M.V. Iordache, P.J. Antsaklis, “DES Abstractions for the Supervisory Control of Hybrid Systems," special issue on Hybrid Systems of the Transactions of the Institute of Measurement and Control.

DES Abstractions for the Supervisory Control of Hybrid Systems

Marian V. Iordache*and Panos J. Antsaklis†

Abstract

An examination of the literature results on timed and untimed discrete event system (DES) models reveals clearly that the supervisory control problem is more tractable on untimed models. Thus it is interesting to consider the extent to which untimed DES models can be used to design controllers for dynamical systems. In order to approach a larger class of systems appropriate abstraction methods are necessary as well as some extensions of the untimed supervisory control methods. This paper proposes an abstraction procedure that can be used to extract untimed DES models from hybrid automata models with control inputs and continuous disturbances. The abstractions obtained using this procedure are state machines in which every state corresponds to a region of the state space of the hybrid automaton and transitions between states correspond to transitions between the regions. Results describing properties of the abstraction procedure are obtained, including a semidecidability result. The procedure is useful not only for sequential problems but also in the context of concurrency.

1 Introduction

Numerous methods have been proposed for the supervisory control of discrete event systems (DESs). Since a DES model is usually not a complete model of a plant, it is important to ensure that it is obtained in such a way that any controller designed based on the DES model will be able to satisfy the specifications when in closed-loop with the actual plant. Plants modeled by DESs are often more accurately modeled by hybrid systems, since they involve not only discrete but also continuous variables.

In this paper we consider the problem of extracting DES models from hybrid system models. We call such DES models abstractions. For the hybrid system models we use hybrid automata with control inputs. Note that using the underlying automaton of a hybrid automaton as the DES abstraction is in general an inappropriate choice. This is due to the interaction between the continuous and discrete variables of a hybrid system. Rather, DES abstractions should contain enough detail to allow the design of DES controllers that operate correctly when in closed-loop with the actual plant.

*Department of Engineering, LeTourneau University, TX 75607-7001, USA. E-mail: MarianIordache@letu.edu.

†Department of Electrical Engineering, University of Notre Dame, IN 46556, USA. E-mail: Antsaklis.1@nd.edu.

Note that we consider a hierarchical control architecture in which the control of the system is separated in two levels: the higher level performs the supervisory control, while the lower level executes the commands issued by the DES controller. Compared to the usual setting of [8, 15], here the lower level could involve a more complex controller, not just selecting an input from a set of predetermined inputs.

The abstractions generated by the procedure described in this paper can be described as follows. Given a hybrid automaton, let Q be the set of discrete states (or modes) and X the set of continuous states. Given $q \in Q$ and $R \subseteq X$, let (q, R) denote the set of hybrid automaton states $z \in \{q\} \times R$. The abstraction is a state machine in which every state s corresponds to a region (q, R) of the hybrid automaton. The abstraction is such that when a transition exists from s_1 to s_2, the corresponding regions (q_1, R_1) and (q_2, R_2) of the hybrid automaton have the property that there is a control law leading any state z of (q_1, R_1) to (q_2, R_2), regardless of the (bounded) disturbances, and such that the state stays in (q_1, R_1) until (q_2, R_2) is reached. In general, several discrete states s_i may correspond to the same mode q, though the regions R_i will be distinct. Compared to the concept of quasideterminism [7], note that the regions produced by the abstraction procedure satisfy a stronger requirement. Thus, here we can ensure that if a switching sequence (s_1, s_2), (s_2, s_3), $\ldots (s_{n-1}, s_n)$ can be induced in the abstraction, the corresponding switching sequence (q_1, q_2), (q_2, q_3), $\ldots (q_{n-1}, q_n)$ can also be induced in the hybrid automaton, regardless of disturbances, from any (q_1, x) with $x \in R_1$. Further, the converse statement is shown to be also true on a certain version of the abstraction procedure. Moreover, note that if there is a transition from s_1 to s_2, then it is labeled by the same discrete event as the transition from q_1 to q_2 of the hybrid automaton. Thus, specifications in terms of the events of the hybrid system can be enforced by the DES controller. Note also that the abstraction procedure gradually builds the abstraction by iteratively adding new states and arcs. Each intermediary result is a valid abstraction. Such intermediary abstractions may be of interest when the procedure does not converge or when it is attempted to reduce the complexity of the supervisory control problem by using a smaller DES abstraction.

In order to make the method more applicable, the abstraction procedure has been formulated in a general setting, without assuming very specific classes of systems. Thus, while the procedure relies on computations of controlled invariant sets and controlled predecessor sets, it does not propose any specific methods for these computations. Rather, any of the methods developed in the literature could be applied, depending on the specific context.

The contribution of this paper is as follows. In our previous work, we have discussed extensions of the supervisory control of DES to the control of hybrid systems and formulated an abstraction procedure (chapters 9 and 10 of [6]). However, a theoretical analysis of the abstraction procedure has not been published before and is presented here. Here we define the properties satisfied by the abstractions and provide a sufficient condition for the termination of the abstraction procedure. Further, a new version of the

abstraction procedure is introduced for which semidecidability is proved.

Related work includes papers on the supervisory control of hybrid systems, such as [8, 15]. The setting of our paper is also related to that of [13, 12, 16], in which supervision can be applied to abstractions of continuous systems. Note that abstractions have been used also in the context of hybrid system verification, such as in [1, 3, 9]. Since our abstraction method can be applied to concurrent systems, the results on hybrid and continuous Petri nets are also related [10, 2, 4], as well as other approaches that use Petri nets for the modeling of continuous systems [11].

The paper is organized as follows. First, section 2 describes the role of controlled invariants for the feasibility and permissiveness of the DES control problem. Then, the abstraction procedure is presented and characterized in section 3. Examples are given in section 4 and final remarks in the conclusion.

2 Significance of Invariants to DES Control

Supervisory control methods can be applied to the design of controllers for hybrid systems in a hierarchical setting involving a high level DES controller and a low level controller. In this approach the control specification is divided into a specification for the DES controller and a specification for the low level controller. The DES controller restricts the control inputs that can be applied at the lower level. The function of the lower level controller is to generate control inputs based on the input from the DES controller, such that the state of the hybrid system is in the specified region of the state space.

The design of the low and high level controllers relies also on a hierarchical model of the plant. At the low level the plant is represented as a hybrid system. At the high level the plant is represented by a DES model (abstraction) of the hybrid system. Each state s of the DES model corresponds to a region R of the state space of the hybrid system.

Of special interest to the design of DES controllers are the controlled invariants of a hybrid system. Note that a region Z of the state space of the hybrid system is a controlled invariant if there is a control law able to keep the state in Z regardless of disturbances. In the context of concurrency, the ability of a DES controller to wait for the occurrence of certain events is linked to the presence of controlled invariants at the low level. If the DES controller can wait at any state s, then virtually any DES control methods can be used to design the controller. However, such a requirement would eliminate many interesting problems in which not all regions of interest are controlled invariants. As it turns out, this ability to wait for events is not always necessary, or not for all discrete states s, as illustrated in the following example.

Figure 1 represents the abstraction of a hybrid system consisting of three subsystems, denoted as A, B, and C. The hybrid subsystems A, B, and C have the modes a_1 and a_2, $b_1 \dots b_4$, and c_1 and c_2, respectively. This is a simple example in which the modes of the subsystems appear explicitly in the abstraction. Of

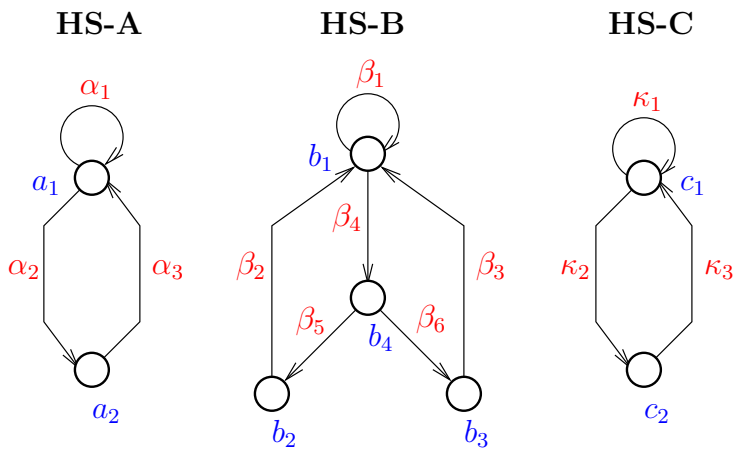


Figure 1: A DES illustrating the significance of invariants to DES control.

course, in general abstractions are much more complex, as more than one place may correspond to the same mode. The abstraction associates with each of the modes a_1, a_2, $\ldots c_2$ the sets A_1, A_2, $\ldots C_2$. Whenever subsystem A is in the mode a_1 of the abstraction, the continuous state of A satisfies $x_a \in A_1$ and whenever it is in the mode a_2, $x_a \in A_2$. The sets $B_1 \ldots C_2$ have the similar meaning. The transitions drawn in the figure can always be taken (they are controllable) when their associated event is enabled by the DES controller. For instance, when the subsystem B is in the mode b_4, any of the transitions β_5 and β_6 can be induced. Further, the self-loop transitions α_1, β_1, and κ_1 correspond to modes that have a controlled invariant set. For instance, as long as the event β_1 is selected, the subsystem B stays in the mode b_1, that is, the continuous state x_b is maintained in the set B_1.

Assume the following specification. The subsystems A and B should not be at the same time in the modes a_2 and b_2, respectively, and the subsystems B and C should not be at the same time in the modes b_3 and c_2, respectively. That is, the forbidden states of the system are (a_2, b_2, c_1), (a_2, b_2, c_2), (a_1, b_3, c_2), and (a_2, b_3, c_2).

The difference between the classical DES setting [14] and the DES setting of the hybrid system abstractions can be illustrated if we assume the system in the state (a_2, b_4, c_2). To ensure the specification is satisfied, both β_5 and β_6 are to be disabled until one of κ_3 or α_3 occurs. Thus, the DES controller should be able to keep the subsystem B in the mode b_4 as long as needed. This ability would be taken for granted in the classical DES setting. However, here b_4 does not have a self-loop, that is, B_4 is not a controlled invariant. This indicates that the controller does not have the option to keep the state in b_4. In fact, it has only two choices, to enable β_5 or β_6. Otherwise, if neither β_5 nor β_6 is enabled, the subsystem B will operate according to unmodeled dynamics, which is clearly an undesirable situation.

Based on this example we can conclude the following.

1. Assuming that all places of the abstractions have self-loops (that is, all associated sets of the hybrid

system are controlled invariants) the methods of the classical DES setting (such as in [14, 5]) can be applied without change.

2. This assumption is not necessary in order to apply untimed supervisory control. Indeed, in our example it is not hard to see that there is a solution if (a_2, b_4, c_2) is added to the set of forbidden states. However, apart from this assumption, some enhancements of the supervisory control methods are required. Enhancements are discussed in chapter 9 of [6].

3. If the mode b_4 had a self-loop, a more permissive controller would be possible, since the state (a_2, b_4, c_2) would be permissible. On the other hand, if b_1 did not have a self-loop, the problem would have no solution. We conclude that the computation of controlled invariants is essential for the permissiveness of the DES controller and for the feasibility of the supervisory control problem.

3 The DES Abstraction

The hybrid automata considered in this paper are defined as follows. A **controlled hybrid automaton** is $H = (Q, X, V, Init, f, Inv, Edg, G, Res, \phi)$ where

1. Q is the set of modes (discrete states).

2. $X \subseteq \mathbb{R}^n$ is the domain of the continuous state variable, denoted by x.

3. $V = U \times D \times \Sigma_C$ is the domain of the input, where U is the domain of the control input, D is the domain of the disturbances and Σ_C is the set of events (or discrete inputs). The events are assumed to be *controllable*, that is, discrete disturbances are not incorporated in our model.

4. $Init$ is the set of initial states (modes).

5. $f : Q \times X \times U \times D \to \mathbb{R}^n$ is the right-hand side of the state equation. Thus, in the continuous-time case: $\dot{x}(t) = f(q, x(t), u(t), d(t))$, and in the discrete-time case: $x(t+1) = f(q, x(t), u(t), d(t))$.

6. $Inv : Q \to 2^X$ maps to each q the invariant of the mode q, that is the set in which x must be when the system is in the mode q.

7. $Edg \subseteq Q \times Q$ is the set of transitions (edges) between modes; (Q, Edg) is a state machine.

8. $G : Edg \to 2^{X \times U \times \Sigma_C}$ maps to each transition a guard, meaning that a transition $e \in Edg$ may occur only if $(x, u, \alpha) \in G(e)$. In particular, when $G(e)$ does not depend on u and α, the transition e is *uncontrollable*. It will be assumed that a transition e occurs if $(x, u, \alpha) \in G(e)$ (where (u, α) is the input applied to the system). We will assume *determinism*, that is $G(q, q_1) \cap G(q, q_2) \neq \emptyset \Rightarrow q_1 = q_2$. Future work may remove this assumption.

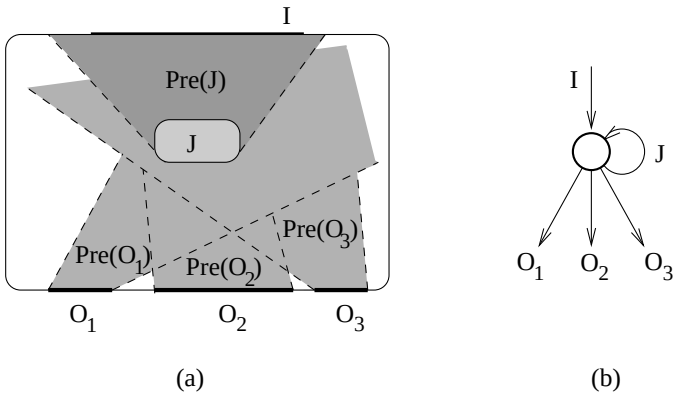


Figure 2: Illustration of a desirable situation in the controlled behavior of a hybrid system. (a) A hybrid system mode with "input" set I and "output" sets O_1, O_2 and O_3 corresponding to the thick lines, controlled invariant set J and $Pre(O_1)$, $Pre(O_2)$, $Pre(O_3)$ and $Pre(J)$ represented through the shaded areas. (b) Equivalent DES abstraction, where the self-loop corresponds to J and the other transitions to the transitions exiting O_1, O_2 and O_3.

9. $Res : Edg \times X \times U \to 2^X$ is the reset map, mapping (e, x, u) with $(x, u) \in G(e)$, to the set in which x may be after the transition e occurs.

10. $\phi : Q \times X \to 2^{U \times D}$ identifies the admissible inputs at every state.

A set I is a **controlled invariant** if for all states $x(0) \in I$ there is some feedback control law ensuring that $x(t) \in I$ for all $t > 0$, regardless of the disturbances. $Pre_q(M)$ is the **controlled predecessor** of M if it contains all values $x(0) \in Inv(q)$ for which there is an admissible control law u leading the state x to M regardless of disturbances. Note that the predecessor was defined with respect to a hybrid system mode q. Since it is obvious from the context what mode is referred, we will denote Pre_q by Pre.

The process of DES abstraction has a favorable situation which we consider below. First we define for every mode $q \in Q$ the following sets:

(i) $J_q \subseteq Inv(q)$.

(ii) For every $(q, p) \in Edg$, let $O_{q \to p} \subseteq Inv(q)$ denote the continuous states of $Inv(q)$ from which it is always possible to switch from mode q to p, regardless of disturbances.

(iii) Let I_q be the set of continuous states in which the mode q may be entered from the modes q_c such that $(q_c, q) \in Edg$.

Note that the set I_q could be reduced by an appropriate control law. An ideal situation for the DES abstraction is when for all $q \in Q$ there is J_q such that:

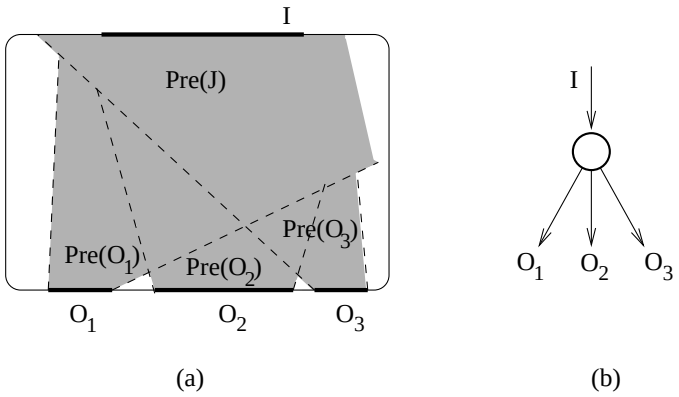


Figure 3: Illustration of the situation in which no controlled invariant set exists. Note that the abstraction has no self-loop.

(a) J_q is a controlled invariant set.

(b) $I_q \subseteq Pre(J_q)$.

(c) $J_q \subseteq \bigcap_{p \in q \rightarrow} Pre(O_{q \rightarrow p})$.

where we write $p \in q \rightarrow$ for $(q, p) \in Edg$. This situation is illustrated in Figure 2, together with the DES abstraction of the mode. Thus, once we have the sets I_q and $O_{q \rightarrow p}$, we are interested to compute the maximal controlled invariant set J_q satisfying (i) and (c). Indeed, if the maximal controlled invariant set does not satisfy (b), there is no controlled invariant set J_q satisfying (a-c). However, even when (b) is not satisfied, we may still be able to reduce the set I_q (through a control law) such that (b) is satisfied.

In the cases, when we find no invariant set J_q satisfying (a-c), we could obtain DES abstractions by simple considerations, such as considering the inclusion relations between I_q on one side and $Pre(O_{q \rightarrow p})$ on the other, or among I_q, $Pre(J_q)$, J_q and $Pre(O_{q \rightarrow p})$. A situation in which a mode has no controlled invariant sets is illustrated in Figure 3.

The abstraction procedure we propose will be defined next. The procedure assumes a hybrid automaton $H = (Q, X, V, Init, f, Inv, Edg, G, Res, \phi)$ to be given. The result of the procedure is a state machine $(S, \rightarrow)$, where S is the set of states and $\rightarrow \subseteq S \times S$ is the transition relation. Here is the notation.

- Similar to $Inv(q)$ in the hybrid automaton, we define also $Inv : S \rightarrow X$ for the states s of the abstraction. Further, the map $C : S \rightarrow Q$ is defined, to associate a mode $q \in Q$ to each $s \in S$. Thus, each $s \in S$ corresponds to a region $(C(s), Inv(s))$ of the hybrid system, where $Inv(s) \subseteq Inv(C(s))$.

- Let $Res_{q' \rightarrow q}$ denote the area in which the state is reset when switching from q' to q for $q', q \in Q$. Technically, let ϕ_u be the restriction of ϕ to 2^U (that is, $\phi_u(p, x)$ is the set of inputs that can be applied

when the mode is p and the state x.) Then $Res_{q' \to q} = \bigcup_{(x,u) \in V} Res((q',q),x,u)$ for $V = \{(x,u) | \, (\exists \alpha \in \Sigma_C : (x,u,\alpha) \in G(q',q)) \wedge x \in Inv(q') \wedge u \in \phi_u(q',x)\}$.

- Given $I \subseteq Inv(q)$, the set $G^{-1}_{q' \to q}(I)$, $q', q \in Q$, denotes the set of states x in mode q' from which there is an input leading to the mode q with the state x reset within I. Formally, $G^{-1}_{q' \to q}(I) = \{x \in Inv(q') | \, \exists u \in \phi_u(q',x), \exists \alpha \in \Sigma_C : Res((q',q),x,u) \subseteq I \wedge (x,u,\alpha) \in G(q',q)\}$.

- The predecessor is defined with respect to each mode $q \in Q$. As the mode q is clear from the context, we use the notation Pre instead of Pre_q. Note that for any set M, $Pre_q(M)$ is by definition a subset of $Inv(q)$. Formally, let $\psi(t)$ denote the solution to $\dot{x} = f(q,x,u,d)$ or $x(t+1) = f(q,x(t),u,d)$, assuming it exists. Then, $Pre_q(M) = Inv(q) \cap \{z \in X : \exists t > 0, \exists u : [0,t] \times X \to \mathbb{R}^m, \forall d : [0,t] \to \mathbb{R}^p : [\forall \tau \in [0,t], u(\tau, \psi(\tau)) \in \phi_u(q, \psi(\tau))] \wedge [(\forall \tau \in [0,t], d(\tau) \in \phi_d(q, \psi(\tau))) \Rightarrow (\exists \tau \in [0,t], \psi(\tau) \in M)] \wedge \psi(0) = z\}$. Similarly to ϕ_u, ϕ_d is the restriction of ϕ to 2^D.

The procedure starts with a number of sets of interest (q_i, J_i) specified by the user, where $q_i \in Q$ and $J_i \subseteq Inv(q_i)$. These could represent portions of the state space that should be reached during the operation of the plant. The abstraction procedure will be called the **AB-procedure**. It is defined next.

Input: The hybrid automaton and the sets of interest (q_i, J_i).

Output: A state machine $(S, \to)$ and the maps C and Inv.

Procedure:

1. Initialize $S = \emptyset$ and $\to = \emptyset$.
2. For all sets of interest (q, J), create a state $s \in S$ with $C(s) = q$ and $Inv(s) = Pre(J)$. If J is a controlled invariant, add (s,s) to $\to$.
3. Initialize $StateList := S$.
4. While $StateList \neq \emptyset$ do
   (a) For all $s \in StateList$
      i. Compute $I_{q \to s} = Inv(s) \cap Res_{q \to C(s)}$ for all $q \in Q$ such that $(q, C(s)) \in Edg$.
      ii. For all $I_{q \to s}$ computed above find $O_{q \to s} = G^{-1}_{q \to C(s)}(I_{q \to s})$.
      iii. For all $O_{q \to s}$ computed above, if $O_{q \to s} \neq \emptyset$, add $O_{q \to s}$ to the list $L(q)$.
   (b) Set $StateList = \emptyset$.
   (c) For all $q \in Q$ with $L(q) \neq \emptyset$ do:
      i. For all $O_{q \to s} \in L(q)$ and $s' \in S$, if $C(s') = q$ and $Inv(s') \subseteq Pre(O_{q \to s})$, add (s', s) to $\to$.

ii. Distribute the states s with $O_{q\to s} \in L(q)$ into k disjoint groups $\Gamma_1 \dots \Gamma_k$, such that $Inv_i \neq \emptyset$ for $i = 1 \dots k$ and $Inv_i \neq Inv_j$ for $i \neq j$, where $Inv_i := \bigcap_{s\in\Gamma_i} Pre(O_{q\to s})$.

iii. Let $s_1 \dots s_{2k} \notin S$ be new discrete states.

iv. Let $cinv(Inv_i)$ denote the maximal controlled invariant set included in Inv_i.

v. For all $i = 1 \dots k$, let $Inv(s_i) = cinv(Inv_i)$ and $Inv(s_{i+k}) = Inv_i$. If $cinv(Inv_i) = Inv_i$, then s_{i+k} will denote s_i: $s_{i+k} \equiv s_i$.

vi. For all $i = 1 \dots 2k$, if $Inv(s_i) = Inv(s')$ and $C(s') = q$ for some $s' \in S$, then s_i will denote s': $s_i \equiv s'$.

vii. Let $\Gamma = \{i : Inv(s_i) \neq \emptyset\}$. For all $i \in \Gamma$, if $s_i \notin S$ then add s_i to S and $StateList$, and set $C(s_i) = q$.

viii. For all $i \in \Gamma$ with $i \leq k$, add (s_i, s_i) to $\to$ and (s_i, s) to $\to$ for all $s \in \Gamma_i$.

ix. For all $i \in \Gamma$ with $i > k$, for all $s \in \Gamma_{i-k}$, add (s_i, s) to $\to$.

(d) Set $L(q) = \emptyset$ for all $q \in Q$.

The procedure is graphically illustrated in Figure 4. Note that the procedure does not assume a set of initial states for the hybrid automaton. Rather, the abstraction could be used to determine the states in which the system could be initialized.

Given $P_1 \subseteq Inv(q_1)$ and $P_2 \subseteq Inv(q_2)$, $q_1, q_2 \in Q$, we write $(q_1, P_1) \to (q_2, P_2)$ if it is always possible to reach (q_2, P_2) from any (q, x) with $x \in P_1$. That is, we write $(q_1, P_1) \to (q_2, P_2)$ if $Pre(G^{-1}_{q_1\to q_2}(P_2)) \supseteq P_1$. On the other hand, we will also write $(q, P) \to (q, P)$ if P is a controlled invariant.

Definition 3.1 *A collection of sets $\mathcal{C} = \{(q, Q_i) : q \in Q$ and $Q_i \subseteq Inv(q)\}$ is said to satisfy the* **abstraction property (AP)**, *if*

1. *For all sets of interest (q, J) given as input to the AB-procedure, $\exists (p, Q_i) \in \mathcal{C}$ such that $p = q$ and $Q_i = Pre(J)$.*

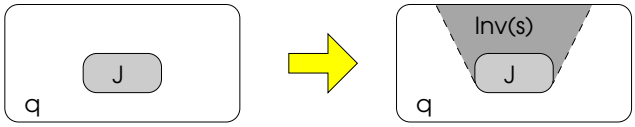
2. *For all $p \in Q$ and $P \subseteq Inv(p)$ satisfying $(p, P) \to (q_i, Q_i)$ for some $(q_i, Q_i) \in \mathcal{C}$ with $p \neq q_i$, $\exists Q_j \in Inv(p)$, $(p, Q_j) \in \mathcal{C}$, such that*

   (a) $P \subseteq Q_j$

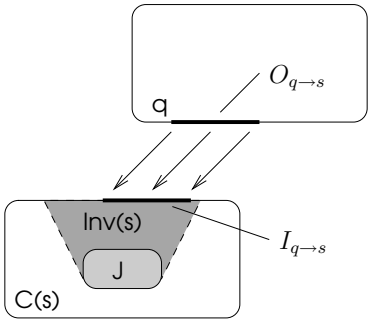
   (b) $\forall (q, Q_k) \in \mathcal{C}$, $[(p, P) \to (q, Q_k)] \Rightarrow [(p, Q_j) \to (q, Q_k)]$.

   (c) $[(p, P) \to (p, P)] \Rightarrow [(p, Q_j) \to (p, Q_j)]$.

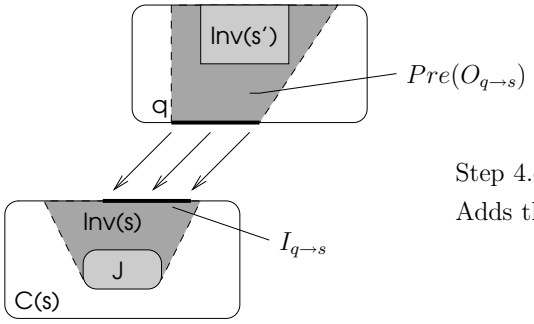
Proposition 3.1 *Let $\mathcal{C}$ be a collection of sets satisfying the AP. Then, the abstraction produced by the AB-procedure satisfies that for all $s \in S$, $(C(s), Inv(s)) \in \mathcal{C}$.*



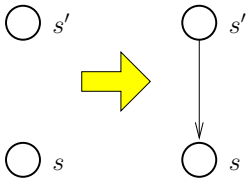
Step 2:
$Inv(s) = Pre(J)$
$C(s) = q$

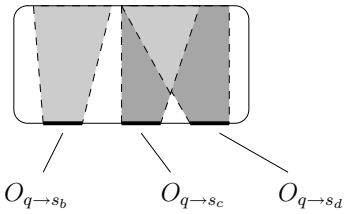


Step 4.a:
Computation of $O_{q\to s}$



Step 4.c.i:
Adds the arc (s', s) to $\to$

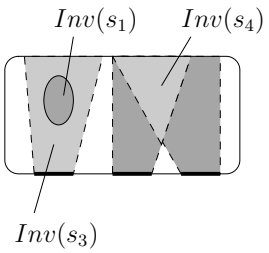




Step 4.c.ii:

$\Gamma_1 = \{s_b\}$, $Inv_1 = Pre(O_{q\to s_b})$

$\Gamma_2 = \{s_c, s_d\}$, $Inv_2 = Pre(O_{q\to s_c}) \cap Pre(O_{q\to s_d})$



Steps 4.c.iii–ix:

$Inv(s_1) = cinv(Inv_1)$
$Inv(s_2) = cinv(Inv_2) = \emptyset$
$Inv(s_3) = Inv_1$
$Inv(s_4) = Inv_2$
$\Gamma = \{1, 3, 4\}$

The abstraction after the step 4.c.ix:

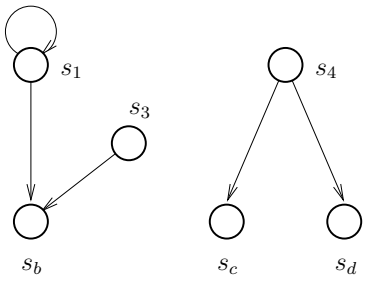


Figure 4: Illustration of the AB-procedure.

Proof: The proof is by induction: assuming that all states s added to S before the iteration k of step 4 satisfy $(C(s), Inv(s)) \in \mathcal{C}$, we show that the states s added to S in the iteration k satisfy also $(C(s), Inv(s)) \in \mathcal{C}$.

Note that the step 2 of the AB-procedure ensures that initially all $s \in S$ satisfy $(C(s), Inv(s)) \in \mathcal{C}$, by part 1 of Definition 3.1. At the iteration k, new discrete states s are added to S in the step 4.c.vii. They have $Inv(s) = Inv_i$ or $Inv(s) = cinv(Inv_i)$ and $C(s) = q$ for some $q \in Q$, where $Inv_i = \bigcap_{s \in \Gamma_i} Pre(O_{q \to s})$ is computed at the step 4.c.ii. There are two cases.

Case 1, $Inv(s) = Inv_i \neq cinv(Inv_i)$: Let $P = Inv_i$. By the definition of Inv_i, $(q, P) \to (C(s), Inv(s))$ $\forall s \in \Gamma_i$. Further, $(q, P) \not\to (q, P)$. By the induction assumption, $\chi = \{(C(s), Inv(s)) : s \in \Gamma_i\} \subseteq \mathcal{C}$. By Definition 3.1, there is Q_j such that $(q, Q_j) \in \mathcal{C}$, $Q_j \supseteq P$, and $(q, Q_j) \to (q_l, Q_l)$ for all $(q_l, Q_l) \in \chi$. Hence, $\forall (q_l, Q_l) \in \chi$: $Pre(G^{-1}_{q \to q_l}(Q_l)) \supseteq Q_j$. Therefore, $Q_j \subseteq Inv_i$, and so $Q_j = P$, as we already know that $Q_j \supseteq P$ and $P = Inv_i$.

Case 2, $Inv(s) = cinv(Inv_i)$: Let $P = cinv(Inv_i)$. As in Case 1, we get $Q_j \subseteq Inv_i$. Then, by part 2.c of Definition 3.1, $(q, Q_j) \to (q, Q_j)$, that is, Q_j is a controlled invariant. By definition, $cinv(Inv_i)$ is the largest controlled invariant contained in Inv_i, and so $Q_j \subseteq cinv(Inv_i)$. It follows $P = Q_j$, since we already know that $P = cinv(Inv_i)$ and $Q_j \supseteq P$.

In both case 1 and 2 we have proved $Q_j = P$. This ends the proof, since $(q, Q_j) \in \mathcal{C}$, $C(s) = q$ and P was selected such that $Inv(s) = P$. □

The previous result can be used to give a sufficient condition for the termination of the AB-procedure. However, note that in principle we could terminate the procedure as soon as the abstraction is fine enough for our purposes, without waiting for a possible convergence. For instance, a termination condition could be that the abstraction has a directed circuit including all $s \in S$ associated with the sets of interest.

Proposition 3.2 *The AB-procedure terminates if there is a finite collection $\mathcal{C}$ with the AP property.*

Proof: This is an immediate consequence of Proposition 3.1, as every state s of S corresponds to an element $(q, Q_j) \in \mathcal{C}$ such that $C(s) = q$ and $Inv(s) = Q_j$. □

Note that the AB-procedure may not produce an abstraction that satisfies the AP. The AB-procedure was constructed such that if a sequence (s_0, s_1), (s_1, s_2), $\ldots (s_{n-1}, s_n)$ can be induced in $(S, \to)$, then the sequence (q_0, q_1), (q_1, q_2), $\ldots (q_{n-1}, q_n)$ can also be induced in the hybrid automaton, regardless of disturbances, for $q_i = C(s_i)$ and any initial value of x in $Inv(s_0)$. Moreover, the AB-procedure guarantees that during the switching sequence the state will satisfy $x \in Inv(s_i)$ in the mode q_i. However, all these are not sufficient to guarantee the AP is satisfied.

It is possible to modify the AB-procedure such that the AP is satisfied. This modified version of the procedure will be called the **AB-L procedure**. The AB-L procedure is identical to the AB-procedure,

except for the following two changes. First, the step 4.d is removed. That is, $L(q)$ will contain all the sets $O_{q\to s}$ computed up to the current iteration. Second, the step 4.c.ii, is changed as follows: Find all sets $\Gamma_i \subseteq S$, $i = 1, 2, \ldots$ such that $Inv_i \neq \emptyset$, $Inv_i \neq Inv_j$ for $i \neq j$, and $Inv_i \neq \bigcap_{s\in Z} Pre(O_{q\to s})$ for all $Z \supset \Gamma_i$, where $Inv_i := \bigcap_{s\in\Gamma_i} Pre(O_{q\to s})$. Note that the condition $Inv_i \neq \bigcap_{s\in Z} Pre(O_{q\to s})$ for all $Z \supset \Gamma_i$ requires that the sets Γ_i be maximal.

From a practical viewpoint, the AB-L procedure may not be as attractive as the AB-procedure, as it has less likely convergence. However, from a theoretical viewpoint, its study reveals properties of interest to this abstraction approach. We show next that the AB-L procedure produces sets $\mathcal{C}$ satisfying the AP.

Lemma 3.1 *If the AB-L procedure terminates, the set $\mathcal{C} = \{(C(s), Inv(s)) : s \in S\}$ satisfies the AP.*

Proof: Part 1 of Definition 3.1 is satisfied because of the step 2 of the AB-L procedure. So we prove part 2 of the Definition 3.1. For convenience, let's denote $Inv(s_k)$ by Q_k, for all $s_k \in S$. Let $q \in Q$ and $P \subseteq Inv(q)$ be chosen arbitrarily, such that $(q, P) \to (q_i, Q_i)$ for some $(q_i, Q_i) \in \mathcal{C}$. Let $\mathcal{L} = \{s_k \in S : (q, P) \to (C(s_k), Q_k)\}$. Note that each time a state s_k is added to S in the AB-L procedure (see step 4.c.vii), $O_{q\to s_k}$ is computed at the next iteration for all $q \in Q$ (see step 4.a.iii). Therefore, for each $s_k \in \mathcal{L}$, there is a step 4.c.ii at which all $O_{q\to s_k}$ are included in $L(q)$. At that step, $P \subseteq Inv_i$ for some $\Gamma_i \supseteq \mathcal{L}$. If P is not a controlled invariant, then $s_j \in S$ with $C(s_j) = q$ and $Q_j = Inv_i$ satisfies the requirement 2 of Definition 3.1. Else, if P is a controlled invariant, then $s_j \in S$ with $C(s_j) = q$ and $Q_j = cinv(Inv_i)$ satisfies the requirement 2 of Definition 3.1. □

Lemma 3.2 *Let $\mathcal{C}$ be a collection of sets satisfying the AP. Then, the abstraction produced by the AB-L procedure satisfies that for all $s \in S$, $(C(s), Inv(s)) \in \mathcal{C}$.*

Proof: The proof is similar to that for the AB-procedure in Proposition 3.1. □

Assuming that all operations involved in the AB-L procedure are computable (i.e. Pre, set comparison, set intersection, complement, and notably, the computation of the maximal controlled invariants), we have the following result.

Theorem 3.1 *The AB-L procedure is semidecidable.*

Proof: By Lemma 3.1, the AB-L procedure computes a set $\mathcal{C}$ satisfying the AP. By Lemma 3.2, given $\mathcal{C}$ satisfying the AP, $s \in S \Rightarrow (C(s), Inv(s)) \in \mathcal{C}$. Therefore, if a finite set $\mathcal{C}$ satisfying the AP exists, the AB-L procedure must terminate. □

The next result shows that the AB-L procedure generates an abstraction that captures all switching sequences that can be robustly enforced and lead to the sets of interest.

Theorem 3.2 *Assume that the AB-L procedure terminates. Let $(S, \rightarrow)$ be the state machine it produces. Let (q_n, J) be one of the sets of interest used in the AB-L procedure, and let $q_0 \in Q$. Assume that from (q_0, x_0), with $x_0 \in Inv(q_0)$, it is possible to reach (q_n, J) with the control law $u = g(q, x)$ and the switching sequence $(q_0, q_1), (q_1, q_2), \ldots (q_{n-1}, q_n)$, $q_{i-1} \neq q_i$, regardless of the disturbances that may occur. Then there are $s_0, s_1, \ldots s_n \in S$ such that $C(s_i) = q_i \ \forall i = 0 \ldots n$, $Inv(s_n) = Pre(J) \cap Inv(q_n)$, $x_0 \in Inv(s_0)$, and the sequence $(s_0, s_1), (s_1, s_2), \ldots (s_{n-1}, s_n)$ can be induced in $(S, \rightarrow)$.*

Proof: Consider the trajectories leading (q_0, x_0) to (q_n, J) through the given control law and switching sequence. Let $Z_0 = \{x\}$. Let $Z_1 \subseteq Inv(q_1)$ be the set of states x in which the mode q_1 is entered. (Z_1 depends on x_0, the control law, the disturbances, and the reset map). Similarly, let $Z_i \subseteq Inv(q_i)$ be the set of states x in which the mode q_i is entered. Then $Z_n \subseteq Pre(J) \cap Inv(q_n)$. Let $s_n \in S$ be such that $Inv(s_n) = Pre(J) \cap Inv(q_n)$. Note that s_n exists by Lemma 3.1 and part 1 of Definition 3.1. Note also that since $(q_{n-1}, Z_{n-1}) \rightarrow (q_n, Inv(s_n))$, there is $s_{n-1} \in S$ such that $C(s_{n-1}) = q_{n-1}$ and $Z_{n-1} \subseteq Inv(s_{n-1})$, by Lemma 3.1 and part 2(a-b) of Definition 3.1. Further, $(s_{n-1}, s_n) \in \rightarrow$. Continuing the same way, we find $s_i \in S$ with $C(s_i) = q_i$, $Z_i \subseteq Inv(s_i)$, and $(s_i, s_{i+1}) \in \rightarrow$, for $i = n-2, n-3, \ldots 1, 0$. □

Note that both the AB-procedure and the AB-L procedure were defined such that the converse of the Theorem 3.2 is satisfied. Thus, if the sequence $(s_0, s_1), (s_1, s_2), \ldots (s_{n-1}, s_n)$ can be induced in $(S, \rightarrow)$, the sequence $(q_0, q_1), (q_1, q_2), \ldots (q_{n-1}, q_n)$, can also be induced in the hybrid system from q_0 and any $x \in Inv(s_0)$, where $q_i = C(s_i)$ for all i. However, unlike to the AB-L procedure, the AB-procedure does not satisfy Theorem 3.2. The AB-procedure does not produce an abstraction describing all possible sequences of events that can be controlled in the hybrid automaton. However, this may not be necessary. In fact, the two procedures could be stopped before convergence if the abstraction contains sufficient information to solve the specification of the DES controller.

4 Examples

In this section the abstraction procedures are illustrated on three examples. We begin with an example in which the abstraction procedures do not terminate. Consider the hybrid system shown in Figure 5. There are no continuous control inputs and no disturbances. The state variable x has dimension 1 and the invariants are $Inv(q_1) = Inv(q_2) = \mathbb{R}$. The state equations in the two modes are defined as follows.

$$\text{(Mode } q_1) \quad \dot{x} = -2(x - 2k) \quad \text{for } x \in [2k-1,\ 2k+1) \text{ and } k \in \mathbb{Z} \tag{1}$$

$$\text{(Mode } q_2) \quad \dot{x} = -x + 2k + 1 \quad \text{for } x \in [2k,\ 2k+2) \text{ and } k \in \mathbb{Z} \tag{2}$$

The transition (q_1, q_2) occurs when $x \in \bigcup_{k\in\mathbb{Z}}[2k - 0.25,\ 2k + 0.25)$ and the controllable event α occurs. Similarly, the transition (q_2, q_1) occurs when $x \in \bigcup_{k\in\mathbb{Z}}[2k + 0.75,\ 2k + 1.25)$ and the controllable event β

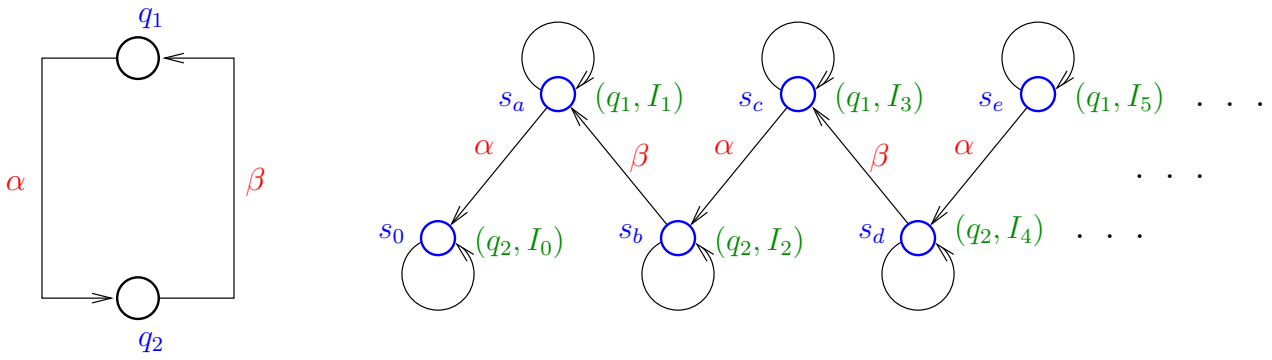


Figure 5: Structure of hybrid system (left) and infinite abstraction (right).

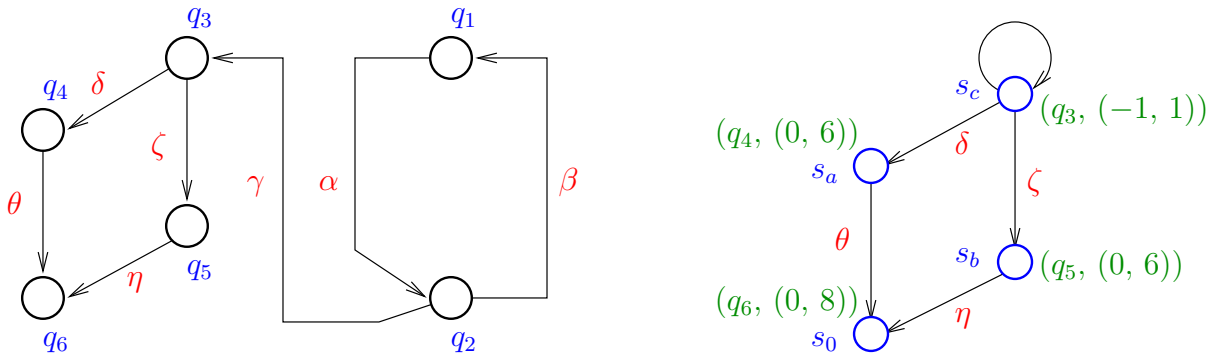


Figure 6: Hybrid system structure in the second example (left) and possible outcome of the AB-procedure (right).

occurs. Further, the reset map is defined as follows.

$$Res((q_1, q_2), x) = [2k - 2,\ 2k) \text{ for } x \in [2k - 0.25,\ 2k + 0.25) \text{ and } k \in \mathbb{Z}$$

$$Res((q_2, q_1), x) = [2k - 1,\ 2k + 1) \text{ for } x \in [2k + 0.75,\ 2k + 1.25) \text{ and } k \in \mathbb{Z}$$

Now, assume that the AB-procedure is started with the set of interest (q_2, J), where $J = [0.9,\ 1.1]$. Given an integer n, let $I_n = [n,\ n + 2)$. Then, $Pre(J) = I_0$. Therefore, the second step of the procedure creates the state s_0 with $C(s_0) = q_2$ and $Inv(s_0) = I_0$. At step 4 it is found that a transition to s_0 is possible from the mode q_1 and the set $O_{q_1 \to s_0} = [1.75,\ 2.25)$. Since $Pre(O_{q_1 \to s_0}) = I_1$, we obtain a new state s_1 with $C(s_1) = q_1$ and $Inv(s_1) = I_1$. The state s_1 has a self-loop, since I_1 is an invariant set. The following iterations of the procedure are similar. The abstraction shown in Figure 5 is obtained. Note that the abstraction is infinite. Therefore, the abstraction procedure cannot terminate.

In view of Theorem 3.1, the AB-L procedure terminates iff there is a finite abstraction satisfying the AP (the property introduced in Definition 3.1). Since the abstractions produced by the AB-procedure may not satisfy the AP, it may be possible for the AB-procedure to terminate when the AB-L procedure does not

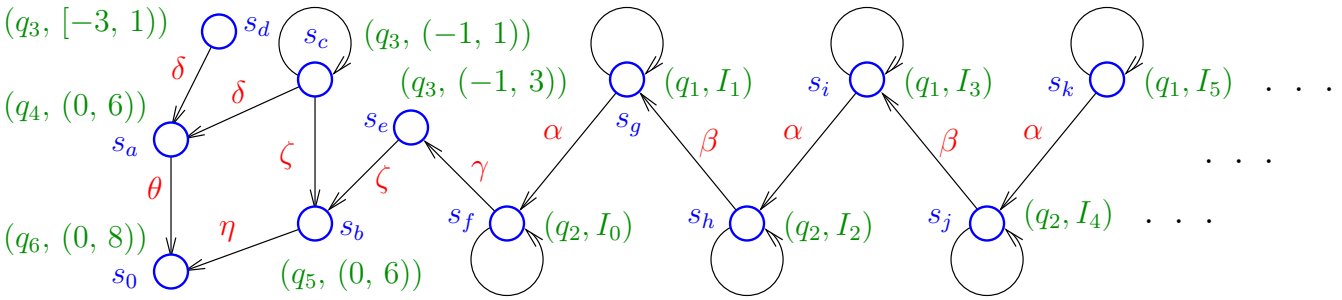


Figure 7: Abstraction of the AB-L procedure.

terminate. The following example illustrates this point.

Consider the hybrid system structure shown in Figure 6. The hybrid system consists of the hybrid system of the first example (Figure 5) and four additional modes, $q_3 \ldots q_6$. The modes q_1 and q_2 have the same properties as in the first example. This time, however, the system has also a bounded continuous control input $u \in I_{-1}$. Note that we continue to use the notation $I_n = [n, n+2)$, where n is an integer. The additional modes have $Inv(q_3) = \ldots = Inv(q_6) = [-9, 9]$ and the same reset map $Res(e, x, u) = I_1$, for all edges e directed towards one of the modes $q_3 \ldots q_6$. Moreover, the guards are defined as follows. $G((q_2, q_3)) = I_0 \times I_{-1} \times \{\gamma\}$, meaning that the transition from q_2 to q_3 takes place when $x \in I_0$, $u \in I_{-1}$, and the controllable event γ occurs. Further, $G((q_3, q_4)) = I_{-3} \times I_{-1} \times \{\delta\}$, $G((q_3, q_5)) = I_1 \times I_{-1} \times \{\zeta\}$, $G((q_4, q_6)) = I_4 \times I_{-1} \times \{\theta\}$, and $G((q_5, q_6)) = I_4 \times I_{-1} \times \{\eta\}$. Note that the transitions between q_1 and q_2 are defined the same way as in the first example and they are not affected by the control input u. The state equations are defined as follows. In mode q_3, $\dot{x} = x + u$. In mode q_4, $\dot{x} = 4x$. In mode q_5, $\dot{x} = 5x$. In mode q_6, $\dot{x} = 6x$. Assume that the abstraction procedure is started with the set of interest (q_6, J), where $J = [7, 8)$.

The iterations of the AB-procedure and the AB-L procedure are summarized in Tables 1 and 2, respectively, where an iteration consists of one execution the steps (a)–(d) of the step 4 of the procedures. The corresponding abstractions are shown in Figures 6 and 7. Note that excepting the final iteration, each iteration adds at least one new state s to the abstraction. Edges outgoing from s are added in the same iteration and possibly also in the next iterations. Edges incoming to s are added in the following iteration and possibly also in the remaining iterations.

In this example the AB-procedure terminates in three iterations, while the AB-L procedure does not terminate. The example could be easily modified so that the abstraction of Figure 7 has only N states associated with the modes q_1 and q_2, where N is an arbitrary positive integer. Then, the total number of states of the abstraction produced by the AB-procedure would be 4, and the total number of states in the case of the AB-L procedure would be $N + 6$. This illustrates the fact that in general there is no relationship

Table 1: In summary, the AB-procedure creates the abstraction of Figure 6 as follows.

Initialization	Add s_0 with $C(s_0) = q_6$ and $Inv(s_0) = Pre(J) = (0, 8)$.
Iteration 1	$O_{q_4 \to s_0} = [4, 6)$, add s_a with $C(s_a) = q_4$ and $Inv(s_a) = Pre(O_{q_4 \to s_0}) = (0, 6)$.
	$O_{q_5 \to s_0} = [4, 6)$, add s_b with $C(s_b) = q_5$ and $Inv(s_b) = Pre(O_{q_5 \to s_0}) = (0, 6)$.
Iteration 2	$O_{q_3 \to s_a} = [-3, -1)$, $Pre(O_{q_3 \to s_a}) = [-3, 1)$, $O_{q_3 \to s_b} = [1, 3)$, $Pre(O_{q_3 \to s_b}) = (-1, 3)$, add s_c with $C(s_c) = q_3$ and $Inv(s_c) = Pre(O_{q_3 \to s_a}) \cap Pre(O_{q_3 \to s_b}) = (-1, 1)$.
Iteration 3	$O_{q_2 \to s_c} = \emptyset$, so no new state s is added.

Table 2: In summary, the AB-L procedure creates the abstraction of Figure 7 as follows.

$\vdots$	$\vdots$ $\vdots$ $\vdots$
Iteration 2	$O_{q_3 \to s_a} = [-3, -1)$, $Pre(O_{q_3 \to s_a}) = [-3, 1)$, $O_{q_3 \to s_b} = [1, 3)$, and $Pre(O_{q_3 \to s_b}) = (-1, 3)$. Add s_c with $C(s_c) = q_3$ and $Inv(s_c) = Pre(O_{q_3 \to s_a}) \cap Pre(O_{q_3 \to s_b}) = (-1, 1)$. Add s_d with $C(s_d) = q_3$ and $Inv(s_d) = Pre(O_{q_3 \to s_a}) = [-3, 1)$. Add s_e with $C(s_e) = q_3$ and $Inv(s_e) = Pre(O_{q_3 \to s_b}) = (-1, 3)$.
Iteration 3	$O_{q_2 \to s_e} = [0, 2)$, add s_f with $C(s_f) = q_2$ and $Inv(s_f) = Pre(O_{q_2 \to s_e}) = [0, 2)$.
Iteration 4	$O_{q_1 \to s_f} = [1.75, 2.25)$, add s_g with $C(s_g) = q_1$ and $Inv(s_g) = Pre(O_{q_1 \to s_f}) = [1, 3)$.
$\vdots$	$\vdots$ $\vdots$ $\vdots$

between the convergence velocities of the two procedures. However, it is clear that the AB-procedure is faster, since the operations it performs are a subset of the operations of the AB-L procedure.

Finally, this example could be used to illustrate that the behavior of the AB-procedure can be very sensitive to the way the step 4.c.ii is performed. Referring to Table 1 and the iteration 2, the step 4.c.ii was performed by taking $k = 1$ and $\Gamma_1 = \{s_a, s_b\}$. If instead the choice $k = 2$, $\Gamma_1 = \{s_a\}$, and $\Gamma_2 = \{s_b\}$ were made, the procedure would have not terminated and it would have generated an abstraction similar to that of the AB-L procedure.

The abstraction procedures are further illustrated on a third example. Consider a hybrid system having the structure shown in Figure 8. This time we will avoid the details related to continuous dynamics and assume that each mode can be abstracted as shown in Figure 9. Each directed arc denotes a controllable transition. Given the set of interest $(q, J) = (q_2, \{a\})$, the abstractions produced by the AB-procedure and the AB-L procedure are shown in Figures 10 and 11, respectively. The operations of the procedures are summarized in Tables 3 and 4. Note that the abstraction procedures do not generate unnecessary edges, but only edges required by the abstraction property (Definition 3.1). In particular, two states s_i and s_j having $(C(s_i), Inv(s_i)) = (q, A_i)$, $(C(s_j), Inv(s_j)) = (q, A_j)$, and $A_i \subset A_j$, could be joined by a directed arc from

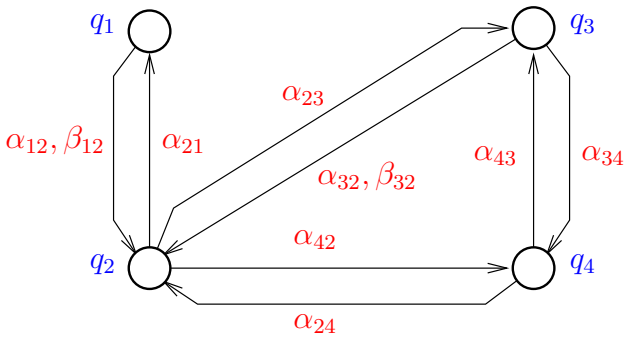


Figure 8: Hybrid system structure in the third example.

s_i to s_j. Such directed arcs are represented with dashed lines in Figures 10 and 11.

Table 3: In summary, the AB-procedure creates the abstraction of Figure 10 as follows.

Initialization	Add s_0 with $C(s_0) = q_2$ and $Inv(s_0) = Pre(J) = \{a, b, c\}$.
Iteration 1	$O_{q_1 \to s_0} = \{i\}$, add s_1^1 with $C(s_1^1) = q_1$ and $Inv(s_1^1) = Pre(O_{q_1 \to s_0}) = \{h, i\}$.
	$O_{q_3 \to s_0} = \{k\}$, add s_2^1 with $C(s_2^1) = q_3$ and $Inv(s_2^1) = Pre(O_{q_3 \to s_0}) = \{j, k\}$.
Iteration 2	$O_{q_2 \to s_1^1} = \{e\}$, $Pre(O_{q_2 \to s_1^1}) = \{e, f\}$, $O_{q_2 \to s_2^1} = \{g\}$, $Pre(O_{q_2 \to s_2^1}) = \{f, g\}$, add s_3^2 with $C(s_3^2) = q_2$ and $Inv(s_3^2) = Pre(O_{q_2 \to s_1^1}) \cap Pre(O_{q_2 \to s_2^1}) = \{f\}$.
	$O_{q_4 \to s_2^1} = \{n\}$, add s_4^2 with $C(s_4^2) = q_4$ and $Inv(s_4^2) = Pre(O_{q_4 \to s_2^1}) = \{m, n\}$.
Iteration 3	$O_{q_2 \to s_4^2} = \{d\}$, add s_5^3 with $C(s_5^3) = q_2$ and $Inv(s_5^3) = Pre(O_{q_2 \to s_4^2}) = \{a, b, c, d\}$.
Iteration 4	$Pre(O_{q_1 \to s_5^3}) = Inv(s_1^1)$ and $Pre(O_{q_3 \to s_5^3}) = Inv(s_2^1)$, so no new state s is added.

5 Conclusions

The paper has presented an abstraction procedure that obtains DES models from hybrid automata. The obtained DES models are appropriate not only for sequential control but also in the context of concurrency. They allow the application of DES control methods for the enforcement of specifications expressed in terms of the events of the hybrid system and of the regions of the state space that should be visited. An example has been used to illustrate the fact that the presence of controlled invariants is important for the permissiveness and feasibility of the DES control problem. Properties of the procedure were characterized and a sufficient condition for termination was given. Further, one version of the procedure was shown to be semidecidable. The procedure assumes that controlled predecessor sets and controlled invariant sets are computable. Depending on the context, various methods from the literature could be used to calculate the controlled predecessors and invariants, including suboptimal methods.

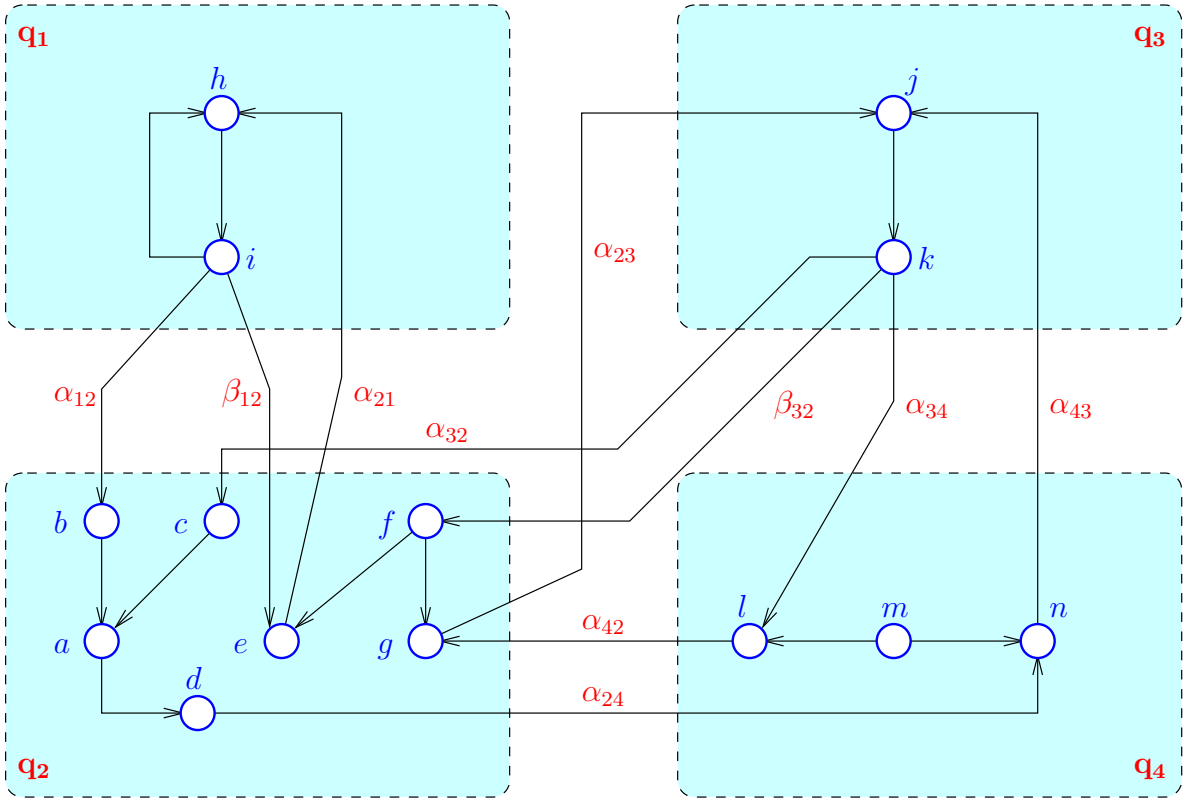


Figure 9: The four modes of the hybrid system.

References

[1] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[2] R. David and H. Alla. Continuous Petri nets. In *8th European Workshop on Application and Theory of Petri Nets*, 1987.

[3] T. Henzinger. Hybrid automata with finite bisimulations. In Z. Füllöp and G. Gécgeg, editors, *ICALP'95: Automata, Languages, and Programming*. Springer-Verlag, 1995.

[4] Y.C. Ho, editor. *Discrete Event Dynamic Systems: Theory and Application,* Special Issue on Hybrid Petri Nets, volume 11(1/2). Kluwer Academic Press, 2001.

[5] M. V. Iordache and P. J. Antsaklis. Supervision based on place invariants: A survey. *Discrete Event Dynamic Systems*, 16:451–492, 2006.

[6] M. V. Iordache and P. J. Antsaklis. *Supervisory Control of Concurrent Systems: A Petri Net Structural Approach*. Birkhäuser, 2006.

[7] X. Koutsoukos and P. Antsaklis. Safety and reachability of piecewise linear hybrid dynamical systems based on discrete abstractions. *Discrete Event Dynamic Systems*, 13(3):203–243, 2003.

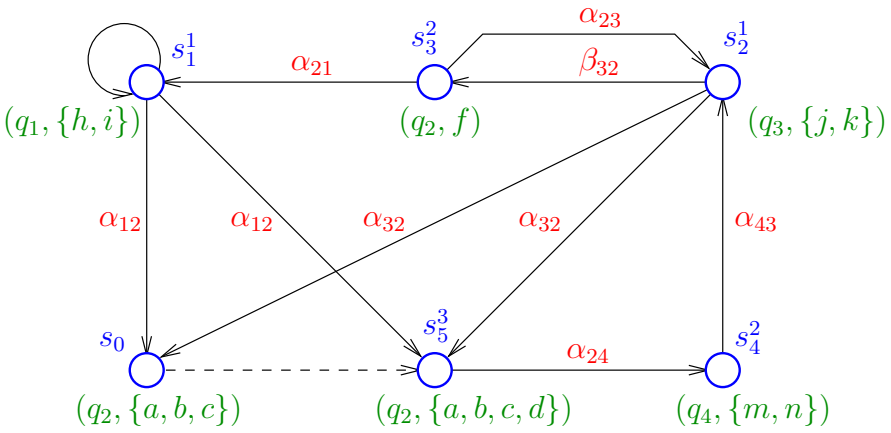


Figure 10: Abstraction produced by the AB-procedure.

[8] X.D. Koutsoukos, P.J. Antsaklis, J.A. Stiver, and M.D. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*, pages 1026–1049, July 2000.

[9] G. Lafferriere, G. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13:1–21, 2000.

[10] M.D. Lemmon, K.X. He, and C.J. Bett. Modeling Hybrid Control Systems Using Programmable Timed Petri Nets. In *3rd International Conference ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems, (ADPM'98)*, pages 177–184, Rheims, France, March 1998.

[11] J. Lunze. A Petri-net approach to qualitative modelling of continuous dynamical systems. *Systems Analysis, Modelling, Simulation*, 9:89–111, 1992.

[12] T. Moor, J.M. Davoren, and J. Raisch. Strategic refinements in abstraction based supervisory control of hybrid systems. In *Proceedings of the 6th International Workshop on Discrete Event Systems*, pages 329–334, 2002.

[13] J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 4(43):568–573, 1998.

[14] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[15] J.A. Stiver, P.J. Antsaklis, and M.D. Lemmon. A logical des approach to the design of hybrid control systems. *Mathematical and Computer Modelling*, pages 55–76, 1996.

[16] P. Tabuada and G. Pappas. Model checking LTL over controllable linear systems is decidable. In *Hybrid Systems: Computation and Control*, volume 2623 of *LNCS*, pages 498–513. Springer-Verlag, 2003.

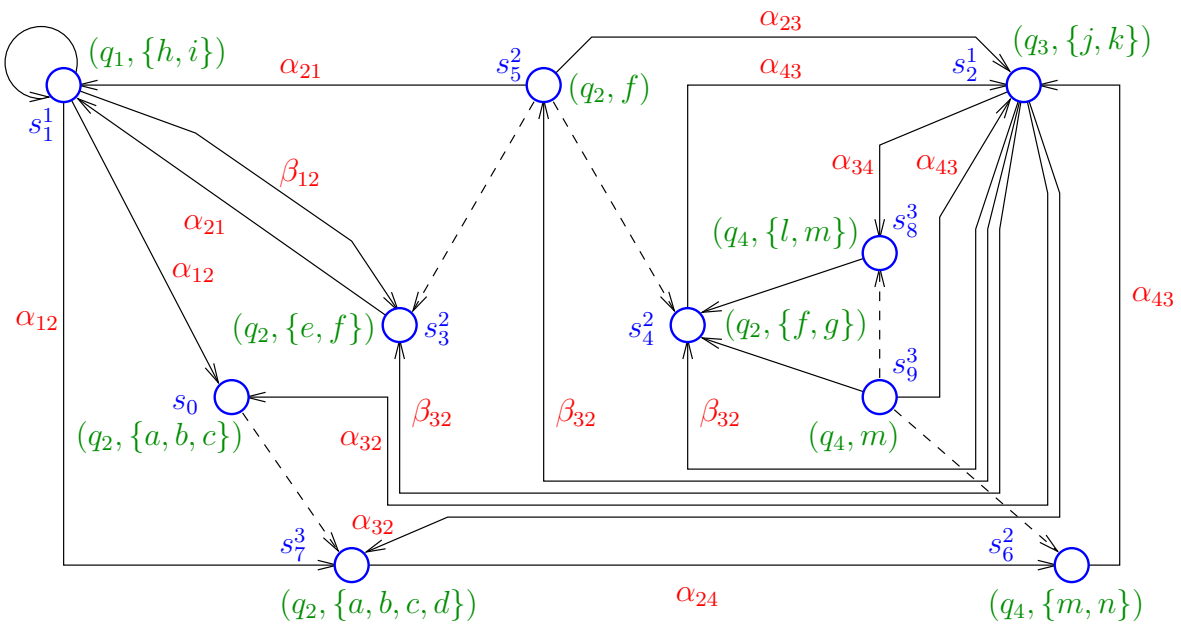


Figure 11: Abstraction produced by the AB-L procedure.

Table 4: In summary, the AB-procedure creates the abstraction of Figure 11 as follows.

Initialization	Add s_0 with $C(s_0) = q_2$ and $Inv(s_0) = Pre(J) = \{a, b, c\}$.
Iteration 1	$O_{q_1 \to s_0} = \{i\}$, add s_1^1 with $C(s_1^1) = q_1$ and $Inv(s_1^1) = Pre(O_{q_1 \to s_0}) = \{h, i\}$.
	$O_{q_3 \to s_0} = \{k\}$, add s_2^1 with $C(s_2^1) = q_3$ and $Inv(s_2^1) = Pre(O_{q_3 \to s_0}) = \{j, k\}$.
Iteration 2	$O_{q_2 \to s_1^1} = \{e\}$, $Pre(O_{q_2 \to s_1^1}) = \{e, f\}$, $O_{q_2 \to s_2^1} = \{g\}$, and $Pre(O_{q_2 \to s_2^1}) = \{f, g\}$. Add s_3^2 with $C(s_3^2) = q_2$ and $Inv(s_3^2) = Pre(O_{q_2 \to s_1^1}) = \{e, f\}$. Add s_4^2 with $C(s_4^2) = q_2$ and $Inv(s_4^2) = Pre(O_{q_2 \to s_2^1}) = \{f, g\}$. Add s_5^2 with $C(s_5^2) = q_2$ and $Inv(s_5^2) = Pre(O_{q_2 \to s_1^1}) \cap Pre(O_{q_2 \to s_2^1}) = \{f\}$.
	$O_{q_4 \to s_2^1} = \{n\}$, add s_6^2 with $C(s_6^2) = q_4$ and $Inv(s_6^2) = Pre(O_{q_4 \to s_2^1}) = \{m, n\}$.
Iteration 3	$O_{q_2 \to s_6^2} = \{d\}$, add s_7^3 with $C(s_7^3) = q_2$ and $Inv(s_7^3) = Pre(O_{q_2 \to s_6^2}) = \{a, b, c, d\}$.
	$O_{q_4 \to s_4^2} = \{l\}$, add s_8^3 with $C(s_8^3) = q_4$ and $Inv(s_8^3) = Pre(O_{q_4 \to s_4^2}) = \{l, m\}$. Add s_9^3 with $C(s_9^3) = q_4$ and $Inv(s_9^3) = Pre(O_{q_4 \to s_4^2}) \cap Pre(O_{q_4 \to s_2^1}) = \{m\}$.
Iteration 4	$Pre(O_{q_1 \to s_7^3}) = Inv(s_1^1)$, $Pre(O_{q_3 \to s_7^3}) = Inv(s_2^1)$, $O_{q_4 \to s_7^3} = \emptyset$, $O_{q_2 \to s_8^3} = \emptyset$, $Pre(O_{q_3 \to s_8^3}) = Inv(s_2^1)$, $O_{q_2 \to s_9^3} = \emptyset$, and $O_{q_3 \to s_9^3} = \emptyset$, so no new state s is added.