Supervisor Reduction for Discrete-Event Systems

R. Su, W.M. Wonham
Edward S. Rogers Sr. Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario M5S 3G4, Canada
surong,wonham@control.utoronto.ca

June 3, 2003

Abstract

In supervisory control theory (SCT) the supremal supervisor (representing the supremal controllable sublanguage) typically has a large state size (of order the product of state sizes of the plant and specification automata). In this paper we propose a polynomial-time algorithm which can significantly reduce supervisor size while preserving control action. We also show that finding a supervisor of minimal size is NP-hard.

1 Introduction

In supervisory control theory (SCT) [RW87] [WR87] [Won02], the control problem associated with a discrete-event system is to enforce controllable and nonblocking behavior of the plant that is admissible by the specification. The optimal solution is the supremal controllable sublanguage (of the specification language), as represented by the *supremal supervisor*, having state size of order the product of state sizes of the plant and specification transition structures.

As the supremal supervisor incorporates redundant information, namely about transition constraints already enforced by the plant itself, state size reduction should be possible without affecting controlled behavior. A reduced supervisor is desirable both for greater economy of implementation, and because its greater simplicity may render its control action more readily comprehensible to the designer. As noted in [QC02], supervisor reduction can be especially useful in the design of modular controls, where optimal local modular supervisors may admit quite small reductions that are simple and practical to implement. Examples in [QC02] for control of a manufacturing cell include reductions from 32 states to 2 and 24 to 4, computed by the algorithm in [VW86]. Other applications (of the algorithm in Sect. 4, below), to automaton models of Petri net controllers from the literature, can be found in [Won02, Sect. 8.14].

The supervisor reduction problem was first explored by Vaz and Wonham [VW86], who introduced the concept of *control cover* from which the structure of a reduced supervisor could be projected. They proved two useful reduction theorems, and proposed a corresponding (exponential time) reduction algorithm. We follow a similar path, but with three important differences. First, our definition of cover is less restrictive than that of [VW86], and leads to a single reduction theorem that subsumes both theorems of [VW86], is easier to prove, and shows that a cover (in our sense) is necessary as well as sufficient for supervisor reduction. Second, using the concept of control congruence (a special but useful case of control cover), we provide a polynomial-time reduction algorithm. Third, we show that the *minimal supervisor problem (MSP)* of computing a supervisor with minimal state size is NP-hard. Thus, while our proposed algorithm may fail (unsurprisingly) to solve MSP, we present numerical evidence to show that it can do so in individual cases, and in any case often achieves significant state size reduction in reasonable time for problem instances of moderate size. As a byproduct of our approach to the complexity issue, we also provide a polynomial-time computable lower bound to the minimal supervisor state size, which can be helpful in estimating how close in size a computed reduced supervisor is to a minimal supervisor and, if the two computed results happen to coincide, proves minimality for the instance at hand.

This paper is self-contained, requiring no familiarity with [VW86] or, for that matter, with SCT. For the latter the reader may consult the original sources [RW87],[WR87] or the updated comprehensive treatment [Won02]. We suggest, however, that the paper will be of interest mainly to readers with an active interest in designing supervisory controls and already with some background in the SCT area.

In Section 2 we provide the theory of supervisor reduction and our main reduction result, the Generalized Quotient Theorem. The complexity issue is addressed in Section 3, and the new reduction algorithm along with numerical results in Section 4. Conclusions are presented in Section 5. Detailed proofs are placed in the Appendix.

2 Theoretical results

2.1 Preliminaries

In SCT, a discrete-event system (DES) to be controlled (or *plant*) is modelled as a discrete transition structure

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

Here Y is the state set, Σ the event set (alphabet, assumed finite) which is partitioned into two subsets - the controllable event set Σ_c and the uncontrollable event set Σ_{uc}, $\eta : Y \times \Sigma \to Y$ the (partial) transition function, $y_0 \in Y$ the initial state of $\mathbf{G}$ and $Y_m \subseteq Y$ the marker subset [Won02]. In the usual way, η is extended to a partial function on strings $s \in \Sigma^*$, giving $\eta : Y \times \Sigma^* \to Y$ (pfn). Writing $\eta(y_0, s)!$ to mean that η is defined for s at y_0, we bring in the *closed behavior* of $\mathbf{G}$,

$$L(\mathbf{G}) := \{s \in \Sigma^* \,|\, \eta(y_0, s)!\}$$

and the *marked behavior* of $\mathbf{G}$,

$$L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) \,|\, \eta(y_0, s) \in Y_m\}$$

A nonempty language $SPEC \subseteq \Sigma^*$ is *represented* by a DES $\mathbf{SPEC}$ if $\mathbf{SPEC}$ is reachable and

$$L(\mathbf{SPEC}) = \overline{SPEC}, \qquad L_m(\mathbf{SPEC}) = SPEC$$

where $\overline{SPEC}$ is the prefix closure of $SPEC$. Let $K \subseteq L_m(\mathbf{G}) \cap SPEC$ be the *supremal controllable sublanguage* of $L_m(\mathbf{G})$ with respect to $SPEC$ (see [Won02]):

$$K = \sup \mathcal{C}\,(L_m(\mathbf{G}) \cap SPEC)$$

In case $K \neq \varnothing$, a *supremal supervisor* for $\mathbf{G}$ (with respect to $SPEC$) is a DES **SUPER** which represents K. In this paper, $L_m(\mathbf{G})$ and $SPEC$ are fixed regular languages, with $\mathbf{G}$ and **SPEC** finite-state DES; then K is also regular [Won02], and we assume $K \neq \varnothing$. Write $|\mathbf{G}|$ for the state size of $\mathbf{G}$, etc. Then **SUPER** can be readily computed (e.g. using the operator **supcon** in [Won02]) in time a polynomial function of $|\mathbf{G}|$, $|\mathbf{SPEC}|$, with result

$$|\mathbf{SUPER}| \leq |\mathbf{G}||\mathbf{SPEC}|$$

Here the conservative bound is typically representative of the actual size of **SUPER**. In applications, engineering intuition may (or may not) suggest that there might exist a DES, say **SIMSUP**, such that

$$|\mathbf{SIMSUP}| << |\mathbf{SUPER}|$$

and the following properties hold,

$$\begin{aligned} L(\mathbf{G}) \cap L(\mathbf{SIMSUP}) &= L(\mathbf{SUPER}) && (1a) \\ L_m(\mathbf{G}) \cap L_m(\mathbf{SIMSUP}) &= L_m(\mathbf{SUPER}) && (1b) \end{aligned}$$

These linguistic relations are displayed in Fig. 1. In the operator notation of [Won02], (1a,b) is equivalent to

$$\mathbf{isomorph}(\mathbf{meet}(\mathbf{G}, \mathbf{SIMSUP}), \mathbf{SUPER}) = true \qquad (2)$$

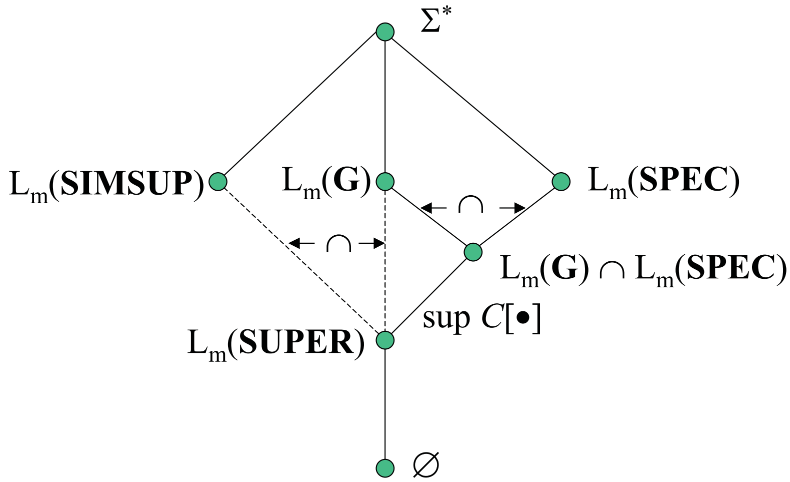

Figure 1: Supremal (Controllable) Sublanguage

Computing (2), hence deciding (1), can be done in time a polynomial function of $|\mathbf{G}|$, $|\mathbf{SUPER}|$. Any DES **SIMSUP** that satisfies (1a,b) is *control equivalent to* **SUPER** *with respect to* **G**.

Ideally, one seeks to construct a control-equivalent **SIMSUP**, say **MINSUP**, with state size as small as possible, that is, to solve the *minimal supervisor problem* (MSP) for the control pair $(\mathbf{G}, \mathbf{SUPER})$. It should be emphasized that MSP differs from classical problems of minimizing the state size of a finite recognizer for a regular language, e.g. [HU79] [Pfl73] [PO98]. In our terminology, if a finite automaton **A** is given, with either partial or total transition function, one might seek to minimize the size of **A** while preserving both $L_m(\mathbf{A})$ (the marked behavior of **A**, i.e., classically, the language 'recognized' by **A**) and $L(\mathbf{A})$ (the closed behavior of **A**, namely the input strings for which the transition function of **A** is defined). This problem (P1, say) is well known to be polynomial-time solvable (cf. [HU79]), and in the operator notation of [Won02] is accomplished by **minstate**. Alternatively, when the transition function of **A** is only a partial function, one may seek to complete it to a total function while preserving the marked, but not necessarily the closed behavior of **A**. Call such a completion 'optimal' if the corresponding reduced automaton (in the sense of P1) is minimal; this problem (P2, say) is known to be NP-hard [Pfl73]. In the present context of supervisor reduction, solving P1 for **SUPER** rarely yields any significant state reduction, for when **SUPER** is computed from **supcon** with minimal-state arguments **G** and **SPEC** it is typically already a minimal state representation of controlled behavior or, if not, is not significantly redundant as a representation of **SUPER**'s closed and marked behaviors. In any case, our objective in computing some **SIMSUP** (or in particular, some **MINSUP**) is not (necessarily) to preserve either the marked or closed behavior of **SUPER**, but instead the control action of **SUPER** as prescribed by (1). This requires suitably exploiting the (fixed) marked and closed behaviors of the controlled plant **G** in addition to the control action of **SUPER**. As languages, the resulting closed and marked behaviors of **SIMSUP** will typically be much larger than those of **SUPER**, namely **SIMSUP** is not reducible from **SUPER** alone, by standard methods.

In this paper we show that MSP is NP-hard, but on the other hand propose an algorithm for computing some (not necessarily minimal) **SIMSUP** which empirically can be considerably smaller than **SUPER** and which is often computable in 'reasonable' time on problems of 'moderate' size. The theoretical basis for reduction is control covers, as discussed next.

2.2 Control Covers

Suppose **SUPER** $= (X, \Sigma, \xi, x_0, X_m)$. Let $E : X \to 2^{\Sigma}$ with

$$x \mapsto E(x) := \{\sigma \in \Sigma \,|\, \xi(x, \sigma)!\}$$

be the (**SUPER**$-$)*enabled event set* at state $x \in X$. Let $D : X \to 2^{\Sigma}$ with

$$x \mapsto D(x) := \{\sigma \in \Sigma \,|\, \neg\xi(x,\sigma)! \,\&\, (\exists s \in \Sigma^*)[\xi(x_0,s) = x \,\&\, \eta(y_0,s\sigma)!]\}$$

be the (**SUPER**$-$)*disabled event set* at state $x \in X$. Let $M : X \to \{true, false\}$ with

$$x \mapsto M(x) = true \text{ if } x \in X_m$$

Finally let $T : X \to \{true, false\}$ with

$$x \mapsto T(x) = true \text{ if } (\exists s \in \Sigma^*)\, \xi(x_0,s) = x \,\&\, \eta(y_0,s) \in Y_m$$

Let $\mathcal{R} \subseteq X \times X$ be the binary relation such that for a pair $x, x' \in X$, $(x,x') \in \mathcal{R}$ iff

1. $E(x) \cap D(x') = E(x') \cap D(x) = \varnothing$
2. $T(x) = T(x') \Rightarrow M(x) = M(x')$

Condition 1 says that for a pair of states (x,x') in $\mathcal{R}$, the associated enable/disable control actions should be consistent, namely no event is enabled at x but disabled at x'. Condition 2 requires that states x,x' in $\mathcal{R}$ be consistently marked either *true* or *false* in **SUPER** if they are reachable by some strings s,s' in $L_m(\mathbf{G})$ (in case $T(x) = T(x') =$*true*), or else if neither is reachable by strings in $L_m(\mathbf{G})$ (i.e., $T(x) = T(x') =$*false*). In the latter case (1b) shows that in fact $M(x) = M(x') =$ *false.* Clearly $\mathcal{R}$ is reflexive and symmetric, but it need not be transitive, and so is generally not an equivalence relation. This fact underlies our next definition.

Recall that a *cover* of a set X is a family of subsets of X whose union is X. Let I be an index set.

Definition 2.1 A cover $\mathcal{C} = \{X_i \subseteq X | i \in I\}$ of X is a *control cover on* **SUPER** if

1. $(\forall i \in I)\, X_i \neq \varnothing \wedge (\forall x, x' \in X_i)\, (x,x') \in \mathcal{R}$

2. $(\forall i \in I)(\forall \sigma \in \Sigma)(\exists j \in I)[(\forall x \in X_i)\, \xi(x,\sigma)! \Rightarrow \xi(x,\sigma) \in X_j]$

The subsets X_i are the *cells* of $\mathcal{C}$. A control cover $\mathcal{C}$ is a *control congruence* if $\mathcal{C}$ is a partition on X, namely the X_i are pairwise disjoint.

Condition 1 requires that each cell of $\mathcal{C}$ be nonempty and each pair of states in the same cell should belong to $\mathcal{R}$, namely their associated control action and marked status should be consistent. Condition 2 states that for each $X_i \in \mathcal{C}$ and each event $\sigma \in \Sigma$, the set of states that can be reached from any state in X_i by a one-step transition σ is covered by some $X_j \in \mathcal{C}$. Note that the cells of a control cover may overlap: $x \in X$ may belong to more than one cell X_i. To avoid ambiguity later we may denote by E_i, etc. the restriction of E to the cell X_i of a cover.

Given a control cover $\mathcal{C} = \{X_i \subseteq X | i \in I\}$ on **SUPER**, we construct an *induced supervisor*

$$\mathbf{J} = (I, \Sigma, \kappa, i_0, I_m)$$

as follows. Let

$$\begin{aligned} i_0 &= \text{some } i \in I \text{ with } x_0 \in X_i \\ I_m &= \{i \in I \,|\, X_i \cap X_m \neq \varnothing\} \end{aligned}$$

$$\kappa : I \times \Sigma \to I \text{ (pfn)}$$

with $\kappa(i,\sigma) = j$ provided, for such choice of $j \in I$,

$$(\exists x \in X_i)\, \xi(x,\sigma) \in X_j \,\&\, (\forall x' \in X_i)[\xi(x',\sigma)! \Rightarrow \xi(x',\sigma) \in X_j]$$

Because of overlapping, i_0 and κ may not be uniquely determined; here a fixed but arbitrary instance of $\mathbf{J}$ is intended. If $\mathcal{C}$ is a control congruence then $\mathbf{J}$ is uniquely determined by $\mathcal{C}$.

Proposition 2.1 **J** is control equivalent to **SUPER** with respect to **G**.

Definition 2.2 A DES **SIMSUP** $:= (Z,\, \Sigma,\, \zeta,\, z_0,\, Z_m)$ is *normal with respect to* **SUPER** if

1. $(\forall z \in Z)(\exists s \in L(\mathbf{SUPER}))\, \zeta(z_0, s) = z$
2. $(\forall z, z' \in Z)(\forall \sigma \in \Sigma)\Big[\zeta(z, \sigma) = z' \Rightarrow (\exists s \in \Sigma^*)\, \big[s\sigma \in L(\mathbf{SUPER})\, \&\, \zeta(z_0, s) = z\big]\Big]$
3. $(\forall z \in Z_m)(\exists s \in L_m(\mathbf{SUPER}))\, \zeta(z_0, s) = z$

Thus **SIMSUP** is normal if (1) its states are all reachable and (2) its transitions all taken, under strings in $L(\mathbf{SUPER})$, namely no state or transition in **SIMSUP** is superfluous; (3) each marked state in **SIMSUP** is reachable by at least one string that is marked by **SUPER**, namely no state in **SIMSUP** is marked unnecessarily. If **SIMSUP** is control equivalent to **SUPER** with respect to **G** but not normal, then simply deleting all superfluous states and transitions, and converting all unnecessarily marked states into unmarked states in **SIMSUP**, will render **SIMSUP** normal, while preserving the control equivalence between **SIMSUP** and **SUPER**. Normalizing **SIMSUP** cannot increase its state size, so a minimal normal supervisor must also be a minimal supervisor. From now on we therefore consider only reduction to normal supervisors.

Before stating our first main result we need one more definition.

Definition 2.3 Let

$$\begin{aligned} \mathbf{G}_A &= (X_A, \Sigma, \xi_A, x_{A,0}, X_{A,m}) \\ \mathbf{G}_B &= (X_B, \Sigma, \xi_B, x_{B,0}, X_{B,m}) \end{aligned}$$

$\mathbf{G}_B$ is a *DES-epimorphic image* of $\mathbf{G}_A$ under *DES-epimorphism* $\theta : X_A \to X_B$ if

1. $\theta : X_A \to X_B$ is surjective

2. $\theta(x_{A,0}) = x_{B,0}$ and $\theta(X_{A,m}) = X_{B,m}$

3. $(\forall x \in X_A)(\forall \sigma \in \Sigma)\, \xi_A(x,\sigma)! \Rightarrow [\xi_B(\theta(x),\sigma)! \,\&\, \xi_B(\theta(x),\sigma) = \theta(\xi_A(x,\sigma))]$

4. $(\forall x \in X_B)(\forall \sigma \in \Sigma)\, \xi_B(x,\sigma)! \Rightarrow [(\exists x' \in X_A)\, \xi_A(x',\sigma)! \,\&\, \theta(x') = x]$

In particular, $\mathbf{G}_B$ is *DES-isomorphic* to $\mathbf{G}_A$ if $\theta : X_A \to X_B$ is bijective.

Theorem 2.1 [**Generalized Quotient Theorem**]

Let **SUPER** be a supremal supervisor for **G** and let **SIMSUP** be any normal supervisor with respect to **SUPER** that is control-equivalent to **SUPER** with respect to **G**. Then there exists a control cover $\mathcal{C}$ on **SUPER** for which some induced supervisor **J** is DES-isomorphic to **SIMSUP**.■

The Generalized Quotient Theorem implies that any minimal normal supervisor is induced by a suitable control cover. Fig. 2 depicts examples of **G** and **SUPER**, along with three control-equivalent simplified supervisors. It is easily checked that each of the

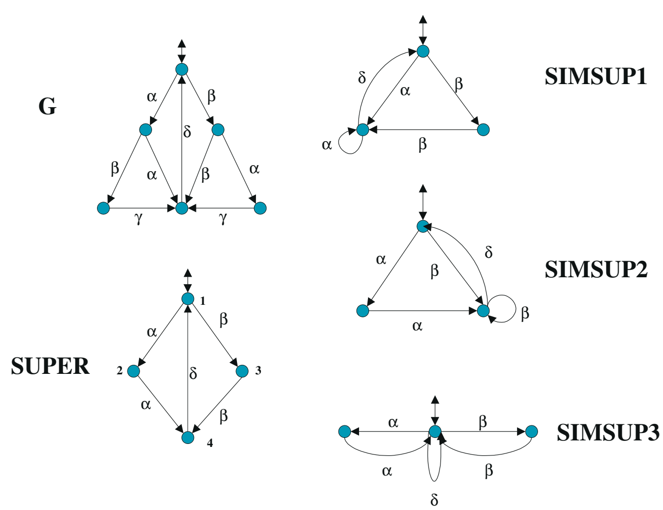

Figure 2: Minimal Supervisors with α, β Controllable and γ, δ Uncontrollable

latter can be constructed from **SUPER** by a control cover, but all three are pairwise non-DES-isomorphic. It can be shown by direct calculation (see Section 4.2 on the lower bound estimate for minimal supervisors) that each is actually state-minimal. From this we conclude that a state-minimal supervisor is in general not unique, even up to DES-isomorphism.

3 NP-hardness of the Minimal Supervisor Problem

It would be desirable to find a polynomial-time algorithm which always computes a minimal supervisor. Unfortunately, as we show in this section, the minimal supervisor problem MSP is NP-hard. Let $\Theta(\mathbf{G}, \mathbf{SUPER})$ be the set of all supervisors that are control equivalent to **SUPER** with respect to **G**. Let $< O >$ denote the encoding of an object O into its representation as a string, and set

$$A := \{< \mathbf{G}, \mathbf{SUPER}, R > \mid (\exists \mathbf{SIMSUP} \in \Theta(\mathbf{G}, \mathbf{SUPER}))\, |\mathbf{SIMSUP}| \leq R\}$$

Lemma 3.1 Deciding A is polynomial time reducible to solving MSP.

Proof: If on solving MSP we find $|\mathbf{MINSUP}| = r$, then

$$< \mathbf{G}, \mathbf{SUPER}, R > \in A \quad \text{iff} \quad R \geq r \ \blacksquare$$

Next we show that deciding A is NP-complete, and therefore solving MSP is NP-hard.

Lemma 3.2 Deciding language A is an NP problem.

Proof: We can (nondeterministically) decide whether or not a triple $< \mathbf{G}, \mathbf{SUPER}, R >$ is in A, as follows.

1. Arbitrarily construct an automaton **S** (over the fixed alphabet Σ) with no more than R states.

2. Check whether or not **S** is deterministic.

3. Check whether or not **S** is control-equivalent to **SUPER** with respect to **G**.

Constructing a concrete automaton (Step 1) and checking properties 2,3 are all polynomial-time computations. Thus deciding A is an NP problem. ■

Consider an undirected graph $\mathbf{Gr} = (Ver, Edg)$, and positive integer $R \leq |Ver|$. The Partition-Into-Cliques problem PIC($\mathbf{Gr}, R$) (designated GT15 in [GJ79]) is to decide whether Ver can be partitioned into $r \leq R$ subsets called *cliques*, such that each pair of vertices in the same clique are connected by an edge in Edg. Deciding PIC($\mathbf{Gr}, R$) is NP-complete [GJ79]. Let

$$B := \{<\mathbf{Gr}, R> \mid \text{PIC}(\mathbf{Gr}, R) \text{ is solvable } \}$$

We shall convert $\mathbf{Gr} = (Ver, Edg)$ into a plant DES **GrG** and supremal supervisor **GrSUPER**. The idea is to define the state set X of **GrSUPER** from the vertex set Ver of **Gr** such that a cliques partition on Ver corresponds to a control congruence on X; as shown in Prop. 2.1, the latter induces a reduced supervisor control equivalent to **GrSUPER**. For this consider the following conversion algorithm TC.

TC:="On input $<\mathbf{Gr}>$,

1. Let $Ver = \{v_0, \cdots, v_n\}$ and bring in new symbols $W := \{v_{ij} \mid 0 \leq i < j \leq n\}$. Let

$$\begin{aligned} X' &:= Ver \cup W \\ Y' &:= X' \\ \Sigma &:= \varnothing \end{aligned}$$

2. Construct partial transition functions ξ and η as follows.

2.1. For $i = 0, 1, \cdots, n-1$, create an event α_i and add it to Σ. Let

$$\begin{aligned} \xi(v_i, \alpha_i) &= v_{i+1} \\ \eta(v_i, \alpha_i) &= v_{i+1} \end{aligned}$$

2.2. For $0 \le i < j \le n$, if $(v_i, v_j) \notin Edg$ then create event β_{ij} and add it to Σ. Let

$$\begin{aligned} \xi(v_j, \beta_{ij}) &= v_{ij} \\ \eta(v_i, \beta_{ij}) &= \eta(v_j, \beta_{ij}) = v_{ij} \end{aligned}$$

3. Imposing reachability, define

$$\begin{aligned} X &:= \{v \in X' \,|\, (\exists s \in \Sigma^*)\xi(v_0, s) = v\} \\ X_m &:= X \\ x_0 &:= v_0 \\ Y &:= \{v \in Y' \,|\, (\exists s \in \Sigma^*)\eta(v_0, s) = v\} \\ Y_m &:= Y \\ y_0 &:= v_0 \end{aligned}$$

4. Return

$$\begin{aligned} \mathbf{GrSUPER} &:= (X, \Sigma, \xi, X_m, x_0) \\ \mathbf{GrG} &:= (Y, \Sigma, \eta, Y_m, y_0) \end{aligned}$$

as the output of TC and set all events in Σ to be controllable."

If **Gr** has n vertices, then **GrG** contains at most $n + n(n-1) = n^2$ states and $n - 1 + n(n-1) = n^2 - 1$ transitions; **GrSUPER** contains at most n^2 states and $n - 1 + \frac{1}{2}n(n-1) = \frac{1}{2}n(n+1) - 1$ transitions. If we take as a measure of computation time the total number of states and transitions in **GrG** and **GrSUPER**, then TC is a polynomial-time computational procedure on **Gr** with respect to the number of vertices in **Gr**. Later we use the fact that the transition graph of **GrSUPER** is a finite tree.

To illustrate TC, let **Gr** be the graph in Fig. 3. With $r = R = 2$, a solution of

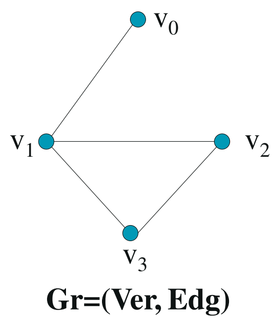

Figure 3: Example 3

PIC($\mathbf{Gr}, 2$) is $\{\{v_0\}, \{v_1, v_2, v_3\}\}$. According to TC, we have

$$\begin{aligned} X &= Y = \{v_0, v_1, v_2, v_3, v_{02}, v_{03}\} \\ \Sigma &= \{\alpha_0, \alpha_1, \alpha_2, \beta_{02}, \beta_{03}\} \end{aligned}$$

The transition graphs for **GrSUPER** and **GrG** are displayed in Fig. 4. With all

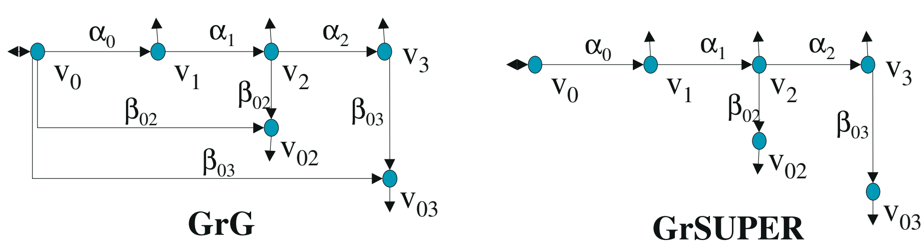

Figure 4: Example 3

events in Σ controllable, it is easy to check that a control congruence on **GrSUPER** is $\mathcal{C} = \{\{v_0, v_{02}, v_{03}\}, \{v_1, v_2, v_3\}\} = \{X_1, X_2\}$, say. The induced supervisor **GrSIMSUP** is displayed in Fig. 5.

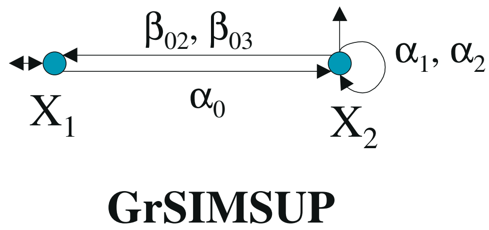

Figure 5: Example 3

In general, the relation between control congruences on **GrSUPER** and clique partitions on **Gr** is described by the following lemma.

Lemma 3.3 PIC(**Gr**, R) is solvable iff **GrSUPER** (obtained by TC) admits a control cover $\mathcal{C}$ (with respect to **GrG**) having at most R cells.

With Lemma 3.3 we can describe the relation between languages A and B as follows.

Proposition 3.1 Deciding B is polynomial-time-mapping-reducible to deciding A.

Theorem 3.1 The problem of deciding language A is NP-complete.

Proof: By Lemma 3.2, deciding A is an NP problem. By Prop. 3.1, deciding B, which is a known NP-complete problem, is polynomially reducible to deciding A. Therefore deciding A is an NP-complete problem. ■

Corollary 3.1 Solving MSP is NP-hard.

Proof: By Lemma 3.1, solving MSP is at least as hard as deciding language A. By Theorem 3.1, deciding A is NP-complete. Therefore solving MSP is NP-hard. ■

4 Reduction and lower bound estimation algorithms

4.1 Reduction algorithm

Having shown in Corollary 3.1 that solving MSP is NP-hard, we cannot expect a polynomial-time algorithm which can find a minimal supervisor. In this section we propose a polynomial-time algorithm which can, however, accomplish supervisor reduction. This algorithm generates a control congruence, whose induced supervisor **J**, as mentioned before, is unique. Notice that the Generalized Quotient Theorem is not valid if "control cover" is replaced by "control congruence". For example, in Fig. 6 we cannot construct a reduced supervisor based on a control congruence of **SUPER** whose size is smaller than |**SUPER**| unless we first split state 1. But we can construct a minimal supervisor based on a control cover because a cover permits "sharing", namely cells of the cover need not be disjoint, effectively allowing unrestricted state splitting. Although the control congruence might yield a result with lower (i.e. less favorable)

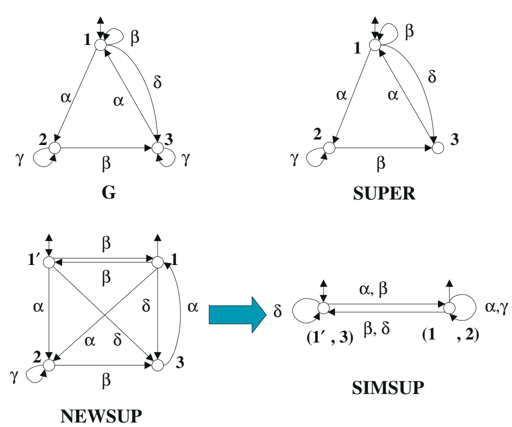

Figure 6: Example 4

compression ratio $\frac{|\mathbf{SUPER}|}{|\mathbf{SIMSUP}|}$, there is some empirical evidence that the sacrifice in compression ratio (if any) is typically rather small.

Let **SUPER** $= (X, \Sigma, \xi, x_0, X_m)$. Suppose $X = \{x_1, \cdots, x_n\}$ and let

$$\mathcal{C} := \{[x] \subseteq X | x \in X \wedge x \in [x]\}$$

be a control congruence on **SUPER**, initially set to be

$$(\forall x \in X)\, [x] := \{x\}$$

Let *waitlist* $\subseteq X \times X$ be a list of state pairs that are waiting to be merged, namely if all subsequent tests return *true* then each state pair will be placed (*merged*) in the same cell of the control congruence that is newly formed.

Reduction algorithm (RA):

main procedure:

for $i = 1$ to $n - 1$

  $\{$**if** $i > \min\{k \in I | x_k \in [x_i]\}$,

    continue;

  for $j = i + 1$ to n

    $\{$**if** $j > \min\{k \in I | x_k \in [x_j]\}$,

      continue;

    $waitlist := \varnothing$;

    flag = Check-Mergibility(x_i, x_j, $waitlist$, x_i);

    if flag $\neq false$

      $\{\tilde{\mathcal{C}} := \left\{[x] \cup \bigcup_{x' : \{(x,x'),(x',x)\} \cap waitlist \neq \varnothing} [x'] \middle| [x], [x'] \in \mathcal{C}\right\}$;

      $\mathcal{C} := \tilde{\mathcal{C}};\}\}\}$

boolean Check-Mergibility(x_i, x_j, *waitlist*, *cnode*)

for each $x_p \in [x_i] \cup \bigcup_{x : \{(x,x_i),(x_i,x)\} \cap waitlist \neq \varnothing} [x]$

2. $\{$**for** each $x_q \in [x_j] \cup \bigcup_{x:\{(x,x_j),(x_j,x)\}\cap waitlist \neq \varnothing}[x]$

3. $\{$**if** $\{(x_p, x_q), (x_q, x_p)\} \cap waitlist \neq \varnothing$,

4. **continue**;

5. **if** $(x_p, x_q) \notin \mathcal{R}$,

6. **return** $false$;

7. $waitlist := waitlist \cup \{(x_p, x_q)\}$;

8. **for** each $\sigma \in \Sigma$ with $\xi(x_p, \sigma)!$ and $\xi(x_q, \sigma)!$

9. $\{$**if** $[\xi(x_p, \sigma)] = [\xi(x_q, \sigma)]$ or $\{(\xi(x_p, \sigma), \xi(x_q, \sigma)), (\xi(x_q, \sigma), \xi(x_p, \sigma))\} \cap waitlist \neq \varnothing$,

10. **continue**;

11. **if** $\min\{k | x_k \in [\xi(x_p, \sigma)]\} < cnode$ or $\min\{k | x_k \in [\xi(x_q, \sigma)]\} < cnode$,

12. **return** $false$;

13. flag = Check-Mergibility($\xi(x_p, \sigma)$, $\xi(x_q, \sigma)$, *waitlist*, *cnode*)

14. **if** flag = $false$,

15. **return** $false$;$\}\}\}$

16. **return** $true$;

For the sake of illustration we apply RA on the supremal supervisor **SUPER** shown in Figure 2. The whole process is depicted in Figure 7. We say "RA terminates" if RA stops after a finite number of line executions in the main procedure and in the subroutine; "Check-Mergibility terminates" has a similar meaning.

Proposition 4.1 RA terminates and the result $\mathcal{C}$ is a control congruence.

Remarks:

1. The function of lines 2-3 and 5-6 in the main procedure and line 11-12 in the subroutine is to maintain pairwise disjointness among elements of $\mathcal{C}$. Only pairs that have not been tested for mergibility can pass through these "filters"; otherwise either redundant

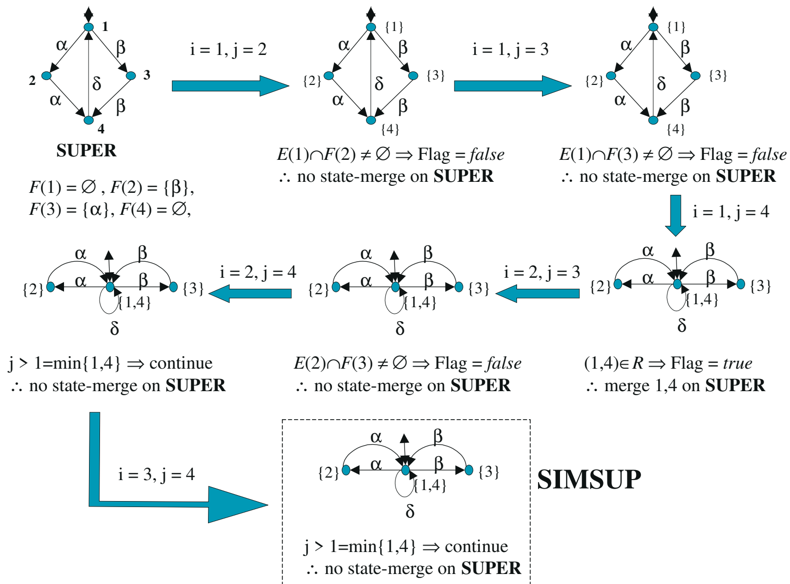

Figure 7: Detailed Reduction Process

tests or element overlapping in $\mathcal{C}$ may occur. This ensures the construction of a control congruence. Such pre-filtering operations can greatly reduce the number of calls to the subroutine Check-Mergibility and speed up the reduction process.

2. In the worst case the main procedure makes $\frac{1}{2}n(n-1)$ calls to subroutine Check-Mergibility, and the latter could make $\frac{1}{2}n(n-1)$ calls to itself. The main operation in the subroutine is to check whether or not $(x_p, x_q) \in \mathcal{R}$. So the overall number of checking operations (calls from the main procedure or recursive calls from the subroutine) is $\frac{1}{2}n(n-1)\frac{1}{2}n(n-1) = \frac{1}{4}n^2(n-1)^2$. Hence the complexity is $O(n^4)$.

4.2 Lower bound on the size of a minimal normal supervisor

In many situations it is useful to estimate the minimal state size because a reasonable (conservative) lower bound could serve as an indicator of when to stop searching. Let K be the supremal controllable sublanguage of $L_m(\mathbf{G}) \cap SPEC$, with $\overline{K}$ its prefix closure.

Let

$$\phi : \overline{K} \to 2^{\Sigma} : s \mapsto \phi(s) := \{\sigma \in \Sigma \,|\, s\sigma \notin \overline{K} \,\&\, s\sigma \in L(\mathbf{G})\}$$

be its control (disablement) function. Let

$$\pi : \overline{K} \to 2^{\Sigma} : s \mapsto \pi(s) := \{\sigma \in \Sigma \,|\, s\sigma \in \overline{K}\}$$

be the exit event function, so that

$$(\forall s \in \overline{K})\, \phi(s) \cap \pi(s) = \varnothing$$

Definition 4.1 For any $s_1,\, s_2 \in \overline{K}$, s_1 and s_2 are *mutually control consistent* if

1. $(\forall s \in \Sigma^*)\, s_1 s,\, s_2 s \in \overline{K} \Rightarrow \phi(s_1 s) \cap \pi(s_2 s) = \phi(s_2 s) \cap \pi(s_1 s) = \varnothing$
2. $s_1,\, s_2 \in L_m(\mathbf{G}) \Rightarrow s_1,\, s_2 \in K$

Definition 4.2 K is *$p-$bounded* if there exists a set $S = \{s_1, s_2, \cdots, s_p\} \subseteq \overline{K}$ such that for any distinct $s, s' \in S$, s and s' are not mutually control consistent. We call p a *lower bound index* of K.

Let Δ_K be the maximum lower bound index of K. Since K is regular it can be represented by a finite-state automaton **SUPER**, which is a supremal supervisor as before.

Proposition 4.2 Let **SIMSUP** be any supervisor which is control equivalent to **SUPER** with respect to **G**. Then $|\mathbf{SIMSUP}| \geq |\Delta_K|$.

Proof: Since **SIMSUP** is control equivalent to **SUPER** with respect to **G**, by definition of control equivalence we have $\overline{K} = L(\mathbf{SUPER}) \subseteq L(\mathbf{SIMSUP})$. Therefore $L(\mathbf{SIMSUP})$ contains at least Δ_K mutually control inconsistent strings, which means **SIMSUP** must have at least Δ_K states to accept those strings individually.

■

To relate Δ_K to a set of states in **SUPER**, we need to impose one further restriction.

Definition 4.3 **SUPER** is *complete* with respect to **G** if

$$(\forall s_1, s_2 \in L(\mathbf{SUPER}))\xi(x_0, s_1) = \xi(x_0, s_2) \Rightarrow \phi(s_1) = \phi(s_2)\,\&\,\eta(y_0, s_1) = \eta(y_0, s_2)$$

Generally, if **SUPER** is not complete then by forming $\mathbf{G} \times \mathbf{SUPER}$ (using the **meet** operation [Won02]) we obtain a complete supremal supervisor. The result of **supcon** is always a complete supremal supervisor. As before let X be the state set of **SUPER**. Let

$$\Omega_{\mathbf{SUPER}} := \max\{k | (\exists X' \subseteq X)\, |X'| = k \wedge (\forall x, x' \in X')\, x \neq x' \Rightarrow (x, x') \notin \mathcal{R}\}$$

Proposition 4.3 If **SUPER** is complete with respect to **G** then $\Omega_{\mathbf{SUPER}} = \Delta_K$.

Proof: By assumption **SUPER** is complete with respect to **G**. It is not difficult to verify that

$$(\forall s, s' \in \overline{K})\ s,\ s' \text{ are mutually control consistent } \iff (\xi(x_0, s), \xi(x_0, s')) \in \mathcal{R}$$

and the proposition follows. ■

We can construct an undirected graph $\mathrm{Gr} = (Ver, Edg)$ with $Ver = X$ and $Edg = \{(x, x') \in X \times X | (x, x') \notin \mathcal{R}\}$ in polynomial time with respect to $|X|$ (this can be done by using the subroutine Check-Mergibility in the reduction algorithm). Calculating $\Omega_{\mathbf{SUPER}}$ is equivalent to finding a maximum clique in Gr, which is NP-hard because the corresponding decision problem is NP-complete [GJ79].

There is extensive literature on maximum clique problems: e.g. algorithms for finding all cliques are given in [BK73], and suboptimal maximum clique algorithms in [PX94]. For simplicity of programming we used a suboptimal maximum clique algorithm based on successive vertex removal. In this we repeatedly remove from the graph a vertex v with the smallest *neighborhood* (set of vertices connected by an edge to v), until the remainder of the graph becomes a clique. The test results in Sect. 4.3 indicate that this simple clique-detection algorithm serves our purpose well. Possible improvements based on the cited references could be explored in future work.

Suppose **SIMSUP** is the result produced by the previous reduction algorithm and lbe is the result of applying our sub-optimal maximum clique algorithm. We know that $lbe \leq \Omega_{\mathbf{SUPER}} \leq |\mathbf{MINSUP}| \leq |\mathbf{SIMSUP}|$. The difference $|\mathbf{SIMSUP}| - lbe$ indicates how close lbe is to $\Omega_{\mathbf{SUPER}}$ and $|\mathbf{MINSUP}|$.

4.3 Test Results

The above reduction algorithm has been realized by a C program. Some test results are provided in Table 1, where computation time is based on an Intel Celeron 300MHz CPU. A pair (k, l) means k states, l transitions; **CRatio** denotes compression ratio, defined as $\frac{|\mathbf{SUPER}|}{|\mathbf{SIMSUP}|}$; lbe is the lower bound estimate for a minimal supervisor; and **CT** is the computation time. In all cases the control-equivalence of **SIMSUP** to **SUPER** was verified directly as indicated in (2), Sect. 2.1. The test cases are ordered by compression ratio. From the table we see that in most cases $\frac{|\mathbf{SIMSUP}| - lbe}{|\mathbf{SIMSUP}|}$ is small, e.g. in cases 8,9, or simply zero, e.g. in cases 3,4,5,7,10, which means **SIMSUP** is close in size to or is actually a minimal supervisor. Further applications, to supervisor design problems from the literature, are provided in [Won02], Sect. 8.14.

Table 1

Item	G	SUPER	SIMSUP	CRatio	CT (sec)	*lbe*	CT (sec)
1	(243,1215)	(341,1005)	(88,492)	3.875	0.051	65	0.596
2	(128,1288)	(776,3826)	(186,1130)	4.172	1.932	60	2.628
3	(9,24)	(183,461)	(41,223)	4.463	0.17	41	0.127
4	(9,24)	(273,691)	(61,333)	4.475	0.352	61	0.291
5	(52,111)	(35,62)	(6,15)	5.833	0	6	0.006
6	(324,1404)	(432,1098)	(56,349)	7.714	0.474	34	0.848
7	(282,604)	(40,80)	(3,18)	13.333	0.019	3	0.011
8	(405,1647)	(872,2112)	(34,182)	25.647	5.053	32	2.479
9	(405,1647)	(1574,3828)	(60,325)	26.233	7.423	58	7.214
10	(390,1089)	(295,701)	(5,64)	59	40.23	5	0.434

5 Conclusion

We have shown that, by revising the definition of control cover in [VW86] but adopting essentially the same approach, a more general treatment of the supervisor reduction problem is possible. Although we have proved that computing a minimal supervisor is NP-hard, we have also provided a useful polynomial-time algorithm for supervisor reduction, based on heuristic search for a control congruence. A corresponding lower bound on the size of a minimal supervisor is a practical index of closeness of the computed supervisor to minimality.

Other related problems of interest for future research include more detailed investigation of algorithmic reduction by covers (see e.g. [Min02]), and supervisor reduction 'on the fly' to avoid the explicit computation of controlled behavior (**SUPER**) altogether.

Acknowledgments : The authors are indebted to R.S. Minhas for helpful discussion. Financial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada, RG7399, is gratefully acknowledged.

Appendix: Proofs of propositions, lemmas and theorems

P-1. Proof of Proposition 2.1:

1. We first claim that $L_m(\mathbf{SUPER}) \subseteq L_m(\mathbf{J})$.

Let $s \in L_m(\mathbf{SUPER})$. If $s = \epsilon$, then $x_0 \in X_m$. Since $x_0 \in X_{i_0}$, we have $X_{i_0} \cap X_m \neq \varnothing$.

Therefore $i_0 \in I_m$, namely $\epsilon \in L_m(\mathbf{J})$. Let $s = \sigma_0 \cdots \sigma_k$ $(k \geq 0)$. Because

$$\xi(x_0, \sigma_0)!,\ \xi(x_0, \sigma_0\sigma_1)!, \cdots,\ \xi(x_0, \sigma_0\sigma_1 \cdots \sigma_k)!$$

we have

$$\xi(x_0, \sigma_0)! \text{ and } \xi(x_j, \sigma_j)! \text{ with } x_{j+1} = \xi(x_0, \sigma_0 \cdots \sigma_j),\ j = 1, \cdots, k$$

Since $\{X_i | i \in I\}$ is a control cover on X, by Def. 2.1 and the definition of κ we have

$$(\forall j : 0 \leq j \leq k)(\exists i_j, i_{j+1} \in I)\, x_j \in X_{i_j} \wedge x_{j+1} \in X_{i_{j+1}} \wedge \kappa(i_j, \sigma_j) = i_{j+1}$$

and therefore

$$\kappa(i_0, \sigma_0\sigma_1 \cdots \sigma_k) = \kappa(i_0, s)!$$

with

$$\xi(x_0, s) \in X_m \Rightarrow \kappa(i_0, s) \in I_m$$

Therefore $s \in L_m(\mathbf{J})$, proving the claim.

It follows from 1. and equation (1) that

$$L_m(\mathbf{SUPER}) \subseteq L_m(\mathbf{G}) \cap L_m(\mathbf{J})$$

and on taking closures that

$$L(\mathbf{SUPER}) = \overline{L_m(\mathbf{SUPER})} \subseteq L(\mathbf{G}) \cap L(\mathbf{J})$$

as required.

2. For the reverse inclusion in (1a), let

$$s \in L(\mathbf{G}) \cap L(\mathbf{J})$$

If $s = \epsilon$ then, as $L(\mathbf{SUPER}) \neq \varnothing$, $s \in L(\mathbf{SUPER})$. Suppose $s = \sigma$. Then $\kappa(i_0, s)!$, so there are $x \in X_{i_0}$ and $x' \in X$ such that $\xi(x, \sigma) = x'$, namely $\sigma \in E_{i_0}(x)$. By definition of the control cover $\mathcal{C}$, $\sigma \notin D_{i_0}(x_0)$, so either $\xi(x_0, \sigma)!$ or

$$(\forall t \in \Sigma^*)\, \xi(x_0, t) = x_0 \Rightarrow \mathit{not}\ \eta(y_0, t\sigma)! \tag{1}$$

But $\xi(x_0, \epsilon) = x_0$, and $s = \sigma \in L(\mathbf{G})$ implies $\eta(y_0, \sigma)!$; thus (3) fails at $t = \epsilon$, and we conclude $\xi(x_0, \sigma)!$. Of course by definition of $\mathcal{C}$, there follows $\xi(x_0, \sigma) = x' \in X_{i'}$ for some $i' \in I$. In general let $s = \sigma_0\sigma_1 \cdots \sigma_k$. Repeating the foregoing argument k-fold, we see that $\kappa(i_0, s)!$ implies $\xi(x_0, s)!$. This shows that $s \in L(\mathbf{SUPER})$.

3. For the inclusion $(\subseteq)$ in (1b) let $s \in L_m(\mathbf{G}) \cap L_m(\mathbf{J})$. As shown above, $\xi(x_0, s)!$ with $\xi(x_0, s) \in \kappa(i_0, s)$. Since $\kappa(i_0, s) \in I_m$, there exists $x' \in X_{\kappa(i_0,s)} \cap X_m$, namely $M(x') = \mathit{true}$. Since $\mathbf{SUPER}$ is reachable and $L_m(\mathbf{SUPER}) \subseteq L_m(\mathbf{G})$, there is $s' \in \Sigma^*$ such that $\xi(x_0, s') = x'$ and $\eta(y_0, s') \in Y_m$. At the same time, $s \in L_m(\mathbf{G})$ implies $\eta(y_0, s) \in Y_m$, resulting in $T_{\kappa(i_0,s)}(x') = T_{\kappa(i_0,s)}(\xi(x_0, s)) = \mathit{true}$. By definition of $\mathcal{C}$, we get

$$M(\xi(x_0, s)) = M(x') = \mathit{true}$$

namely $\xi(x_0, s) \in X_m$, and $s \in L_m(\mathbf{SUPER})$ as required. ■

P-2. Proof of Theorem 2.1:

With $\mathbf{SUPER} = (X, \Sigma, \xi, x_0, X_m)$, $\mathbf{SIMSUP} = (Z, \Sigma, \zeta, z_0, Z_m)$, and $z \in Z$, let

$$X(z) := \{x \in X \mid (\exists s \in L(\mathbf{SUPER}))\, \xi(x_0, s) = x \,\&\, \zeta(z_0, s) = z\}$$

and define

$$\mathcal{C} := \{X(z) \mid z \in Z\}$$

1. We check that $\mathcal{C}$ is a control cover on **SUPER**.

By normality $X(z) \neq \varnothing$ for all $z \in Z$. Also, if $x \in X$, by reachability of **SUPER** there is $s \in L(\mathbf{SUPER}) = L(\mathbf{G}) \cap L(\mathbf{SIMSUP})$ with $\xi(x_0, s) = x$ and $z := \zeta(z_0, s)!$, hence $x \in X(z)$. This shows that $\mathcal{C} = \{X(z) \,|\, z \in Z\}$ covers X.

Next, fix $z \in Z$ and let $a, b \in X(z)$ with $\sigma \in E_z(a)$. Then for some $s \in L(\mathbf{SUPER})$, $\xi(x_0, s) = a$ and $\xi(a, \sigma)!$. So $s\sigma \in L(\mathbf{SUPER})$, hence $s\sigma \in L(\mathbf{SIMSUP})$, so $\zeta(\zeta(z_0, s), \sigma)!$, namely $\zeta(z, \sigma)!$. Thus for any $t \in \Sigma^*$ with $\xi(x_0, t) = b \in X(z)$ and $\eta(y_0, t\sigma)!$, i.e. $t\sigma \in L(\mathbf{G})$, we have $\zeta(z_0, t) = z$ and

$$t\sigma \in L(\mathbf{G}) \cap L(\mathbf{SIMSUP}) = L(\mathbf{SUPER})$$

So $\xi(x_0, t\sigma) = \xi(b, \sigma)!$. This shows that $\sigma \notin D_z(b)$, namely $E_z(a) \cap D_z(b) = \varnothing$ as required.

Next let $z \in Z$, $\sigma \in \Sigma$. We must find $\tilde{z} \in Z$ with the property

$$(\forall x \in X(z))\, \xi(x, \sigma)! \Rightarrow \xi(x, \sigma) \in X(\tilde{z})$$

Assume $\xi(x_0, s) = x \in X(z)$ and $\xi(x, \sigma)!$. We have $\zeta(z_0, s) = z$ and $s\sigma \in L(\mathbf{SUPER}) = L(\mathbf{G}) \cap L(\mathbf{SIMSUP})$, so $\zeta(z, \sigma)!$. Choose $\tilde{z} := \zeta(z, \sigma)$. Then $\xi(x, \sigma) = \xi(x_0, s\sigma)$ with $\zeta(z_0, s\sigma) = \tilde{z}$, namely $\xi(x, \sigma) \in X(\tilde{z})$ as required.

Finally let $z \in Z$ and $a, b \in X(z)$ with $M_z(a) \neq M_z(b)$, say $M_z(a) = \mathit{true}$ and $M_z(b) = \mathit{false}$. Since $M_z(a) = \mathit{true}$ and $a \in X(z)$, there exists $s \in L_m(\mathbf{SUPER})$ with $\xi(x_0, s) = a$ and $\zeta(z_0, s) = z$. Since $L_m(\mathbf{SUPER}) = L_m(\mathbf{G}) \cap L_m(\mathbf{SIMSUP})$, we have $s \in L_m(\mathbf{G}) \cap L_m(\mathbf{SIMSUP})$, namely $T_z(a) = \mathit{true}$ and $z \in Z_m$. With $b \in X(z)$, let $s' \in L(\mathbf{SUPER})$ be any string with $\xi(x_0, s') = b$ and $\zeta(z_0, s') = z \in Z_m$. From $M_z(b) = \mathit{false}$ we get that $s' \notin L_m(\mathbf{SUPER})$ which means either $s' \notin L_m(\mathbf{G})$ or $s' \notin L_m(\mathbf{SIMSUP})$. Since $z \in Z_m$ implies that $s' \in L_m(\mathbf{SIMSUP})$, we get $s' \notin L_m(\mathbf{G})$. Therefore $T_z(b) = \mathit{false}$. So

$$M_z(a) \neq M_z(b) \Rightarrow T_z(a) \neq T_z(b)$$

or equivalently $T_z(a) = T_z(b) \Rightarrow M_z(a) = M_z(b)$, as required.

2. To complete the proof it must be shown that the control cover $\mathcal{C}$ has the required DES-isomorphism property: namely some one of its (possibly many) induced supervisors, say

$$\mathbf{J} = (I, \Sigma, \kappa, i_0, I_m)$$

(in the notation used before) is DES-isomorphic to **SIMSUP**. For this let

$$I = Z \text{ and } i_0 = z_0$$

As described before,

$$I_m = \{z \,|\, X(z) \cap X_m \neq \varnothing\}$$

Since **SIMSUP** is normal, $(\forall z \in Z_m)(\exists s \in L_m(\mathbf{SUPER}))\, \zeta(z_0, s) = z$, namely for any $z \in Z_m$, $X(z) \cap X_m \neq \varnothing$, hence $Z_m \subseteq I_m$. By definition of $X(z)$ and the assumption that $L_m(\mathbf{SUPER}) \subseteq L_m(\mathbf{SIMSUP})$, we have

$$\begin{aligned} X(z) \cap X_m \neq \varnothing \quad &\Rightarrow \quad (\exists s \in \Sigma^*) s \in L_m(\mathbf{SUPER}) \,\&\, \zeta(z_0, s) = z \\ &\Rightarrow \quad s \in L_m(\mathbf{SIMSUP}) \,\&\, \zeta(z_0, s) = z \\ &\Rightarrow \quad z \in Z_m \end{aligned}$$

So $I_m \subseteq Z_m$, and therefore $I_m = Z_m$. Let

$$\kappa : I \times \Sigma \to I$$

be the partial transition function with

$$\kappa(z, \sigma) = z' \text{ if } \zeta(z, \sigma) = z' \,\&\, (\exists x)\, x \in X(z) \,\&\, \xi(x, \sigma) \in X(z')$$

As shown in part 1 of the proof, if $\zeta(z, \sigma) = z'$ then

$$(\forall x)\, x \in X(z) \,\&\, \xi(x,\sigma)! \Rightarrow \xi(x,\sigma) \in X(z')$$

so κ is well-defined. Obviously

$$\kappa(z,\sigma) = z' \Rightarrow \zeta(z,\sigma) = z'$$

Since **SIMSUP** is normal, $\zeta(z,\sigma) = z'$ implies that

$$(\exists s \in L(\mathbf{SUPER}))\, \zeta(z_0, s) = z \,\&\, s\sigma \in L(\mathbf{SUPER})$$

so $\xi(x_0, s) \in X(z)$ and $\xi(\xi(x_0,s),\sigma)!$. By definition of κ we have $\kappa(z,\sigma) = z'$. Now we have

$$\kappa(z,\sigma) = z' \iff \zeta(z,\sigma) = z'$$

So the identity is a DES-isomorphism between **J** and **SIMSUP**. ■

P-3. Proof of lemma 3.3:

1. (Only if) Suppose **Gr** has a partition into r cliques $\{V_1, \cdots, V_r\}$, $r \leq R$. We may assume $v_0 \in V_1$, and set

$$\begin{aligned} V_0 &:= \{v_{ij} \,|\, v_{ij} \in X\} \\ \mathcal{C} &:= \{V_0 \cup V_1, V_2, \cdots, V_r\} \end{aligned}$$

Obviously $\mathcal{C}$ is a partition on X.

Claim: $(\forall v_i, v_j \in Ver)\, (v_i, v_j) \in Edg \iff E(v_j) \cap D(v_i) = E(v_i) \cap D(v_j) = \varnothing$.

Proof of Claim: By TC we see that (1) If $i \geq j$ then

$$E(v_j) \cap D(v_i) \subseteq \{\alpha_j, \beta_{pj} \in \Sigma | p < j\} \cap \{\alpha_i, \beta_{iq} \in \Sigma | q > i\} = \varnothing$$

(2) If $i < j$ then $(v_i, v_j) \notin Edg \iff \beta_{ij} \in E(v_j) \cap D(v_i)$. Since $E(v_j) \cap D(v_i) \subseteq \{\beta_{ij}\}$, we get that $(v_i, v_j) \in Edg \iff \beta_{ij} \notin E(v_j) \cap D(v_i) \iff E(v_j) \cap D(v_i) = \varnothing$.

By (1) and (2) the Claim is true.

By the Claim with the fact that $E(v_{ij}) = D(v_{ij}) = \varnothing$ for all $v_{ij} \in V_c$ and $X_m = X$, we obtain that

$$(\forall V \in \mathcal{C})(\forall v_i, v_j \in V)\,(v_i, v_j) \in \mathcal{R}$$

Considering that $(\forall \sigma \in \Sigma)(\exists! x \in X)\,\xi(x, \sigma)!$, we conclude that $\mathcal{C}$ is a control congruence, in particular a control cover by Definition 2.1.

2. (If) Assume that for some $r \leq R$,

$$\mathcal{C} = \{W_1, \cdots, W_r\}$$

is a control cover on **GrSUPER**, and set $V_l := W_l \cap Ver$ $(l = 1, \cdots, r)$. Suppose $v_i, v_j \in V_l$ with $i < j$; it must be shown that $(v_i, v_j) \in Edg$. By the Claim, the latter is true if $E(v_j) \cap D(v_i) = E(v_i) \cap D(v_j) = \varnothing$; but this follows from $V_l \subseteq W_l$ and condition 1 in the definition of $\mathcal{R}$. Therefore for each l $(l = 1, \cdots, r)$, V_l is a clique. If V_l and V_q are not disjoint, then simply take $V_l' := V_l - V_q$ and $V_q' := V_q$. Iterate until each pair of sets in $\{V_l' | l = 1, \cdots, r\}$ are disjoint. Clearly this can be done in finite steps and the resulting set $\{V_l' | l = 1, \cdots, r\} - \{\varnothing\}$ is a clique partition on Ver. Therefor PIC$(\mathbf{G_r}, R)$ is solvable.■

P-5. Proof of Proposition 3.1:

Given a graph $\mathbf{Gr} =< Ver, Edg >$, apply conversion algorithm TC to it and produce **GrG** and **GrSUPER**. We need to show that

$$< \mathbf{Gr}, R > \in B \iff < \mathbf{GrG}, \mathbf{GrSUPER}, R > \in A$$

(1) If $< \mathbf{Gr}, R > \in B$, namely **Gr** contains a partition of cliques with no more than R cells, then by Lemma 3.3, **GrSUPER** has a control cover with no more than R cells.

By Proposition 2.1, the supervisor induced by the control cover is control-equivalent to the supremal supervisor **GrSUPER**. Therefore **GrG** has an equivalent supervisor with no more than R states, namely

$$< \mathbf{GrG}, \mathbf{GrSUPER}, R >\in A$$

(2) If $< \mathbf{GrG}, \mathbf{GrSUPER}, R >\in A$, then **GrG** has a supervisor **SIMSUP** with no more than R states, which is control-equivalent to **GrSUPER**. By Theorem 2.1, there is a control cover $\mathcal{C}$ on **GrSUPER** such that the induced supervisor is DES-isomorphic to **SIMSUP**. By Lemma 3.3 we have $< \mathbf{Gr}, R >\in B$.

By (1) and (2), we have

$$< \mathbf{Gr}, R >\in B \iff < \mathbf{GrG}, \mathbf{GrSUPER}, R >\in A$$

Since TC is a polynomial-time computation as mentioned before, B is polynomial-time-mapping-reducible to A. ■

P-6. Proof of Proposition 4.1:

(Sketch) Proof: (1) To show that RA terminates, it is sufficient to show that for each pair (i, j) in the main procedure, Check-Mergibility(x_i, x_j, *waitlist*, *cnode*) terminates. By the pseudocode, Check-Mergibility can execute line 8-16 (recursive call) only after executing line 7 (increase the size of *waitlist*). Therefore we have a sequence of recursive-call subroutines

$$[waitlist_1, \mathrm{CM}_1^1, \cdots, \mathrm{CM}_1^{r_1}] \rightarrow \cdots \rightarrow [waitlist_m, \mathrm{CM}_m^1, \cdots, \mathrm{CM}_m^{r_m}] \rightarrow \cdots$$

where $r_1 = 1$, CM_1^1 is the subroutine called by the main procedure, $waitlist_1 = \varnothing$ initialized in the main procedure and CM_k^l $(1 \leq l \leq r_k,\ k = 2, \cdots)$ the subroutine recursively called. By line 3-4 and line 7 we get that $(\forall k)|waitlist_{k+1}| = |waitlist_k| + 1$,

and by line 8 that $(\forall k)\, r_k \leq |\Sigma|$. Since *waitlist* can have at most $\frac{1}{2}n(n-1)$ elements, the above sequence is finite, say with $q \leq \frac{1}{2}n(n-1)$ terms. Then for $k = q, q-1, \cdots, 1$ and $l = k, \cdots, 1$, each copy $\mathrm{CM}_k^{r_l}$ will return either $true$ or $false$ in a finite number ($\leq n^2(5+7|\Sigma|)+1$) of line executions. Therefore after a finite number of line executions Check-Mergibility(x_i, x_j, *waitlist*, *cnode*) returns a value and terminates, as required.
(2) Second we show that when RA terminates, the resulting $\mathcal{C}$ is a control congruence. Initially $\mathcal{C} := \{[x] = \{x\} | x \in X\}$ which is obviously a control congruence. When $i = 1$ and $j = 2$, if Check-Mergibility(x_1, x_2, *waitlist*, *cnode*)$=$ $false$, then $\mathcal{C}$ won't change (no execution of line 10-13 in the main procedure), namely it is still a control congruence. If Check-Mergibility(x_1, x_2, *waitlist*, *cnode*)$=$ *true*, then by line 1,2 and 5-6 in CM we can derive that

$$(\forall U \in \tilde{\mathcal{C}})(\forall x, x' \in U)\, (x, x') \in \mathcal{R}$$

Also by line 1,2 and 8-16 in CM we can derive that

$$(\forall U \in \tilde{\mathcal{C}})(\forall \sigma \in \Sigma)(\exists U' \in \tilde{\mathcal{C}})[(\forall x \in U)\xi(x,\sigma)! \Rightarrow \xi(x,\sigma) \in U']$$

Hence by Def. 2.1 $\tilde{\mathcal{C}}$ is a control cover. Furthermore by line 2-3, 5-6 in the main procedure and line 11-12 in the subroutine, there is no state sharing among elements in $\tilde{\mathcal{C}}$. Hence $\tilde{\mathcal{C}}$ is a control congruence. By line 12 $\mathcal{C}$ is a control congruence. Applying the same argument we can show that for each pair (i, j), the resulting $\mathcal{C}$ (generated at line 12 in the main procedure) is a control congruence. Therefore when RA terminates the resulting $\mathcal{C}$ is a control congruence, as required. ■

References

[BK73] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *CACM*, 16(9):575–577, 1973.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H.Freeman and Company, New York, USA, 1979.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[Min02] R.S. Minhas. *Complexity Reduction in Discrete Event Systems*. PhD Thesis, ECE Dept., Univ. of Toronto, 2002.

[Pfl73] C.P. Pfleeger. State reduction in incompletely specified finite-state machines. *IEEE Trans. on Computers*, C-22(12):1099–1102, 1973.

[PO98] J.M. Pena and A.L. Oliveira. A new algorithm for the reduction of incompletely specified finite-state machines. In *Proc. ICCAD98*, San Jose, CA, November 1998.

[PX94] P.M. Pardalos and J. Xue. The maximum clique problem. *Global Optimization*, 4:301–328, 1994.

[QC02] M.H. de Queiroz and J.E.R. Cury. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In: *M. Silva, A. Giua, J.M. Colom (Eds.), Proc. 6th International Workshop on Discrete Event Systems (WODES'02), Zaragoza, Spain, 2-4 October 2002*, pages 377–382. IEEE Computer Society, 2002.

[RW87] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.

[VW86] A.F. Vaz and W.M. Wonham. On supervisor reduction in discrete-event systems. *International J. Control*, 44(2):475–491, 1986.

[Won02] W. M. Wonham. *Notes on Control of Discrete-Event Systems: ECE 1636F/1637S 2002-2003.* Systems Control Group, Dept. of ECE, University of Toronto, 2002. Posted at URL: www.control.utoronto.ca/people/profs/wonham/wonham.html.

[WR87] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, 1987.