Effective Control Synthesis for DES Under Partial Observations

Hichem Lamouchi and John Thistle
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

thistle@kingcong.uwaterloo.ca

Abstract

A procedure is given for the effective solution of an infinite-string supervisory control problem under partial observations, for the case where the plant and specification languages are represented by finite ω-automata (automata on infinite strings), and the observation mask by a finite Moore automaton. This solves an ω-language version of the standard centralized supervisory control problem under partial observations. It is shown that a natural extension to decentralized control is undecidable, even if the plant, the specification languages and the masks are represented by finite automata. This undecidability result carries over to the finite-string case.

1 Introduction

This article reports positive and negative results on effective computation of supervisors for partially-observed discrete event systems (DES). First, a supervisory control problem under partial observations is effectively solved. This extends earlier results by taking account of infinite-string languages, represented by finite automata on infinite strings, or ω-*automata*. While partial observations naturally induce a nondeterministic automaton model (or, more precisely, a universal or $\forall$-automaton model), it turns out that the complex procedure for "determinization" of ω-automata can be avoided in the computation of a solution to our problem. The result also extends earlier work by allowing for a more general observation mechanism – that of arbitrary causal, or "prefix-preserving," maps from the event alphabet to an observations alphabet; because such maps do not merely project event strings onto substrings of observable events, this extension requires a proper account of the role of control inputs in observation.

The main idea of our construction is to reduce the problem to the case of complete observations. In general, the control mechanism for the complete-observations version is such that the set of "control patterns" is not closed under union; this reflects the fact that (even under the standard control mechanism) the partial-observations problem need not have a maximally permissive solution.

The aforementioned procedure constitutes a control synthesis algorithm for centralized supervision. A decentralized version of the synthesis problem turns out to be undecidable. This negative result is established by adapting results on synthesis of reactive systems and multiple-person games of incomplete information [1, 2]. The result applies not only to infinite-string synthesis problems, but also to simple variations of finite-string problems that have already been studied in the literature.

In section 2 we recall the necessary preliminary notions. Section 3 defines a framework for centralized supervisory control problem with partial observations. In section 4 a control synthesis problem

is defined and it is shown how results from the ω-language theory of supervisory control can be used to compute solutions to this control problem. Section 5 shows that the extension of the problem to decentralized control is undecidable. Section 6 gives some concluding remarks.

2 Preliminaries

Let Σ be a finite alphabet. Σ^* denotes the set of all finite strings over Σ, including the empty string, 1_Σ. The expression Σ^ω represents the set of infinite strings over Σ. The union of Σ^* and Σ^ω is denoted by Σ^∞.

Any $L \subseteq \Sigma^*$ is called a $*$-*language* over Σ and any $S \subseteq \Sigma^\omega$ an ω-*language* over Σ.

For any $k \in \Sigma^*$ and $v \in \Sigma^\infty$, kv stands for the concatenation of the two strings.

Define the map pre : $2^{\Sigma^\infty} \longrightarrow 2^{\Sigma^*}$ by $\text{pre} : V \mapsto \{k \in \Sigma^* : (\exists v \in V)(k \leq v)\}$. Thus, $\text{pre}(V)$ is the set of all finite prefixes of strings in V. For any $R \subseteq \Sigma^\omega$, we call $\text{pre}(R) \subseteq \Sigma^*$ the *prefix* of R. For any $K \subseteq \Sigma^*$, we call $\text{pre}(K)$ the $*$-*closure* of K. If $K = \text{pre}(K)$ we say that K is $*$-*closed*.

The *limit* of a $*$-language K is the set of all infinite strings whose prefixes are all contained in K. The operator $\text{clo} : 2^{\Sigma^\omega} \longrightarrow 2^{\Sigma^\omega}$ gives for every ω-langage R the set of all infinite strings all of whose prefixes are contained in $\text{pre}(R)$. We call $\text{clo}(R)$ the ω-*closure* of R. If $R = \text{clo}(R)$ we say that R is ω-*closed*; if $R = \text{clo}(R) \cap S$ (where $S \subseteq \Sigma^\omega$) we say that R is closed *relative to* or *with respect to* S.

3 Control of DES under partial observations

3.1 Discrete event systems

We model a DES as a pair $G = (L(G), S(G)) \in 2^{\Sigma^*} \times 2^{\Sigma^\omega}$ consisting of a $* - language$ $L(G)$ called the $* - behaviour$ of G and an $\omega - language$ $S(G)$ called the $\omega - behaviour$ of G [3].

The $*-$behaviour $L(G)$ is assumed to be $*-closed$, i.e., $\text{pre}\,(L(G)) = L(G)$. We also assume that $\text{pre}\,(S(G)) \subseteq L(G)$. If the reverse inclusion holds, then we say that G is *deadlock-free*.

We represent a DES by a Rabin automaton (see, for example, [4, 5]):

$$\mathcal{G} = (\Sigma, X, \delta, x_0, \{(R_p, I_p), p \in P\})$$

where Σ is the alphabet of events, X is the set of states, $\delta{:}\Sigma \times X \longrightarrow X$ is the transition function, x_0 is the initial state and $\forall p \in P$, $\{(R_p, I_p) \in 2^X \times 2^X$ defines a *Rabin acceptance condition*: an infinite string is accepted by the automaton if a corresponding path through the transition structure of the automaton visits R_p infinitely often and I_p "almost always," for some $p \in P$. The ω-languages accepted by such Rabin "ω-automata" are called ω-*regular*. The Rabin ω-automaton $(\Sigma, X, \delta, x_0, \{(R_p, I_p), p \in P\})$ accepts $S(G)$ and the $* - automaton$ $(\Sigma, X, \delta, x_0, X)$ accepts $L(G)$. These languages will also be denoted $S(\mathcal{G})$ and $L(\mathcal{G})$, respectively.

The Rabin automaton introduced above is deterministic, in the sense that the transition function assigns (at most) a single successor state to the current state and alphabet symbol. Consequently, a given symbol string has at most a single corresponding path through the transition structure of the automaton. One may more generally employ a transition function of the form

$$\delta : \Sigma \times X \longrightarrow 2^X$$

(where 2^X denotes the powerset of X), in which case there may be many successor states; consequently, a given string may have many possible paths through the transition structure. A *universal* automaton accepts such a string if *all* corresponding paths satisfy the Rabin acceptance condition.

3.2 Partially-observed DES

We assume that G is only partially observed, which means that the supervisor cannot observe some events and cannot distinguish among others. We model this situation by introducing an observation

mask between the DES and the supervisor :

$$\begin{array}{rcl} M : L(G) & \longrightarrow & \Sigma_o^* \\ 1_\Sigma & \longmapsto & 1_{\Sigma_o} \\ \sigma_1\sigma_2...\sigma_n & \longmapsto & M(\sigma_1\sigma_2...\sigma_{n-1})a \\ & \text{or} & \\ \sigma_1\sigma_2...\sigma_n & \longmapsto & M(\sigma_1\sigma_2...\sigma_{n-1}) \end{array}$$

where

Σ_o is the *observable* alphabet,

and

a is the symbol observed by V when G generates the event σ_n after generating the sequence $\sigma_1\sigma_2...\sigma_{n-1}$.

When $a = \sigma_n$, we say that σ_n is *completely observable* after the sequence $\sigma_1\sigma_2...\sigma_{n-1}$. Otherwise ($a \neq \sigma_n$) we say that σ_n is *partially observable* after the generation of $\sigma_1\sigma_2...\sigma_{n-1}$. The last clause of the definition applies if there is no observation, in which case σ_n is said to be *completely unobservable* after $\sigma_1\sigma_2...\sigma_{n-1}$. Note how the observation mask has memory in the sense that the observation of an event depends on the preceding actions that had been made.

We extend M to $*$-langages in the standard way:

$$M(K) := \{M(s) : s \in K\}, \forall K \subseteq L(G).$$

It is also natural to extend M to ω-languages by defining

$$M(K) := \lim(M(\text{pre}(K))), \forall K \subseteq S(G).$$

3.3 Supervisors

To the DES G we add a means of control: we associate with the alphabet Σ a nonempty family $C \subseteq 2^\Sigma$ of *control patterns*, $C = \{\Gamma_i, 1 \leq i \leq n\}$. A *supervisor* of (G, M) (G partially observed through the observation mask M) is a partial function $V : \Sigma_o^* \longrightarrow C$.

The action of the supervisor V on the DES G is given by the DES V/G, where

(i) $L(V/G)$, the $*$-language synthesized by V, is defined by the following recursion:

(a) $1_\Sigma \in L(V/G)$.

(b) For all $k \in \Sigma^*$, $\sigma \in \Sigma$

$$k\sigma \in L(V/G) \iff k \in L(V/G) \text{ and } M(k) \in dom(V) \text{ and } k\sigma \in L(G) \text{ and } \sigma \in V(M(k))$$

(ii) $S(V/G)$, the ω-language synthesized by V, is given by

$$S(V/G) = \lim(L(V/G)) \cap S(G).$$

We say that V is a *complete* supervisor for G if $M(L(V/G)) \subseteq \text{dom}(V)$, and we say that V is *deadlock-free* if V/G is a deadlock-free DES.

4 Effective control synthesis

4.1 The supervisory control and observation problem for ω-languages (SCOP$^\omega$)

Our aim is to solve the following problem :

Problem 4.1 Given the DES $G = (L(G), S(G))$ (which is the process under control), defined on the alphabet Σ, and an $\omega-$language $E \subseteq \Sigma^\omega$ such that $E \subseteq S(G)$, construct a complete, deadlock-free supervisor V for G such that

$$S(V/G) \subseteq E$$

To simplify the computation, we restrict attention to the case where the ω-behaviour $S(G)$ of the plant is $\lim(L(G))$. Thus we assume that G is represented by a finite Rabin automaton with a trivial acceptance condition:

$$\mathcal{G} = (\Sigma, X_G, \delta_G, x_{0_G}, (X_G, X_G))$$

such that the $*$-automaton $(\Sigma, X_G, \delta_G, x_{0_G}, X_G)$ recognizes $L(G)$. We assume also, without loss of generality [4], that the specification E is represented by a finite, universal Rabin automaton with a total transition function and a single accepting state-subset pair.

$$\mathcal{R} = (\Sigma, X_E, \delta_E, x_{0_E}, (R_E, I_E)).$$

We finally assume that the observation mask $M : \Sigma^* \longrightarrow \Sigma_o^*$ is represented by a finite Moore automaton :

$$\widehat{\mathcal{A}}_M = \left(\Sigma, X_M, \widehat{\delta}_M, \widehat{x}_{0_M}, \Sigma_o, \widehat{\lambda}\right)$$

4.2 Controlled Discrete-Event Systems (CDES)

The control synthesis problem is complicated by the fact that among the information at the supervisor's disposal is not only that provided by the mask M but also that contained in the sequence of control actions that the supervisor itself has applied. To render this information more explicit, we enrich the alphabet by adding the control patterns to the event symbols, bringing in the *controlled DES*:

$$G_C = (\Sigma \times C, X, \delta_C, x_0, \{(R_p, I_p), p \in P\})$$

where $\delta_C : (\Sigma \times C) \times X \longrightarrow X$ is defined as:

$$\delta_C((\sigma, \Gamma), x) = \begin{cases} \delta(\sigma, x) & \text{if } \delta(\sigma, x) \text{ is defined} \\ & \text{and } \sigma \in \Gamma \\ \text{undefined} & \text{otherwise} \end{cases}$$

We associate with the alphabet $\Sigma \times C$ the following family of control patterns $C_C = \{\Gamma_{C_i}, 1 \leq i \leq n\}$ where $\Gamma_{C_i} = \{(\sigma, \Gamma_i) : \sigma \in \Gamma_i\}$. (We say that Γ_{C_i} is induced by Γ_i.)

We complete the enrichment of the alphabet by extending the mask M to strings over $\Sigma \times C$. We define the map $\widetilde{M}_C$ which produces a "place-holder," ϵ, for completely unobservable events:

$$\begin{array}{rrcl} \widetilde{M}_C : & L(G_C) & \longrightarrow & ((\Sigma_o \cup \{\epsilon\}) \times C)^* \\ & 1_{\Sigma \times C} & \longmapsto & 1_{(\Sigma_o \cup \{\epsilon\}) \times C} \\ & (\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) & \longmapsto & \widetilde{M}_C((\sigma_1, \Gamma_1)...(a, \Gamma_n) \\ & & \text{or} & \\ & (\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) & \longmapsto & \widetilde{M}_C((\sigma_1, \Gamma_1)...(\epsilon, \Gamma_n) \end{array}$$

Of course, these ϵ events must not be exploited in a control law, so we adapt the definition of a supervisor accordingly:

Definition 4.2 Let G be DES, G_C the controlled DES associated to G, and $\widetilde{M}_C$ the observation mask defined above. A *supervisor for* $\left(G_C, \widetilde{M}_C\right)$ is a partial function

$$\widetilde{V}_C : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_C$$

which satisfies the following condition :
$\forall m_1, m_2 \in (\Sigma \times C)^*$

$$\pi_\epsilon(\widetilde{M}_C(m_1)) = \pi_\epsilon(\widetilde{M}_C(m_2)) \quad \Rightarrow \quad \begin{array}{l} \widetilde{V}_C\left(\widetilde{M}_C(m_1)\right) = \\ \widetilde{V}_C\left(\widetilde{M}_C(m_2)\right). \end{array}$$

where π_ϵ is the natural projection from $(\Sigma_o \cup \{\epsilon\}) \times C$ to $\Sigma_o \times C$

4.3 Control of Rabin automata

We can prove that the supervision of (G, M) is equivalent to the supervision of $(G_C, \widetilde{M}_C)$, so in order to resolve our intial problem, we have to find a complete deadlock-free supervisor for $\left(G_C, \widetilde{M}_C\right)$.

The first thing to do is to construct the automaton

$$\mathcal{G}_C = (\Sigma \times C, X_G, \delta_{G_C}, x_{0_G}, (X_G, X_G))$$

which represents the CDES G_C, and also the universal Rabin automaton

$$\mathcal{R}_C = (\Sigma \times C, X_E, \delta_{E_C}, x_{0_E}, (R_E, I_E)).$$

which represents the specification on the closed-loop behaviour $\widetilde{V}_C/G_C$ (which is $E_C = P^{-1}(E)$).

Then we combine the automata representing G_C and the specification language E_C, so we compute the product $\mathcal{G}_C \times \mathcal{R}_C$. The result of this operation is the automaton

$$\mathcal{A}_C = (\Sigma \times C, X, \delta_C, x_0, (R, I)).$$

Our control synthesis problem is equivalent to that of controlling $\mathcal{A}_C$ to the satisfaction of its own acceptance condition. However, $\mathcal{A}_C$ being partially observed (and, in general, universal rather than deterministic), we cannot apply the known control synthesis methods of [5] to it directly. Our next step is therefore to transform $\mathcal{A}_C$ into an automaton defined over the observation alphabet.

4.3.1 Control of a universal automaton over $(\Sigma_o \cup \{\epsilon\}) \times C$

If we apply the observation mask $\widetilde{M}_C$ to the automaton $\mathcal{A}_C$ we obtain a universal automaton $\mathcal{A}_o$ that accepts precisely the set of observation sequences that are produced only by the generation of legal strings.

Consider the Rabin automaton of the previous section

$$\mathcal{A}_C = (\Sigma \times C, X, \delta_C, x_0, \{(R_p, I_p),\, p \in P\})$$

Assume that $\widetilde{M}_C$ is represented by a finite Moore automaton :

$$\mathcal{A}_M = (\Sigma \times C, X_M, \delta_M, x_{0_M}, \Sigma_o \cup \{\epsilon\}, \lambda)\,.$$

then it is possible to construct the automaton $\mathcal{A}_o = \left((\Sigma_o \cup \{\epsilon\}) \times C, Y_o, \delta_o, y_0, \left\{(R_{o_p}, I_{o_p}), p \in P\right\}\right)$: that represents $G_o = \left(\widetilde{M}_C\left(L(G_C)\right), \widetilde{M}_C\left(S(G_C)\right)\right)$. We associate with G_o the following family of control patterns, induced in a natural way by those of C:

$$C_o = \{\Gamma_{o_i}, 1 \leq i \leq n\},$$

where
$\Gamma_{i_o} = \{(\sigma, \Gamma_i) : \sigma \in \Sigma_o$ and $\exists \sigma' \in \Gamma_i, x_M \in X_M$
such that $\delta_M\left((\sigma', \Gamma_i), x_M\right)$ is defined and
$\lambda\left(\delta_M\left((\sigma', \Gamma_i), x_M\right)\right) = \sigma\}$

We write $\Gamma_{i_o} = \widetilde{M}_C(\Gamma_{C_i})$, where $\Gamma_{C_i} = \{(\sigma, \Gamma_i) : \sigma \in \Gamma_i\}$ and we say that Γo_i is the *observed image* of Γ_{C_i}.

4.3.2 Control of a deterministic Rabin automaton

While the automaton $\mathcal{A}_o$ can be considered completely observed (since it is defined on the observation alphabet) it is, in general, universal. This complicates control synthesis, as there may be many paths corresponding to a given sequence of observations, and a controller must naturally apply the same sequence of control patterns along any two such paths. We shall solve this problem by converting $\mathcal{A}_o$ into a deterministic automaton that, for our purposes, is equivalent.

The construction is inspired by the determinization procedure for finite automata on words, and also by results on determinization of Büchi automata [4]. Thus, after generation of a given string k, the state z of the deterministic automaton will record the set of all states in which the universal automaton may find itself after generating strings that have the same projection as k . In fact, the state of the deterministic automaton $\mathcal{A}_d$ will be a pair $z = (z^1, z^2)$ of two state subsets: z^1 will be the subset of $\mathcal{A}_o$-states that are reachable by the word in question and z^2 those that are reachable by words that have the same projection as the given word.
We further enrich the state set by "colouring" the states of $\mathcal{A}_o$ according to the following rules:

- At the beginning, all the elements are coloured "white".
- Each element of an $\mathcal{A}_d$-state is coloured "green" if it is in R_o or all its predecessors are coloured "green".
- Each element of an $\mathcal{A}_d$-state is coloured "red" if it is not in I_o.

While reading a word w we identify "breakpoints". The initial state is one breakpoint. Furthermore, a "red breakpoint" occurs whenever any $\mathcal{A}_o$-state is marked red; a "green breakpoint" occurs when all $\mathcal{A}_o$-states in the current $\mathcal{A}_d$-state are marked green. At breakpoints, the corresponding colours are removed from the $\mathcal{A}_o$-states. Intuitively, an $\mathcal{A}_o$-state is green if *all* paths that lead to it have passed through R_o since the last green breakpoint; an $\mathcal{A}_o$-state is red if *some* path that leads to it has passed outside of I_o since the last red breakpoint. The word w is accepted if, on reading w, the automaton passes through infinitely many green breakpoints but only finitely many red breakpoints (Note that this is a Rabin acceptance condition).

4.3.3 Epsilon-invariance

We must force the controller for our deterministic Rabin automaton to ignore ϵ-events. For this we compute a product of $\mathcal{A}_d$ with the automaton $\mathcal{A}_i$ that rejects any strings along which there is a change of

control pattern after an ϵ-event. This will ensure the *ϵ-invariance* of the languages generated by the resulting automaton:

The stucture of the Büchi automaton $\mathcal{A}_i$ is the following :

$$\mathcal{A}_i = ((\Sigma \cup \{\epsilon\}) \times C, X_{co}, \delta_{co}, x_{0_{co}}, T_{co})$$

with

$$\begin{array}{l} X_{co} = \{x_{0_{co}}, x_1, x_2, ..., x_n, x_p\},\ n = |C|. \\ T = X_{co} \setminus \{x_p\}. \\ \delta_{co}: \quad ((\Sigma \cup \{\epsilon\}) \times C) \times X_{co} \longrightarrow X_{co} \\ \qquad\qquad (x_{0_{co}}, (\sigma, \Gamma_i)) \longmapsto x_{x_{co}} \\ \qquad\qquad (x_{0_{co}}, (\epsilon, \Gamma_i)) \longmapsto x_i \\ \qquad\qquad (x_i, (\epsilon, \Gamma_i)) \longmapsto x_i \\ \qquad\qquad (x_i, (\epsilon, \Gamma_j)) \longmapsto x_p \\ \qquad\qquad (x_i, (\sigma, \Gamma_i)) \longmapsto x_{co} \\ \qquad\qquad (x_p, (\sigma, \Gamma_j)) \longmapsto x_p \end{array}$$

The last thing to do is to compute the product of $\mathcal{A}_d$ and $\mathcal{A}_i$. The result of this operation is the deterministic Rabin automaton $\mathcal{A}_{d_i}$.

It can be proved that the supervision of $\mathcal{A}_{d_i}$ (in the classic fashion [5]) is equivalent to the supervision of $\mathcal{A}_o$.

Together, the results of this section show that SCOP$^\omega$ can be reduced to the problem of controlling a deterministic, completely observed Rabin automaton to the satisfaction of its own acceptance condition. The latter control problem, which is in fact equivalent to other well-known problems in automata theory, is solved in [5]. This yields an effective procedure for deciding solvability of SCOP$^\omega$ and for computing a solution if one exists.

In the next section we show that a decentralized version of SCOP$^\omega$ admits no such effective solution.

5 Decentralized control synthesis

In this section we show, by the methods of [1, 2], that a decentralized version of the above problem is undecidable. For this we assume that C is closed under intersection and bring in the standard definition of the *conjunction* of two supervisors V_1 and V_2; namely,

$$\begin{array}{rl} V_1 \wedge V_2: & L(G) \mapsto C \\ & s \mapsto V_1(s) \cap V_2(s), \quad \forall s \in L(G) \end{array}$$

We can now define a decentralized supervision problem for ω-languages (DCP$^\omega$):

Problem 5.1 Given $A \subseteq E \subseteq S(G)$, observation alphabets $T_i, i = 1, 2$ and masks $M_i : L(G) \longrightarrow T_i$, find complete, deadlock-free supervisors $V_i : L(G) \longrightarrow C$ such that $\ker M_i \leq \ker V_i, i = 1, 2$, the conjunction $V_1 \wedge V_2$ is deadlock-free, and

$$A \subseteq S(V_1 \wedge V_2/G) \subseteq E$$

Theorem 5.2 The solvability of DCP$^\omega$ is undecidable.

Proof sketch: Following [1, 2], we show that, given a Turing machine $\mathcal{M}$, we can construct an instance of DCP$^\omega$ that has a solution if and only if $\mathcal{M}$ halts on the empty input. It follows that the existence of solutions to DCP$^\omega$ is undecidable.

Suppose that the event alphabet Σ contains two disjoint subalphabets, say $\mathcal{K}_1$ and $\mathcal{K}_2$, each sufficiently rich to encode sequences of configurations of $\mathcal{M}$. We shall formulate an instance of SCP$^\omega$ whose solution requires both supervisors to use the respective subalphabets to simulate the sequence of configurations through which $\mathcal{M}$ passes on reading the empty input. In order to do so we assume that we can extend the subalphabets $\mathcal{K}_i$ to

$$\Sigma_i := \mathcal{K}_i \dot{\cup} \{S_i, N_i\}, i = 1, 2$$

Intuitively, the symbol S_i will be used by the plant to "signal" supervisor V_i to move to the next symbol in its encoding of the sequence of configurations, while N_i will signal it to skip directly to the next configuration.

The generator is so constructed that symbols in $\{S_i, N_i : i = 1, 2\}$ alternate with symbols from $\mathcal{K}_1 \dot{\cup} \mathcal{K}_2$. Furthermore, any symbol in $S_i \cup N_i$ is immediately followed by a symbol in $\mathcal{K}_i, i = 1, 2$. By the definition of the observation and control mechanisms,

the latter symbol may be "chosen" from $\mathcal{K}_i$ by the supervisor V_i: for this, the supervisor chooses a control pattern containing only the appropriate symbol, together with S_i, N_i and all the symbols in $\Sigma_j, j \neq i$. The mask M_i will be the natural projection of Σ onto Σ_i. Indeed, neither supervisor observes or controls any symbols of the other supervisor's subalphabet: supervisor S_i observes only events in Σ_i, and all of its control patterns include Σ_j.

The generator will perform "random" checks on the two substrings over $\mathcal{K}_1$ and $\mathcal{K}_2$ in an effort to disqualify them as representing the trajectory that $\mathcal{M}$ follows on the empty input. It will elicit the next symbol or configuration in a sequence by means of the symbols S_i and N_i, and in so doing check whether the appropriate substrings do in fact represent configurations of $\mathcal{M}$, and whether the $(i+1)$th configuration is in fact the successor of the ith. Tests of the latter sort are carried out by checking that the structure of the state transition function is obeyed, that tape head movement is correct, and so forth. Because the generator can compare one sequence against the other, it requires only a finite number of states to perform these tests (despite the fact that the encoding of a configuration of $\mathcal{M}$ may require an arbitrarily long substring). If ever it detects an inconsistency in the sequences, it enters a "dead end" state, where no transitions are possible; if it detects an accepting configuration of $\mathcal{M}$, it enters a state where it loops, generating strings containing only some symbol $\alpha \in \Sigma \setminus (\Sigma_1 \cup \Sigma_2)$. This defines the finite behaviour $L(G)$ of the generator; its infinite behaviour is defined by $S(G) = \lim(L(G)) \cap \Sigma^* \alpha^\omega$. The minimal acceptable sublanguage $A \subseteq S(G)$ is empty; the maximal legal language is $E = S(G)$.

Because neither supervisor can observe (either directly or indirectly) the actions of the other, the only control strategy that is sure to avoid the generator's dead-end state is that whereby each supervisor V_i ensures that the $\mathcal{K}_i$ substring indeed represents the sequence of configurations through which $\mathcal{M}$ passes upon reading the empty input; moreover, this strategy yields deadlock-free supervision if and only if the empty input is accepted by $\mathcal{M}$.

It follows that solvability of DCP$^\omega$ is undecidable.

The ω-language framework is not essential to this result; absence of blocking can play the role of deadlock-freedom in the finite-string case. Note also that a standard control mechanism, based on a partition of the event alphabet into controllable and uncontrollable symbols, suffices for the construction used in the proof.

6 Conclusion

We have shown how to solve effectively a supervisory control synthesis problem with partial observations in the (ω-regular) infinite-string setting. A straightforward generalization of this problem to the case of decentralized control is undecidable. As the undecidability result applies also to finite-string problems, it may not be possible to relax the significant assumptions that underlie existing synthesis algorithms.

References

[1] Roni Rosner. *Modular Synthesis of Reactive Systems.* PhD thesis, The Weizmann Institute of Science, Rehovot, Israel, February 1991.

[2] Gary L. Peterson and John H. Reif. Multiple-person alternation. In *Proceedings of the 1979 Twentieth Annual Symposium on Foundations of Computer Science*, pages 348–363, 1979.

[3] Peter J. G. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Trans. Automatic Control*, 34(1):10–19, January 1989.

[4] Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B: Formal Models and Semantics*, pages 134–191. Elsevier, The MIT Press, 1990.

[5] J. G. Thistle and W. M. Wonham. Control of infinite behaviour of finite automata. *SIAM J. Control and Optimization*, 32(4):1075–1097, July 1994.