

# Finite Bisimulation of Reactive Untimed Infinite State Systems Modeled as Automata with Variables

Ratnesh Kumar<sup>†</sup>, Changyan Zhou<sup>†</sup>, Samik Basu<sup>‡</sup>

<sup>†</sup>Department of Electrical & Computer Engineering

<sup>‡</sup>Department of Computer Science

Iowa State University, Ames, IA 50011

**Abstract—** Some discrete event systems such as software are typically infinite state systems, and a commonly used technique for performing formal analysis such as automated verification is based on their finite abstractions. In this paper, we consider a model for reactive untimed infinite state systems called *input-output extended finite automaton (I/O-EFA)*, which is an automaton extended with discrete variables such as inputs, outputs, and data. Using I/O-EFA as a model many value-passing processes can be represented by finite graphs. We study the problem of finding a finite abstraction that is bisimilar to a given I/O-EFA. We present a sufficient condition under which the underlying transition system of an I/O-EFA admits a finite bisimilar quotient. This sufficient condition is *existential* as it relies on the existence of a suitable partition of the state space. We then identify a class of I/O-EFAs for which a partition satisfying our sufficient condition can be *constructed* by inspecting the structure of the given I/O-EFA.

**Keywords:** Extended automata, symbolic transition systems, formal verification, bisimulation equivalence, software modeling, software abstraction, software verification.

## I. INTRODUCTION

In recent years there has been extensive research on symbolic modeling and automated verification of infinite state systems. Examples of symbolic models for untimed discrete event systems include symbolic transition graph (STG) [6] and its extension STGA (STG with assignment) [10], and extended finite state machines (EFSMs) [5]. These models are extensions of automata through incorporation of variables and possess an underlying infinite transition system such as those categorized in [9]. Further these models can be regarded as a special type of hybrid automata [8] with no continuous dynamics (i.e., flow rate of each variable is zero). Untimed infinite state systems deserve a separate attention since a considerable class of software systems can be viewed as consisting of a finite control component and infinite (integer-valued) data component, such as communication protocols, counters, queues, buffers, stacks, parameterized systems (e.g.,  $N$ -philosophers,  $N/M$  readers/writers), mobile networks, etc. Such systems evolve data values from a potentially infinite domain and as a result possess an infinite number of different states.

Verification methods such as model-checking have been invented for the analysis of finite state systems. An important technique for verifying an infinite state system is its reduction

to an equivalent finite state system through abstraction. An abstraction is exact when the abstracted system is bisimulation equivalent to the original system. Approaches to obtain exact finite abstractions have been pursued for timed systems [2], [4], for linear and nonlinear systems [13], [12], and for hybrid systems [3].

In this paper, we consider a model for reactive untimed infinite state systems, called *input-output extended finite automaton (I/O-EFA)*, which is an automaton extended with discrete variables such as inputs, outputs, and data, and study the problem of finding their exact finite abstractions. For I/O-EFAs, besides the usual notion of bisimilarity, one can define a stronger notion, namely that of “late”-bisimilarity [6], [10]. (In [6], [10], the term “early-bisimilarity” is used for what one would define to be bisimilarity for I/O-EFAs; we avoid using “early-bisimilarity” as that can create an illusion.) According to the usual notion of bisimilarity, a system can use its knowledge about the current input in choosing a transition to bisimulate a transition of another system, whereas in the “late” setting the input is read only after choosing a transition for bisimulating a transition of another system. In general, the problem of finding a finite quotient that is bisimilar to an I/O-EFA is undecidable. We present a sufficient condition under which an I/O-EFA admits a finite bisimilar quotient. The sufficient condition we present is for the existence of a finite late-bisimilar quotient, and by virtue of late-bisimilarity being stronger than bisimilarity, it also serves as a sufficient condition for the existence of a finite bisimilar quotient. The condition we present is *existential* as it relies on the existence of a suitable partition of the state space. Next we identify a class of I/O-EFAs for which a partition satisfying our sufficient condition can be *constructed* by inspecting the structure of the given I/O-EFA.

The problem of bisimulation-checking between infinite-state systems modeled as STGAs has been studied in [6], [10]. [10] computes a set of predicate equations whose largest solution produces the condition under which the two STGAs are bisimilar. However, such largest solutions are not automatically computable in general. [11], [7] proposed proof systems for model-checking value passing CCS processes, which again are not decidable in general. The main goal of these works is to develop semi-algorithmic approaches for bisimulation and/or model checking for infinite-state systems and not to identify a restricted subclass for

This work was supported in part by the National Science Foundation under the grants NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, and NSF-ECS-0424048.

which their algorithms are guaranteed to terminate. In contrast we present sufficient conditions under which an I/O-EFA with an infinite state space possesses a finite bisimilar quotient.

Also [9] studies a class of systems called, symbolic transition systems (STSs), and reports a semi-algorithm to compute a finite-index bisimulation relation by recursively performing a partition refinement. The semi-algorithm terminates if and only if the STS possesses a finite bisimilar quotient. A difference between our work and that reported in [9] is that in [9] there is no notion of initial states and hence all states are reachable. We do have a notion of initial states, and so not all states may be reachable, and as a result the conditions we provide is only sufficient since the unreachable states do not have to satisfy any condition for a finite bisimilar quotient to exist. [1] defines a class of infinite state systems, called well-structured systems, for which a finite simulation quotient exists.

Rest of the paper is organized as follows. In Section 2, we define I/O-EFA model. In Section 3, we introduce the notions of bisimilarity and quotient systems. In Section 4, we present a sufficient condition under which an I/O-EFA admits a finite bisimilar quotient. In Section 5, we identify a subclass of I/O-EFAs possessing a finite bisimilar quotient. The paper concludes in Section 6.

## II. INPUT/OUTPUT EXTENDED FINITE AUTOMATA

An input/output extended finite automaton or state machine (I/O-EFA or I/O-EFSM) is a symbolic description of reactive untimed infinite state systems in form of automata extended with discrete variables such as inputs, outputs, and data. Using I/O-EFA as a model many value-passing processes can be represented by finite graphs. An I/O-EFA consists of locations ( $L$ ), data ( $D$ ), inputs ( $U$ ), outputs ( $Y$ ), transition labels ( $\Sigma \cup \{\epsilon\}$ ), transitions ( $E$ ), and initial locations ( $L_0$ ) and data values ( $D_0$ ). Locations together with data form the state-space of I/O-EFAs. Locations are finite and form the vertices of the automaton graph. Edges of the graph represent transitions between locations and are guarded by data and inputs. Occurrence of a transition triggers a data update and an output assignment. Transition labels are used for inter-system synchronization. An  $\epsilon$  label is used for a transition if it is to occur asynchronously. An I/O-EFA is formally defined as follows.

*Definition 1:* An input/output extended finite automaton (I/O-EFA) is an eight-tuple

$$P = (L, D, U, Y, \Sigma, E, L_0, D_0),$$

where

- $L$  is the set of locations,
- $D = D_1 \times \dots \times D_p$  is the set of  $p$ -dimensional data,
- $U = U_1 \times \dots \times U_q$  is the set of  $q$ -dimensional input,
- $Y = Y_1 \times \dots \times Y_r$  is the set of  $r$ -dimensional output,
- $\Sigma \cup \{\epsilon\}$  is the set of transition labels,
- $E$  is the set of edges, and each  $e \in E$  is a 6-tuple,

$$e = (o_e, t_e, \sigma_e, G_e, f_e, h_e),$$

where

- $o_e \in L$  is the origin location,
- $t_e \in L$  is the terminal location,
- $\sigma_e \in \Sigma \cup \{\epsilon\}$  is the transition label,
- $G_e \subseteq D \times U$  is the enabling guard,
- $f_e : D \times U \rightarrow D$  is the data update function,
- $h_e : D \times U \rightarrow Y$  is the output assignment function,
- $L_0$  is the set of initial location, and
- $D_0 = D_{10} \times \dots \times D_{p0}$  is the set of initial data values.

All variables range over countable sets and can be taken to be the set of integers. We use  $\vec{u}$ ,  $\vec{y}$ , and  $\vec{d}$  to denote an input, an output, and a data respectively. We use  $d(i)$  to denote the  $i$ th component of  $\vec{d}$ , i.e.,  $\vec{d} = (d(1), \dots, d(p))$ , where  $p$  is the number of data variables.  $\vec{d}_1 = \vec{d}_2$  means component-wise equality, i.e.,  $(d_1(i) = d_2(i), \forall i \in \{1, \dots, p\})$ . We use  $f_e(i)(\vec{d}, \vec{u})$  (resp.,  $h_e(j)(\vec{d}, \vec{u})$ ) to represent the update function of the  $i$ th data (resp., assignment function of the  $j$ th output). The edge enabling guard  $G_e(D, U)$  is a predicate over data and inputs. Sometimes we write “ $(\vec{d}, \vec{u}) \in G_e(D, U)$ ” also as “ $G_e(\vec{d}, \vec{u})$ ”. We define  $G_e(D, \{\vec{u}\}) := \{\vec{d} \in D \mid (\vec{d}, \vec{u}) \in G_e(D, U)\}$  to be the predicate over data such that for any data satisfying this predicate the edge  $e$  is enabled on input  $\vec{u} \in U$ . Functions  $f_e$  and  $h_e$  can naturally be extended to be defined over sets:  $\forall \hat{D} \subseteq D, \hat{U} \subseteq U$ :

$$f_e(\hat{D}, \hat{U}) := \cup_{\vec{d} \in \hat{D}, \vec{u} \in \hat{U}} \{f_e(\vec{d}, \vec{u})\};$$

$$h_e(\hat{D}, \hat{U}) := \cup_{\vec{d} \in \hat{D}, \vec{u} \in \hat{U}} \{h_e(\vec{d}, \vec{u})\}.$$

Note that there is no requirement that data update and output assignment occur in the same transition (i.e., one or both functions can be identity), and so the model is powerful enough to capture both synchronous and asynchronous systems.

*Remark 1:* I/O-EFA model is more general than STG [6] and its extension STGA [10]. For doing a comparison here we follow our own notation. In a STGA, each edge is associated with a tuple  $(G, f, \alpha)$ .  $G \subseteq D$  is a guard,  $f : D \rightarrow D$  is an update, and  $\alpha \in \{c?x, c!exp, \tau\}$  is a transition label.  $c?x$  denotes an input of a value for a variable  $x$  from a channel  $c$ , and can be represented as a data update  $d(j)(= x) := u(i)(= c)$  in our model.  $c!exp$  denotes an output of the value of the expression  $exp$  in a channel  $c$ , and can be represented as an output assignment  $y(i)(= c) := h_e(\vec{d})(= exp)$  in our model.  $\tau$  denotes an “internal” transition (similar to transition labeled  $\epsilon$  in our setting). Also  $f \neq Id$  ( $Id$  denotes the identity function) if and only if  $\alpha = \tau$ , i.e., a data update on a transition not involving communication occurs only if the transition is labeled by  $\tau$ .

## III. BISIMULATION, TRANSITION AND QUOTIENT SYSTEMS

The section defines bisimilarity and late-bisimilarity for I/O-EFAs, the underlying transition system of an I/O-EFA, and a quotient system of a transition system. We assume, without loss of generality, that none of the transitions are labeled by  $\epsilon$ . We introduce the following notations.

$\forall l, l' \in L, \vec{d}, \vec{d}' \in D, e \in E, \sigma \in \Sigma \cup \{\epsilon\}, \vec{u} \in U, \vec{y} \in Y$ :

$$\begin{aligned} [l \xrightarrow{e} l'] &\Leftrightarrow [o_e = l] \wedge [t_e = l'], \\ [(l, \vec{d}) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l', \vec{d}')] &\Leftrightarrow [\exists e = (l, l', \sigma, G, f, h) \in E \mid \\ &G(\vec{d}, \vec{u}), \vec{d}' = f(\vec{d}, \vec{u}), \vec{y} = h(\vec{d}, \vec{u})]. \end{aligned}$$

*Definition 2:* Given an I/O-EFA  $P$ , a simulation relation over its states is a binary relation  $\Phi \subseteq (L \times D) \times (L \times D)$  such that  $((l_1, \vec{d}_1), (l_2, \vec{d}_2)) \in \Phi$  implies

$$\begin{aligned} &\forall e_1, \forall \vec{u}, \exists e_2 : \sigma_{e_2} = \sigma_{e_1} := \sigma \text{ and} \\ &[(l_1, \vec{d}_1) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_1, \vec{d}'_1), l_1 \xrightarrow{e_1} l'_1] \Rightarrow \\ &\exists [(l_2, \vec{d}_2) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_2, \vec{d}'_2), l_2 \xrightarrow{e_2} l'_2] \text{ s.t.} \\ &((l'_1, \vec{d}'_1), (l'_2, \vec{d}'_2)) \in \Phi. \end{aligned}$$

On the other hand, a late-simulation relation over states of  $P$  is a binary relation  $\Phi \subseteq (L \times D) \times (L \times D)$  such that  $((l_1, \vec{d}_1), (l_2, \vec{d}_2)) \in \Phi$  implies

$$\begin{aligned} &\forall e_1, \exists e_2 : \sigma_{e_2} = \sigma_{e_1} := \sigma \text{ and} \\ &\forall \vec{u}, [(l_1, \vec{d}_1) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_1, \vec{d}'_1), l_1 \xrightarrow{e_1} l'_1] \Rightarrow \\ &\exists [(l_2, \vec{d}_2) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_2, \vec{d}'_2), l_2 \xrightarrow{e_2} l'_2] \text{ s.t.} \\ &((l'_1, \vec{d}'_1), (l'_2, \vec{d}'_2)) \in \Phi. \end{aligned}$$

A symmetric simulation (resp., late-simulation) relation is called bisimulation (resp., late-bisimulation) relation. Two states  $(l_1, \vec{d}_1), (l_2, \vec{d}_2) \in L \times D$  are bisimilar (resp., late-bisimilar), denoted  $(l_1, \vec{d}_1) \simeq (l_2, \vec{d}_2)$  (resp.,  $(l_1, \vec{d}_1) \simeq_l (l_2, \vec{d}_2)$ ), if exists a bisimulation (resp., late-bisimulation) relation  $\Phi$  such that  $((l_1, \vec{d}_1), (l_2, \vec{d}_2)) \in \Phi$ . Two systems  $P_1$  and  $P_2$  are said to be bisimilar (resp., late-bisimilar) if exists a bisimulation (resp., late-bisimulation) relation  $\Phi$  such that for each  $(l_{10}, \vec{d}_{10}) \in L_{10} \times D_{10}$  exists  $(l_{20}, \vec{d}_{20}) \in L_{20} \times D_{20}$  such that  $[(l_{10}, \vec{d}_{10}), (l_{20}, \vec{d}_{20})] \in \Phi$ .

*Remark 2:* The notion of late-bisimulation is strictly finer than bisimulation [6], [10]. It follows that if a system possesses a finite late-bisimulation quotient, it also possesses a finite bisimulation quotient. For this reason we concentrate mainly on the late-bisimulation relation.

The next two definitions define underlying transition systems and quotient systems.

*Definition 3:* Given an I/O-EFA  $P = (L, D, U, Y, \Sigma, E, L_0, D_0)$ , its underlying transition system  $\mathcal{P}$  is a 6-tuple  $\mathcal{P} = (S, U, Y, \Sigma, \mathcal{E}, S_0)$ , where  $S := L \times D$  is its states,  $\mathcal{E} := \{((l_1, \vec{d}_1), \sigma, \vec{u}, \vec{y}, (l_2, \vec{d}_2)) \mid (l_1, \vec{d}_1) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l_2, \vec{d}_2)\}$  is its set of edges (transitions),  $S_0 := L_0 \times D_0$  is its initial states, and the remaining components in the tuple are the same as those in  $P$ .

Recall that a partition of a set  $S$  is a set  $\mathcal{C} \subseteq 2^S$  of non-empty subsets of  $S$  such that all members of  $\mathcal{C}$  are disjoint and  $\mathcal{C}$  covers  $S$ . Also recall that an equivalence induces a partition. Given a partition of the set of states, one can obtain a quotient system as follows.

*Definition 4:* Given an I/O-EFA  $P$  and a partition  $\mathcal{C}$  of the state set  $L \times D$ , the *quotient system* of  $\mathcal{P}$  with respect to the

partition  $\mathcal{C}$  is the transition system  $\mathcal{P}_{\mathcal{C}} = (\mathcal{C}, U, Y, \Sigma, \mathcal{E}_{\mathcal{C}}, \mathcal{C}_0)$ , where

$$\begin{aligned} \mathcal{E}_{\mathcal{C}} &= \{(C_1, \sigma, \vec{u}, \vec{y}, C_2) \mid \exists (l_i, \vec{d}_i) \in C_i \in \mathcal{C}, i = 1, 2, \text{ s.t.} \\ &((l_1, \vec{d}_1), \sigma, \vec{u}, \vec{y}, (l_2, \vec{d}_2)) \in \mathcal{E}\} \end{aligned}$$

and  $\mathcal{C}_0 = \{C \in \mathcal{C} \mid C \cap (L_0 \times D_0) \neq \emptyset\}$ . The quotient system of  $\mathcal{P}$  with respect to the partition induced by an equivalence “eq” over  $L \times D$  is denoted  $\mathcal{P}_{eq}$ .

#### IV. CONDITION FOR FINITE BISIMILAR QUOTIENT

An I/O-EFA possibly has infinitely many states and its reachability set can also be infinite (see for example the I/O-EFA shown in Figure 1). In order to be able to apply existing finite state system verification methods to a system modeled as an I/O-EFA, its underlying transition system should possess a finite bisimilar quotient. In general, the problem of finding a finite bisimilar quotient is undecidable. In this section, we present a sufficient condition under which  $P$  admits a finite bisimilar quotient.

Given an I/O-EFA, we can always partition its data space into a finite number of bounded and unbounded regions. Due to the countable nature of the data space, each bounded region contains a bounded number of data values. Our idea is to have a partition such that each unbounded region and each data value in a bounded region is its own late-bisimulation equivalence class. This will then ensure that the given I/O-EFA possesses a finite late-bisimilar quotient. For a finite partition of the data space to satisfy the above mentioned property, it should hold that for any input the data update (resp., output assignment) along any enabled edge map an unbounded region to either another unbounded region or to some fixed data value (resp., be identical over an entire unbounded region). This is stated and proved in the following theorem.

*Theorem 1:* Given an I/O-EFA  $P$ , it admits a finite late-bisimilar quotient if the data space  $D$  can be partitioned into a finite number of regions,  $\Pi_1, \dots, \Pi_n \subseteq D$  (where  $\Pi_1, \dots, \Pi_m$  are unbounded, and  $\Pi_{m+1}, \dots, \Pi_n$  are bounded) such that

$$\begin{aligned} \forall i \leq m, \forall e, \forall \vec{u} : & [G_e(D, \{\vec{u}\}) \cap \Pi_i \neq \emptyset \Rightarrow \\ & 1. \Pi_i \subseteq G_e(D, \{\vec{u}\}), \text{ and} \\ & 2. (\exists j \leq m : f_e(\Pi_i, \vec{u}) \subseteq \Pi_j) \vee \\ & (\forall \vec{d} \in \Pi_i, f_e(\Pi_i, \vec{u}) = \{f_e(\vec{d}, \vec{u})\}), \text{ and} \\ & 3. \forall \vec{d} \in \Pi_i, h_e(\Pi_i, \vec{u}) \{h_e(\vec{d}, \vec{u})\}. \end{aligned}$$

**Proof:** We define a finite partition of the state space  $L \times D$  of  $P$  based on the given partition of the data space, and show that each member of the partition is a late-bisimulation equivalence class. Consider,

$$\begin{aligned} \mathcal{C} &:= \mathcal{C}_1 \cup \mathcal{C}_2; \\ \mathcal{C}_1 &:= \{\{l\} \times \Pi_i \mid l \in L, i \leq m\}; \\ \mathcal{C}_2 &:= \{\{l, \vec{d}\} \mid l \in L, \vec{d} \in \Pi_i, m < i \leq n\}. \end{aligned}$$

We show that each  $C \in \mathcal{C}$  is a late-bisimulation equivalence class. For this we define a relation  $\Phi \subseteq (L \times D) \times (L \times D)$

as follows,

$$\Phi := \{((l_1, \vec{d}_1), (l_2, \vec{d}_2)) \in C \times C \mid C \in \mathcal{C}\}.$$

We claim that  $\Phi$  is a late-bisimulation relation. Clearly  $\Phi$  is symmetric, and so it suffices to show that it is a simulation relation. Pick  $((l_1, \vec{d}_1), (l_2, \vec{d}_2)) \in \Phi$ , then by definition of  $\mathcal{C}$ ,  $l_1 = l_2 := l$ . Further if  $(l, \vec{d}_1), (l, \vec{d}_2) \in C \in \mathcal{C}_2$ , then  $\vec{d}_1 = \vec{d}_2$ , and so obviously  $(l, \vec{d}_1)$  simulates  $(l, \vec{d}_2)$ . Consider next the case when  $(l, \vec{d}_1), (l, \vec{d}_2) \in C_1$ . Pick any  $e_1 \in E$ . We claim that we can choose  $e_2 = e_1 := e$  to satisfy the definition of a simulation relation. Obviously  $\sigma_{e_2} = \sigma_{e_1} =: \sigma_e$ . We need to further show that for all  $\vec{u} \in U$ ,

$$\begin{aligned} & [(l, \vec{d}_1) \xrightarrow{\sigma_e, \vec{u}, \vec{y}} (l', \vec{d}_1), l \xrightarrow{e} l'_1] \Rightarrow \\ & \exists [(l, \vec{d}_2) \xrightarrow{\sigma_e, \vec{u}, \vec{y}} (l'_2, \vec{d}_2), l \xrightarrow{e} l'_2] \text{ s.t.} \\ & ((l'_1, \vec{d}_1), (l'_2, \vec{d}_2)) \in \Phi. \end{aligned}$$

First note that  $l'_1 = t_e = l'_2$ , i.e.,  $l'_1 = l'_2 := l'$ . Next note that

$$\begin{aligned} [(l, \vec{d}_1) \xrightarrow{\sigma_e, \vec{u}, \vec{y}} (l', \vec{d}_1)] & \Rightarrow [l = o_e, \vec{d}_1 \in G_e(D, \{\vec{u}\}), \\ & \vec{d}_1 = f_e(\vec{d}_1, \vec{u}), \vec{y} = h_e(\vec{d}_1, \vec{u})]. \end{aligned}$$

Since  $\vec{d}_1, \vec{d}_2 \in C \in \mathcal{C}_1$ , exists  $i \leq m$  such that  $\vec{d}_1, \vec{d}_2 \in \Pi_i$ . Since  $\vec{d}_1 \in G_e(D, \{\vec{u}\})$ , it follows that  $\Pi_i \cap G_e(D, \{\vec{u}\}) \neq \emptyset$ , and so from the first part of our sufficient condition,  $\Pi_i \subseteq G_e(D, \{\vec{u}\})$ . So we have,  $\vec{d}_2 \in \Pi_i \subseteq G_e(D, \{\vec{u}\})$ , i.e.,  $e$  is enabled at  $(\vec{d}_2, \vec{u})$ . We claim that if we let  $\vec{d}'_2 := f_e(\vec{d}_2, \vec{u})$ , then

$$\begin{aligned} & [(l, \vec{d}_2) \xrightarrow{\sigma_e, \vec{u}, \vec{y}} (l', \vec{d}'_2), l \xrightarrow{e} l'] \text{ is such that} \\ & ((l', \vec{d}_1), (l', \vec{d}'_2)) \in \Phi. \end{aligned}$$

Since  $h_e(\vec{d}_1, \vec{u}) = \vec{y}$ , from the third part of our sufficient condition  $h_e(\Pi_i, \vec{u}) = \{\vec{y}\}$ . Since  $\vec{d}_2 \in \Pi_i$ , this further implies that  $h_e(\vec{d}_2, \vec{u}) = \vec{y}$ . Thus indeed it holds that,  $[(l, \vec{d}_2) \xrightarrow{\sigma_e, \vec{u}, \vec{y}} (l', \vec{d}'_2)]$ . It remains to be shown that  $((l', \vec{d}_1), (l', \vec{d}'_2)) \in \Phi$ . By the second part of our sufficiency condition, either  $\vec{d}'_1 = f_e(\vec{d}_1, \vec{u}), \vec{d}'_2 = f_e(\vec{d}_2, \vec{u}) \in \Pi_j$  for some  $j \leq m$  or  $\vec{d}'_1 = f_e(\vec{d}_1, \vec{u}) = \vec{d}'_2 = f_e(\vec{d}_2, \vec{u}) \in \Pi_j$  for some  $j \leq n$ . It follows that in both cases,  $((l', \vec{d}_1), (l', \vec{d}'_2)) \in \Phi$ . ■

The following example illustrates Theorem 1.

*Example 1:* Consider the I/O-EFA  $P$  shown in Figure 1.  $P$  has an infinite reachability set, a portion of which is shown in Figure 2. Figure 2 also shows a partition of the data space  $D$  satisfying the condition of Theorem 1, where the members of the partition are as follows:

$$\begin{aligned} \Pi_1 &= [2d(1) + d(2) > 2] \wedge [d(1) > 1], \\ \Pi_2 &= [2d(1) + d(2) > 2] \wedge [d(1) = 1], \\ \Pi_3 &= [2d(1) + d(2) > 2] \wedge [d(1) = 0], \\ \Pi_4 &= [2d(1) + d(2) > 2] \wedge [d(1) < 0] \\ \Pi_5 &= [2d(1) + d(2) = 2] \wedge [d(1) < 0], \\ \Pi_6 &= [2d(1) + d(2) < 2] \wedge [d(1) < 0], \\ \Pi_7 &= [2d(1) + d(2) < 2] \wedge [d(1) = 0], \\ \Pi_8 &= [2d(1) + d(2) < 2] \wedge [d(1) = 1], \\ \Pi_9 &= [2d(1) + d(2) < 2] \wedge [d(1) > 1], \end{aligned}$$

$$\begin{aligned} \Pi_{10} &= [2d(1) + d(2) = 2] \wedge [d(1) > 1], \\ \Pi_{11} &= [2d(1) + d(2) = 2] \wedge [d(1) = 1], \\ \Pi_{12} &= [2d(1) + d(2) = 2] \wedge [d(1) = 0]. \end{aligned}$$

Let  $e_1, e_2$  and  $e_3$  denote the edges from  $A$  to  $B$ ,  $B$  to  $C$  and  $C$  to  $A$ , respectively. It can be seen from Table I that the sufficient condition of Theorem 1 holds (the entry “T” in column for  $h_{e_i}$  indicates that the third part of the sufficient condition holds True). It follows that  $P$  admits a finite late-bisimilar quotient; it is shown in Figure 1, with the unreachable states omitted.

## V. I/O-EFA SUBCLASS WITH FINITE BISIMILAR QUOTIENT

The condition of Theorem 1 for the existence of a finite late-bisimilar quotient is *existential* as it relies on the existence of a certain partition. In this section, we identify a subclass of I/O-EFAs for which a partition satisfying the sufficient condition of Theorem 1 can be constructed by inspecting the structure of a given I/O-EFA of the subclass. The subclass is obtained by imposing two conditions on the class of the I/O-EFAs. The first condition restricts the manner in which the transition guards are formed, and is specified by the following grammar.

*Condition 1:*  $G(D, U) \longrightarrow G(U) \mid d(i) \leq c \mid d(i) \geq c \mid \neg G(D, U) \mid G_1(D, U) \wedge G_2(D, U)$ ,

where  $c$  is an integer constant (it can also be a rational constant but there is no loss of generality in restricting it to the domain of integers). According to the above grammar, an “atomic” guard is either a predicate over only the inputs, or it is a predicate over only the data and in which case it can be written as a single inequality constraint over one of the data components. A generic guard is a boolean combination of the atomic guards.

When guards are formed using the above grammar, there exists a natural partition of the data space into a finite number of regions. In order to define such a partition, we let  $d(i)^{\max}$  and  $d(i)^{\min}$  denote the largest and smallest integer against which the  $i$ th data variable is compared over all the guards. Then the domain of  $i$ th data component is naturally partitioned into up to three regions: One where  $d(i)$  is below  $d(i)^{\min}$ , another where it is in between  $d(i)^{\min}$  and  $d(i)^{\max}$ , and the last one where  $d(i)$  exceeds  $d(i)^{\max}$ . This ( $\leq 3$ )-way partition of the domain of the  $i$ th data component naturally yields a ( $\leq 3^p$ )-way partition of the entire data space (recall  $p$  is the dimension of data space), only at most one of which is bounded.

Since each element within a bounded region is to be its own late-bisimulation equivalence class, for each  $\vec{d} \in D$  we define its equivalence class,  $[\vec{d}] \subseteq D$  as in the following definition. We first define index sets  $I^{\max}(\vec{d})$  and  $I^{\min}(\vec{d})$  containing indices  $i \leq p$  for which  $d(i)$  is above  $d(i)^{\max}$  and below  $d(i)^{\min}$  respectively. We let  $I := \{1, \dots, p\}$  denote the set of data components.

*Definition 5:* Given an I/O-EFA satisfying Condition 1, for  $\vec{d} \in D$  define  $I^{\max}(\vec{d}), I^{\min}(\vec{d}) \subseteq I = \{1, \dots, p\}$  as:

$$[i \in I^{\max}(\vec{d})] \Leftrightarrow [d(i) > d(i)^{\max}],$$

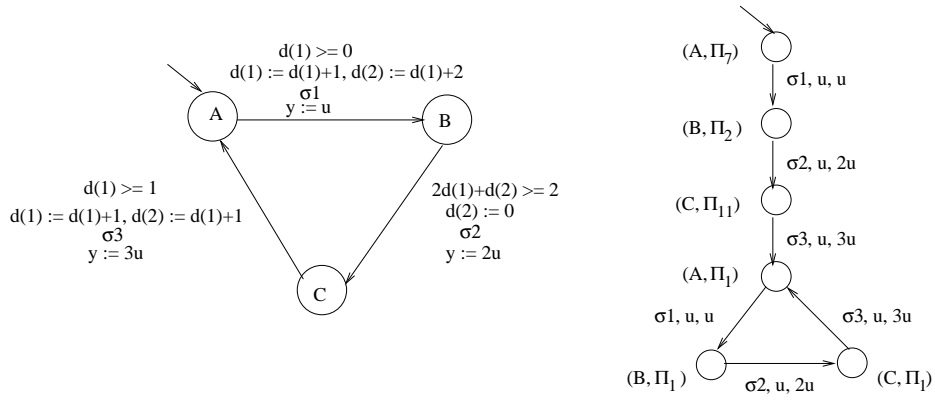


Fig. 1. I/O-EFA  $P$  (left); Finite bisimilar quotient of  $P$  (right)

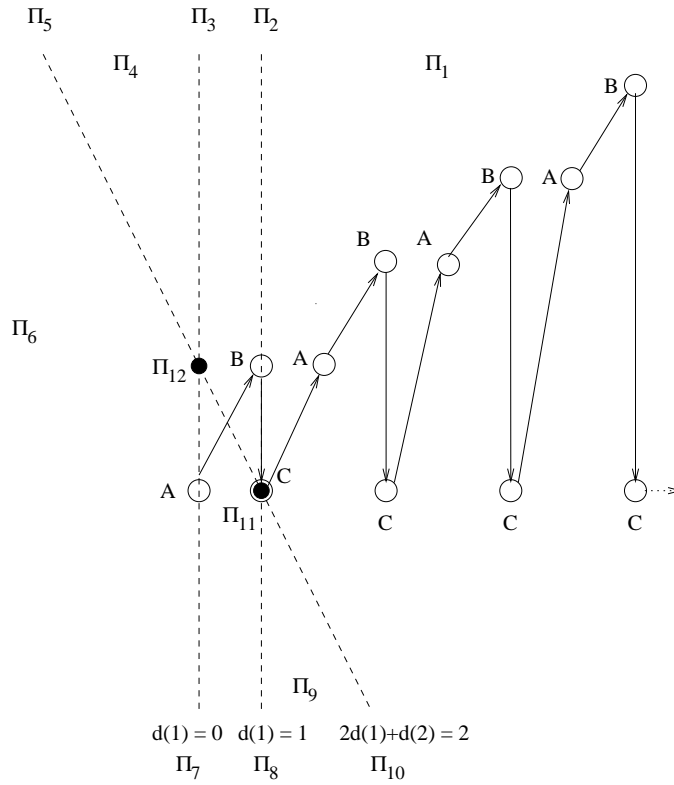


Fig. 2. A partition of data space of  $P$  of Figure 1

$\Pi_i$	Bounded	$\Pi_i \cap G_{e_1}$	$f_{e_1}(\Pi_i, \cdot)$	$h_{e_1}$	$\Pi_i \cap G_{e_2}$	$f_{e_2}(\Pi_i, \cdot)$	$h_{e_2}$	$\Pi_i \cap G_{e_3}$	$f_{e_3}(\Pi_i, \cdot)$	$h_{e_3}$
$\Pi_1$	No	$\Pi_1$	$\subseteq \Pi_1$	$T$	$\Pi_1$	$\subseteq \Pi_1$	$T$	$\Pi_1$	$\subseteq \Pi_1$	$T$
$\Pi_2$	No	$\Pi_2$	$\subseteq \Pi_1$	$T$	$\Pi_2$	$\subseteq \Pi_{11}$	$T$	$\Pi_2$	$\subseteq \Pi_1$	$T$
$\Pi_3$	No	$\Pi_3$	$\subseteq \Pi_2$	$T$	$\Pi_3$	$\subseteq \Pi_7$	$T$	$\emptyset$	N/A	N/A
$\Pi_4$	No	$\emptyset$	N/A	N/A	$\Pi_4$	$\subseteq \Pi_6$	$T$	$\emptyset$	N/A	N/A
$\Pi_5$	No	$\emptyset$	N/A	N/A	$\Pi_5$	$\subseteq \Pi_7$	$T$	$\emptyset$	N/A	N/A
$\Pi_6$	No	$\emptyset$	N/A	N/A	$\emptyset$	N/A	N/A	$\emptyset$	N/A	N/A
$\Pi_7$	No	$\Pi_7$	$\subseteq \Pi_2$	$T$	$\emptyset$	N/A	N/A	$\emptyset$	N/A	N/A
$\Pi_8$	No	$\Pi_8$	$\subseteq \Pi_1$	$T$	$\emptyset$	N/A	N/A	$\Pi_8$	$\subseteq \Pi_1$	$T$
$\Pi_9$	No	$\Pi_9$	$\subseteq \Pi_1$	$T$	$\emptyset$	N/A	N/A	$\Pi_9$	$\subseteq \Pi_1$	$T$
$\Pi_{10}$	No	$\Pi_{10}$	$\subseteq \Pi_1$	$T$	$\Pi_{10}$	$\subseteq \Pi_1$	$T$	$\Pi_{10}$	$\subseteq \Pi_1$	$T$
$\Pi_{11}$	Yes	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$\Pi_{12}$	Yes	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE I

$$[i \in I^{\min}(\vec{d})] \Leftrightarrow [d(i) < d(i)^{\min}].$$

Define an equivalence class of  $\vec{d} \in D$ , denoted  $[\vec{d}] \subseteq D$ , as:

$$[\vec{d}] := \left\{ \vec{d}' \in D \mid \begin{aligned} & [I^{\max}(\vec{d}') = I^{\max}(\vec{d}) := I^{\max}] \\ & \wedge [I^{\min}(\vec{d}') = I^{\min}(\vec{d}) := I^{\min}] \\ & \wedge [d'(i) = d(i), \forall i \in I - I^{\max} - I^{\min}] \end{aligned} \right\}.$$

Note that for any  $\vec{d} \in D$ ,  $I^{\max}(\vec{d}) \cap I^{\min}(\vec{d}) = \emptyset$ , or equivalently,  $I^{\max}(\vec{d}) \subseteq I - I^{\min}(\vec{d})$ , or equivalently,  $I^{\min}(\vec{d}) \subseteq I - I^{\max}(\vec{d})$ .

The following lemma states that the equivalence class of  $\vec{d}$  is well defined. Due to space consideration, its proof is omitted.

*Lemma 1:* Consider the definition of equivalence class  $[\vec{d}]$  for  $\vec{d} \in D$  given in Definition 5. Then for any  $\vec{d}, \vec{d}' \in D$ ,

$$[\vec{d}' \in [\vec{d}]] \Rightarrow [[\vec{d}'] \subseteq [\vec{d}]] (\Leftrightarrow [\vec{d}'' \in [\vec{d}]] \Rightarrow \vec{d}'' \in [\vec{d}']).$$

*Remark 3:* It can be verified that the total number of equivalence classes, i.e.,  $|\{[\vec{d}] \mid \vec{d} \in D\}|$  is given by,

$$\sum_{I_1 \subseteq I, I_2 \subseteq I - I_1} \prod_{i \in I - I_1 - I_2} (d(i)^{\max} - d(i)^{\min} + 1).$$

Also note that an equivalence class  $[\vec{d}]$  is unbounded if for some  $i$  either  $d(i) > d(i)^{\max}$  or  $d(i) < d(i)^{\min}$ , or equivalently if  $I^{\max}(\vec{d}) \cup I^{\min}(\vec{d}) \neq \emptyset$ . So there are a total of  $\prod_{i \in I} (d(i)^{\max} - d(i)^{\min} + 1)$  number of bounded equivalence classes, each one of which is a singleton. The remaining equivalence classes are all unbounded. We denote the set of all equivalence classes by  $\mathcal{D}$ , and the subset of unbounded and bounded ones by  $\mathcal{D}_u$  and  $\mathcal{D}_b$  respectively.

The second condition given below imposes restriction on the data update functions and output assignment functions depending on the region of the partition where they are defined. It requires that if the  $i$ th data component is above  $d(i)^{\max}$  (resp., below  $d(i)^{\min}$ ), then an update of the  $i$ th data component be either an increment (resp., a decrement) or be independent of the data component  $j$  that does not lie in the range  $d(j)^{\min}$  and  $d(j)^{\max}$ . Further, the output assignment is required to be independent of the data component  $j$  that does not lie in the range  $d(j)^{\min}$  and  $d(j)^{\max}$ . This is formally stated as follows.

*Condition 2:*  $\forall \vec{d}, \forall e, \forall \vec{u}$ :

$$\begin{aligned} & [(\vec{d}, \vec{u}) \in G_e(D, U)] \Rightarrow \\ & 1. \forall i \in I^{\max}(\vec{d}) : \left( f_e(i)(\vec{d}, \vec{u}) \geq d(i) \right) \vee \\ & \quad \left( \forall \vec{d}' \in [\vec{d}] : f_e(i)([\vec{d}], \vec{u}) = \{f_e(i)(\vec{d}', \vec{u})\} \right) \\ & 2. \forall i \in I^{\min}(\vec{d}) : \left( f_e(i)(\vec{d}, \vec{u}) \leq d(i) \right) \vee \\ & \quad \left( \forall \vec{d}' \in [\vec{d}] : f_e(i)([\vec{d}], \vec{u}) = \{f_e(i)(\vec{d}', \vec{u})\} \right) \\ & 3.  $h_e(\vec{d}, \vec{u})$  is constant function of  $d(i), \forall i \in I^{\max}(\vec{d}) \cup I^{\min}(\vec{d})$ . \end{aligned}$$

The condition “ $\forall i \in I^{\max}(\vec{d}) : [f_e(i)(\vec{d}, \vec{u}) \geq d(i)]$ ” can be read as: In regions where  $d(i)$  is above  $d(i)^{\max}$ , it can only be incremented. The condition “ $\forall i \in I^{\min}(\vec{d}) : [f_e(i)(\vec{d}, \vec{u}) \leq$

$d(i)]$ ” is dual and can be understood similarly. The condition “ $[\forall \vec{d}' \in [\vec{d}] : f_e(i)([\vec{d}], \vec{u}) = \{f_e(i)(\vec{d}', \vec{u})\}]$ ” states that the update of  $d(i)$  is uniform over all elements in the equivalence class of  $\vec{d}$ , which is the same thing as saying that the update of  $d(i)$  is constant with respect to all  $d(j), j \in I^{\max}(\vec{d}) \cup I^{\min}(\vec{d})$ .

The next theorem shows that under Conditions 1 & 2, the partition induced by the equivalence over data defined by Definition 5 satisfies the sufficient condition of Theorem 1.

*Theorem 2:* Consider an I/O-EFA  $P$  satisfying Conditions 1 and 2, and the partition  $\{[\vec{d}] \mid \vec{d} \in D\}$  of the data space. Then it holds that,

$$\begin{aligned} \forall [\vec{d}] \in \mathcal{D}_u, \forall e, \forall \vec{u} : & \quad [ \vec{d} \in G_e(D, U) \Rightarrow \\ & 1. [\vec{d}] \subseteq G_e(D, U), \text{ and} \\ & 2. \left( \exists [\vec{d}'] \in \mathcal{D}_u : f_e([\vec{d}], \vec{u}) \subseteq [\vec{d}'] \right) \vee \\ & \quad \left( f_e([\vec{d}], \vec{u}) = \{f_e(\vec{d}, \vec{u})\} \right), \text{ and} \\ & 3.  $h_e([\vec{d}], \vec{u}) = \{h_e(\vec{d}, \vec{u})\}$ . \end{aligned}$$

**Proof:** By Condition 1, every guard  $G_e(D, U)$  is a boolean combination of atomic guards of the type,  $G_e(U), [d(i) \leq c], [d(i) \geq c]$ . Now suppose the boolean combination of atomic guards is written in disjunctive normal form (DNF), then for the guard to be satisfied, one of the disjuncts in the DNF must be satisfied. Since each disjunct in a DNF is a conjunct of atomic guards (or their negations), for a disjunct in a DNF to be satisfied, each atomic guard (or its negation) in the disjunct must be satisfied. So in order to show that whenever  $\vec{d}$  satisfies a guard  $G_e(D, U)$ , it implies that  $[\vec{d}] \subseteq G_e(D, U)$ , it suffices to show that whenever  $\vec{d}$  satisfies an atomic guard, all elements of the equivalence class  $[\vec{d}]$  satisfies an atomic guard. Clearly this holds when the atomic guard is predicate over only the inputs. So now consider an atomic guard of the type,  $[d(i) \leq c]$ . Then by definitions of  $d(i)^{\min}$  and  $d(i)^{\max}$ , we have  $d(i)^{\min} \leq c \leq d(i)^{\max}$ . So  $\vec{d}$  satisfies  $[d(i) \leq c]$  if and only if either  $[d(i) < d(i)^{\min} \leq c]$  or  $[d(i)^{\min} \leq d(i) \leq c \leq d(i)^{\max}]$ . In the first case,  $i \in I^{\min}(\vec{d})$ , whereas in the second case  $i \in I - I^{\max}(\vec{d}) - I^{\min}(\vec{d})$ . If former, then it follows from definition of equivalence class that  $i \in I^{\min}(\vec{d}')$  for all  $\vec{d}' \in [\vec{d}]$ ; whereas if latter, then again it follows from the definition of equivalence class that  $i \in I - I^{\max}(\vec{d}') - I^{\min}(\vec{d}')$  for all  $\vec{d}' \in [\vec{d}]$ . Thus an atomic guard of the type  $[d(i) \leq c]$  is satisfied by  $\vec{d}$  if and only if that guard is satisfied by each element in  $[\vec{d}]$ . From an analogous argument, the same property is enjoyed by an atomic guard of the form  $[d(i) \geq c]$ . This establishes the first condition appearing in the theorem.

Now to prove the second condition appearing in the theorem, we need to show that either first disjunct of the condition holds or the second disjunct of the condition holds. The first two parts of Condition 2 can be combined and rewritten as:

$$[(\forall i \in I^{\max}(\vec{d}) : f_e(i)(\vec{d}, \vec{u}) \geq d(i)) \wedge$$

$$\begin{aligned} & (\forall i \in I^{\min}(\vec{d}) : [f_e(i)(\vec{d}, \vec{u}) \leq d(i)]) \\ \vee & \left[ \forall i \in I^{\max}(\vec{d}) \cup I^{\min}(\vec{d}), \forall \vec{d}' \in [\vec{d}] : \right. \\ & \left. f_e(i)([\vec{d}], \vec{u}) = \{f_e(i)(\vec{d}', \vec{u})\} \right]. \end{aligned}$$

If in the above the first disjunct holds, then every data in  $[\vec{d}]$  is updated to an equivalence class  $[\vec{d}']$  such that  $I^{\max}(\vec{d}') \supseteq I^{\max}(\vec{d})$  and  $I^{\min}(\vec{d}') \supseteq I^{\min}(\vec{d})$ . It follows that if  $[\vec{d}] \in \mathcal{D}_u$ , then  $[\vec{d}'] \in \mathcal{D}_u$ . So the first disjunct of the second condition appearing in the theorem holds.

On the other hand, if in the above the second disjunct holds, then an update of the  $i$ th data component is identical for all data in  $[\vec{d}']$  for all  $i \in I^{\max}(\vec{d}') \cup I^{\min}(\vec{d}')$ . Since for every data in  $[\vec{d}']$  it holds that the  $i$ th component is identical for all  $i \in I - I^{\max}(\vec{d}') - I^{\min}(\vec{d}')$ , it can be concluded that the second disjunct of the second condition appearing in the theorem holds.

A similar argument as the one in the previous paragraph can be constructed to show that the third part of Condition 2 implies the third condition appearing in the theorem. This completes the proof. ■

The following corollary follows from Theorems 1 and 2.

*Corollary 1:* Given an I/O-EFA  $P$  satisfying Conditions 1 and 2,  $P$  admits a finite late-bisimilar quotient.

*Remark 4:* It follows from the development above that each state in a finite late-bisimilar quotient of  $P$  satisfying Conditions 1 and 2 is of the form  $\{l\} \times [\vec{d}]$ , where  $[\vec{d}] \subseteq D$  is defined in Definition 5. Thus the number of states in a finite late-bisimilar quotient of  $P$  is given by,

$$|L| \times \left( \sum_{I_1 \subseteq I, I_2 \subseteq I - I_1} \prod_{i \in I - I_1 - I_2} (d(i)^{\max} - d(i)^{\min} + 1) \right).$$

*Remark 5:* If there are no inputs and outputs (i.e., the case of closed systems), then Conditions 1 and 2 can be simplified as follows.

$$\text{C1: } G(D) \longrightarrow d(i) \leq c \mid d(i) \geq c \mid \neg G(D) \mid G_1(D) \wedge G_2(D), \text{ and}$$

$$\text{C2: } \forall \vec{d}, \forall e: [\vec{d} \in G_e(D) \Rightarrow$$

$$\begin{aligned} & 1. \forall i \in I^{\max}(\vec{d}) : (f_e(i)(\vec{d}) \geq d(i)) \vee \\ & (\forall \vec{d}' \in [\vec{d}] : f_e(i)([\vec{d}]) = \{f_e(i)(\vec{d}')\}) \\ & 2. \forall i \in I^{\min}(\vec{d}) : (f_e(i)(\vec{d}) \leq d(i)) \vee \\ & (\forall \vec{d}' \in [\vec{d}] : f_e(i)([\vec{d}]) = \{f_e(i)(\vec{d}')\}) \end{aligned}$$

Then from Corollary 1 an EFA  $P = (L, D, \Sigma, E, L_0, D_0)$  admits a finite late-bisimilar quotient if the two conditions stated above in this remark hold.

*Example 2:* Consider an EFA  $P$  shown in Figure 3. One can see by inspection that all guards satisfy Condition 1 of Remark 5, and  $d(1)^{\min} = d(2)^{\min} = 0$  and  $d(1)^{\max} = d(2)^{\max} = 2$ . Let  $e_1, e_2$  and  $e_3$  denote edges from  $A$  to  $B$ ,  $B$  to  $C$  and  $C$  to  $A$ , respectively. We next verify whether Condition 2 of Remark 5 also holds, as follows.

$$1) \{ \vec{d} \mid I^{\max}(\vec{d}) = \{1\} \} = \{ \vec{d} \mid d(1) > 2 \} =: R_1.$$

$$e = e_1: G_e(D) \cap R_1 = \{d(1) > 2, d(2) \geq 0\} \neq \emptyset;$$

$$f_e(1) = d(1) + 1 \geq d(1).$$

$$e = e_2: G_e(D) \cap R_1 = \{d(1) > 2, d(2) > 2\} \neq \emptyset;$$

$$f_e(1) = d(1) \geq d(1).$$

$$e = e_3: G_e(D) \cap R_1 = \{d(1) > 2, d(2) \geq 1\} \neq \emptyset;$$

$$f_e(1) = d(1) \geq d(1).$$

$$2) \{ \vec{d} \mid I^{\max}(\vec{d}) = \{2\} \} = \{ \vec{d} \mid d(2) > 2 \} =: R_2.$$

$$e = e_1: G_e(D) \cap R_2 = \{d(1) \geq 0, d(2) > 2\} \neq \emptyset;$$

$$f_e(2) = d(2) \geq d(2).$$

$$e = e_2: G_e(D) \cap R_2 = \{d(1) \geq 1, d(2) > 2\} \neq \emptyset;$$

$$f_e(2) = 1.$$

$$e = e_3: G_e(D) \cap R_2 = \{d(1) \geq 2, d(2) > 2\} \neq \emptyset;$$

$$f_e(2) = d(1) + d(2) \geq 2 + d(2) \geq d(2).$$

$$3) \{ \vec{d} \mid I^{\min}(\vec{d}) = \{1\} \} = \{ \vec{d} \mid d(1) < 0 \} =: R_3.$$

$$e = e_1: G_e(D) \cap R_3 = \emptyset.$$

$$e = e_2: G_e(D) \cap R_3 = \emptyset.$$

$$e = e_3: G_e(D) \cap R_3 = \emptyset.$$

$$4) \{ \vec{d} \mid I^{\min}(\vec{d}) = \{2\} \} = \{ \vec{d} \mid d(2) < 0 \} =: R_4.$$

$$e = e_1: G_e(D) \cap R_4 = \emptyset.$$

$$e = e_2: G_e(D) \cap R_4 = \emptyset.$$

$$e = e_3: G_e(D) \cap R_4 = \emptyset.$$

We have verified that desired conditions hold and so  $P$  admits a finite late-bisimilar quotient even though the reachability set of  $P$  is infinite. The equivalence classes representing the states of a finite late-bisimilar quotient and some of the reachable states are shown in Figure 4. The finite late-bisimilar quotient system is shown in Figure 3, with the unreachable states omitted.

## VI. CONCLUSION

We considered a model of reactive untimed infinite state systems, called I/O-EFA, which is an automaton extended with discrete variables such as inputs, outputs, and data, and presented a sufficient condition under which such a model admits a finite late-bisimilar quotient (and hence also a finite bisimilar quotient as late-bisimilarity implies bisimilarity). This sufficient condition is *existential* as it relies on the existence of a suitable partition of the state space. We then identify a class of I/O-EFAs for which a partition satisfying our sufficient condition can be *constructed* by inspecting the structure of the given I/O-EFA.

If a given I/O-EFA fails to satisfy the Condition 1 or 2, then starting from the initial partition  $\{\{ \vec{d} \} \mid \vec{d} \in D\}$  one can apply a partition refinement algorithm, where the refinement will be done with the goal of satisfying the conditions of Theorem 1. Also relaxed versions of Theorems 1 and Corollary 1 may be developed to seek a finite bisimilar quotient of a given I/O-EFA.

## REFERENCES

- [1] P. A. Abdulla, K. Cerans, B. Jonsson, and Y. Tsay. General decidability theorems for infinite-state systems. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 313–321, Washington, DC, USA, 1996. IEEE Computer Society.
- [2] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

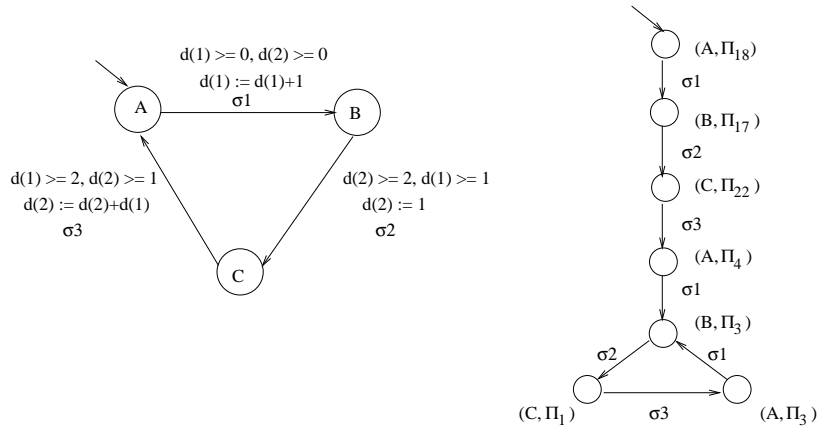


Fig. 3. EFA  $P$  (left); Finite late-bisimilar quotient system (right)

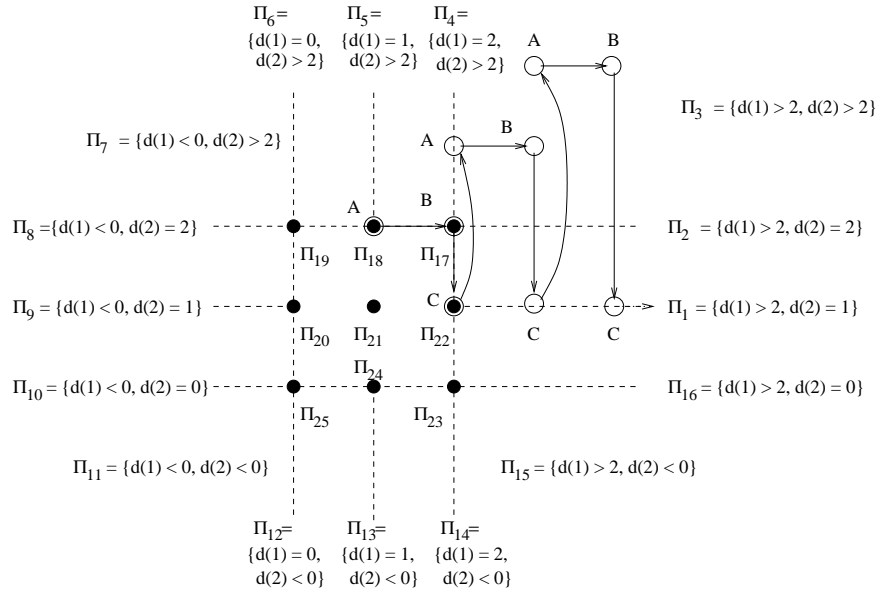


Fig. 4. Equivalent data sets of  $P$  of Figure 3

- [3] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971 – 984, July 2000.
- [4] P. Bouyer, C. Dufour, E. Fleury, and A. Petit. Updatable timed automata. *Theor. Comput. Sci.*, 321(2-3):291–345, 2004.
- [5] K. Cheng and A. Krishnakumar. Automatic functional test generation using the extended finite state machine model. In *DAC '93: Proceedings of the 30th international conference on Design automation*, pages 86–91, New York, NY, USA, 1993. ACM Press.
- [6] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.
- [7] Matthew Hennessy, Huimin Lin, and Julian Rathke. Unique fixpoint induction for message-passing process calculi. *Science of Computer Programming*, 41:241–275, 2001.
- [8] T. Henzinger. The theory of hybrid automata. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, Washington, DC, USA, 1996. IEEE Computer Society.
- [9] T. Henzinger, R. Majumdar, and J. Raskin. A classification of symbolic transition systems. *ACM Transactions on Computational Logic*, 6(1):1–31, 2005.
- [10] H. Lin. Symbolic transition graph with assignment. In *CONCUR*, pages 50–65, 1996.
- [11] Julian Rathke and Matthew Hennessy. Local model checking for value-passing processes. In *Proc. TACS'97, International Symposium on Theoretical Aspects of Computer Software*, Sendai. Springer-Verlag, 1997.
- [12] P. Tabuada and G. J. Pappas. Finite bisimulations of controllable linear systems. In *Proceedings 42nd IEEE Conference on Decision and Control*, volume 1, pages 634–639, Dec. 2003.
- [13] A.J. van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE Transactions on Automatic Control*, 49(12):2160–2172, Dec. 2004.