2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

On Identification of Input/Output Extended Automata with Finite Bisimilar Quotients

Changyan Zhou, *Member, IEEE,* and Ratnesh Kumar[†], *Fellow, IEEE*
[†]Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
Email: rkumar@iastate.edu

***Abstract*— We study the problem of finding a finite bisimilar abstraction for a class of reactive untimed infinite-state systems, modeled as input-output extended finite automata (I/O-EFA). We identify a lower bound abstraction (that is coarser than any finite bisimilar abstraction), and present an iterative refinement algorithm whose termination guarantees the existence of a finite bisimilar abstraction. The termination condition is weaker than the one given in [15] for the existence of a finite bisimilar quotient, and thus, the paper identifies a larger class of I/O-EFAs possessing a finite bisimilar abstraction (than that given in [15]).**

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

I. Introduction

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 [17], [14], [1]. Another approach is to directly analyze an infinite state system by symbolically encoding the states and the transitions as formulas of a suitable logic, see for example [7], [3]. [9], [5], [17], [4] employed predicate abstraction technique for extracting finite state models from infinite state systems. Given a concrete infinite state system and a set of abstraction predicates, a conservative finite state abstraction is generated. 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 [1], [6], for linear and nonlinear systems [20], [19], and for hybrid systems [2].

Some discrete event systems such as software are typically infinite state systems and symbolic models, such as symbolic transition graph (STG) [10] and its extension STGA (STG with assignment) [16], and extended finite state machines (EFSMs) [8], are proposed so that untimed infinite state systems can be represented as a finite graph. 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. An I/O-EFA possibly has infinitely many states and its set of reachable states can also be infinite. In order to be able to apply existing finite state system verification methods to a system modeled as an I/O-EFA, it is desirable that its underlying transition system possesses a finite bisimilar abstraction (also called quotient).

Much attention has been given to the identification of classes of decidable hybrid automata that admit finite bisimilar quotient [1], [6], [2], [13]. Although I/O-EFA is a special type of a hybrid automaton [12] with no continuous dynamics (i.e., flow rate of each variable is zero), it deserves a separate attention since (i) it represents a large class of untimed infinite state systems such as embedded software and (ii), owing to the non-existence of continuous dynamics in I/O-EFAs, restrictions on guards, data update and assignment functions needed for the existence of a finite bisimilar quotient can be more relaxed than those needed for the decidability of a hybrid automaton (see our work [15] for example).

For an I/O-EFA to admit a finite bisimilar quotient, its data space should possess a finite partition over which the data update and assignment functions are bisimilar. For I/O-EFAs, besides the usual notion of bisimilarity, one can define a stronger notion, namely that of "late"-bisimilarity [10], [16]. (In [10], [16], the term "early-bisimilarity" is used for what one would define to be bisimilarity; we avoid using "early-bisimilarity" as that can cause confusion.) 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 setting of late-bisimulation the input is read only after choosing a transition for bisimulating a transition of another system.

In general, the problem of finding a finite bisimilar quotient is undecidable. [13] studied a class of systems called, symbolic transition systems (STSs), and reported 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. Note in [13] there is no notion of initial states, and any state is treated an initial state and so all states are reachable. We do have a notion of initial states, and so not all states may be reachable. Then the unreachable states do not have to satisfy any condition for the existence of a finite bisimilar quotient. This situation does not arise in [13] since there the set of unreachable states is empty. In our earlier

The research was supported in part by the National Science Foundation under the grants NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, NSF-0424048, and NSF-ECS-0601570.

C. Zhou is with Magnatech LLC, East Granby, CT 06026 (Email: changyan.zhou@gmail.com).

work [15], we presented a sufficient condition under which an I/O-EFA admits a finite bisimilar quotient ([15, Theorem 1]) and identified a class of I/O-EFAs for which a finite bisimilar partition satisfying our sufficient condition can be *constructed* by inspecting the structure of the given I/O-EFA ([15, Theorem 2]). In this paper, we present an algorithm that identifies a larger class of I/O-EFAs that satisfies the sufficient condition.

There exist other works related to the topic of the paper. The problem of bisimulation-checking between infinite-state systems modeled as STGAs has been studied in [10], [16]. [16] 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. [18], [11] proposed proof systems for model-checking value passing processes, which again are not decidable in general.

II. Notation and Preliminaries

In this section, we introduce the I/O-EFA model, and the notions of bisimilarity and quotient systems. We present a sufficient condition under which an I/O-EFA admits a finite bisimilar quotient that is equivalent to the result of [15, Theorem 1]. (The equivalent form is found easier to use for a later analysis.)

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 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.

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, $D = D_1 \times \ldots \times D_p$ is the set of p-dimensional data, $U = U_1 \times \ldots \times U_q$ is the set of q-dimensional input, $Y = Y_1 \times \ldots \times Y_r$ 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, $f_e : D \times U \to D$ is the data update function, $h_e : D \times U \to Y$ is the output assignment function. Finally, L_0 is the set of initial location, and $D_0 = D_{10} \times \ldots \times D_{p0}$ 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 ith component of $\vec{d}$, i.e., $\vec{d} = (d(1), ..., d(p))$, where p is the number of data variables. $\vec{d_1} = \vec{d_2}$ means component-wise equality, i.e., $(d_1(i) = d_2(i), \forall i \in \{1, \ldots, p\})$.

The semantics of an input-output automaton P can be understood as follows. P starts with an initial location $l \in L_0$ and an initial data value $\vec{d} \in D_0$. While at a certain state $(l, \vec{d}) \in L \times D$, if P receives an input $\vec{u}$, a transition $e \in E$ such that $o_e = l$ is enabled if the guard $G_e(\vec{d}, \vec{u})$ holds. An enabled transition can be executed. The execution of an enabled transition e at the state $(l, \vec{d})$ causes P to transit to the location t_e, the data value is updated to $f_e(\vec{d}, \vec{u})$, and the output $\vec{y} = h_e(\vec{d}, \vec{u})$ is produced.

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.

We assume, without loss of generality, that none of the transitions are labeled by ϵ. In order to define bisimilarity over states of an I/O-EFA P, we introduce the following notations.

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

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

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

$$\begin{aligned}
\forall e_1, \forall \vec{u}, \exists e_2 : \quad & \sigma_{e_2} = \sigma_{e_1} := \sigma, \text{ and} \\
& [(l_1, \vec{d_1}) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_1, \vec{d'_1}), l_1 \xrightarrow{e_1} l'_1] \Rightarrow \\
& \exists [(l_2, \vec{d_2}) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_2, \vec{d'_2}), l_2 \xrightarrow{e_2} l'_2] \text{ s.t.} \\
& ((l'_1, \vec{d'_1}), (l'_2, \vec{d'_2})) \in \Phi.
\end{aligned}$$

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

$$\begin{aligned}
\forall e_1, \exists e_2 : \quad & \sigma_{e_2} = \sigma_{e_1} := \sigma, \text{ and} \\
& \forall \vec{u}, \ [(l_1, \vec{d_1}) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_1, \vec{d'_1}), l_1 \xrightarrow{e_1} l'_1] \Rightarrow \\
& \exists [(l_2, \vec{d_2}) \xrightarrow{\sigma, \vec{u}, \vec{y}} (l'_2, \vec{d'_2}), l_2 \xrightarrow{e_2} l'_2] \text{ s.t.} \\
& ((l'_1, \vec{d'_1}), (l'_2, \vec{d'_2})) \in \Phi.
\end{aligned}$$

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 there exists a bisimulation (resp., late-bisimulation) relation Φ 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 there exists a bisimulation (resp., late-bisimulation) relation Φ such that for each $(l_{10}, \vec{d_{10}}) \in L_{10} \times D_{10}$ there exists $(l_{20}, \vec{d_{20}}) \in L_{20} \times D_{20}$ such that $((l_{10}, \vec{d_{10}}), (l_{20}, \vec{d_{20}})) \in \Phi$.

The notion of late-bisimulation is strictly stronger than bisimulation [10], [16]. It follows that if a system possesses a finite late-bisimilar quotient, it also possesses a finite bisimilar 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 Π of the state set $L \times D$, the *quotient system* of $\mathcal{P}$ with respect to the partition Π is the transition system $\mathcal{P}_\Pi = (\Pi, U, Y, \Sigma, \mathcal{E}_\Pi, \Pi_0)$, where

$$\mathcal{E}_\Pi = \{(\pi_1, \sigma, \vec{u}, \vec{y}, \pi_2) \mid \exists (l_i, \vec{d}_i) \in \pi_i \in \Pi, i = 1, 2, \text{ s.t. } ((l_1, \vec{d}_1), \sigma, \vec{u}, \vec{y}, (l_2, \vec{d}_2)) \in \mathcal{E}\}$$

and $\Pi_0 = \{\pi \in \Pi \mid \pi \cap (L_0 \times D_0) \neq \emptyset\}$. The quotient system of $\mathcal{P}$ with respect to the partition induced by an equivalence "eq" over $L \times D$ is denoted $\mathcal{P}_{eq}$.

Remark 1: Given an I/O-EFA $P = (L, D, U, Y, \Sigma, E, L_0, D_0)$, we can modify P to absorb the initial condition D_0 according to Figure 1. Then without loss of generality, we assume that D_0 is same as True.

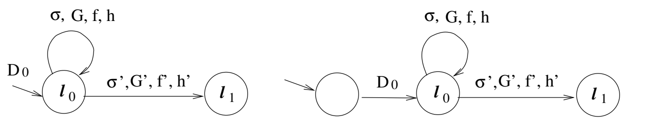

Fig. 1. I/O-EFA P (left) and P' that absorbs initial condition D_0 of P (right)

A partition satisfying the condition of the theorem given below guarantees that it induces a finite late-bisimilar quotient. For a predicate π over the data variables, the notation $\pi(f_e(\vec{d}, \vec{u}))$, where $f_e : D \times U \to D$ is the update function associated with a certain edge $e \in E$, denotes the predicate π with the variable $\vec{d}$ substituted with $f_e(\vec{d}, \vec{u})$. $\pi(f_e(\vec{d}, \vec{u}))$ is the predecessor of π along the edge e under the input $\vec{u}$.

Theorem 1: Given an I/O-EFA P, it admits a finite late-bisimilar quotient if there exists a partition Π of its data space such that

$$\begin{aligned} &\forall \pi \in \Pi, e \in E, \vec{u} \in U : \\ &\pi \Rightarrow [\neg G_e(\vec{d}, \vec{u}) \vee [G_e(\vec{d}, \vec{u}) \wedge \exists \pi' \in \Pi, \vec{y} \in Y : \\ &\quad \pi'(f_e(\vec{d}, \vec{u})) \wedge [h_e(\vec{d}, \vec{u}) = \vec{y}]]]. \end{aligned}$$

Theorem 1 states that exists a partition Π such that for each class $\pi \in \Pi$, each edge $e \in E$, and input $\vec{u} \in U$, either π is stronger than $\neg G_e(\vec{d}, \vec{u})$ (so e is enabled no where in $\{o_e\} \times \pi$), or π is stronger than $G_e(\vec{d}, \vec{u})$ (so e is enabled every where in $\{o_e\} \times \pi$) and at the same time exists an equivalence class $\pi' \in \Pi$ and output $\vec{y} \in Y$ such that along edge e and under input $\vec{u}$ successors of π are contained in π', while the same output $\vec{y}$ is produced everywhere in π. The above result can be shown to be equivalent to [15, Theorem 1]. (The equivalent form is found easier to use for a later analysis.)

The following example illustrates Theorem 1.

Example 1: Consider the I/O-EFA P shown in Figure 2 with edges e_1 (between locations A and B), e_2 (between locations B and C), and e_3 (between locations C and A). There exists a partition $\Pi = \cup_{i=1}^{6} \pi_i$ (shown in Figure 2), where

$$\begin{aligned} \pi_1 &= [d(1) \geq 1] \wedge [2d(1) + d(2) \geq 2], \\ \pi_2 &= [d(1) = 0] \wedge [2d(1) + d(2) \geq 2], \\ \pi_3 &= [d(1) < 0] \wedge [2d(1) + d(2) \geq 2], \\ \pi_4 &= [d(1) < 0] \wedge [2d(1) + d(2) < 2], \\ \pi_5 &= [d(1) = 0] \wedge [2d(1) + d(2) < 2], \\ \pi_6 &= [d(1) \geq 1] \wedge [2d(1) + d(2) < 2], \end{aligned}$$

such that the following holds for all u:

π_i	G_{e1}	$\exists \pi'_i, \vec{y} : \pi'_i(f_{e1}) \wedge [h_{e1} = \vec{y}]$
π_1	$\pi_1 \Rightarrow G_{e1}$	π_1, u
π_2	$\pi_2 \Rightarrow G_{e1}$	π_1, u
π_3	$\pi_3 \Rightarrow \neg G_{e1}$	N/A
π_4	$\pi_4 \Rightarrow \neg G_{e1}$	N/A
π_5	$\pi_5 \Rightarrow G_{e1}$	π_1, u
π_6	$\pi_6 \Rightarrow G_{e1}$	π_1, u
π_i	G_{e2}	$\exists \pi'_i, \vec{y} : \pi'_i(f_{e2}) \wedge [h_{e2} = \vec{y}]$
π_1	$\pi_1 \Rightarrow G_{e2}$	$\pi_1, 2u$
π_2	$\pi_2 \Rightarrow G_{e2}$	$\pi_5, 2u$
π_3	$\pi_3 \Rightarrow G_{e2}$	$\pi_4, 2u$
π_4	$\pi_4 \Rightarrow \neg G_{e2}$	N/A
π_5	$\pi_5 \Rightarrow \neg G_{e2}$	N/A
π_6	$\pi_6 \Rightarrow \neg G_{e2}$	N/A
π_i	G_{e3}	$\exists \pi'_i, \vec{y} : \pi'_i(f_{e3}) \wedge [h_{e3} = \vec{y}]$
π_1	$\pi_1 \Rightarrow G_{e3}$	$\pi_1, 3u$
π_2	$\pi_2 \Rightarrow \neg G_{e3}$	N/A
π_3	$\pi_3 \Rightarrow \neg G_{e3}$	N/A
π_4	$\pi_4 \Rightarrow \neg G_{e3}$	N/A
π_5	$\pi_5 \Rightarrow \neg G_{e3}$	N/A
π_6	$\pi_6 \Rightarrow G_{e3}$	$\pi_1, 3u$

Thus, P admits a finite late-bisimilar quotient.

The condition of Theorem 1 is *existential* as it relies on the existence of a certain partition. In [15], we identified a class 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 class (included as Theorem 3 in the Appendix). Two conditions were imposed on the class of I/O-EFAs. The first condition restricts the manner in which the transition guards are formed, whereas the second condition restricts the data update functions and output assignment functions.

III. Algorithm for Computing Late-Bisimilar Quotient

In this section, we present an algorithm that upon termination yields a finite late-bisimilar quotient. Starting from the coarsest partition candidate that may satisfy the condition of Theorem 1, the algorithm iteratively computes a finer partition guided by the condition of Theorem 1 until the partition converges, and in which case the given I/O-EFA admits a finite late-bisimilar quotient.

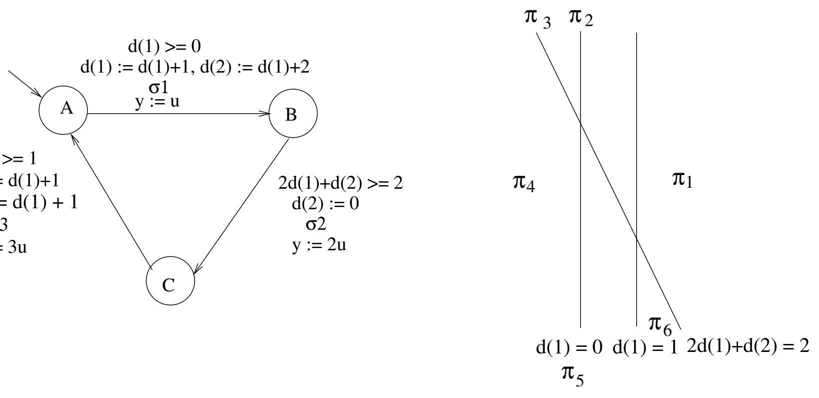

Fig. 2. An I/O-EFA P (left) and a partition of data space of P (right)

We first show that any partition Π that satisfies the condition of Theorem 1 must be finer than the one *induced* by the set of data predicates appearing as the guard and assignment conditions of an I/O-EFA, where a partition induced by a set of predicates is defined as follows.

Definition 5: Given a set of data predicates Θ defined over a data space D, the *partition of the data space induced* by Θ, denoted Π^{Θ}, is defined by

$$\Pi^{\Theta} := \{\bigwedge_{\theta\in\hat{\Theta}} \theta \bigwedge_{\theta'\in\Theta-\hat{\Theta}} \neg\theta' \mid \hat{\Theta}\subseteq\Theta\}.$$

It can be easily verified that Π^{Θ} is a partition, i.e., (i) $\pi,\pi'\in\Pi^{\Theta}$ and $\pi\neq\pi'$ imply that $\pi\wedge\pi' = False$, and ii) $\bigvee_{\hat{\Theta}\subseteq\Theta}[\bigwedge_{\theta\in\hat{\Theta}}\theta\bigwedge_{\theta'\in\Theta-\hat{\Theta}}\neg\theta'] = D$.

Proposition 1: Consider an I/O-EFA $P = (L, D, U, Y, \Sigma, E, L_0, D_0)$ with a set of data predicates of the guards and the assignments: $\Theta := \bigcup_{e\in E,\vec{u}\in U,\vec{y}\in Y}\{G_e(\vec{d},\vec{u}) \wedge [h_e(\vec{d},\vec{u}) = \vec{y}]\}$. If a partition Π satisfies the condition of Theorem 1, then Π is finer than Π^{Θ}.

Proof: Pick $\pi\in\Pi$. We need to show that exists $\overline{\pi}\in\Pi^{\Theta}$ such that $\pi\Rightarrow\overline{\pi}$. Since $\pi\in\Pi$ for each $e\in E,\vec{u}\in U$, either $\pi\Rightarrow\neg G_e(\vec{d},\vec{u})$ or exist $\pi'\in\Pi,\vec{y}\in Y$ such that $\pi\Rightarrow G_e(\vec{d},\vec{u})\wedge\pi'(f_e(\vec{d},\vec{u}))\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]$. If former, we can choose $\overline{\pi} = \neg G_e(\vec{d},\vec{u})$ and if latter, we can choose $\overline{\pi} = G_e(\vec{d},\vec{u})\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]$. Since $G_e(\vec{d},\vec{u})\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]\in\Theta$, and $\neg G_e(\vec{d},\vec{u}) = \neg[\vee_{\vec{y}\in Y}[G_e(\vec{d},\vec{u})\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]]]$, it follows that in either case $\overline{\pi}\in\Pi^{\Theta}$, as desired. ■

The above proposition suggests that it may be possible to refine the partition induced by the set of predicates appearing as guard and assignment conditions to obtain a partition satisfying Theorem 1. The algorithm we present in the following performs such a refinement. A possible way to achieve refinement is to introduce additional data predicates as suggested by the following lemma.

Lemma 1: Given two sets of data predicates Θ_1 and Θ_2 defined over a data space D such that $\Theta_1\subseteq\Theta_2$, then $\forall\pi_1\in\Pi^{\Theta_1}$, $\exists\pi_2\in\Pi^{\Theta_2}$ such that $\pi_2\Rightarrow\pi_1$, where Π^{Θ_1} and Π^{Θ_2} are computed by Definition 5.

Proof: Pick $\pi_1\in\Pi^{\Theta_1}$. Then exists $\hat{\Theta}_1\subseteq\Theta_1$ such that $\pi_1 = \bigwedge_{\theta\in\hat{\Theta}_1}\theta\bigwedge_{\theta'\in\Theta_1-\hat{\Theta}_1}\neg\theta'$. Define $\pi_2 = \bigwedge_{\theta\in\hat{\Theta}_1}\theta\bigwedge_{\theta'\in\Theta_2-\hat{\Theta}_1}\neg\theta'$. Then we have $\pi_2\in\Pi^{\Theta_2}$ and $\pi_2\Rightarrow\pi_1$. ■

Next we present an iterative algorithm that upon termination yields a finite late-bisimilar quotient. At each iteration of the algorithm, a new set of predicates are introduced (so that a partition that is finer than the one from the previous iteration is obtained). The computation of the new set of predicates is guided by the condition of Theorem 1, and thus, the partition obtained in each iteration is guided "closer" towards a partition that satisfies Theorem 1 (if one exists).

Algorithm 1: Consider an I/O-EFA $P = (L, D, U, Y, \Sigma, E, L_0, D_0)$.

1) Let $k = 0$. $\Theta^k = \cup_{\vec{u}\in U,\vec{y}\in Y,e\in E}\ \{G_e(\vec{d},\vec{u})\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]\}$.
2) $\Theta^{k+1} = \Theta^k\cup_{\vec{u}\in U,\vec{y}\in Y,e\in E,\pi'\in\Pi^{\Theta^k}}\ \{G_e(\vec{d},\vec{u})\wedge\pi'(f_e(\vec{d},\vec{u}))\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]\}$.
3) If $\Pi^{\Theta^{k+1}} = \Pi^{\Theta^k}$, then stop; else let $k := k+1$ and go to step 2.

Note Θ^k is infinite in general. There are special cases when it is finite: (i) When input and output sets are finite, and (ii) When guard condition is generated by the following grammar:

$$G(\vec{d},\vec{u})\rightarrow G(\vec{d})\mid G(\vec{u})\mid\neg G(\vec{d},\vec{u})\mid G_1(\vec{d},\vec{u})\wedge G_2(\vec{d},\vec{u}),$$

and the output set is finite. Also note when there are no inputs and outputs, Θ^0 and Θ^{k+1} are reduced to $\cup_{e\in E}\{G_e(\vec{d})\}$ and $\Theta^k\cup_{e\in E,\pi'\in\Pi^{\Theta^k}}\ \{\pi'(f_e(\vec{d}))\}$, respectively.

Theorem 2: An I/O-EFA satisfies Theorem 1 if and only if Algorithm 1 terminates.

Proof: ($\Leftarrow$) Suppose that the algorithm terminates in the kth step, and suppose for contradiction that the condition of Theorem 1 is not satisfied by Π^{Θ^k}. Then exists $\pi\in\Pi^{\Theta^k}, e\in E,\vec{u}\in U$ such that $\pi\not\Rightarrow\neg G_e(\vec{d},\vec{u})$ and for all $\pi'\in\Pi^{\Theta^k},\vec{y}\in Y$, it holds that $\pi\not\Rightarrow G_e(\vec{d},\vec{u})\wedge\pi'(f_e(\vec{d},\vec{u}))\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]$. Since $\pi\in\Pi^{\Theta^k}$ and for each $\vec{u}\in U, e\in E$, $G_e(\vec{d},\vec{u})\wedge[h_e(\vec{d},\vec{u}) = \vec{y}]\in\Theta^0\subseteq\Theta^k$, it must be the case that $\pi'(f_e(\vec{d},\vec{u}))$ not in Θ^k for all $\pi'\in\Pi^{\Theta^k}, e\in E,\vec{u}\in U,\vec{y}\in Y$. It follows that $\Theta^k\neq\Theta^{k+1}$, a contradiction.

($\Rightarrow$) Suppose exists a partition Π such that the condition of Theorem 1 holds. We first show using induction on k that Π^{Θ^k} is coarser than Π. The base step ($k = 0$) holds from the definition of Θ^0 and Proposition 1. For the induction step,

u mod 10	$G_e \wedge \pi_1(f_e) \wedge [h_e = y]$	$G_e \wedge \pi_2(f_e) \wedge [h_e = y]$	$G_e \wedge \pi_3(f_e) \wedge [h_e = y]$	$G_e \wedge \pi_4(f_e) \wedge [h_e = y]$
0	$[d=2]$, $[d=3]$	F	F	F
1	$[d=2]$, $[d=3]$	F	F	F
2	F	$[d=2]$, $[d=3]$	F	F
3	F	F	$[d=2]$, $[d=3]$	F
4	F	F	F	$[d=2]$, $[d=3]$
5	$[d=2]$, $[d=3]$	F	F	F
6	$[d=2]$, $[d=3]$	F	F	F
7	F	$[d=2]$, $[d=3]$	F	F
8	F	F	$[d=2]$, $[d=3]$	F
9	F	F	F	$[d=2]$, $[d=3]$

TABLE I
TABLE FOR COMPUTING Θ^1

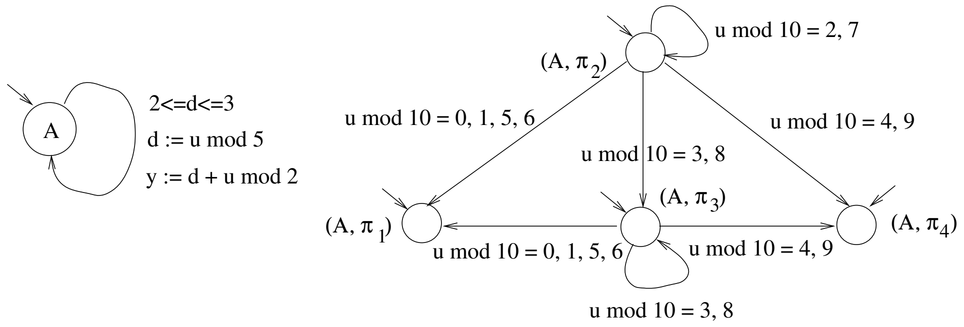

Fig. 3. I/O-EFA P (left) and its finite late-bisimilar quotient (right)

we suppose as part of the induction hypothesis that Π^{Θ^k} is coarser than Π, and pick $\pi \in \Pi$. Then we need to show that exists $\overline{\pi} \in \Pi^{\Theta^{k+1}}$ such that $\pi \Rightarrow \overline{\pi}$. Since $\pi \in \Pi$ for each $e \in E, \vec{u} \in U$, either $\pi \Rightarrow \neg G_e(\vec{d}, \vec{u})$ or exist $\pi' \in \Pi, \vec{y} \in Y$ such that $\pi \Rightarrow G_e(\vec{d}, \vec{u}) \wedge \pi'(f_e(\vec{d}, \vec{u})) \wedge [h_e(\vec{d}, \vec{u}) = \vec{y}]$. If former, we can choose $\overline{\pi} = \neg G_e(\vec{d}, \vec{u})$ and if latter, we can choose $\overline{\pi} = G_e(\vec{d}, \vec{u}) \wedge \pi'(f_e(\vec{d}, \vec{u})) \wedge [h_e(\vec{d}, \vec{u}) = \vec{y}]$. Since $G_e(\vec{d}, \vec{u}) \wedge \pi'(f_e(\vec{d}, \vec{u})) \wedge [h_e(\vec{d}, \vec{u}) = \vec{y}] \in \Theta^{k+1}$, and $\neg G_e(\vec{d}, \vec{u}) = \neg[\vee_{\pi' \in \Pi^{\Theta^k}, \vec{y} \in Y} [G_e(\vec{d}, \vec{u}) \wedge \pi'(f_e(\vec{d}, \vec{u})) \wedge [h_e(\vec{d}, \vec{u}) = \vec{y}]]]$, it follows that in either case $\overline{\pi} \in \Pi^{\Theta^{k+1}}$, as desired. Finally we show that exists k such that $\Pi^{\Theta^k} = \Pi^{\Theta^{k+1}}$. Suppose not true, i.e., for each k, $\Pi^{\Theta^{k+1}}$ is strictly finer than Π^{Θ^k}. Then since Π is a finite partition (satisfying the condition of Theorem 1), exists k such that $|\Pi^{\Theta^k}| > |\Pi|$. This contradicts the fact that Π is finer than Π^{Θ^k} for all k. ■

Since Theorem 1 applies to a larger class of I/O-EFAs as compared to Theorem 3, Theorem 2 suggests that the set of I/O-EFAs identified by Algorithm 1 subsumes that identified by Theorem 3. So if the condition of Theorem 3 does not apply, Algorithm 1 may be used. The following example illustrates Algorithm 1.

Example 2: Consider the I/O-EFA P shown in Figure 3. Then

$$G_e(\vec{d}, \vec{u}) \wedge [h_e(\vec{d}, \vec{u}) = y] = \begin{cases} \cup_{y \in [-\infty, \infty]} [2 \leq d \leq 3] \wedge [1 + d = y] & \text{if } u \text{ mod } 2 = 1 \\ \cup_{y \in [-\infty, \infty]} [2 \leq d \leq 3] \wedge [d = y] & \text{if } u \text{ mod } 2 = 0 \end{cases}$$

It can be verified that the condition of Theorem 3 does not apply.

The predicate $[2 \leq d \leq 3] \wedge [1 + d = y]$ is equal to $[d = 2]$, $[d = 3]$, and False, when $y = 3$, $y = 4$, and $y \notin \{3, 4\}$, respectively. On the other hand, $[2 \leq d \leq 3] \wedge [d = y]$ is equal to $[d = 2]$, $[d = 3]$, and False, when $y = 2$, $y = 3$, and $y \notin \{2, 3\}$, respectively. Thus $\Theta^0 = \{[d = 2], [d = 3]\}$. It follows that $\Pi^{\Theta^0} = \cup_{i=1}^4 \pi_i$, where $\pi_1 = [d < 2], \pi_2 = [d = 2], \pi_3 = [d = 3]$, and $\pi_4 = [d > 3]$.

The computation of Θ^1 is shown in the Table I. We have $\Theta^1 = \Theta^0$, and so Algorithm 1 terminates. The finite late-bisimilar quotient is shown in Figure 3.

IV. CONCLUSION

In this paper we presented an algorithm for computing a finite late-bisimilar quotient for reactive untimed infinite-state systems modeled as I/O-EFAs. We showed that any partition that satisfies the condition of Theorem 1 must be finer than the one induced by the set of data predicates appearing as the guard and assignment conditions of an I/O-EFA. We also showed that it is possible to achieve refinement by introducing additional data predicates. Based on these results, we presented an algorithm that performs refinement by adding new set of predicates at each iteration of the algorithm. The new set of predicates are obtained guided by the condition of Theorem 1, and thus, the partition obtained in each iteration is guided "closer" towards a partition that satisfies Theorem 1 (if one exists). The algorithm terminates if and only if a finite bisimilar quotient satisfying the condition of Theorem 1 exists.

REFERENCES

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[2] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971 – 984, July 2000.

[3] R. Alur, T. A. Henzinger, and P. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Softw. Eng.*, 22(3):181–201, 1996.

[4] T. Ball, R. Majumdar, T. Millstein, and S. K. Rajamani. Automatic predicate abstraction of c programs. In *PLDI '01: Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation*, pages 203–213, New York, NY, USA, 2001.

[5] T. Ball and S. Rajamani. Bebop: A symbolic model checker for boolean programs. *LNCS*, 1885:113–130, 2000.

[6] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Updatable timed automata. *Theor. Comput. Sci.*, 321(2-3):291–345, 2004.

[7] T. Bultan, R. Gerber, and W. Pugh. Symbolic model checking of infinite state systems using presburger arithmetic. In *CAV*, pages 400–411, 1997.

[8] 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.

[9] S. Graf and H. Saidi. Construction of abstract state graphs with pvs. In *Conference on Computer Aided Verification CAV'97*, LNCS 1254, Springer Verlag, 1997.

[10] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[11] Matthew Hennessy, Huimin Lin, and Julian Rathke. Unique fixpoint induction for message-passing process calculi. *Science of Computer Programming*, 41:241–275, 2001.

[12] 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.

[13] T. Henzinger, R. Majumdar, and J. Raskin. A classification of symbolic transition systems. *ACM Transactions on Computational Logic*, 6(1):1–31, 2005.

[14] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *Proceedings of the 29th Annual Symposium on Principles of Programming Languages*, pages pp. 58–70. ACM Press, 2002.

[15] R. Kumar, C. Zhou, and S. Basu. Finite bisimulation of reactive untimed infinite state systems modeled as automata with variables. In *Proceedings of 2006 American Control Conference*, pages 6057–6062, Minneapolis, MN, June 2006.

[16] H. Lin. Symbolic transition graph with assignment. In *CONCUR*, pages 50–65, 1996.

[17] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1):11–44, 1995.

[18] 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.

[19] 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.

[20] A.J. van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE Transactions on Automatic Control*, 49(12):2160–2172, Dec. 2004.

APPENDIX

[15, Theorem 2] identified 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.

The second condition given below imposes restriction on the data update functions and output assignment functions. We use $f_e(i)(\vec{d},\vec{u})$ to represent the update function of the ith data. Let $d(i)^{\max}$ and $d(i)^{\min}$ denote the largest and smallest integer against which the ith data variable is compared over all the guards. 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 $I := \{1,\ldots,p\}$ denote the set of data components.

Definition 6: Given an I/O-EFA satisfying Condition 1, for $\vec{d} \in D$ define
$I^{\max}(\vec{d}),\ I^{\min}(\vec{d}) \subseteq I = \{1,\ldots,p\}$ as:

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

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

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

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

$$\begin{aligned}&\Big[(\vec{d},\vec{u}) \in G_e(D,U) \Rightarrow \\ &1.\ \forall i \in I : \Big(f_e(i)([\vec{d}],\vec{u}) \geq d(i)^{max}\Big) \\ &\vee \Big(f_e(i)([\vec{d}],\vec{u}) \leq d(i)^{min}\Big) \\ &\vee \Big(f_e(i)([\vec{d}],\vec{u}) = \{f_e(i)(\vec{d},\vec{u})\}\Big) \\ &2.\ \ h_e([\vec{d}],\vec{u}) = \{h_e(\vec{d},\vec{u})\}\Big].\end{aligned}$$

Theorem 3: [15, Theorem 2] Given an I/O-EFA P satisfying Conditions 1 and 2, P admits a finite late-bisimilar quotient.