Consistent Reduction in Discrete-Event Systems

Kai Cai [a], Alessandro Giua [b], Carla Seatzu [b]

[a] *Department of Electrical and Information Engineering, Osaka City University, Japan*

[b] *Department of Electrical and Electronic Engineering, University of Cagliari, Italy*

Abstract

In this paper we develop a general framework, called "consistent reduction", for formalizing and solving a class of state minimization/reduction problems in discrete-event systems. Given an arbitrary finite-state automaton and a cover on its state set, we propose a consistent reduction procedure that generates a reduced automaton, preserving certain special properties of the original automaton. The key concept of the consistent reduction procedure is the *dynamically consistent cover*; in each cell of this cover, any two states, as well as their future states reached by the same system trajectories, satisfy the binary relation induced from the given cover. We propose a new algorithm that computes a dynamically consistent cover that refines a given cover. We demonstrate the developed general framework on state reduction problems in different application areas.

Published as:

K. Cai, A. Giua, C. Seatzu, "Consistent reduction in discrete-event systems," *Automatica*, Vol. 142, 2022. DOI: 10.1016/j.automatica.2022.110333

⋆ This work was supported in part by the JSPS KAKENHI Grant no. 21H04875, the JSPS Invitational Fellowships no. S19091, the Visiting Professor Program 2017 of the University of Cagliari, and Region Sardinia (RAS) with project MOSIMA, RASSR05871, FSC 2014-2020, Annualita' 2017, Area Tematica 3, Linea d'Azione 3.1.

Email addresses: kai.cai@eng.osaka-cu.ac.jp (Kai Cai), giua@unica.it (Alessandro Giua), carla.seatzu@unica.it (Carla Seatzu).

1 Introduction

A fundamental notion in the theory of computation is that of *minimal automaton* [Hopcroft et al., 2006], i.e., a deterministic finite automaton $\mathbf{G}$ with a minimal number of states which accepts the same language accepted by a given automaton $\mathbf{H}$. Defining an equivalence relation $\approx$, called *congruence*, on the state space of $\mathbf{H}$ — such that two states are congruent if the languages accepted starting from each of them are identical — it is well known that the minimal automaton $\mathbf{G}$ is exactly the quotient $\mathbf{H}/\approx$.

In more general terms one can describe this problem as follows: one aims to classify the infinite set of strings on a given alphabet into two classes (accepted or not accepted), and a finite structure, e.g., an automaton, is used to do the classification. Introducing a suitable equivalence relation on the state set of the automaton allows one to solve the problem in an optimal fashion, i.e., with a minimal structure.

In more recent years, the above approach has been generalized to the setting of *state transition systems*. In this case several types of *bisimulation relation* on the state set have been defined to capture interesting features of the system trajectories [Milner, 1989, Alur et al., 2000, Tabuada, 2009]. Since bisimulation is also an equivalence relation, a minimal quotient system may be constructed. We note that bisimulation has also been used as an abstraction method in discrete-event systems (DES) [Mohajerani et al., 2014, Su et al., 2010].

There are other problems where the minimization (or at least reduction) of the number of states plays a fundamental role in DES. One such problem is *supervisor localization* [Cai and Wonham, 2016] (as well as its parent problem *supervisor reduction* [Su and Wonham, 2004, Su and Wonham, 2018]). Another (perhaps less familiar) problem is *model identification*, i.e., inferring the structure of a deterministic finite automaton $\mathbf{G}$ from examples and counterexamples of its accepted language $L_m(\mathbf{G})$ [Oncina and Garcia, 1992, Bugalho and Oliveira, 2005, Verwer et al., 2006]. The latter problem can be described as follows. Given a set of strings S_+ that must be accepted and a set of strings S_- that must not be accepted, a *prefix tree acceptor* $\mathbf{H}$ (a loopless deterministic finite automaton) is first constructed. In $\mathbf{H}$ states reached by accepted strings are labeled "A" (these are the accepting states), states corresponding to non-accepted strings are labeled "N", and all other "don't care" states are not labeled. Then one tries to reduce the number of states of $\mathbf{H}$ by merging state pairs that are *consistent*: i.e., it is possible to merge a state labeled "A" (resp., "N") with either a state "A" (resp., "N") or a "don't care" state; it is also possible to merge two "don't care" states. By this state merging (and determinization), the reduced automaton $\mathbf{G}$ is ensured to accept a language $L_m(\mathbf{G})$ that satisfies $S_+ \subseteq L_m(\mathbf{G})$ and $S_- \cap L_m(\mathbf{G}) = \emptyset$, namely all strings in S_+ are accepted while no string in S_- is accepted.

Although the problems and approaches mentioned above may appear rather different, we point out a similarity among them. They can all be described in terms of a classification problem of the infinite set of strings, solved by means of the finite structure of an automaton. One tries to minimize/reduce the size of the automaton by aggregating states that are pairwise consistent under a suitable relation that depends on the considered classification problem. However, while the Myhill-Nerode and the bisimulation approaches consider *equivalence relations*, in model identification and supervisor reduction/localization the relations are not *transitive*. In model identification, a state x_1

labeled "A" (accepting state) can be merged with a "don't care" state x_2, and x_2 can be merged with another state x_3 labeled "N" (non-accepting state), but x_1 and x_3 cannot be merged. The non-transitive relation in supervisor reduction/localization is similar.

In this paper we develop a general framework, called "consistent reduction", for formalizing and solving a large class of minimization/reduction problems in DES. In particular, this framework includes minimal automaton, minimal bisimulation, model identification, and supervisor localization/reduction as special cases. Concretely, the consistent reduction problem starts with an arbitrary deterministic finite automaton and a (non-redundant) cover on the state set of that automaton (every cell containing at least one distinct state). Note that such a cover naturally induces a (binary) relation on the state set: two states are related if there exists a cell of the cover to which they both belong. Then we present a consistent reduction procedure that generates a reduced automaton, achieving *consistency* in a certain sense with the original automaton and cover. The key concept of the consistent reduction procedure is *dynamically consistent cover*: in each cell of this special cover, any two states, as well as their future states reached by the same system trajectories, satisfy the induced relation. We design a new algorithm to compute a dynamically consistent cover that is *finer* than the given cover.

The main contributions of our work are as follows.

(i) We establish a unified framework for formulating and solving state minimization/reduction problems in DES. Any problem that can be formulated in terms of an automaton and a cover on its state set may be cast into our formulation including a set of existing minimization/reduction problems as special cases. Moreover, within this general framework, it is possible to define interesting new problems in DES that find immediate solutions in this framework. For example, in state estimation of an automaton $\mathbf{G}$ we may consider the problem of localizing a monolithic observer with respect to a particular subset $\bar{Q}$ of states in $\mathbf{G}$, with the purpose of distinguishing three cases: (a) $\mathbf{G}$ is in $\bar{Q}$; (b) $\mathbf{G}$ is not in $\bar{Q}$; (c) $\mathbf{G}$ may be in $\bar{Q}$ (Section 5.2).

(ii) To solve our formulated consistent reduction problem, a key issue is how to compute a dynamically consistent cover *refining* a given cover. We design a new algorithm that effectively does so, and prove that the complexity is $O(n^4)$, where n is the number of states of the given automaton. Note that the supervisor reduction algorithm in [Su and Wonham, 2004], designed to solve a special instance of the consistent reduction problem, generally fails to solve our problem (illustrated by Example 12). This is a consequence of that algorithm's lack of a mechanism to account for the initial given cover, and hence the generated partition need not refine the given cover in general.

(iii) Our new algorithm's capability of processing and generating covers can potentially achieve further state reduction than the partition-based algorithm in [Su and Wonham, 2004], for problems where both algorithms can solve. In particular, there exist cases where *minimal* state reduction cannot be achieved by the algorithm in [Su and Wonham, 2004] because it is based on partitions, but can be achieved by our cover-based algorithm. An illustration is given in Example 13.

(iv) Moreover, since our designed algorithm generates a cover, different reduced automata may be obtained. This provides an additional degree of flexibility as compared to the supervisor reduction algorithm in [Su and Wonham,

2004] which can generate just a single partition and thus a single reduced automaton. To corroborate this claim, we discuss in Example 14 a case where the flexibility may allow one to select reduced automata with many selfloop events. A reduced automaton with this property can save observation/communication costs since selfloop events causing no state changes do not need to be observed or communicated: this can be particularly beneficial for designing communication resource aware control and estimation schemes in networked distributed DES.

Preliminary results concerning our consistent reduction approach are presented in [Cai et al., 2019]. In this paper, besides a thorough reformulation of the problem, we propose a new algorithm that solves the consistent reduction problem in all generality. In addition, we provide a new section with different application areas to demonstrate the generality of our framework.

The remainder of this paper is organized as follows. In Section 2 we provide preliminaries on binary relations and covers, and in Section 3 formulation of the consistent reduction problem. In Section 4 we present a consistent reduction procedure that solves the formulated problem, and an algorithm is designed to compute a dynamically consistent cover. In Section 5 we illustrate our consistent reduction solutions with different application areas, and provide detailed comparisons with [Su and Wonham, 2004]. Finally in Section 6 we state our conclusions and point out our future lines of research in this framework.

2 Binary relations and covers

In this section we provide some background on binary relations and covers, relevant to the rest of the paper.

2.1 Binary relations

Given a set X, $\mathcal{R} \subseteq X \times X$ is a *binary relation* on X, and we write $(x, x') \in \mathcal{R}$ to denote that x is related with x'.

In the following we will consider binary relations satisfying the following two properties:

(i) $(\forall x \in X)\,(x, x) \in \mathcal{R}$ (reflexivity)
(ii) $(\forall x, x' \in X)\,(x, x') \in \mathcal{R} \Rightarrow (x', x) \in \mathcal{R}$ (symmetry).

Given $\mathcal{R} \subseteq X \times X$, the *reflexive and symmetric closure* of $\mathcal{R}$ is $\tilde{\mathcal{R}} := \mathcal{R} \cup \{(x, x) \mid x \in X\} \cup \{(x', x) \mid (x, x') \in \mathcal{R}\}$.

We say that $\mathcal{R}$ is an *equivalence relation* if it is reflexive, symmetric, and transitive, namely it satisfies properties (i) and (ii), plus the following additional property:

(iii) $(\forall x, x', x'' \in X)(x, x') \in \mathcal{R}\ \&\ (x', x'') \in \mathcal{R} \Rightarrow (x, x'') \in \mathcal{R}$ (transitivity).

2.2 Covers

Definition 1 *Given a set X and an index set I, an* (I-) cover *on X is a set $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ such that (i) $(\forall i \in I)X_i \neq \emptyset$ and (ii) $\bigcup_{i \in I} X_i = X$. Each X_i is referred to as a* cell *of $\mathcal{C}$.*

Given a cover $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$, we define the associated *indicator* function $O : X \to Pwr(I)$ (Pwr denotes powerset) by $O(x) = \{i \in I \mid x \in X_i\}$. Thus $O(x)$ is the set of indices for the cells that x belongs to. Note that $O(x) \neq \emptyset$ for all $x \in X$, and $O(x)$ is not a singleton set iff x is shared by two or more cells.

We point out that a cover $\mathcal{C}$ on a set X can be viewed as a multi-criterion classifier for the elements in X, which assigns (via the indicator function O) to each $x \in X$ those criteria in I that x satisfies.

Example 2 *Consider a set* $X = \{x_1, x_2, x_3, x_4\}$*, an index set* $I = \{1, 2, 3\}$*, and a cover* $\mathcal{C} = \{X_1 = \{x_1, x_2\}, X_2 = \{x_2, x_4\}, X_3 = \{x_3, x_4\}\}$*. Thus one obtains* $O(x_1) = \{1\}$*,* $O(x_2) = \{1, 2\}$*,* $O(x_3) = \{3\}$*, and* $O(x_4) = \{2, 3\}$*.* ■

Definition 3 *A cover* $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ *is* non-redundant *if* $(\forall i \in I)\ (\exists x \in X_i)\ (\forall j \neq i)\ x \notin X_j$*, namely each cell contains at least one element of* X *that does not belong to other cells.*

Note that in a non-redundant cover removing any cell destroys the covering property $\bigcup_{i \in I} X_i = X$. It follows that if the cover $\mathcal{C}$ is non-redundant, then $|I| \leq |X|$.

A cover $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ is a *partition* on X if $(\forall i, j \in I)\ i \neq j \Rightarrow X_i \cap X_j = \emptyset$. Thus a partition is a special cover that has pairwise disjoint cells. Given two covers $\mathcal{C}_1 = \{X_{i_1} \subseteq X \mid i_1 \in I_1\}$, $\mathcal{C}_2 = \{X_{i_2} \subseteq X \mid i_2 \in I_2\}$ on X, we say that $\mathcal{C}_2$ *refines* $\mathcal{C}_1$ (written $\mathcal{C}_1 \succcurlyeq \mathcal{C}_2$) if $(\forall i_2 \in I_2)(\exists i_1 \in I_1) X_{i_2} \subseteq X_{i_1}$.

2.3 $\mathcal{R}$-consistent covers

Definition 4 *Given a reflexive and symmetric binary relation* $\mathcal{R} \subseteq X \times X$ *we say that a cover* $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ *of* X *is* $\mathcal{R}$-consistent *if*

$$(\forall i \in I)(\forall x, x' \in X_i)(x, x') \in \mathcal{R} \tag{1}$$

i.e., all elements that belong to the same cell are pairwise related with respect to $\mathcal{R}$*.*

An $\mathcal{R}$*-consistent cover is called:*

- minimal *if there does not exist another* $\mathcal{R}$*-consistent cover* $\mathcal{C}'$ *on* X *with a number of cells* $|I'| < |I|$*;*
- complete *if* $(\forall i \in I)\ (\forall x' \in X \setminus X_i)\ (\exists x \in X_i)\ (x, x') \notin \mathcal{R}$*, i.e., one cannot add a new element to any cell without destroying the* $\mathcal{R}$*-consistency.*

It can be easily shown that a minimal $\mathcal{R}$-consistent cover is also non-redundant, while the opposite is not necessarily true. In addition we remark that given a $\mathcal{R}$-consistent cover $\mathcal{C}$ one can always, by iteratively adding new elements to cells, obtain a complete cover $\mathcal{C}'$: if $\mathcal{C}$ is minimal then $\mathcal{C}'$ is also minimal.

Given a non-redundant cover $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ on X, one may define an *induced* relation

$$\mathcal{R}_{\mathcal{C}} = \{(x, x') \mid (\exists i \in I) x, x' \in X_i\}. \tag{2}$$

Note that $\mathcal{C}$ is a complete $\mathcal{R}_{\mathcal{C}}$-consistent cover. To see this, suppose that there exist $X_i \in \mathcal{C}$ and $x' \in X \setminus X_i$ such that for every $x \in X_i$ it holds that $(x, x') \in \mathcal{R}_{\mathcal{C}}$. Then every $x \in X_i$ belongs to another cell X_j for some $j \in I$, which means that X_i is redundant. This contradicts the assumption that $\mathcal{C}$ is non-redundant.

Among the many possible $\mathcal{R}$-consistent covers, a particularly interesting problem is that of determining a minimal cover. This problem does not admit a unique solution and furthermore it is NP-hard. To show this, we observe that with a reflexive and symmetric binary relation $\mathcal{R} \subseteq X \times X$ one can (univocally) associate an undirected graph $\mathcal{G}$. The set of vertices of $\mathcal{G}$ coincides with the set X and, given two vertices x and x', with $x \neq x'$, an edge connecting x and x' exists if and only if $(x, x') \in \mathcal{R}$ (for simplicity, we neglect self-loops originating from the reflexivity property). Given an undirected graph, a clique is a subset of vertices such that its induced subgraph is complete, i.e., any two distinct vertices in the clique are connected by an edge. The problem of determining a minimal $\mathcal{R}$-consistent cover is thus equivalent to the problem of determining a minimum clique cover, a problem that has been shown NP-hard [Karp, 1972].

Example 5 *Consider a set* $X = \{x_1, x_2, x_3, x_4, x_5\}$ *and a binary relation* $\mathcal{R}$ *that is the reflexive and symmetric closure of* $\{(x_1, x_2), (x_1, x_4), (x_2, x_3), (x_2, x_5), (x_4, x_5), (x_3, x_5)\}$. *Relation* $\mathcal{R}$ *is clearly not transitive. Indeed,* (x_1, x_2), $(x_1, x_4) \in \mathcal{R}$ *while* $(x_2, x_4) \notin \mathcal{R}$.

Cover $\mathcal{C}_1 = \{\{x_1, x_4\}, \{x_2, x_3, x_5\}\}$ *is minimal, non-redundant, and complete. Another non-redundant and complete cover is* $\mathcal{C}_2 = \{\{x_1, x_2\}, \{x_2, x_3, x_5\}, \{x_4, x_5\}\}$ *but it is not minimal. Cover* $\mathcal{C}_3 = \{\{x_1, x_4\}, \{x_1, x_2\}, \{x_3, x_5\}\}$ *is still non-redundant, but it is neither complete nor minimal. Finally, cover* $\mathcal{C}_4 = \{\{x_1, x_4\}, \{x_1, x_2\}, \{x_2, x_3, x_5\}\}$ *is complete, but neither minimal nor non-redundant. Indeed, cell* $\{x_1, x_2\}$ *can be removed without destroying the covering property.* ■

In the particular case in which $\mathcal{R}$ is an equivalence relation on X, there exists a unique non-redundant complete $\mathcal{R}$-consistent cover that is also minimal: this is the partition of the set X into equivalence classes for relation $\mathcal{R}$.

3 Problem Formulation

Let $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$ be a finite-state *automaton*, where X is a finite state set, $x_0 \in X$ an initial state, $X_m \subseteq X$ a set of marker states [1], Σ a finite event set, and $\xi : X \times \Sigma \to X$ the (partial) state transition function. In the usual way, ξ is extended to $\xi : X \times \Sigma^* \to X$, and we write $\xi(x, s)!$ to mean that $\xi(x, s)$ is defined. Finally, given a set $X' \subseteq X$, define $\xi(X', \sigma) := \bigcup_{x' \in X'} \xi(x', \sigma)$.

Let $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ be a non-redundant cover on X with the associated indicator function $O : X \to Pwr(I)$. Recall that $\mathcal{C}$ is a complete $\mathcal{R}_{\mathcal{C}}$-consistent cover for the induced relation $\mathcal{R}_{\mathcal{C}}$ defined in (2).

[1] The explicit inclusion of X_m in the automaton definition is not necessary, because the partition $\{X_m, X \setminus X_m\}$ can be viewed as a classification of states/strings in addition to a given cover. However, since X_m is commonly used in classical problems of automaton minimization and supervisory control, we choose to include X_m in the definition.

Our goal is to construct a 'reduction' of $\mathbf{H}$, say $\mathbf{G}$ with an associated indicator function P which provides a classification of strings generated by $\mathbf{H}$ consistent with that provided by O based on $\mathbf{H}$. Formally, we state our problem as follows:

Consistent Reduction Problem (CRP): Given an automaton $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$, a non-redundant cover $\mathcal{C}$ on X and the associated indicator function O, construct a reduced automaton $\mathbf{G} = (Y, \Sigma, \eta, y_0, Y_m)$ and an associated indicator function $P : Y \to Pwr(I)$ such that the following three properties hold:

$$|Y| \leq |X| \tag{3}$$

$$(\forall s \in \Sigma^*)\xi(x_0, s)! \Rightarrow \eta(y_0, s)! \tag{4}$$

$$(\forall s \in \Sigma^*)\xi(x_0, s)! \Rightarrow \emptyset \neq P(\eta(y_0, s)) \subseteq O(\xi(x_0, s)). \tag{5}$$

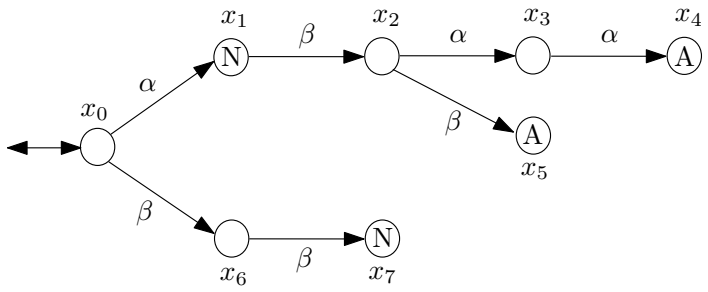
Condition (3) requires that the state size of $\mathbf{G}$ be smaller than (or at worst equal to) that of $\mathbf{H}$. Indeed, for practical concern, one would often expect that $|Y|$ be much smaller than $|X|$. Condition (4) requires that every string that can be generated by (and classified in) $\mathbf{H}$ can also be generated by (and classified in) $\mathbf{G}$; namely the generated language of $\mathbf{G}$ is larger than or equal to that of $\mathbf{H}$. Finally, condition (5) means that for any string s generated by $\mathbf{H}$, the classification of the state $\eta(y_0, s)$ by P is nonempty and moreover is a subset of the classification of $\xi(x_0, s)$ by O. Thus if the state $\xi(x_0, s)$ belongs to a unique cell of $\mathcal{C}$ under O (satisfying a single criterion), the corresponding state $\eta(y_0, s)$ will still belong to that unique cell under P; while if $\xi(x_0, s)$ belongs to multiple cells of $\mathcal{C}$ under O (satisfying multiple criteria), $\eta(y_0, s)$ will belong to a (nonempty) subset of these cells under P. In exactly this sense we say that P based on $\mathbf{G}$ provides a *consistent* classification with O based on $\mathbf{H}$.

Example 6 *To illustrate the formulated CRP, we provide an example of model identification, adapted from [Verwer et al., 2006]. Given a positive sample $S_+ = \{\alpha\beta\beta, \alpha\beta\alpha\alpha\}$ and a negative sample $S_- = \{\alpha, \beta\beta\}$ on alphabet $\Sigma = \{\alpha, \beta\}$, consider the prefix tree acceptor $\mathbf{H}$ displayed in Fig. 1: states x_4, x_5 corresponding to accepted strings in S_+ are labeled "*A*", states x_1, x_7 corresponding to non-accepted strings in S_- are labeled "*N*", and all other "don't care" states are not labeled. Define $A : X \to \{0, 1\}$ by $A(x) = 1$ if and only if x is labeled "*A*". Also define $N : X \to \{0, 1\}$ by So $N(x) = 1$ if and only if x is labeled "*N*".*

*Now consider a non-redundant cover $\mathcal{C}$ with just two cells: $\mathcal{C} = \{X_1, X_2\}$, where $X_1 = \{x \in X \mid A(x) = 0\}$ and $X_2 = \{x \in X \mid N(x) = 0\}$. Here X_1 is the subset of states which are either labeled "*N*" or not labeled, while X_2 is the subset of states which are either labeled "*A*" or not labeled. The associated indicator function $O : X \to Pwr(\{1, 2\})$ is given by*

$$O(x) = \begin{cases} \{1\}, & \text{if } A(x) = 1 \\ \{2\}, & \text{if } N(x) = 1 \\ \{1, 2\}, & \text{if } N(x) = A(x) = 0. \end{cases}$$

Thus for the prefix tree acceptor $\mathbf{H}$ in Fig. 1 we have $O(x_4) = O(x_5) = \{1\}$, $O(x_1) = O(x_7) = \{2\}$ and $O(x_0) = O(x_2) = O(x_3) = O(x_6) = \{1, 2\}$.



H

Fig. 1. The prefix tree acceptor **H** for positive samples $S_+ = \{\alpha\beta\beta, \alpha\beta\alpha\alpha\}$ and negative samples $S_- = \{\alpha, \beta\beta\}$. States x_4, x_5 corresponding to accepted strings in S_+ are labeled "A", states x_1, x_7 corresponding to non-accepted strings in S_- are labeled "N", and all other "non-determined" states are not labeled.

The model identification problem aims to obtain a reduced automaton $\mathbf{G}$ *such that every string classified in* $\mathbf{H}$ *can be consistently classified in* $\mathbf{G}$*. This problem falls under the formulation of CRP, which is to construct a reduced automaton* $\mathbf{G}$ *and an associated indicator function* P *such that the properties (3)-(5) are satisfied. We will present such a* $\mathbf{G}$ *in Fig. 2 and the associated indicator function* P *in Example 10 in the next section (p.6).* ■

4 Consistent Reduction Procedure

In this section we present a procedure that generates a solution to CRP. The key concept is that of *dynamically consistent cover*, based on which consistent reduction is carried out.

4.1 Dynamically consistent cover

Given an automaton $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$ and a binary relation $\mathcal{R}$ on X, we introduce the following key concept.

Definition 7 *Let* $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$ *be an automaton,* $\mathcal{R}$ *be a reflexive and symmetric binary relation on the state set* X *of* $\mathbf{H}$*, and* $\mathcal{C}_{dyn} = \{X_a \subseteq X \mid a \in A\}$ *(*A *is some finite index set) be a non-redundant cover on* X*. We say that cover* $\mathcal{C}_{dyn}$ *is* dynamically consistent with $\mathbf{H}$ and $\mathcal{R}$ *, or for short an* $(\mathbf{H}, \mathcal{R})$-consistent cover*, if the following two conditions hold:*

$$\text{(i) } (\forall a \in A)(\forall x, x' \in X)[x, x' \in X_a \Rightarrow (x, x') \in \mathcal{R}]$$
$$\text{(ii) } (\forall a \in A)\ (\forall \sigma \in \Sigma)\ (\exists a' \in A)\ \xi(X_a, \sigma) \subseteq X_{a'}.$$

In this definition, condition (i) requires that all states in the same cell of $\mathcal{C}_{dyn}$ be pairwise consistent with respect to $\mathcal{R}$ (i.e. $\mathcal{C}_{dyn}$ is an $\mathcal{R}$-consistent cover as in Definition 4). Condition (ii) can be paraphrased as follows: all states that can be reached from any states in some cell X_a by the same one-step transition σ must belong to some cell $X_{a'}$. Inductively, any two states in the same cell of $\mathcal{C}_{dyn}$ are consistent with respect to $\mathcal{R}$, and all their future states reached by the same strings are also consistent with respect to $\mathcal{R}$. Hence $\mathcal{C}_{dyn}$ 'respects' the dynamics of $\mathbf{H}$, and for

this reason we call $\mathcal{C}_{dyn}$ *dynamically consistent cover*. In the case where the cells of $\mathcal{C}_{dyn}$ are pairwise disjoint, we call $\mathcal{C}_{dyn}$ a *dynamically consistent partition* on X.

Example 8 *For the example displayed in Fig. 1, the cover* $\mathcal{C} = \{\{x_0, x_2, x_3, x_4, x_5, x_6\}, \{x_0, x_1, x_2, x_3, x_6, x_7\}\} = \{X_1, X_2\}$ *is not dynamically consistent because condition (ii) is violated. Indeed, for the pair* (x_2, x_6) *in the same cell (*X_1 *or* X_2*) there hold* $\xi(x_2, b)!$ *and* $\xi(x_6, b)!$*, but* $\xi(x_2, b), \xi(x_3, b)$ *are not covered by the same cell:* $\xi(x_2, b) \in X_1$*,* $\xi(x_6, b) \in X_2$*. Another pair that violates condition (ii) is* (x_0, x_3)*. Refine* $\mathcal{C}$ *to*

$$\begin{aligned}\mathcal{C}_{dyn} &= \{\{x_0, x_4, x_5\}, \{x_4, x_5, x_6\}, \{x_1, x_3, x_7\}, \{x_2, x_7\}\} \\ &= \{X_1, X_2, X_3, X_4\}\end{aligned}$$

and one may check that (ii) is satisfied and $\mathcal{C}_{dyn}$ *is a (non-redundant) dynamically consistent cover.* ■

Given an automaton $\mathbf{H}$ with state set X and a non-redundant cover $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ on X, let $\mathcal{R}_{\mathcal{C}}$ be the induced relation as defined in (2). We present in the next subsection an algorithm that computes a non-redundant $(\mathbf{H}, \mathcal{R}_{\mathcal{C}})$-consistent cover $\mathcal{C}_{dyn} = \{X_a \subseteq X \mid a \in A\}$ which refines $\mathcal{C}$, i.e., $(\forall a \in A)(\exists i \in I) X_a \subseteq X_i$.

Having computed such an $(\mathbf{H}, \mathcal{R}_{\mathcal{C}})$-consistent cover $\mathcal{C}_{dyn} = \{X_a \subseteq X \mid a \in A\}$ on X, we construct a reduced version of $\mathbf{H}$,

$$\mathbf{G} = (Y, \Sigma, \eta, y_0, Y_m) \tag{6}$$

as follows:

(i) $Y := A$;

(ii) Let $A_0 := \{a \in A \mid x_0 \in X_a\}$. Pick $a_0 \in A_0$ and let $y_0 := a_0$.

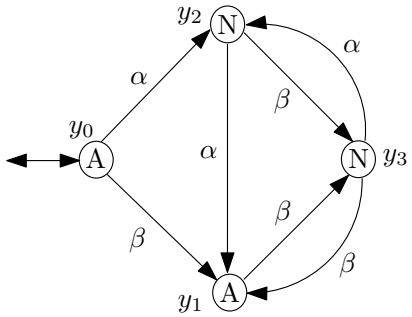
(iii) $Y_m := \{a \in A \mid X_a \cap X_m \neq \emptyset\}$;

(iv) For all $a \in Y$ and $\sigma \in \Sigma$, with $\xi(X_a, \sigma) \neq \emptyset$, let

$$\mathcal{T}(a, \sigma) := \{a' \in A \mid \xi(X_a, \sigma) \subseteq X_{a'}\}.$$

Pick $a' \in \mathcal{T}(a, \sigma)$ and let $\eta(a, \sigma) = a'$.

In words, the state set Y of $\mathbf{G}$ is simply the index set of $\mathcal{C}_{dyn}$; the initial state y_0 is the index of a cell where the initial state x_0 belongs; the subset Y_m of marker states is the subset of indices of those cells containing some marker states in X_m; and finally a transition $\eta(a, \sigma) = a'$ is defined if there exists a state x in the cell labeled a transitions on the event σ to another state x' in the cell labeled a', whenever σ is defined at x via ξ. The transition function $\eta : Y \times \Sigma \to Y$ in (iv) is well-defined owing to condition (ii) of $(\mathbf{H}, \mathcal{R}_{\mathcal{C}})$-consistent cover (in Definition 7); hence $\mathbf{G}$ is a deterministic finite automaton. Note that, due to covering, the choices for y_0 and η are non-unique in general; in that case we pick an arbitrary instance of y_0 and of η (by picking a' as in (iv) above), respectively. If $\mathcal{C}_{dyn}$ is a



G

Fig. 2. Reduced automaton **G** for the model identification problem in Example 6, accepting $S_+ = \{\alpha\beta\beta, \alpha\beta\alpha\alpha\}$ and rejecting $S_- = \{\alpha, \beta\beta\}$

partition on X, then y_0 and η can be uniquely determined. The indicator function $P : Y \to Pwr(I)$ associated with **G** is defined by

$$P(y) = \bigcap_{x \in X_y} O(x). \tag{7}$$

The main result of this subsection is stated below.

Theorem 9 *The reduced automaton* **G** *in (6) and the associated indicator function* P *in (7) solve CRP, i.e. properties (3)-(5) hold.*

Proof. In CRP, we are given an automaton $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$, a non-redundant cover $\mathcal{C} = \{X_i \subseteq X \mid i \in I\}$ and the associated indicator function O. Let $\mathcal{R}_\mathcal{C}$ be the induced relation (as in (2)) and $\mathcal{C}_{dyn} = \{X_a \subseteq X \mid a \in A\}$ a non-redundant $(\mathbf{H}, \mathcal{R}_\mathcal{C})$-consistent cover on X that refines $\mathcal{C}$ (such $\mathcal{C}_{dyn}$ can be computed by an algorithm presented in the next subsection). Referring to (i)-(iv) that define **G** in (6), it follows from (i) $Y = A$ that $|Y| = |A| \leq |X|$. Thus property (3) holds. Next, it follows from η in (iv) by induction that for every $s \in \Sigma^*$, if $\xi(x_0, s)!$ then $\eta(y_0, s)!$. Hence property (4) also holds.

It remains to prove property (5). Let $s \in \Sigma^*$ such that $\xi(x_0, s)!$. It will be shown that $\emptyset \neq P(\eta(y_0, s)) \subseteq O(\xi(x_0, s))$. Since all states x in the cell $X_{\eta(y_0,s)}$ are pairwise consistent with respect to $\mathcal{R}$ and $\mathcal{C}_{dyn}$ is finer than $\mathcal{C}$, $X_{\eta(y_0,s)}$ is a subset of some cell of the original non-redundant cover $\mathcal{C}$ that is complete $\mathcal{R}_\mathcal{C}$-consistent, by (7) it holds that $P(\eta(y_0, s)) = \bigcap_{x \in X_{\eta(y_0,s)}} O(x) \neq \emptyset$. Now let $i \in P(\eta(y_0, s))$. It follows from the definition of η in (iv) that $\xi(x_0, s) \in X_{\eta(y_0,s)}$, and thereby we derive again by (7) that $i \in O(\xi(x_0, s))$. Therefore $P(\eta(y_0, s)) \subseteq O(\xi(x_0, s))$, and property (5) holds. The proof is now complete. □

Example 10 *For the example displayed in Fig. 1, with the (non-redundant) dynamically consistent cover* $\mathcal{C} = \{\{x_0, x_4, x_5\}, \{x_4, x_5, x_6\}, \{x_1, x_3, x_7\}, \{x_2, x_7\}\} = \{X_1, X_2, X_3, X_4\}$ *found in Example 8, we construct by (6) the reduced automaton* **G***, displayed in Fig. 2. The state set of* **G** *is* $Y = \{y_0, y_1, y_2, y_3\}$*, where* $y_0 = 1$*,* $y_1 = 2$*,* $y_2 = 3$*, and* $y_3 = 4$*. Due to covering (*X_1 *and* X_2 *share states* x_4, x_5*;* X_3 *and* X_4 *share states* x_7*),* **G** *is not the unique reduced*

automaton; indeed there are two choices of $\mathbf{G}$*, one as displayed, the other having transition* $\eta(y_1, \beta) = y_2$ *instead of* $\eta(y_1, \beta) = y_3$*. For the displayed* $\mathbf{G}$*, it is easily verified that properties (3) and (4) hold. Then by (7), the associated indicator function* P *is such that*

$$
\begin{aligned}
P(y_0) &= O(x_0) \cap O(x_4) \cap O(x_5) = \{1,2\} \cap \{1\} \cap \{1\} = \{1\} \\
P(y_1) &= O(x_4) \cap O(x_5) \cap O(x_6) = \{1\} \cap \{1\} \cap \{1,2\} = \{1\} \\
P(y_2) &= O(x_1) \cap O(x_3) \cap O(x_7) = \{2\} \cap \{1,2\} \cap \{2\} = \{2\} \\
P(y_3) &= O(x_2) \cap O(x_7) = \{1,2\} \cap \{2\} = \{2\}.
\end{aligned}
$$

Hence property (5) may also be verified to hold. According to acceptance of strings in S_+ *and rejection of strings in* S_-*, we may label* y_0, y_1 *as "*A*" and* y_2, y_3 *as "*N*". Note that the reduced automaton* $\mathbf{G}$ *is an identified model that accepts strings in* $S_+ = \{\alpha\beta\beta, \alpha\beta\alpha\alpha\}$ *and rejects those in* $S_- = \{\alpha, \beta\beta\}$*, which is consistent with* $\mathbf{H}$*.* ■

4.2 Computation of a dynamically consistent cover

In this subsection, we present a new algorithm to compute a non-redundant [2] $(\mathbf{H}, \mathcal{R}_\mathcal{C})$-consistent cover $\mathcal{C}_{dyn}$ which refines a given non-redundant cover $\mathcal{C}$ with the induced relation $\mathcal{R}_\mathcal{C}$.

The main idea is to split a cell for which the condition (ii) of Definition 7 is violated. This happens when for a cell π and an event σ such that $\xi(\pi, \sigma) \neq \emptyset$, there exists no cell that contains $\xi(\pi, \sigma)$ as a subset. In such a case we say that π *is inconsistent with* σ and given an arbitrary π' such that $\xi(\pi, \sigma) \cap \pi' \neq \emptyset$ we call (π, σ, π') a *splitter*.

Consider a cell π inconsistent with σ and one of its splitters (π, σ, π'). Cell π can be partitioned as: $\pi = \pi_\emptyset \cup \pi_{in} \cup \pi_{out}$ where

$$\pi_\emptyset := \{x \in \pi \mid \neg\xi(x, \sigma)!\} \tag{8}$$

$$\pi_{in} := \{x \in \pi \mid \xi(x, \sigma)! \ \&\ \xi(x, \sigma) \in \pi'\} \tag{9}$$

$$\pi_{out} := \{x \in \pi \mid \xi(x, \sigma)! \ \&\ \xi(x, \sigma) \notin \pi'\}. \tag{10}$$

Thus $\pi_\emptyset$ is the subset of π where σ is not defined; π_{in} is the subset where σ is defined and transitions to π'; and π_{out} is the subset where σ is defined but does not transition to π'. As far as condition (i) of Definition 7 (namely $\mathcal{R}_\mathcal{C}$-consistency) is concerned, the first subset $\pi_\emptyset$ can go with both π_{in} and π_{out} without violation so one can split π into two sets — non-disjoint iff $\pi_\emptyset \neq \emptyset$ — both consistent with σ:

$$\pi_1 := \pi_\emptyset \cup \pi_{in}, \qquad \pi_2 := \pi_\emptyset \cup \pi_{out}.$$

Finally, we observe that it is always possible to replace π in $\mathcal{C}$ with either both cells or with just one, so as to obtain an updated non-redundant $\mathcal{R}_\mathcal{C}$-consistent cover that refines the old one.

[2] Non-redundancy is key to ensure state reduction. An algorithm for supervisor reduction that computes a possibly redundant cover is reported in [Minhas, 2002], which however is not guaranteed to achieve state reduction.

Now we are ready to present the algorithm to compute a dynamically consistent cover by refinement. The main idea, given an initial cover $\mathcal{C}$, is to construct the set $\mathcal{V} \subseteq \mathcal{C} \times \Sigma \times \mathcal{C}$ that contains a *single* splitter for each pair (π, σ) such that cell π is inconsistent with event σ. Note that $|\mathcal{V}| \leq |\mathcal{C}| \cdot |\Sigma|$, since a single splitter is considered for each violation of condition (ii) in Definition 7. We then proceed splitting cells that cause a violation.

After each splitting, $\mathcal{C}$ and $\mathcal{V}$ need to be updated. The algorithm halts when $\mathcal{V}$ is empty which means that the corresponding $\mathcal{C}$ is dynamically consistent.

Before we present the algorithm, let us introduce some notation. Given an automaton with state set

$$X = \{x_0, x_1, \ldots, x_{n-1}\}$$

we represent a cell π of a cover by its characteristic vector, i.e., an n-dimensional vector $\text{char}(\pi) := [u_1\ u_2\ \cdots u_n]$ where $u_i = 1$ if $x_{i-1} \in \pi$ else $u_i = 0$. We also write $\text{char}(\emptyset)$ for the n-dimensional 0 vector.

For a cover $\mathcal{C}$, we denote by $U := \sum_{\pi \in \mathcal{C}} \text{char}(\pi)$ the sum of all characteristic vectors of its cells; thus vector U represents the *multiset* [Blizard, 1989] defined by the union of all cells $\pi \in \mathcal{C}$ counting multiplicities of shared elements. Note that $\text{char}(\pi) \leq U$ holds iff π is a subset of the multiset represented by U.

Algorithm 1 is the main procedure which calls three functions described in Fig. 3. We start by describing these functions [3].

Function *splitter* requires input arguments π, σ and π' and determines if (π, σ, π') can be used as a splitter if π is inconsistent with σ. This can be done with a *for* loop that checks all states in π. The function returns: value 1 if $\emptyset \subsetneq \xi(\pi, \sigma) \subseteq \pi'$, thus ensuring π is consistent with σ; value -1 if $\emptyset \subsetneq \xi(\pi, \sigma) \cap \pi' \subsetneq \xi(\pi, \sigma)$, thus showing that (π, σ, π') can be used as a splitter; and value 0 otherwise.

Function *split* requires input arguments π, σ and π' and returns the two subsets (π_1, π_2) obtained by splitting π according to (σ, π').

Function *update* requires input arguments $\mathcal{V}, \mathcal{C}$ and π_{new}. This function returns the updated set of violating conditions $\mathcal{V}$ and the updated cover $\mathcal{C}$ when a new cell π_{new} is added to $\mathcal{C}$. This is done by checking for all cells π' in the updated cover and all symbols σ in the alphabet if a violating condition involving π_{new} exists.

We finally describe the main algorithm. Lines 1-12 initialize vector U and the splitter set $\mathcal{V}$. The while loop at lines 13-32 is executed until there are no more splitters in $\mathcal{V}$. At line 14, a splitter is selected. At lines 15-17, all splitters with π as the first element are removed from $\mathcal{V}$. At line 18, π is removed from cover $\mathcal{C}$. At line 19, vector U is updated. Cell π is split into two cells π_1 and π_2 by calling function *split* at line 20. Line 21 checks if cell π_1 is necessary to obtain a non-redundant cover: if this is the case, sets $\mathcal{V}$ and $\mathcal{C}$ are updated at line 22 while vector U

[3] Note that the functions require the structure of automaton $\mathbf{H}$, because they use its transition function ξ or its alphabet Σ. We assume that ξ and Σ are globally defined.

Algorithm 1. Computation of a dynamically consistent cover

Require: An automaton $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$, a non-redundant cover $\mathcal{C}$ of X.

Ensure: A non-redundant dynamically consistent cover $\mathcal{C}_{dyn}$ that refines $\mathcal{C}$.

$U := \text{char}(\emptyset)$;
for $\pi \in \mathcal{C}$ **do** $U := U + \text{char}(\pi)$;
$\mathcal{V} := \emptyset$;
for $\pi \in \mathcal{C}$ **do**
    for $\sigma \in \Sigma$ **do**
        $\mathcal{V}_{spt} := \emptyset$;
        for $\pi' \in \mathcal{C}$ **do**
            if $\text{splitter}(\pi, \sigma, \pi') = -1$ **then**
                $\mathcal{V}_{spt} := \{(\pi, \sigma, \pi')\}$;
            if $\text{splitter}(\pi, \sigma, \pi') = 1$ **then**
                $\mathcal{V}_{spt} := \emptyset$ and **break**;
        $\mathcal{V} := \mathcal{V} \cup \mathcal{V}_{spt}$;
while $\mathcal{V} \neq \emptyset$ **do**
    Select $(\pi, \sigma, \pi') \in \mathcal{V}$;
    for $(\bar{\pi}, \bar{\sigma}, \bar{\pi}') \in \mathcal{V}$ **do**
        if $\bar{\pi} = \pi$ **then**
            $\mathcal{V} := \mathcal{V} \setminus \{(\bar{\pi}, \bar{\sigma}, \bar{\pi}')\}$;
    $\mathcal{C} := \mathcal{C} \setminus \{\pi\}$;
    $U := U - \text{char}(\pi)$;
    $(\pi_1, \pi_2) := \text{split}(\pi, \sigma, \pi')$;
    if $\text{char}(\pi_1) \not\leq U + \text{char}(\pi_2)$ **then**
        $(\mathcal{V}, \mathcal{C}) := \text{update}(\mathcal{V}, \mathcal{C}, \pi_1)$;
        $U := U + \text{char}(\pi_1)$;
        for $(\bar{\pi}, \bar{\sigma}, \bar{\pi}') \in \mathcal{V}$ **do**
            if $\bar{\pi}' = \pi$ & $\text{splitter}(\bar{\pi}, \bar{\sigma}, \pi_1) = -1$ **then**
                $\mathcal{V} := (\mathcal{V} \cup \{(\bar{\pi}, \bar{\sigma}, \pi_1)\}) \setminus \{(\bar{\pi}, \bar{\sigma}, \bar{\pi}')\}$;
    if $\text{char}(\pi_2) \not\leq U$ **then**
        $(\mathcal{V}, \mathcal{C}) := \text{update}(\mathcal{V}, \mathcal{C}, \pi_2)$;
        $U := U + \text{char}(\pi_2)$;
        for $(\bar{\pi}, \bar{\sigma}, \bar{\pi}') \in \mathcal{V}$ **do**
            if $\bar{\pi}' = \pi$ & $\text{splitter}(\bar{\pi}, \bar{\sigma}, \pi_2) = -1$ **then**
                $\mathcal{V} := (\mathcal{V} \cup \{(\bar{\pi}, \bar{\sigma}, \pi_2)\}) \setminus \{(\bar{\pi}, \bar{\sigma}, \bar{\pi}')\}$;
$\mathcal{C}_{dyn} := \mathcal{C}$.

function SPLITTER(π, σ, π')

▷ Checks if (π, σ, π') may be used to split cell π

$\pi_{in} := \emptyset$;
$\pi_{out} := \emptyset$;
for $x \in \pi$ **do**
  if $\xi(x, \sigma)$is defined **then**
    if $\xi(x, \sigma) \in \pi'$ **then** $\pi_{in} := \pi_{in} \cup \{\xi(x, \sigma)\}$;
    else $\pi_{out} := \pi_{out} \cup \{\xi(x, \sigma)\}$;
if $|\pi_{in}| > 0$ and $|\pi_{out}| > 0$ **then return** -1
else if $|\pi_{in}| > 0$ **then return** 1
else return 0

---

function SPLIT(π, σ, π')

▷ Splits cell π into $\pi = \pi_1 \cup \pi_2$ according to (σ, π')

$\pi_1 := \emptyset$;
$\pi_2 := \emptyset$;
for $x \in \pi$ **do**
  if $\xi(x, \sigma)$is not defined **then**
    $\pi_1 := \pi_1 \cup \{\xi(x, \sigma)\}$;
    $\pi_2 := \pi_2 \cup \{\xi(x, \sigma)\}$;
  else
    if $\xi(x, \sigma) \in \pi'$ **then** $\pi_1 := \pi_1 \cup \{\xi(x, \sigma)\}$;
    else $\pi_2 := \pi_2 \cup \{\xi(x, \sigma)\}$;
return (π_1, π_2)

---

function UPDATE($\mathcal{V}, \mathcal{C}, \pi_{\text{new}}$)

▷ Updates sets $\mathcal{V}$ and $\mathcal{C}$ when new cell π_{new} is added to $\mathcal{C}$

$\mathcal{C} := \mathcal{C} \cup \{\pi_{\text{new}}\}$;
for $\sigma \in \Sigma$ **do**
  $\mathcal{V}_{spt} := \emptyset$;
  for $\pi' \in \mathcal{C}$ **do**
    if splitter$(\pi_{\text{new}}, \sigma, \pi') = -1$ **then**
      $\mathcal{V}_{spt} := \{(\pi_{\text{new}}, \sigma, \pi')\}$;
    if splitter$(\pi, \sigma, \pi') = 1$ **then**
      $\mathcal{V}_{spt} := \emptyset$ and **break**;
  $\mathcal{V} := \mathcal{V} \cup \mathcal{V}_{spt}$;
return $(\mathcal{V}, \mathcal{C})$

Fig. 3. Functions for Algorithm 1

is updated at line 23. Moreover, all splitters in $\mathcal{V}$ with π as the third element are replaced, if possible, by a splitter with π_1 as the third element at lines 24-26. Line 27 checks if, given the previous choice, cell π_2 is still necessary to obtain a non-redundant cover: if this is the case, sets $\mathcal{V}$ and $\mathcal{C}$ are updated at line 28 while vector U is updated at line 29. Again, all splitters in $\mathcal{V}$ with π as the third element are replaced, if possible, by a splitter with π_2 as the third element at lines 30-32. Note that at the end all spitters with π will be eventually replaced by a splitter with π_1 or π_2. When the set of splitters $\mathcal{V}$ is empty the algorithm ends outputting the computed dynamically consistent cover.

The main result of this subsection is stated below.

Theorem 11 *Algorithm 1 terminates in a finite number of steps and outputs a non-redundant dynamically consistent cover $\mathcal{C}_{dyn}$ that refines $\mathcal{C}$. The complexity of Algorithm 1 is $O(|X|^4 \cdot |\Sigma|)$.*

Proof. In Algorithm 1, each execution of the while-loop corresponds to one split of a cell in the current cover, thereby producing an updated refined cover. First, since each split of a cell removes at least one pair (x_i, x_j) $(i \neq j)$ from the induced relation $\mathcal{R}_{\mathcal{C}}$ and the number of such pairs is finite, the number of executions of the while-loop is finite. Thus Algorithm 1 terminates in a finite number of steps. Moreover, since each while-loop generates a refined cover of the previous one, the output cover $\mathcal{C}_{dyn}$ is a refinement of the input cover $\mathcal{C}$. The fact that $\mathcal{C}_{dyn}$ is non-redundant follows from the updates in lines 21-32, where redundant subsets are discarded. To see that $\mathcal{C}_{dyn}$ is dynamically consistent, first note that each split preserves $\mathcal{R}_{\mathcal{C}}$-consistency, so $\mathcal{C}_{dyn}$ is $\mathcal{R}_{\mathcal{C}}$-consistent which satisfies condition (i) of Definition 7. Further, when the algorithm halts, the set of violating conditions $\mathcal{V}$ is empty, so condition (ii) of Definition 7 is also satisfied. Hence $\mathcal{C}_{dyn}$ is dynamically consistent.

Finally we analyze the time complexity of Algorithm 1. First, we determine the complexity of the three functions. Functions *splitter* and *split* both require a for-loop that checks all states in π; hence their complexity is $O(|X|)$. Function *update* requires checking for all cells π' in the updated cover and all symbols σ in the alphabet if a violating condition involving π_{new} exists: for each pair (σ, π') two calls to function *splitter* are required. Thus the time complexity of the function is $O(|\mathcal{C}| \cdot |\Sigma| \cdot |X|) = O(|X|^2 \cdot |\Sigma|)$, since for a non-redundant cover $|\mathcal{C}| \leq |X|$.

We conclude with the complexity of the main procedure. Lines 1-2 (initialization of U) have a total complexity $O(|X|^2)$. Lines 3-12 (initialization of the set of splitters $\mathcal{V}$) have a total complexity (including the test at lines 8 and 10) $O(|X|^3 \cdot |\Sigma|)$. The lines inside the while-loop with the highest complexity are lines 22-26 and lines 28-32, both of which have complexity $O(|X|^2 \cdot |\Sigma|)$. The while-loop can be executed at most $O(|X|^2)$ times, because each split removes at least one pair (x_i, x_j) $(i \neq j)$ from the induced relation $\mathcal{R}_{\mathcal{C}}$ and the number of such pairs is bounded by $|X|(|X|-1)/2$. Therefore in total the complexity of the algorithm is $O(|X|^4 \cdot |\Sigma|)$ corresponding to the repeated executions of the while-loop. □

5 Applications and Comparisons

In this section we demonstrate that different problems may be cast into our formulated consistent reduction problem, and provide detailed comparisons with the work of supervisor reduction in [Su and Wonham, 2004].

5.1 Supervisory control

Given a monolithic supervisor $\mathbf{H}$, we wish to understand the control logic of $\mathbf{H}$ with respect to a particular controllable event σ. Thus the goal is to construct a local supervisor $\mathbf{G}$ for σ such that $\mathbf{G}$ and $\mathbf{H}$ have consistent control decisions for σ.

Formally, let $\mathbf{M} = (Q, \Sigma, \delta, q_0, Q_m)$ be a plant to be controlled, where Σ is partitioned into a subset Σ_c of controllable events and a subset Σ_u of uncontrollable events. Suppose that $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$ is the monolithic supervisor for $\mathbf{M}$ (enforcing some imposed specification), which is maximally permissive and nonblocking. Let $\sigma \in \Sigma_c$ be a controllable event, and we consider the problem of constructing from $\mathbf{H}$ a *local controller* for σ.

Define $E : X \to \{0, 1\}$ by $E(x) = 1$ iff $\xi(x, \sigma)!$, i.e., σ is enabled at x. Also define $D : X \to \{0, 1\}$ by

$$D(x) = \begin{cases} 1, \text{ if } \textit{not } \xi(x,\sigma)! \ \& \\ \qquad (\exists s \in \Sigma^*)[\xi(x_0, s) = x \ \& \ \delta(q_0, s\sigma)!] \\ 0, \text{ otherwise (i.e. } \xi(x,\sigma)! \text{ or} \\ \qquad (\forall s \in \Sigma^*)[\xi(x_0, s) = x \Rightarrow \textit{not } \delta(q_0, s\sigma)!]) \end{cases}$$

So $D(x) = 1$ iff σ is disabled at x. A state $x \in X$ such that $E(x) = D(x) = 0$ is called a *'don't care' state*. Consider the (non-redundant) cover $\mathcal{C} = \{X_1, X_2\}$ with just two cells, where $X_1 = \{x \in X \mid D(x) = 0\}$, $\quad X_2 = \{x \in X \mid E(x) = 0\}$. Note that $X_1 \cap X_2 = \{x \in X \mid E(x) = D(x) = 0\}$, the set of 'don't care' states. The associated indicator function $O : X \to Pwr(\{1, 2\})$ is such that $O(x) = \{1\}$ iff $E(x) = 1$; $O(x) = \{2\}$ iff $D(x) = 1$; and $O(x) = \{1, 2\}$ iff $E(x) = D(x) = 0$. Given $\mathbf{H}$ and $\mathcal{C}$ described above, our consistent reduction procedure will construct a local controller $\mathbf{G}$ for σ such that after every string $s \in L(\mathbf{H})$, the decision as to disable or enable σ made by $\mathbf{H}$ and by $\mathbf{G}$ are consistent.

5.2 State estimation

While supervisor localization is a known problem that can be cast as a special case of our general formulation, in this subsection we demonstrate that interesting new reduction problems may be defined and solved in our framework. Let $\mathbf{H} = (X, \Sigma, \xi, x_0, X_m)$ be an *observer* (or *state estimator*), constructed from some non-deterministic automaton (say) $\mathbf{M} = (Q, \Sigma, \delta, q_0, Q_m)$. Thus $\mathbf{H}$ is a deterministic automaton where each state is a subset of states of $\mathbf{M}$ and for all $s \in \Sigma^*$ the state $\xi(x_0, s)$ reached in $\mathbf{H}$ by string s coincides with the set of states that can be reached in $\mathbf{M}$ by s: this set is called the *current state estimate* of $\mathbf{H}$ given observation s. Assume, however, one is only interested in monitoring a particular subset $\bar{Q} \subseteq Q$ of states of $\mathbf{M}$, which may be of some critical importance. Thus the goal is to construct a *reduced observer* $\mathbf{G}$ such that $\mathbf{G}$ and $\mathbf{H}$ provide consistent estimation for $\bar{Q}$.

Define $O : X \to \{\{1\}, \{2\}, \{3\}\}$ such that $O(x) = \{1\}$ if $x \cap \bar{Q} = \emptyset$; $O(x) = \{2\}$ if $x \subseteq \bar{Q}$; and $O(x) = \{3\}$ otherwise. Thus $O(x) = \{1\}, \{2\}, \{3\}$ means respectively that $\mathbf{M}$ is not at a state in $\bar{Q}$, is at a state in $\bar{Q}$, or may be at a state

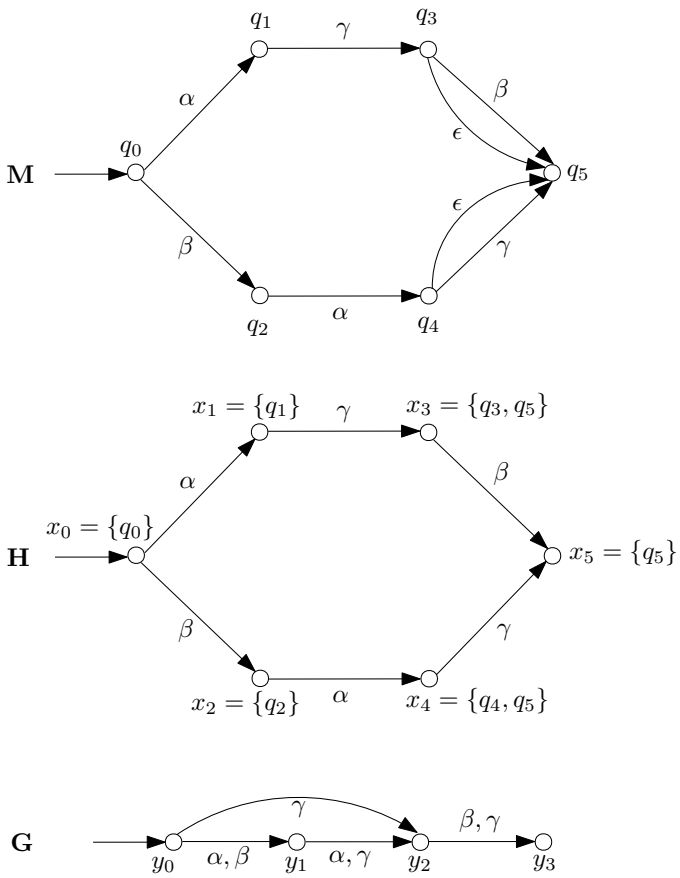


Fig. 4. A non-deterministic automaton **M** and its observer **H** (transition ϵ in **M** is 'silent' or unobservable). Reduced estimator **G** for state set $\bar{Q} = \{q_5\}$.

in $\bar{Q}$. Consider the partition $\mathcal{C} = \{X_1, X_2, X_3\}$ with three (disjoint) cells, where $X_1 = \{x \in X \mid O(x) = \{1\}\}, X_2 = \{x \in X \mid O(x) = \{2\}\}, X_3 = \{x \in X \mid O(x) = \{3\}\}$. Given **H** and $\mathcal{C}$ described above, our consistent reduction procedure will construct a reduced estimator **G** for $\bar{Q}$ such that after every string $s \in L(\mathbf{H})$, the decision (as to whether **M** is not at a state in $\bar{Q}$, is in $\bar{Q}$, or may be in $\bar{Q}$) made by **H** and by **G** are consistent.

An example of reduced estimator is displayed in Fig. 4 where, as previously mentioned, **H** is an observer for the non-deterministic automaton **M**. We are interested in constructing a reduced estimator for a (critical) state q_5 in **M**. Let $O(x_0) = O(x_1) = O(x_2) = \{1\}$ (**M** is not at q_5), $O(x_5) = \{2\}$ (**M** is exactly at q_5), and $O(x_3) = O(x_4) = \{3\}$ (**M** may be at q_5). Then the partition on X is $\mathcal{C} = \{\{x_0, x_1, x_2\}, \{x_3, x_4\}, \{x_5\}\}$. However, $\mathcal{C}$ is not a dynamically consistent partition, because condition (ii) of Definition 7 is violated for states x_0, x_2 and event α. Applying Algorithm 1 we obtain a refined dynamically consistent cover $\mathcal{C}_{dyn} = \{\{x_0, x_1\}, \{x_1, x_2\}, \{x_3, x_4\}, \{x_5\}\}$. Then construct by (6) the local estimator **G**, displayed in Fig. 4. By (7) the associated indicator function P is such that $P(y_0) = O(x_0) \cap O(x_1) = \{1\} \cap \{1\} = \{1\}$, $P(y_1) = O(x_1) \cap O(x_2) = \{1\} \cap \{1\} = \{1\}$, $P(y_2) = O(x_3) \cap O(x_4) = \{3\} \cap \{3\} = \{3\}$, $P(y_3) = O(x_5) = \{2\}$. Observe that **G** makes decisions (as to whether **M** is not at q_5, is at q_5, or may be at q_5) consistently with that done by **H**; thus one may use **G** instead of **H**, with the benefit of having fewer states. It is worth noting, in this example, that even though the initial $\mathcal{C}$ is a partition on X, the resulting $\mathcal{C}_{dyn}$ is a (dynamically

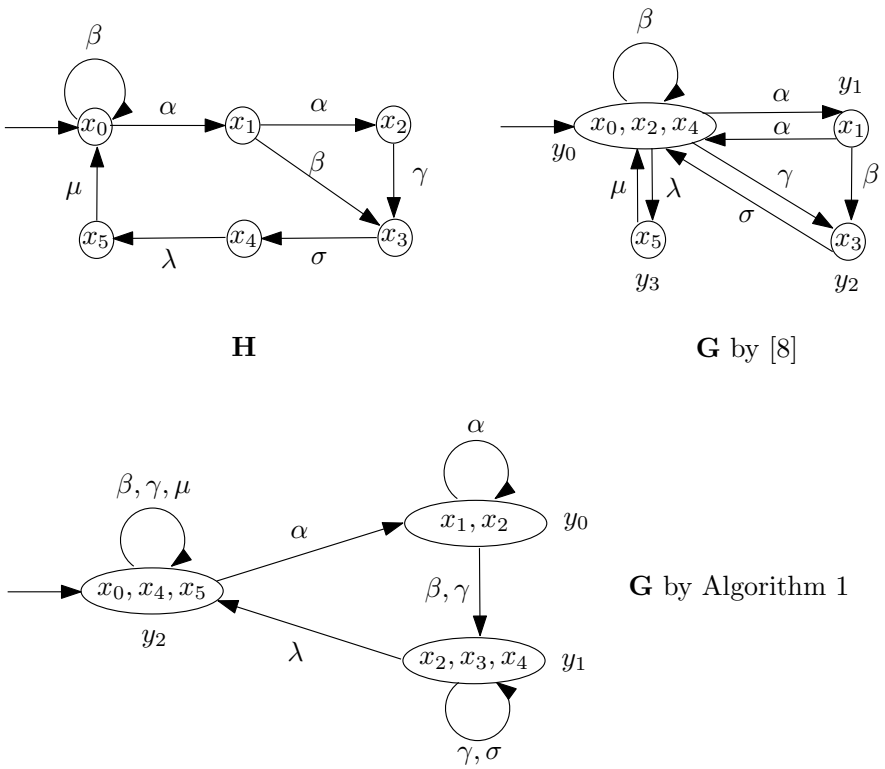


Fig. 5. Example 12: a CRP that cannot be solved by the algorithm in [Su and Wonham, 2004]

consistent) cover rather than a partition.

5.3 Comparisons with supervisor reduction

In [Su and Wonham, 2004] the supervisor reduction problem is studied (cf. Subsection 5.1), and a solution algorithm is proposed to reduce the states of a given supervisor while keeping consistent control logic. This problem is a special case of our more general CRP, because only a special type of cover is considered (cf. the 2-cell cover in Subsection 5.1). The algorithm in [Su and Wonham, 2004] works on a binary relation, which is the induced one from the considered special cover. If we apply the algorithm in [Su and Wonham, 2004] to our CRP, it may fail to find a solution. This is illustrated by the example below.

Example 12 *Consider the automaton* **H** *displayed in Fig. 5, and the non-redundant cover* $\mathcal{C} = \{X_1 = \{x_0, x_1, x_2\}, X_2 = \{x_2, x_3, x_4\}, X_3 = \{x_0, x_4, x_5\}\}$. *This cover may represent a classification of states (or strings) with different features, e.g. markers, secrets, and outputs (where output symbols are generated). The CRP is to construct a reduced automaton* **G** *and an associated indicator function* P *such that the classification is consistent with* **H***; technically, the properties (3)-(5) are satisfied,*

Applying the algorithm in [Su and Wonham, 2004], the state pairs will be checked if they can be merged in the fixed order: $(x_0, x_1), (x_0, x_2), \ldots, (x_4, x_5)$. *The resulting partition is* $\mathcal{C}' = \{X_1' = \{x_0, x_2, x_4\}, X_2' = \{x_1\}, X_3' = \{x_3\}, X_4' = \{x_5\}\}$, *which does not refine the given cover* $\mathcal{C}$*; and the corresponding reduced automaton* **G** *is displayed in Fig. 5. For*

this $\mathbf{G}$*, however, it is impossible to define an indicator function* P *to satisfy property (5). Specifically, for* $y_0 = 1$ *(the index of the first cell in* $\mathcal{C}'$*), there are eight possible choices for* $P(y_0)$*: these are all the subsets of cell indices* $\{1, 2, 3\}$ *in* $\mathcal{C}$*). None of the eight choices, however, can satisfy (5). For example,* $P(y_0) = \emptyset$ *violates (5) for* $s = \beta$*, because* $P(\eta(y_0, \beta)) = P(y_0) = \emptyset$*;* $P(y_0) = \{1\}$ *violates (5) for* $s = \alpha\beta\sigma$*, because* $P(\eta(y_0, \alpha\beta\sigma)) = \{1\}$ *while* $O(\xi(x_0, \alpha\beta\sigma)) = \{2, 3\}$*;* $P(y_0) = \{2, 3\}$ *violates (5) for* $s = \alpha\alpha$*, because* $P(\eta(y_0, \alpha\alpha)) = \{2, 3\}$ *while* $O(\xi(x_0, \alpha\alpha)) = \{1, 2\}$*. Similarly one may verify that all the eight choices for* $P(y_0)$ *fail to satisfy (5), and therefore no function* P *exists to solve the CRP.*

The reason why the algorithm in [Su and Wonham, 2004] fails is because, designed to solve the special instance of supervisor reduction, it does not have a mechanism to take into account the given cover $\mathcal{C}$*. Indeed, that algorithm always starts from the* singleton *partition, trying to merge states in a* fixed *order, and thus the resulting partition need not be a refinement of the given* $\mathcal{C}$*. To satisfy (5), however, a refinement of* $\mathcal{C}$ *is crucial. This is achieved by our designed Algorithm 1, which yields the non-redundant dynamically consistent cover* $\mathcal{C}_{dyn} = \{X_1' = \{x_1, x_2\}, X_2' = \{x_2, x_3, x_4\}, X_3' = \{x_0, x_4, x_5\}\}$*. This* $\mathcal{C}_{dyn}$ *gives rise to multiple reduced automata all with three states* $y_0 = 1, y_1 = 2, y_2 = 3$ *(one such automaton is displayed in Fig. 5). The associated indicator function* P *defined as in (7) is such that* $P(y_0) = O(x_1) \cap O(x_2) = \{1\}$*,* $P(y_1) = O(x_2) \cap O(x_3) \cap O(x_4) = \{2\}$*,* $P(y_2) = O(x_0) \cap O(x_4) \cap O(x_5) = \{3\}$*. It is readily verified that with this* P*, the required properties (3)-(5) of CRP are all satisfied.*

For problems like supervisor reduction where the algorithm in [Su and Wonham, 2004] and our Algorithm 1 can both be applied, due to their different approaches (merging versus splitting) it is not easy to compare them in terms of reduction efficiency. There are cases, however, where our Algorithm 1 achieves (much) greater reduction than the algorithm in [Su and Wonham, 2004], thanks to the use of covers instead of partitions. This is illustrated in the following example.

Example 13 *As in Fig. 6, automaton* $\mathbf{M}$ *is a plant to be controlled,* $\mathbf{H}$ *a monolithic supervisor for* $\mathbf{M}$*, and* σ *the only controllable event. All states are marked. The only disablement of* σ *by* $\mathbf{H}$ *is at state* x_n*; and* σ *is "don't care" at* x_0*. For this example, applying the algorithm in [Su and Wonham, 2004] does not achieve any state reduction. In fact, similar to [Su and Wonham, 2004, Example 4], the reason for failing to reduce any states is due to the use of partitions in that algorithm, and state reduction or even* minimal *state reduction may be achieved* only *by using* covers*, as our Algorithm 1 does.*

Consider the 2-cell cover $\mathcal{C} = \{\{x_0, x_1, \ldots, x_{n-1}\}, \{x_0, x_n\}\} =: \{X_1, X_2\}$ *(as in Subsection 5.1). It is readily verified that* $\mathcal{C}$ *is already a dynamically consistent cover (inputting* $\mathbf{H}$ *and* $\mathcal{C}$ *to Algorithm 1 will output* $\mathcal{C}$ *itself). Then* $\mathcal{C}$ *yields a reduced* 2*-state automaton* $\mathbf{G}$ *displayed in Fig. 6. In fact, this 2-state* $\mathbf{G}$ *is the minimal state automaton that satisfies the three properties of CRP.*

Finally, while the algorithm in [Su and Wonham, 2004] computes a partition which corresponds to a single solution, our Algorithm 1 computes a cover that corresponds to a *family* of solutions, offering extra flexibility to define multiple reduced automata. This flexibility may be useful in exploring additional features in different problems. The following example demonstrate this point from the perspective of reduction of event observation (or communication).

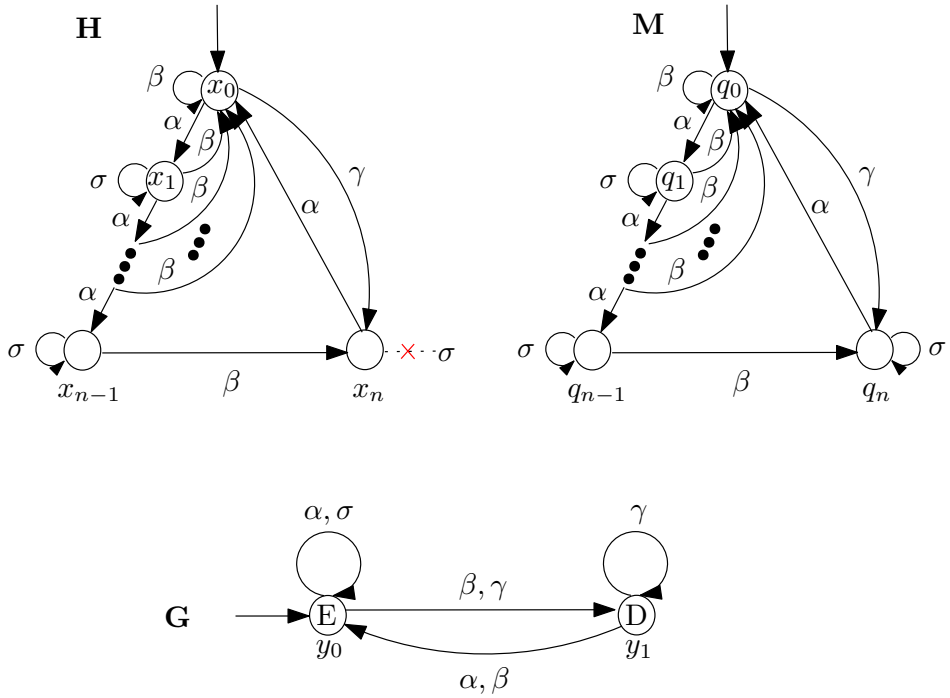


Fig. 6. Example 13: monolithic supervisor **H** for plant **M**; a minimal-state supervisor **G** achieved by Algorithm 1

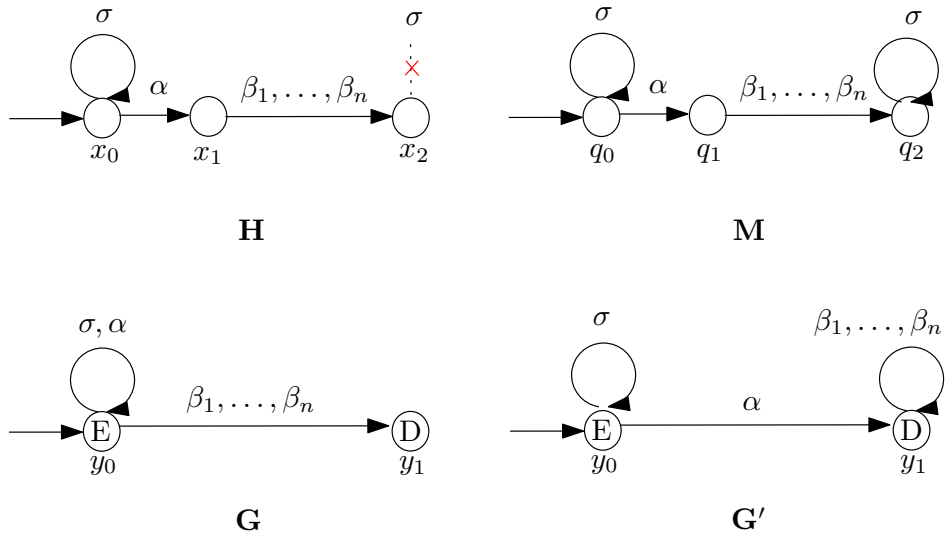


Fig. 7. Example 14: monolithic supervisor **H** for plant **M**; two reduced supervisors **G**, **G**′

Example 14 *In Fig. 7, automaton* **M** *is a plant to be controlled,* **H** *a monolithic supervisor for* **M**, *and* σ *the only controllable event. All states are marked.* **H** *enables* σ *at* x_0, *disables* σ *at* x_2, *and* x_1 *is a "don't care" state. Applying the algorithm in [Su and Wonham, 2004] results in the partition* $\{\{x_0, x_1\}, \{x_2\}\}$, *which leads to a single solution* **G** *as displayed in Fig. 7. By contrast, applying our Algorithm 1 results in the cover* $\{\{x_0, x_1\}, \{x_1, x_2\}\}$, *which gives rise to different partitions* $\{\{x_0, x_1\}, \{x_2\}\}$ *and* $\{\{x_0\}, \{x_1, x_2\}\}$ *which are both dynamically consistent; the latter yield two solutions* **G** *and* **G**′ *as displayed in Fig. 7. Both* **G** *and* **G**′ *can be used as a supervisor for plant* **M** *to correctly enable/disable event* σ; *however,* **G** *must observe* n *events* $\beta_1, \ldots, \beta_n$, *whereas* **G**′ *merely needs to observe*

one event α. Suppose a priori *that events $\beta_1, \ldots, \beta_n$ are unobservable (no sensors for them), or a design objective is to observe as few events as possible to minimize costs (but still can control σ correctly), then one may explore the flexibility offered by the cover from Algorithm 1 and choose to use $\mathbf{G}'$. This flexibility does not exist in the algorithm in [Su and Wonham, 2004].*

6 Conclusions

We have developed a consistent reduction procedure which, given an arbitrary finite automaton and a binary relation, determines a reduced one while preserving the possibility of correctly classifying generated strings. In future work, we are interested in extending this consistent reduction framework to handle more general state transition systems with possibly infinite number of states and/or nondeterministic transitions.

References

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

[Blizard, 1989] Blizard, W. (1989). Multiset theory. *Notre Dame Journal of Formal Logic*, 30(1):36–66.

[Bugalho and Oliveira, 2005] Bugalho, M. and Oliveira, A. (2005). Inference of regular languages using state merging algorithms with search. *Pattern Recognition*, 38(9):1457–1467.

[Cai et al., 2019] Cai, K., Giua, A., and Seatzu, C. (2019). On consistent reduction in discrete-event systems. In *Proc. 15th IEEE Int. Conf. Automation Science and Engineering*, pages 474–479, Vancouver, Canada.

[Cai and Wonham, 2016] Cai, K. and Wonham, W. M. (2016). *Supervisor Localization: A Top-Down Approach to Distributed Control of Discrete-Event Systems.* Lecture Notes in Control and Information Sciences, vol. 459, Springer.

[Hopcroft et al., 2006] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2006). *Introduction to Automata Theory, Languages and Computation, 3rd ed.* Addison-Wesley.

[Karp, 1972] Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

[Milner, 1989] Milner, R. (1989). *Communication and Concurrency.* Prentice Hall.

[Minhas, 2002] Minhas, R. (2002). *Complexity Reduction in Discrete Event Systems.* PhD thesis, ECE Dept., University of Toronto.

[Mohajerani et al., 2014] Mohajerani, S., Malik, R., and Fabian, M. (2014). A framework for compositional synthesis of modular nonblocking supervisors. *IEEE Trans. Autom. Control*, 59(1):150–162.

[Oncina and Garcia, 1992] Oncina, J. and Garcia, P. (1992). Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis, World Scientific*, 1:49–61.

[Su et al., 2010] Su, R., van Schuppen, J. H., Rooda, J. E., and Hofkamp, A. T. (2010). Non-conflict check by using sequential automaton abstractions based on weak observation equivalence. *Automatica*, 46(6):968–978.

[Su and Wonham, 2004] Su, R. and Wonham, W. M. (2004). Supervisor reduction for discrete-event systems. *Discrete Event Dyna. Syst.*, 14(1):31–53.

[Su and Wonham, 2018] Su, R. and Wonham, W. M. (2018). A generalized theory on supervisor reduction. In *Proc. 57th IEEE Conf. on Decision and Control*, pages 3950–3955, Miami, FL.

[Tabuada, 2009] Tabuada, P. (2009). *Verification and Control of Hybrid Systems: A Symbolic Approach.* Springer.

[Verwer et al., 2006] Verwer, S., de Weerdt, M., and Witteveen, C. (2006). Identifying an automaton model for timed data. In *Proc. of the 15th Annual Machine Learning Conference of Belgium and the Netherlands*, pages 57–64, Ghent, Belgium.