Z$, where $Z$ is a finite equivalence relation on $Y_r, H(q)_r(x) := \{ y \in Y_r \ | y \cap H_r(x_r) \neq \emptyset \}$, and condition (1) holds for $S_1 := S_r$ and $S_2 := S_q$.

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

$$x_a \sim x_b \iff (H(x_a), H(x_b)) \in \bar{Z} \quad (5)$$

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 $S_r \preceq Z S_q$.

Theorem V.3: Let $S_r$ be defined as in (2). Also, let $S_q$ be defined as in (4) for some equivalence relations $Q$ on $X_r$ and $Z$ on $H_r(x, X_r)$. Then,

$$Z := \{ (y_1, y_2) \in Y_r \times Y_r \ | y \in H_r(x_r) \}$$

is an OFRR such that $S_r \preceq Z S_q$ and

$$Q := \{ (x, x') \in X_r \times X_r \ | x \in X_r \}$$

is its underlying FRR.

Proof: First, we show that $Q$ is an FRR. Clearly, conditions (i) and (iii) in Definition 4.1 hold since $S_r$ represents a control system. See Remark 5.2 for more details. We show that condition (ii) holds. Consider any $(x, x') \in Q$ and any input $u \in U_q(x')$. Also consider any successor state $x'' \in \text{Post}_{x}^{S_r}(x)$. 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 $([x], u, [x'])$ in $\rightarrow_q$. Since $[x'] \in Q(x')$, we have that $[x'] \in \text{Post}_{x}^{S_q}(x)$. Consequently, $Q$ is an FRR from $S_r$ to $S_q$.

Now, we show that $Z$ is an OFRR. Again, condition (i) in Definition 4.2 holds since $S_r$ represents a control system.

We show that condition (ii) in Definition 4.2 holds. Consider any $(x, y) \in Q$. Since $x \in X_r$, there exists on observation $y := H_r(x)$. Note that $y \in X_r$. Now, by the definition of $Y_r$ in (4), we know there exists $y \in Y_r$ such that $y = H_r(x)$. 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 4.2.

We show that condition (iii) in Definition 4.2 holds. Consider any $(y, z) \in Z$. Note that $z \in H_r(x, X_r)$ (i.e., inside the image of $X_r$ using $H_r$). From the definition of system $S_r$, in (2), we know that there exists $x \in X_r$ such that $x = H_r^{-1}(y)$. Also, we know from condition (1) and the definition of $X_q$ that there exists $x' \in X_r$ such that $x' = H_r^{-1}(y)$. 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 4.2.

D. Synthesis and Refinement of Symbolic Controllers

Let $\psi_{\text{in}}$ be a given output-based specification on $S_q$ as introduced in (4). $\psi_{\text{in}}$ is the corresponding concrete specification that should be enforced on $S_r$, and it is interpreted as follows:

$$\psi_{\text{in}} := \{ s \in \Gamma_{S_r} \ | \exists y \in \psi_{\text{in}} \ | y = 0 \}$$

Here, $S_r$ and $\psi_{\text{in}}$ represent together a concrete control problem $(S_r, \psi_{\text{in}})$, whereas $(S_q, \psi_{\text{in}})$ represents an abstract control problem. To algorithmically design controllers solving $(S_r, \psi_{\text{in}})$, we utilize $(S_q, \psi_{\text{in}})$ to automatically synthesize a symbolic controller $C_q$ that can be refined to solve $(S_r, \psi_{\text{in}})$. Later in Section VI, 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 $(S_r, \psi_{\text{in}})$.

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 $C_r$ be a controller that solves $(S_q, \psi_{\text{in}})$. Then,

- i) $(C_q \circ Z)$ is feedback-composable with $S_r$;
- ii) $Z(B_{\text{int}}((C_q \circ Z) \times S_r)) \subseteq B_{\text{int}}(C_q \times S_r)$;
- iii) $Z(B_{\text{ext}}((C_q \circ Z) \times S_r)) \subseteq B_{\text{ext}}(C_q \times S_r)$.

Proof: Consider a state $y \in \psi_{\text{in}}$. From condition (i) in Definition 4.2 and considering $Z$ as a serially composed static map with $C_q$, we get

$$y = H_r(x) \land \exists u \in U_c \land H_C(x) \land \text{Post}_{x}^{S_q}(x) = \emptyset$$

which completes the proof of (i).

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

For the proof of (iii), consider any external run $r_{C_r \times S_r, ext} \in B_{\text{ext}}(C_r \times S_r)$ defined as

$$r_{C_r \times S_r, ext} := (u^0_{\text{ext}}, y^0_{\text{ext}}, 0)(u^1_{\text{ext}}, y^1_{\text{ext}})0 \cdots (u^i_{\text{ext}}, y^i_{\text{ext}})0 \cdots$$

where $i \in \mathbb{N}$. According to Definition 3.1 and [4, Definition III.3], there exist two external runs:

$$r_{C_r, ext} := (u^0_{\text{ext}}, y^0_{\text{ext}}, 0)(u^1_{\text{ext}}, y^1_{\text{ext}})0 \cdots$$

and

$$r_{S_r, ext} := (y^0_{\text{ext}}, u^1_{\text{ext}}, y^1_{\text{ext}}0 \cdots)$$

(7)
Then, one can easily show that $Z$ is a strict relation. Now, using the given relation $Z$ and for any $y_i, i \in \mathbb{N}$, we know that there exists a corresponding $y_i' \in Y_i$ such that $(y_i, y_i') \in Z$. This allows us to apply $Z$ on the concrete output elements of each of the runs in (7).

Now, by applying Proposition 4.4 inductively to (7) starting with $(y_1, y_1') \in Z$, we conclude that the following external run $r_{S_{q, \text{ext}}}$ in $B_{\text{ext}}(S_q)$ exists:

$$r_{S_{q, \text{ext}}} := y_1' a_1 y_2' a_2 \cdots y_n' a_n$$

Also, since map $Z$ is strict, and it interfaces the input to $C_q$, one can assume that run $r_{C_q, \text{ext}}$ in $B_{\text{ext}}(C_q)$ is synchronized with an run $r_{C_{q, \text{ext}}}$ given by

$$r_{C_{q, \text{ext}}} := u_1' b_1 u_2' b_2 \cdots u_n' b_n$$

Again, according to Definition 3.1 and [4, Definition III.3], the two runs $r_{C_q, \text{ext}}$ and $r_{S_{q, \text{ext}}}$ imply the existence of the external run of the feedback-controlled system $C_q \times S_q$:

$$r_{C_q \times S_q, \text{ext}} := (u_1', y_1') \cdot (u_2', y_2') \cdots (u_n', y_n')$$

where $i \in \mathbb{N}$, which proves that $r_{(C_q \times Z) \times S_q, \text{ext}} \in B_{\text{ext}}(C_q \times S_q)$, 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 $S_r \subseteq Z S_q$. Then,

$$B_{\text{int}}((C_q \circ Z) \times S_r) \subseteq B_{\text{int}}(C_q \times S_q)$$

$$\implies B_{\text{ext}}((C_q \circ Z) \times S_r) \subseteq B_{\text{ext}}(C_q \times S_q)$$

Proof: The proof is similar to that of part (iii) in Theorem 5.4 by mapping the internal sequences to external sequences.

Remark V.6: Given two systems $S_r$ and $S_q$ such that $S_r \subseteq Z S_q$, 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 5.4 and Corollary 5.5 provide general results for output-feedback symbolic control. They can be applied to any methodology that can synthesize controllers (cf., Definition 3.5) for the outputs of symbolic models [cf., the definition in (4)] to enforce output-based specifications (cf., Definition 3.3).

The extended version of this article [15] provides example methodologies that realize the introduced framework. One methodology is based on games of imperfect information to synthesize output-feedback controllers for output-based specifications. Using the results in [16], [17], and [18], a knowledge-based system is constructed from $S_q$. Then, the abstract control problem is solved as presented in [18]. Finally, the synthesized controller is refined to work for $S_r$. Another methodology constructs observers to estimate the states of $S_r$ with some quantifiable error. A state-based symbolic model is then related to the observed system and used for controller synthesis. The symbolic controllers are refined with interfaces that use the observer to enforce the specifications on $S_r$. In the next section, we present the third example methodology.

VI. CONSTRUCTING DETECTORS FOR SYMBOLIC MODELS

We revise the notion of detectability of non-deterministic finite transition systems [19], and use it to design detectors for $S_q$. First, we introduce non-deterministic finite automata (NFA).

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

The transition relation $\delta$ of NFA $A$ is extended to $\delta' \subseteq Q \times \Delta \times Q$ in the usual way: for all $q, q' \in Q$, $(q, e, q') \in \delta'$ iff $q = q'$; and for all $q, q' \in Q$ and $\sigma_0, \ldots, \sigma_{n-1} \in \Delta \setminus \{e\}$, $(q, \sigma_0, q_1, \sigma_1, q_2, \ldots, q_{n-1}, \sigma_{n-1}, q' \in \delta'$ iff there exists $q_1, \ldots, q_{n-1} \in Q$ such that $(q, \sigma_0, q_1, \sigma_1, q_2, \ldots, q_{n-1}, \sigma_{n-1}, q' \in \delta$. 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 $q \in Q$ 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 [20] 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 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 $(S_r, \psi_r)$ and its abstract control problem $(S_q, \psi_q)$ such that $S_r \subseteq Z S_q$, for some OFRR $Z$, and $\psi_q$ is constructed as introduced in (6). We first introduce the concept of detectability for symbolic models.

Definition VI.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 $\sigma \in U^*$, $|\sigma| \geq N$, and all output sequences $\beta \in Y^*$, $|\beta| = |\alpha| + 1$, we have that $|\text{Post}^n_u(X_q)| \leq 1$.

We introduce Algorithm 6.3 that takes $S_q$ as input, and returns NFA $A$, which is used to check the detectability of $S_q$.

Algorithm VI.3: Receive a symbolic model $S_q := (X_q, \mathcal{X}_q, U_q, Y_q, H_q, \text{init})$, and initiate an NFA $A := (Q, \Delta, \delta, q_0, F)$, where $Q := \{\emptyset\}$, $\emptyset$ is a dummy symbol not in $Y_q$.

1) For each $y \in Y_q$, denote $X_y := \{x \in X_q | H_q(x) = y\}$.
2) If $|X_y| = 1$, then $Q_1 := Q_1 \cup \{X_y\}$, $\Delta := \emptyset \cup \{(\emptyset, y), X_y\}$, $\delta := \delta \cup \{(\emptyset, y), X_y\}$, $Q := Q \cup Q_1, Q_2 := Q_2 \cup Q_1, Q_1 := \emptyset$.

3) If $Q_2 = \emptyset$, stop. Else, for each $q_2 \in Q_2$, denote $y_0 := H_q(x), x \in q_2$, for each $u \in U_q$ and each $y \in Y_q$.
   a) If $|\text{Post}^n_u(y_0(x))| = 1$, then $\Delta := \emptyset \cup \{(u, y), \emptyset \}$, $\delta := \delta \cup \{(q_2, (u, y), \text{Post}^n_u(y_0(x)))\}$, then $Q_1 := Q_1 \cup \{\emptyset \cup \text{Post}^n_u(y_0(x))\}$.
   b) Else if $|\text{Post}^n_u(y_0(x))| > 1$, then $\Delta := \emptyset \cup \{(u, y), \text{Post}^n_u(y_0(x))\}$, then $Q_1 := Q_1 \cup \{\emptyset \cup \text{Post}^n_u(y_0(x))\}$.

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

Let $\mathcal{A}$ be the NFA resulting from Algorithm 6.3. We denote by $T_1$ the transient period of $S_q$ and define it as follows:
$T_t := \min \{ t \in \mathbb{N} \mid \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 \text{LP}(A)\}.

Next, we show how to check the detectability of $S_q$. 

**Theorem VI.4:** Let $S_q$ be a symbolic model as introduced in (4). Let $A$ be the NFA resulting from running Algorithm 6.3 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 $ii)$ if $S_q$ 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 $|\text{Post}_{\beta}(X_q)| \leq 1$.

**Proof:** The proof of (i) is given in [19, Th. 8.1]. The proof of (ii) is given in [19, Prop. 8.1].

### 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 6.3 with $S_q$ as input. We introduce the detector system as follows:

$D := (X_D, X_{D,0}, U_q \times Y_q, \xrightarrow{D}, Y_D, H_D)$

where

- $X_D := X_q \times \mathbb{Q} \times \{0, 1\}$;
- $X_{D,0} := \{(x_0, 0, 0) \mid x_0 \in X_q\}$;
- $\xrightarrow{D} := \{(\{(x_q, q, 0), (u_q, y_q), (x_q', q', 1)\} \mid (x_q, u_q, x_q') \in X_q \wedge (q, (u_q, y_q), q') \in \Delta \wedge (q' \leq 1) \cup \{(x_q, q, f), (u_q, y_q), (x_q', q', f)\} \mid (x_q, u_q, x_q') \in X_q \wedge (q, (u_q, y_q), q') \in \Delta \wedge (q' > 1 \lor f = 1)\}$;
- $Y_D := X_q \cup \{p\}$, 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 VI.5:** After $T_t$ sampling periods of providing inputs and observations of $S_q$ to $D$, we have the following:

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 $S_q$;
3) $R_{\text{out}}(D \circ S_q) = B_{\text{init}}(S_q)$.

A controller $C_{p}$, as defined in Definition 3.5, can be synthesized to solve $(S_q, V_q)$, as discussed in Section V-D. Then, using Theorem 5.4 and Remark 6.5(3), $C_p$ is refined using the detector system $D$ and the static map $Z$ as interface, as shown in Fig. 1. We only need to encapsulate $C_q$, in the following system $C_m$, to handle the detection signal $p$:

$C_m := (X_{C_m}, X_{C_m,0}, U_{C_m}, \xrightarrow{C_m}, Y_{C_m} \cup \{\kappa\}, H_{C_m})$

where

- $\kappa$ is a dummy symbol for unavailability of control inputs;
- $X_{C_m} := X_q \cup \{0\}$;
- $X_{C_m,0} := \{(x_0, 0) \mid x_0 \in X_q\}$;
- $U_{C_m} := \{U_q \cup \{p\}\}$, where $p$ is the symbol from (8);
- $\xrightarrow{C_m} := \{(x_0, 0), U_{C_m}, (x_{C_m}', 1)\} \mid (x_{C_m}', U_{C_m}, x_{C_m}') \in \xrightarrow{C_q} \wedge u_{C_m} \neq p\} \cup \{(x_{C_m}, 1), U_{C_m}, (x_{C_m}', 1)\} \mid (x_{C_m}', U_{C_m}, x_{C_m}') \in \xrightarrow{C_q} \wedge u_{C_m} \neq p\}.$

![Fig. 1. Output-feedback symbolic control using detectors.](image-url)

**VII. CASE STUDY**

Given a concrete system $S_q$, we construct a symbolic model $S_q$. We use tool SCOTS [11] to construct $S_q, x_q$. SCOTS can only construct $S_q, x_q$ with an FRR $Q$ in the form:

$Q := \{(x, u) \mid x \in X_q \wedge x_q \wedge x \in X_q\}$

where $x_q$ is a partition on $X_r$ 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 $X_r$. 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 $H_{C_q}^{-1}(y_q)$, $y_q \in Y_q$, complies with the hyper-rectangular structure of $X_q$ needed by SCOTS.

Now, consider a pendulum system [21]:

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & -\frac{q}{m} \sin(x_2) \\ -\frac{1}{m} x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \ y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

where $x_1 \in [-1, 1]$ is the angular position, $x_2 \in [-1, 1]$ is the angular velocity, $u \in [-1, 1, 1.5]$ 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 alternates between two regions $\theta_l := [0.3, 0.4]$ and $\theta_r := [-0.4, -0.3]$. When it reaches one region, the pendulum should hold for ten consequent time steps.

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

\[ Z := \{(y_r, y_q) \in Y_r \times Y_q \mid y_q = y((y_r + 1)/0.04)\}. \]

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

We implemented Algorithm 6.3 in C++ and ran it with \( S_q \) as input. NFA \( A \) has 60 states and 1485 transitions. System \( S_q \) is detectable with \( T_q = 1 \). A controller is synthesized using SCOTS and map \( H_q^{-1} \) 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. 2.

VIII. RELATED WORKS

The work in [22] 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 [23] for probabilistic safety specifications and in [24] for nonlinear systems. The latter is limited to a class of feedback-linearizable systems and the results are limited to safety.

The work in [25] 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 VI. 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 [26] and [27], 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 [27] is different from ours in the following 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, Sec. IV] for a comparison between both types of relations).

3) We introduce multiple practical methodologies that realize the framework we introduced; two in the extended version [15] and one in Section VI.

In [28], 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 [29], 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 cannot 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 6.3 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 6.3 runs in polynomial time, while the KAM algorithm runs in exponential time. Hence, although the KAM algorithm can work for undetectable systems, Algorithm 6.3 is significantly more efficient for detectable systems. Having both algorithms available to the designer of symbolic controllers offers a tradeoff between decidability and applicability.

IX. 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 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.

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 an example methodology in Section VI to design output-based symbolic controllers using detectors of symbolic models. Two additional methodologies are given in the extended version of the article [15].

REFERENCES

[1] P. Tabuada, Verification and Control of Hybrid Systems: A Symbolic Approach. Berlin, Germany: Springer, 2009.

[2] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada, “Symbolic models for nonlinear control systems without stability assumptions,” IEEE Trans. Autom. Control, vol. 57, no. 7, pp. 1804–1809, Jul. 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, Germany: Springer, 2012, pp. 362–377.

[4] G. Reissig, A. Weber, and M. Rungger, “Feedback refinement relations for the synthesis of symbolic controllers,” IEEE Trans. Autom. Control, vol. 62, no. 4, pp. 1781–1796, Apr. 2017.

[5] C. Baier and J. P. Katoen, Principles of Model Checking. Cambridge, MA, USA: MIT Press, Apr. 2008.

[6] A. Pr u c i e l l i and R. Rosner, “On the synthesis of an asynchronous reactive module,” in Proc. 16th Int. Colloq. Automata, Lang. Program., 1989, pp. 652–671.

[7] M. Y. Vardi, An Automata-Theoretic Approach to Fair Realizability and Synthesis. Berlin, Germany: Springer, 1995, pp. 267–278.
[8] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa’ar, “Synthesis of reactive(1) designs,” Special Issue in Commemoration of Amir Pnueli, J. Comput. Syst. Sci., vol. 78, no. 3, pp. 911–938, 2012.

[9] M. Khaled, M. Rungger, and M. Zamani, “Symbolic models of networked control systems: A feedback refinement relation approach,” in Proc. 54th Annu. Allerton Conf. Commun., Control, Comput., 2016, pp. 187–193.

[10] M. Zamani, M.M. Jr, M. Khaled, and A. Abate, “Symbolic abstractions of networked control systems,” IEEE Trans. Control Netw. Syst., vol. 5, no. 4, pp. 1622–1634, Dec. 2018.

[11] M. Rungger and M. Zamani, “Scots: A tool for the synthesis of symbolic controllers,” in Proc. 19th Int. Conf. Hybrid Syst., Comput. Control, 2016, pp. 99–104.

[12] S. Mouelhi, A. Girard, and G. Gössler, “Cosyma: A tool for controller synthesis using multi-scale abstractions,” in Proc. 16th Int. Conf. Hybrid Syst.: Comput. Control, 2013, pp. 83–88.

[13] M. Khaled and M. Zamani, “pFaces: An acceleration ecosystem for symbolic control,” in Proc. 22nd ACM Int. Conf. Hybrid Syst., Comput. Control, 2019, pp. 252–257.

[14] E. D. Sontag, Mathematical Control Theory: Deterministic Finite Dimensional Systems (Texts in Applied Mathematics), vol. 6, 2nd ed. New York, NY, USA: Springer-Verlag, 1999.

[15] M. Khaled, K. Zhang, and M. Zamani. “A framework for output-feedback symbolic control,” 2020, arXiv:2011.14848.

[16] J. H. Reif, “The complexity of two-player games of incomplete information,” J. Comput. Syst. Sci., 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,” Log. Methods Comput. Sci., vol. 3, pp. 4–23, 2007.

[18] D. Berwanger, K. Chatterjee, L. Doyen, T. A. Henzinger, S. Raje, and M. Chechik, Strategy Construction for Parity Games With Imperfect Information. Berlin, Germany: Springer, 2008, pp. 325–339.

[19] K. Zhang, L. Zhang, and L. Xie, Detectability of Nondeterministic Finite-Transition Systems. Cham, Switzerland: Springer Int. Publ., 2020, pp. 165–175, doi: 10.1007/978-3-030-25972-3_8.