Optimal Control of Discrete Event Systems under Partial Observation*

Hervé Marchand†, Olivier Boivineau‡, Stéphane Lafortune**

† EP-ATR Project, Inria Rennes, Campus Univ. de Beaulieu, 35042 Rennes Cedex, France, E-mail: hmarchan@irisa.fr

‡ Electricité de France, Division Recherche et Développement 1, avenue du Général De Gaulle 92141 Clamart Cedex - France

** Dept. of Elec. Eng. & Computer Science, Univ. of Michigan, 1301 Beal avenue, Ann Arbor, Michigan, USA 48109-2122. E-mail: stephane@eecs.umich.edu

Report No CGR-00-10
College of Engineering • Control Group Reports • University of Michigan, Ann Arbor
September 2000

Abstract

We are interested in a new class of optimal control problems for Discrete Event Systems (DES). We adopt the formalism of supervisory control theory [12] and model the system as the marked language generated by a finite state machine (FSM). Our control problem follows the theory in [14] and is characterized by the presence of uncontrollable events, the notion of occurrence and control costs for events and a worst-case objective function. However, compared to the work in [14], we wish to take into account partial observability. Our solution approach consists of two steps. The first step is the derivation of an observer for the partially unobservable FSM, called a C-observer, which allows us to mask the underlying nondeterminism and to construct an approximation of the unobservable trajectory costs. We then define the performance measure on this observer rather than on the original FSM itself. In the second step, we use the algorithm presented in [14] to synthesize an optimal submachine of the C-observer. This submachine leads to the desired supervisor for the system.

Keywords
Discrete Event Systems, Optimal Control, Partial Observation.

Also appear as Research Report IRISA, No 1359. November 2000

*This research is supported in part by INRIA and by the Department of Defense Research & Engineering (DDR&E) Multidisciplinary University Research Initiative (MURI) on Low Energy Electronics Design for Mobile Platforms and managed by the Army Research Office (ARO) under grant DAAH04-96-1-0377.

1 Introduction and Motivation

In general, the purpose of optimal control is to study the behavioral properties of a system (also called plant), modeled as a Discrete Event Systems (DES), to take advantage of a particular structure, and to generate a supervisor which constrains the system to a desired behavior according to quantitative and qualitative requirements. In the field of control of DES, the supervisory control theory initiated by Ramadge and Wonham [12] provides an appropriate theoretical environment for starting to reason about optimal control. In this framework, the behaviors of systems are modeled by sequences of events over a finite alphabet. The plant is represented by some kind of automaton, a Finite State Machine (FSM), that generates sequences of events (or actions) through its execution. The control of the plant is then performed by inhibiting some events in Σ_c, the set of controllable events as opposed to the set Σ_{uc} (uc stands for uncontrollable) of events that cannot be directly prevented from occurring. However, in the basic setup of supervisory control theory, optimality is with respect to set inclusion and thus all legal behaviors are equally good (zero cost) and illegal behaviors are equally bad (infinite cost). The work in [14] enriches this setup by the addition of quantitative measures in the form of occurrence and control cost functions, to capture the fact that some legal behaviors are better than others.

In this paper, we are interested in a new class of optimal control problems for Discrete Event Systems (DES). We adopt the formalism of supervisory control theory [12] and model the system as a finite state machine (FSM). Our control problem follows the theory in [14] and is characterized by the presence of uncontrollable events, the notion of occurrence and control costs for events and a worst-case objective function. However, compared to the work in [14] and compared to the other works dealing with optimal control of DES [6, 9, 10], we wish to take into account partial observability; in contrast, the previous theories only deal with totally observable system representations. Several concepts and properties of the supervisory control problem under partial observation were studied in [2, 4, 7, 5] among other. However, they only propose a qualitative theory for the control of Discrete Event Systems.

Our solution approach consists of two steps. The starting point is a FSM which represents the global behavior of a given system, including its unobservable dynamics. From this FSM, our aim is to generate a supervisor that verifies any property that we would wish to associate to it, from the set of acceptable controllers according to some minimization criterion. This is what has been called the DP-Optimality property[1]. of a FSM [14]. In the framework of partial observation, the first step is the derivation of an observer for the partially unobservable FSM, called a C-observer. This step is necessary since unobservable events alone cannot trigger a specific behavior of a controller. We define the performance measure on the C-observer rather than on the original FSM itself. However, we will make the necessary efforts to keep track of the information that has disappeared with the initial structure. Especially, this observer allows us to remember an approximation of the costs of the unobservable costs between two observable events. This approximation basically correspond to the worst, i.e., the highest, cost of the

[1]DP-Optimality stands for Dynamic Programming Optimality.

different unobservable trajectories than can occur between two observable events. Indeed, since these trajectories are not observable it is not possible to recover their costs. We therefore need to assume that the worst case occurs. Moreover as we want to deal with optimal control we need to store these costs in the observer. In the second step, we use the theory in [14] to synthesize an optimal controller corresponding to the optimal restricted behavior, insofar as it is achievable by an admissible (i.e., physically constructible) supervisor. We use back-propagation from the (only) goal state of the observer to generate the supervisor, based on event cost functions. The supervisor is synthesized in a manner that gives them optimal sub-structure, consistent with the notion of DP-Optimality of [14]. The objective function has a worst-case form. The total worst-case computational complexity of this second step is cubic in the number of states in the observer.

This paper is organized as follows. In Section 2, the necessary notations are introduced. Section 3 is devoted to the presentation of a type of observer for a partially observed FSM, called a C-observer. The C-observer is itself an FSM whose states contain information that allows us to keep track (in a certain way) of the cost of the unobservable trajectories of the initial system G. Finally, in this section, we extend the notion of controllability. Section 4 is devoted to a new notion of optimal control. We introduce the notion of cost of a trace leading to the definition of an optimal (as well as DP-Optimal) submachine of the C-observer. Finally, we show how we can solve the Optimal Supervisory Control Problem within our new framework using the theory presented in [14].

2 Preliminaries

In this section, the main concepts and notations are defined. More definitions will be given when necessary in the following sections. The reader is referred to [3] for any undefined concepts.

The Model. The system to be controlled is a finite state machine (FSM) [3] defined by a 5-tuple $G = \langle \Sigma, Q, q_0, q_m, \delta \rangle$, where Σ is the finite set of events, Q is the finite set of states, q_0 is the initial state, q_m is the unique marked state (in general, an FSM may have several marked states), and δ is the partial transition function defined over $\Sigma^* \times Q \longrightarrow Q$. The notation $\delta(\sigma, q)!$ means that $\delta(\sigma, q)$ is defined, i.e., there is a transition labeled by event σ out of state q in machine G. Likewise, $\delta(s, q)$ denotes the state reached by taking the sequence of events defined by trace s from state q in machine G. We can think of G as an uncontrolled plant that starts at q_0 and executes/generates a sequence of events that are accepted by δ. It is assumed that two events can not be triggered simultaneously; moreover the time between two consecutive events executed by G is not determined but is finite.

With a slight but common abuse of notation, we denote by δ both the partial transition function and the active event set function. δ is defined over $\Sigma^* \times Q \longrightarrow Q$ when it represents the partial transition function, or over $Q \longrightarrow \Sigma$ when it represents the active event set function (i.e. when it returns the set of events that can be executed in a given state). We denote by δ^*, defined over $Q \longrightarrow \Sigma^*$, the active *trace* set function (as opposed to the active *event* set function).

The behavior of the system is described by a pair of languages $\mathcal{L}(G)$ and $\mathcal{L}_m(G)$. The prefix-closed language $\mathcal{L}(G)$ [12] is the language generated by G. It is a subset of Σ^*, where Σ^* denotes the Kleene closure of the set Σ [3]. Similarly, the language $\mathcal{L}_m(G)$ corresponds to the marked behavior of the FSM G, i.e., the set of trajectories of the system ending in q_m. State q_m can be thought of as the state reached when a given task is completed.

Blocking. An FSM is said to be *blocking* if $\mathcal{L}(G) \neq \overline{\mathcal{L}_m(G)}$ and non-blocking if $\mathcal{L}(G) = \overline{\mathcal{L}_m(G)}$, where $\overline{K}$ denotes the prefix closure of the language K. These concepts deserve further comments. If G is blocking, it can reach a state q, where $\delta(q) = \emptyset$ but $q \neq q_m$. This is called a *deadlock* because no event can be triggered. According to our interpretation of marking, we state that the plant blocks because it enters a deadlock state without having terminated the task, i.e., reached q_m. Another issue is when there is a set of unmarked states in G that forms a strongly connected component, but with *no transition going out of the set*. If the plant enters this set of states, then we get what is called a *livelock*.

By definition of $\mathcal{L}(G)$ and $\mathcal{L}_m(G)$, we assume that G is *trim* with respect to q_0 and q_m. This assumption means that all the states of G are accessible from the initial state q_0 and coaccessible to the marked state q_m[2]. In other words, if the system has executed some $s \in \mathcal{L}(G) - \overline{\mathcal{L}_m(G)}$, then in finite time, it will execute t such that $st \in \mathcal{L}_m(G)$ since all behaviors have the opportunity to terminate properly. Plants having behaviors which terminate at unmarked states are illegal and excluded from our solution space.

Submachines. The set of behaviors of the controlled system will be a subset of $\mathcal{L}(G)$. With this in mind, we state the following definition of a submachine of an FSM (that can possibly be partially observable) [3].

Definition 1 *An FSM* $A = \langle \Sigma, Q_A, q_{0A}, \delta_A \rangle$ *is a submachine of* G*, denoted by* $A \subseteq G$*, if* $\Sigma_A \subseteq \Sigma$*,* $Q_A \subseteq Q$*, and* $\forall \sigma \in \Sigma_A, q \in Q_A \quad \delta_A(\sigma, q)! \Rightarrow (\delta_A(\sigma, q) = \delta(\sigma, q))$. •

It is clear that "$\subseteq$" is a partial order on the set of FSMs. We also say that A is a submachine of G at q whenever $q_{0A} = q \in Q$ and $A \subseteq G$. Moreover, we will use

$$\mathcal{M}(G, q) \;=\; \{A \subseteq G : A \;\text{ is trim with respect to } q_m \text{ and } q_{0_A} = q\} \tag{1}$$

to represent the set of trim submachines of G at q with respect to q_m.

For technical reasons, we need to define an algebraic operation called *merge* that combines FSMs.

Definition 2 *Let* A *and* B *be FSMs and let* $q_{0_C} \in Q_A \cap Q_B$*. Then the merge* $A \oplus B$ *of* A *and*

[2]For explicit mathematical definitions, the reader may refer to [11].

B is built as follows:

$$
\begin{array}{rrcl}
A \oplus B = & C & = & \langle \Sigma_C, Q_C, q_{0_C}, Q_{mC}, \delta_C \rangle, \quad \textit{where} \\
& \Sigma_C & = & \Sigma_A \cup \Sigma_B \\
& Q_C & = & Q_A \cup Q_B \\
& Q_{mC} & = & Q_{mA} \cup Q_{mB} \\
& \delta_C(\sigma, q) & = & \begin{cases} \delta_A(\sigma, q) \textit{ if it exists} \\ \delta_B(\sigma, q) \textit{ if it exists} \\ \textit{undefined otherwise.} \end{cases}
\end{array} \tag{2}
$$

•

The merge of A and B produces a machine C whose transitions are the union of the transitions of A and B. The merge of two trim FSMs is not necessarily a trim FSM. Moreover, the transition function is not necessarily well defined. However, we can state the following lemma. Its proof is straightforward [14].

Lemma 1 *Let $A, B \subseteq G$. Then, the transition function for $C = A \oplus B$ in Definition 2 is well defined. Moreover, if $q_{0A} \in Q_B$ and A and B are trim, then $C = \langle \Sigma_C, Q_C, q_{0B}, Q_{mC}, \delta_C \rangle$ is a trim submachine of G.* ◇

The requirements that A and B be submachines of G ensures that if $\delta_A(\sigma, q)$ and $\delta_B(\sigma, q)$ both exist, then $\delta_A(\sigma, q) = \delta_B(\sigma, q)$. Now, for all A, $B \in \mathcal{M}(G, q)$ this lemma implies that $A \oplus B \in \mathcal{M}(G, q)$, which implies that $\mathcal{M}(G, q)$ has a maximal element (in the sense that all others are submachines of it). It is denoted by $M(G, q)$.

Event status. Our aim in the following sections is to synthesize a supervisor for G w.r.t. a specific control objective. It is then important to take into account the possibility that certain events cannot be disabled by the supervisor or that certain events may be not observed by the supervisor. Therefore, some of the events in Σ are said to be uncontrollable, i.e., their occurrence cannot be prevented by a controller, while the others are controllable. An uncontrollable event can for example model a change of sensor readings, the tick of a clock, etc (see [3] for more details). Likewise, control will be applied on a plant that is partially observable, i.e. the supervisor will observe only a subset of the events generated by plant G. Hence some of the events in Σ are unobservable whereas the others will be unobservable. An unobservable event could model a failure event, an internal event, etc. In this regard, Σ can be partitioned as

$$\Sigma = \Sigma_c \cup \Sigma_{uc} \text{ with } \Sigma_c \cap \Sigma_{uc} = \emptyset \text{ and as } \Sigma = \Sigma_o \cup \Sigma_{uo} \text{ with } \Sigma_o \cap \Sigma_{uo} = \emptyset, \tag{3}$$

where Σ_c, Σ_{uc}, Σ_o and Σ_{uo} represent the set of controllable, uncontrollable, observable and unobservable events, respectively. Moreover, unobservable events are assumed to be uncontrollable, i.e., $\Sigma_{uo} \subseteq \Sigma_{uc}$ (Dually, this implies $\Sigma_c \subseteq \Sigma_o$).

Unobservable cycles. In order to consider the control problem under partial observation, we need to make sure that it can have a solution. If the initial FSM G has an unobservable cycle, even if it may be possible to determine its existence, it would be impossible to alleviate the fact that it could make the system run indefinitely in that cycle, without the supervisor noticing. Hence Assumption 1, which entails Property 1.

Assumption 1 *The initial FSM G has no unobservable cycles.* •

Property 1 *Under Assumption 1,* $\forall q \in Q,\ \{s \in \Sigma_o\Sigma_{uo}^* \ / \ \delta(s,q)!\}$ *has finite cardinality.* ◇

Control Cost functions. We now include the last ingredient to be able to discuss optimality, namely a cost (or objective) function. As stated in [14], in order to take into account the numerical aspect of the optimal control problem, two cost values are associated to each event of Σ. To this effect, we introduce

- an occurrence cost function $c_e : \Sigma \to \mathbb{R}^+ \cup \{0\}$. Occurrence cost functions are used to model the cost incurred in executing an event (energy, time, etc.). This function can be easily extented to a trace $s = \sigma_1 \dots \sigma_n$ as follows : $c_e(s) = \sum_{i=1}^{n} c_e(\sigma_i)$.
- a control cost function $c_c : \Sigma \to \mathbb{R}^+ \cup \{0, \infty\}$. Control costs are used to represent the fact that disabling a transition possibly incurs a cost. The control cost function is infinity for events in Σ_{uc} $(\forall\sigma((\sigma \in \Sigma_{uc}) \Leftrightarrow (c_c(\sigma) = \infty)))$.

These cost functions are used to introduce a cost on the trajectories of a submachine A of G as we shall see in Section 4.

3 The C-Observer with respect to Σ_{uo}

The framework in which we develop our control theory is that of partially observable FSMs. The supervisor that will be generated should be able to take decisions (enabling or disabling events) based on the states and/or events that it observes. Consequently, we base our model upon a partially observed system, seen through an observer. However, in order to take into account unobservable events in the optimality under which we apply our control, we must keep track of their costs. Indeed by abstracting away the unobservable events of the plant G, we lose the cost of the unobservable trajectories between two observable events. The idea is to collect an approximation of these costs in the states of the observer we want to build. For example, consider two states p and q of G, connected by (at least) a trace of the form $\sigma s \in \Sigma_o\Sigma_{uo}^*$. As we only observe the first event, it is not be possible to know which trajectory has been taken between these two states. Hence, from an optimal control point of view, we have to consider that the plant evolves from p to q through the trajectory with the highest cost (as this trajectory is unobservable and therefore uncontrollable, whenever the system chooses to trigger σ in p, there is no way to control the system in such a way that this trajectory is not taken).

Example 1 *To illustrate this point, let us consider the following example. Let* $\Sigma_o = \{\sigma\}$ *and* $\Sigma_{uo} = \{\sigma_{uo_1}, \sigma_{uo_2}, \sigma_{uo_3}\}$.

When G is in state p, it can trigger event σ. *Further, it can choose to evolve into q either by triggering the sequence* $\sigma_{uo_1}\sigma_{uo_3}$ *or the sequence* $\sigma_{uo_2}\sigma_{uo_1}$ *and there is no way to determine which trajectory the system evolves trough. Therefore, from an optimal point of view, we have to assume that the trajectory with the highest cost occurred (i.e.,* $max\{c_e(\sigma_{uo_1}) + c_e(\sigma_{uo_3}), c_e(\sigma_{uo_2}) + c_e(\sigma_{uo_2})\}$. *This is this approximation, that will be stored in the observer.*

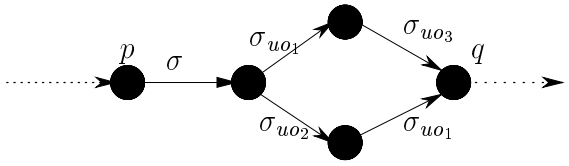


Figure 1: Part of G

In order to collect these costs, we build a deterministic observer, named C-observer (Observer with Costs), and define the notion of a macro-state, allowing both to mask the underlying nondeterminism by abstracting away from the nondeterministic submachine and to keep track of the unobservable event costs of trajectories between two states. The way these costs are computed and attached to the macro-states is explained in Section 3.1. The C-observer constitutes the basic model on which the optimal control will be applied.

Before giving formally the definition of the C-observer, denoted by G_c, we need to check the original FSM G in order to account for unobservable events that may lead to q_m in G. Indeed, if an unobservable event leads to q_m in G, it may be impossible to determine whether or not the system has actually reached q_m. We therefore update G by adding a self-loop at q_m, labeled φ. Formally, $\Sigma \leftarrow \Sigma \cup \varphi$ and $\delta(\varphi, q_m) = q_m$. The φ event is just an (observable) indicator event. The intuition behind the φ event is that we want to be able to know when a given task has been achieved. Without loss of generality, we can assume it is controllable and has zero occurrence and control costs. The event φ can be thought of as a sensor that signals that q_m has been reached.

3.1 The C-Observer definition

The new structure that we define is called a C-observer. It is denoted by $G_c = \langle \Sigma_o, X, x_0, x_m, f \rangle$, where Σ_o is the set of observable events, X is the set of macro-states, x_0 is the initial macro-state, x_m is the marked macro-state, and f is the partial transition function defined over $\Sigma_o^* \times X \longrightarrow X$.

Starting from the initial plant $G = \langle \Sigma, Q, q_0, q_m, \delta \rangle$, the set X of macro-states of G_c will be constituted of pairs in $Q \times \mathbb{R}^+$. More specifically, the admissible states that are considered are states that can be reached by a trace of events constituted by an observable first event followed by a sequence of unobservable events. In language formalism, the latter trace should be in $\Sigma_o \Sigma_{uo}^*$. One way of looking at this choice of projection is that an observable action can lead to a sequence of unobservable events. If one initial observable event is taken, it is possible that several other events take place as a direct consequence.

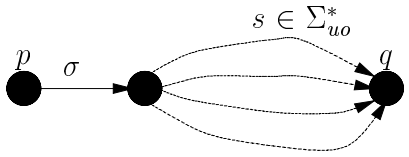
Remark 1 *Imagine a car moving at a given speed (state =* moving*). The driver brakes (event =* brake*). The car stops (state =* stopped*). The* brake *event might have been followed by the* ABS-active *event, but it was not observable by the driver.*

We present more formally the way the states of the system G_c are built. First, we introduce the set of triples $\mathcal{D}$ defined by :

$$\mathcal{D} = \{(p, q, \sigma) \in Q \times Q \times \Sigma_o \ / \ \exists s \in \Sigma_{uo}^*, \ \delta(\sigma s, p) = q\}. \tag{4}$$

A triple (p, q, σ) belongs to set $\mathcal{D}$ if there is a trace between p and q whose first event is σ and whose following events are all unobservable. Note that more than one trace s could verify this condition. In order to generalize this definition and take into account this possibility, we now define the set of traces that verify the above conditions, for a given triple (p, q, σ) :

$$\forall (p, q, \sigma) \in \mathcal{D}, \quad \mathcal{S}(p, q, \sigma) = \{s \in \Sigma_{uo}^* \ / \ \delta(\sigma s, p) = q\}. \tag{5}$$



Using (5), we can easily deduce the following property:

Property 2 *$\forall (p, q, \sigma) \in \mathcal{D}$, the cardinality of $\mathcal{S}(p, q, \sigma)$ is finite.* $\diamond$

Finally, we do not want to lose the cost of the unobservable events that have been projected. To this effect we introduce the notion of *locally computed cost* associated with a triple (p, q, σ) of $\mathcal{D}$. Formally, it is given by a function, denoted by c_o, over $\mathcal{D} \longrightarrow \mathbb{R}^+$, and defined in Equation (6).

$$\forall (p, q, \sigma) \in \mathcal{D}, \quad c_o(p, q, \sigma) = \max_{s \in \mathcal{S}(p,q,\sigma)} c_e(s) \tag{6}$$

The intuition behind the computation of this cost is that we project unobservable events, but do not wish to consider their cost as null. We then keep track of the worst (i.e., inducing the greatest cost) unobservable trace that could lead from p to q.

Using the previous notations, G_c is a FSM, defined as follows.

Definition 3 *Given an FSM G, the associated* C-observer *G_c is given by a tuple $\langle \Sigma_o, X, x_0, x_m, f \rangle$. It is an FSM whose elements are defined as follows:*

1. *X is the set of macro-states. $x \in X$ is defined by a set of pairs $(q, c) \in Q \times \mathbb{R}^+$, called micro-states of x; Formally $X \subseteq 2^{Q \times \mathbb{R}^+}$.*

2. *The final macro-state is defined by $x_m = (q_m, 0)$ and the initial macro-state x_0 as :*

$$x_0 = \{(q, c_q), \ \exists s \in \Sigma_{uo}^*, \delta(s, q_0) = q \text{ and } c_q = \max_{\substack{t \in \Sigma_{uo}^*, \\ \delta(t, q_0) = q}} c_e(t)\}$$

3. *$\forall x \in X$ and $\forall \sigma \in \Sigma_o$, define*

$$\forall (p, c_p) \in x, \ \text{s.t. } \delta(\sigma, p)!, \ A_\sigma^x(p) = \{(q, c_o(p, q, \sigma)) \ / \ (p, q, \sigma) \in \mathcal{D}\}.$$

$A_\sigma^x(p)$ basically constitutes the set of states of G that can be reached via a trace $\sigma\Sigma_{uo}^*$ (from a micro-state of x), together with the associated approximation of the unobservable trace cost.

4. *The transition function f is recursively defined by:*

$$\forall x \in X \ \forall \sigma \in \Sigma_o, \ f(\sigma, x) = \{(q, c_q) \in \bigcup_{(p,c_p)\in x} A^x_\sigma(p), \ c_q = \max_{s\in(p,q,\sigma)\in\mathcal{D}} c_e(s)\}$$

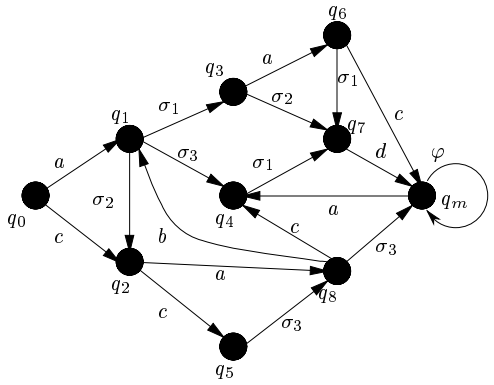
Hence, if there exists different micro-states of the form $(q, .)$ in $f(\sigma, x)$, then we only consider the pair with the maximal cost.

5. *We only build the accessible part of the system (i.e. the states $x \in X$ that are reachable from x_0 by f).* •

The way G_c is built masks the nondeterministic nature of the projected FSM. x_0 is computed from the unobservable reach of $(q_0, 0)$. x_m is a single marked state, namely, $\{(q_m, 0)\}$. Finally, f can be constructed recursively from the initial state. Indeed, we can construct the set of states of G_c using point *(2)* and then point *(3)* and *(4)* of Definition 3 recursively. Note that due to Property 2, the recursion terminates (this property entails that the new set of state X is finite as well as the number of transitions). The structure that we obtain is another deterministic FSM, whose events are taken in Σ_o. States of G_c are macro-states with respect to G. However, we have done more than simply abstract away from the initial FSM by projecting partially unobservable traces. We have computed and kept a local cost to avoid losing track of the costs of the unobservable events that have disappeared from the structure.

Remark 2 *In the above construction procedure, we assumed knowledge of the initial state of the system, since the C-observer is assumed to run in parallel with the system from the start of the supervisory control operation.*

Example 2 *Figure (2) gives an example of a cyclic FSM containing unobservable events along with the characteristics of all its events (we recall that if the control cost of an event is infinite, it means that this event is uncontrollable). Also, the initial state of G is chosen to be q_0, whereas the unique final state of G is q_m. Note the self-loop φ onto q_m that allows to know when the system actually reaches the marked state q_m.*



event	c_e	c_c	Status
a	c_a	C_a	Obs.
b	c_b	C_b	Obs.
c	c_c	∞	Obs.
d	c_d	C_d	Obs.
φ	0	0	Obs.
σ_1	c_1	∞	Unobs.
$\sigma 2$	c_2	∞	Unobs.
σ_3	c_3	∞	Unobs.

Figure 2: The initial system G and the cost functions

Following Definition 3, we derive from G the C-observer G_c with respect to Σ_{uo}. Figure (3) shows the resulting C-Observer. Remark that in the macro-state x_1, the cost associated with q_7 will be the maximum between $c_1 + c_2$ and $c_3 + c_1$. When the system is in state q_7, it is not possible for the observer to determine whether G reaches q_7 through q_3 or through q_4.

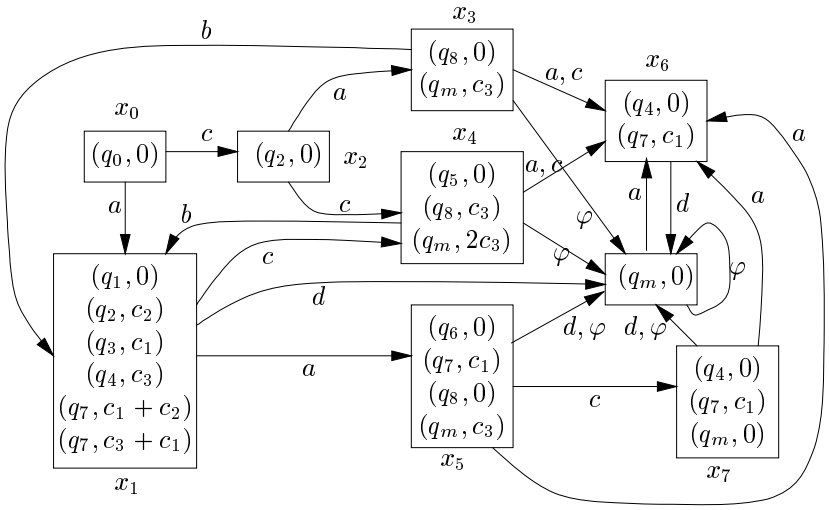


Figure 3: The resulting C-observer G_c

3.2 Properties of the C-observer

We now give a few lemmas and properties that hold for G_c, in order to use them in subsequent results.

Lemma 2 *Let $x \in X - \{x_m\}$ be a state of G_c, and let $(q, c_q) \in x$ be a micro-state of x. We can state that*

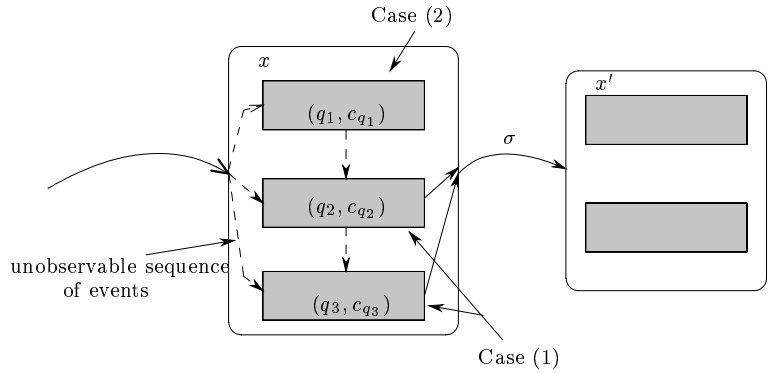
(1) *either $\exists \sigma \in \Sigma_o$, $\exists q' \in Q$, $\delta(\sigma, q) = q'$ and, in this case, $\exists x' \in X$, s.t. $f(\sigma, x) = x'$, and $(q', .) \in x'$*

(2) *or $\exists \sigma \in \Sigma_{uo}$ and $\exists q' \in Q$ s.t. $\delta(\sigma, q) = q'$ and, in this case, $\exists (q', .) \in x$.*

Moreover, there exists at least one micro-state $(q, c_q) \in X$ s.t. $\exists \sigma \in \Sigma_o$, $\exists q' \in Q$, $\exists x' \in X$, s.t. $\delta(\sigma, q) = q'$ and $f(\sigma, x) = x'$

Proof: We first prove (1). Assume $\exists \sigma \in \Sigma_o$, $\delta(\sigma, q) = q'$. Then $(q', q, \sigma) \in \mathcal{D}$. Therefore, as $\sigma \in \Sigma_o$, according to point (3) of Definition 3, $(q', .) \in A^x_\sigma(q)$. Hence, according to point (4) of Definition 3, there exists $x' \in X$, such that $f(\sigma, x) = x'$, for which $(q', .) \in x'$.

We now prove (2). Assume that $\exists \sigma \in \Sigma_{uo}$, s.t. $\delta(\sigma, q) = q'$. Consider now a state x'' for which there exists an observable transition σ_1, such that $f(x'', \sigma_1) = x$. Consider now the trajectory t in G s.t. $q'' \xrightarrow{t} q$ (it is of the form $\sigma_1 t'$ where $t' \in \Sigma^*_{uo}$). As $\sigma \in \Sigma_{uo}$ and $\delta(\sigma, q) = q'$, it means that q' is in the unobservable reach of q'' and finally according to (3) and (4) of Definition 3, there exists a micro-state $(q'', c_{q''})$ that belongs to x.



Using Items (1) and (2), we now prove the last assertion of this lemma. Assume the system is in x. Then, either the system is in a micro-state such that there exists an observable outgoing transition (in G). In this case, the system G_c can possibly leave the state x (Case (1)). Or the system is in a micro-state such that there exists an admissible unobservable transition (in G). In this case the system G_c will evolve into another micro-state of x (Case (2)). As there are no unobservable cycles in G, there is no way of cycling indefinitely in a set of micro-states of a macro-state. So, the system will actually reach a micro-state in x with an admissible observable transition, which brings us to the previous case. ◇

What the above lemma states is that whatever the state x that can be reached during the execution of the plant, there eventually exists a way out of this state (either directly via an observable event or by an unobservable trajectory which reaches a micro-state of x having the previous property.

Next, we state that the C-observer realized from G inherits properties of G. Indeed, according to the definition of *deadlock* and *livelock* we presented in Section 2, we can prove that:

Proposition 1 *G_c is non-blocking.*

Proof: By construction of G_c, all the states are reachable (in particular x_m is reachable). Now, according to the definition of non-blocking, we only have to prove that G_c is both deadlock free and livelock free. Lemma 2 ensures that G is deadlock free. Finally, since G is trim, it contains no livelocks. Since all traces of G project onto traces of G_c, cycles of G project onto cycles of G_c. Since G has no livelock (i.e., cycle with no outgoing events), G_c has no such cycle either (even internally, i.e., in G). Therefore, G_c is livelock free and thereby non-blocking. ◇

3.3 Extented notion of Controllability

In this section, we formalize the method used (by a supervisor) to generate a submachine from a C-observer. We define the notion of a controllable submachine as well as the control policies that are acceptable once a given state has been reached.

3.3.1 Submachines of a C-observer

The machine G_c that we obtain is an FSM that simply reflects what an observer sees in system G. We wish to apply some control to the original system in order to verify a certain performance

criterion. In other words, we wish to reduce the system G_c, and therefore G, to a particular behavior. This leads us to define the notion of a submachine of G_c. In fact, even if the worlds in which they are defined (for G and G_c) are different, the notion of submachine is the same as the one given in Definition 1 (i.e. a submachine of G_c is any structure that has its states in those of G_c, the same initial state and final state and its events and transitions in those of G_c).

Moreover, as explained in Section 2, we are only interested in complete behavior, in the sense that we wish to obtain a controlled system that achieves a task (i.e reaches the state x_m and therefore the state q_m in the world of G). Hence, we wish to consider in our framework the submachine of G_c that have this property. We then introduce the notion of *G-live* submachines that correspond to an adapted definition of Lemma 2.

Definition 4 *Let* $G_c = \langle \Sigma_o, X, x_0, x_m, f \rangle$ *be the C-observer associated with* $G = \langle \Sigma, Q, q_0, q_m, \delta \rangle$. *A submachine* $H = \langle \Sigma_o, X_H, x_{0,H}, x_m, f_H \rangle$ *of* G_c *is said to be* G-live *if the following condition holds:*

$$\forall x_H \in X_H \setminus \{x_m\},\ \forall (q, c_q) \in x_H,\ \exists (q', c_{q'}) \in x_H \ s.t.$$
$$\{[\exists s \in \Sigma_{uo}^*, \delta(s, q) = q'] \wedge [\exists \sigma \in f_H(x_H),\ \delta(\sigma, q')!]\}.$$

Note that (q, c_q) *could be equal to* (q', c'_q). •

A submachine H of G_c is G-live whenever any micro-state of x_H has a transition that is either an observable transition for the initial FSM G, or an unobservable transition that leads to another micro-state of x_H from which there is a possibility of exiting the macro-state (except for the marked state). Quite naturally, using Lemma 2, we can state that :

Proposition 2 *If* G_c *is the C-observer associated with* G, G_c *is G-live.* ◇

3.3.2 Controllability in this framework

The structure on which control will be applied is FSM G_c (and not directly the initial machine G). We first have to adapt the classical definition of controllability introduced by Ramadge & Wonham [12]. Indeed, even if the control policy remains the same (we do not want to disable uncontrollable events), we have to take care of the fact that, by removing controllable transitions, the obtained submachine of G inherits some properties of the initial FSM $= G_c$. In particular, we have to avoid the blocking of G without observing it. Hence the new definition of controllability:

Definition 5 *Let* $G_c = \langle \Sigma_o, X, x_0, x_m, f \rangle$ *be the C-observer associated with* $G = \langle \Sigma, Q, q_0, q_m, \delta \rangle$. $H = \langle \Sigma_o, X_H, x_{0,H}, x_m, f_H \rangle$ *is said to be a* controllable submachine of G_c *if the following conditions hold:*

1. $H \subseteq G_c$,
2. $\forall x_H \in X_H$ *that can be reached via a trace of* $\mathcal{L}(H), \forall \sigma \in \Sigma_{uc} \cap \Sigma_o,\ f(\sigma, x_H)! \Rightarrow f_H(\sigma, x_H)!$,
3. H *is G-live.* •

Condition *(2)* imposes that any transition that needs to be disabled in G_c to generate H needs to be controllable. Condition *(3)* imposes that no submachine of a C-observer presents any deadlocks or livelocks. This condition imposes that any micro-state of a state x_H must have an active outgoing trace (in the original FSM from which G_c was derived) that is either unobservable (thereby leading to another micro-state of x_H and eventually leading to a state from which there is an observable outgoing event) or observable (thereby leading to another macro-state of H).

3.3.3 The supervisor

Now that we have the definition of a controllable submachine of a C-observer, it is interesting to determine how such a submachine can be obtained via a supervisor acting upon the system now represented by G_c. However, control restriction cannot be blindly performed. Disabling an event that was admissible in a state x of G_c can induce a deadlock in the initial FSM G. Hence, before defining formally the notion of a supervisor, we introduce the notion of *Admissible Control Actions (ACA)*.

Definition 6 *Let* $G_c = \langle \Sigma_o, X, x_0, x_m, f \rangle$ *be the C-observer associated with* $G = \langle \Sigma, Q, q_0, q_m, \delta \rangle$. *We define the set of* Admissible Control Actions (ACA) *at state* $x \in X$ *as a function :*

$$\Gamma_x = \{\gamma \subseteq \Sigma_c,\ \forall (q, c_q) \in x,\ \exists (q', c_{q'}) \in x \ s.t. \\ \{[\exists s \in \Sigma_{uo}^*,\ \delta(s, q) = q'] \wedge [\exists \sigma \in f(x) \setminus \gamma,\ \delta(\sigma, q')!]\}$$

Note that micro-state $(q', c_{q'})$ *could be equal to* (q, c_q). •

More precisely, Γ_x gives, for a state x of G_c all the possible sets of controllable events that can be disabled without risk of deadlock. In other words, given a state x of G_c and a given γ in Γ_x, if σ belongs to γ, it means that σ can be disabled because there actually exists at least one trajectory $s \in \Sigma_{uo}^*$ that leads the system in another micro-state of x' for which there exists an observable event σ' that makes the system leave the macro-state x and eventually reach a state $x' = f(x, \sigma')$ of G_c.

Remark 3 *According to this definition, we can remark that a controllable event can, for certain states, become uncontrollable because, should it be disabled, it could induce blocking of the plant* G. *Therefore, in the following, an event* $\sigma \in \Sigma$ *is said to be* conditionally controllable *with respect to a state* x *whenever* $\sigma \in \Sigma_c$ *and there exists* $\gamma \in \Gamma_x$ *such that* $\sigma \in \gamma$.

Using Definition 6, a *supervisor* of G_c can be defined as follows:

Definition 7 *Let* G_c *be the C-observer associated with* G *and* $\Pi = (\Gamma_x)_{x \in X}$ *be the set of admissible control actions, then a supervisor* S *is a function given by :*

$$\begin{aligned} S : X &\rightarrow 2^{\Sigma_c} \\ x &\mapsto \gamma \in \Gamma_x \end{aligned} \tag{7}$$

•

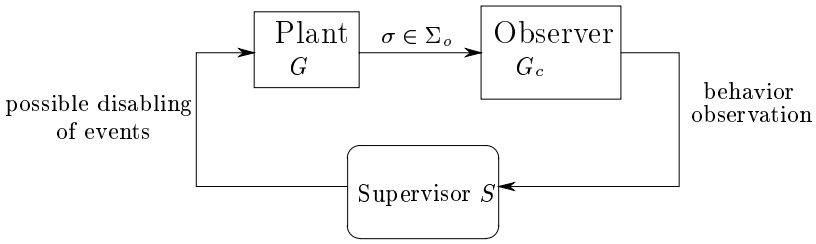


Figure 4: Supervisory S in feedback with G and G_c

In other words, a supervisor of G_c (consequently of G) is obtained by choosing a particular γ in a state x. By definition, the control action will always belongs to Σ_c, which ensures that S never disables an uncontrollable event.

Conceptually, the supervisor controlling the plant G is placed in feedback with G and G_c according to Figure 3. Only the observable events can be seen by S. Therefore G_c plays the role of an observer that will somehow rebuild a part of the state in which the system has evolved. According to this information, the supervisor determines whether the observation corresponds to a (conditionally) controllable event and if it has to enable/disable this event in order to keep the closed loop system behaving "desirably". We write S/G for the closed loop system, consisting of the initial plant G and the C-observer controlled by the supervisor S.

To conclude this section, let us remark that Definition 7 is consistent with the definition of a controllable submachine of the C-observer G_c. This is summarized by the following proposition:

Proposition 3 *$H = \langle \Sigma_o, X_H, x_{0,H}, x_{m,H}, f_H \rangle \subseteq G_c$ is a controllable submachine of G_c if and only if there exists a supervisor S, such that*

$$\forall x_H \in X_H,\ f_H(x_H) = f(x_H) \setminus S(x_H). \tag{8}$$

$\diamond$

4 Optimal Supervisory Control Problem

In general, the aim of optimal control is to study the behavioral properties of a system, to take advantage of a particular structure, and to generate a controller which constrains the system to a desired behavior according to quantitative and qualitative aspects [6, 9, 10, 14]. This is performed by the addition of quantitative measures in the form of occurrence and control cost functions, to capture the fact that some legal behaviors are better than others. In this paper, we enrich this setup by the addition of unobservable events in the plant that has to be controlled. We have generated a new type of FSM, namely, a C-observer derived from the original plant G. It is a type of observer, but it holds information from the original submachine that would not be available to a direct observer. This structure cannot be used as such, but leads to the construction of an optimal supervisor, which is defined in this section.

4.1 Transformation of G_c

We first need to transform the C-observer, in order to exactly fit within the framework developed by [14]. Indeed, unlike in the case of total observability where costs are defined in events only, we have incorporated cost information in the macro-states of the G_c. These costs were attached to the states in order to keep track of the unobservable cost of the trajectory between two macro-states (see Section 3.1). Basically, the transformation we will perform on G_c, consists in "shifting" the cost of the macro-state to the events that can be executed in this macro-state. However, we do not blindly take the worst cost of the micro-states contained in the macro-state. For a given macro-state x, and a given event σ admissible in x, we only consider the worst cost of the pairs $(q, c_q) \in x$ such that σ belongs to the active event set of q in G; hence reducing the possible worst cost.

The transformation is performed as follows: let $x \in X$ and let $f(x)$ be the set of events that G_c can execute in x. For each $\sigma \in f(x)$, we rename σ as σ_x and we attach to this new event the cost $c_e(\sigma_x)$ defined by :

$$c_e(\sigma_x) = \max_{\substack{(q, c_q) \in x \\ \delta(\sigma, q)!}} \{c_q\} + c_e(\sigma) \tag{9}$$

The controllability status of the event as well as the control cost of the events do not change (namely, we have $c_c(\sigma_x) = c_c(\sigma)$). Call Σ'_o the new set of event. The transition function f remains the same (i.e. $f(x, \sigma_x)$ is defined and equal to x' whenever $x' = f(x, \sigma)$).

The new C-observer G'_c we obtain is still a FSM. It is defined by $\langle \Sigma', X, x_0, x_m, f \rangle$. Compared to G_c, the global structure of G'_c does not change. The only difference is that we change the original alphabet of G_c in such a way that costs are now defined on events only, as carried out in [14]. From now on, G'_c is a deterministic and trim FSM. All the events of G'_c are observable. Some are controllable, some are uncontrollable. To each event is attached two values, which respectively correspond to its event and control costs. The only difference with [14] lies in the notion of controllability that, in our framework, takes into account the notion of liveness of the underlying system G. However, this does not affect the use of the theory of [14] to compute the optimal supervisor of G_c, and therefore the optimal supervisor of G. Indeed, as in our case, the theory is based on the notion of acceptable control actions that have to be computed at first. In [14], a control action in a state x is admissible whenever it does not disable uncontrollable events and it does not produce local deadlock (i.e. no output event.)

Remark 4 *Note that given G_c, the way we are shifting the costs of the unobservable trajectories in order to obtain G'_c constitutes the best approximation that we can do without memory. A better approximation could be obtained by unfolding the C-observer G_c in order to take into account the history of the plant (e.g. the last* n *observable events that occurred in the system). However, even if the approximation would be better (at each step, the number of admissible pair (q, c_q) in a macro-state would have been lower leading to a possibly lower unobservable cost), the counterpart would be the complexity of the transformed C-observer G'_c.*

4.2 Trajectory costs of a submachine of G'_c

In order to be able to discuss optimality, we now explain how to compute the cost of a trajectory of G'_c.

Control cost function over the state of the system. In order to model this particular aspect, let us define the control cost of an event according to a state. We first introduce

$$\Sigma_d(x, H) = f_{G'_c}(x) \setminus f_H(x) \tag{10}$$

as the set of disabled events at state x for the system to remain in submachine H of G'_c.

Whereas in [14] the control cost function was defined on an event, in the case of partial observation, it is defined on a state as follows: considering a submachine H of G'_c, we have

$$C_c(x, H) = \begin{cases} \infty \text{ if } \Sigma_d(H, x) \notin \Gamma_x \\ \\ \displaystyle\sum_{\sigma' \in \Sigma_d(H,x)} c_c(\sigma') \text{ otherwise} \end{cases} \tag{11}$$

In other words, the cost of a state x is equal to ∞ whenever there does not exist a particular control policy $\gamma \in \Gamma_x$ that restricts the behavior of G'_c to H (i.e. when an uncontrollable event has been removed or when a controllable event has been removed, then inducing a deadlock).

The global cost of a trajectory and of a submachine of G'_c. We are now ready to define the cost of a trajectory s of a submachine H as well as the objective cost function of a submachine H of G'_c.

Definition 8 *Let $H = \langle \Sigma'_o, X_H, x_{0,H}, x_{m,H}, f_H \rangle$ be a submachine of G'_c derived from G and $\mathcal{L}_m(H)$ be the marked language generated by H, then*

1. *for all y in H and trajectory $s = \sigma'_1 ... \sigma'_n$, $\forall i, 1 \leq i \leq n, \sigma'_i \in \Sigma'_o$ such that $f_H(y, s)$ exists, the cost of s is given by :*

$$C_O(y, H, s) = \sum_{i=1}^{n} c_e(\sigma_i) + \sum_{i=0}^{n} C_c(f_H(y, \|s\|_i), H), \tag{12}$$

   where $\|s\|_i$ denotes the prefix of s of length i,

2. *the objective cost function denoted by $C_{Sup}(H)$ is given by:*

$$C_{Sup}(H) = \sup_{s \in \mathcal{L}_m(H)} (C_O(x_0, H, s)) \tag{13}$$

•

Basically, the cost of a trajectory is the sum of the occurrence costs of the events composing it, to which is added the cost of controlling events on the way to remain in machine H. If an uncontrollable event is disabled, the cost of a trajectory becomes infinite because of the second term of (12). Finally, $C_{Sup}(H)$ represents the worst case behavior that is possible in submachine H.

Remark 5 *Due to the transformation of G_c, the event cost of the trajectory s starting in x and ending at a state corresponds to the sum of the costs of the individual observable events of the trajectory s to which is added the cost of the unobservable trajectories traversed by s (i.e. the costs accumulated in the micro-states of the macro-states visited by s).*

In fact, without the transformation, and with the notations of Definition 8, the cost of a trajectory $s = \sigma_1 \dots \sigma_n \in \Sigma_o^*$ in a submachine H of G_c, such that $y = x_0 \xrightarrow{\sigma_1} x_2 \xrightarrow{\sigma_2} \dots \xrightarrow{\sigma_n} x_n$, would have been given by:

$$C'_O(y, H, s) = \sum_{i=1}^{n} c_e(\sigma_i) + \sum_{i=0}^{n-1} \max_{\substack{(q, c_q) \in x_i \\ \delta(\sigma_{i+1}, q)!}} (c_q) + \sum_{i=0}^{n} C_c(f_H(y, \|s\|_i), H), \tag{14}$$

It is easy to verify that $C'_O(y, H, s) = C_O(y, H, s)$.

The next lemma characterizes the interaction of event and control costs:

Lemma 3 *Let $H_1 \subseteq H_2 \in \mathcal{M}(G'_c, x)$ and $s \in \mathcal{L}_m(H_1)$, then*

$$C_O(x, H_1, s) \geq C_O(x, H_2, s).$$

Proof: Let $x_1, \dots, x_n$ be the states visited by s on the way. Now, the fact that $H_1 \subseteq H_2$ easily entails that $\forall x_i,\ \Sigma_d(H_2, x_i) \subseteq \Sigma_d(H_1, x_i)$. From Relation 11, we can deduce that for all the states x_i, $C_c(H_1, x_i) \geq C_c(H_2, x_i)$. Remark now that the cost (without control) of a trajectory is independent of the machine. Therefore $C_O(x, H_1, s) \geq C_O(x, H_2, s)$. ◇

This lemma states that the cost associated with a trajectory admissible in a machine is lower than the cost of the same trajectory generated by one of its submachines. The purpose of "contracting a submachine" is to remove trajectories with high event costs. However this process is accompanied by rising control costs, hence the optimization problem we now define.

4.3 The optimization problem

In this section, we are only interested in machines that achieve a task (we only consider plants having a behavior which terminates at a marked state). Among all the trim and controllable submachines of G'_c, since we want to deal with optimal solutions, we want to extract the submachines that have a minimal objective cost function.

4.3.1 The optimal submachines of G'_c

Considering the trim hypothesis, we denote $\mathcal{M}(G'_c, x)$ as the set of trim submachines of G'_c starting at state x with respect to the unique final state x_m and denote by $M(G'_c, x)$ its maximal element (see Section 2). We now define the optimization problem.

Definition 9 *$\forall x \in X, H_o \in \mathcal{M}(G'_c, x)$ is an optimal submachine of the FSM G'_c if*

$$C_{Sup}(H_o) = \min_{H \in \mathcal{M}(G'_c, x)} C_{Sup}(H) < \infty.$$

•

The cost $C_{Sup}(H_o)$ of H_o represents the minimum worst case cost incurred to reach x_m from x_0 when the behavior of G'_c is restricted to a submachine of it. As some events in some states are not controllable (which induces an infinite cost), optimality is met when there is no other control policy with lower worst-case cost that allows to reach the marked state x_m certainly. At a lower level (in the world of G), the control policy induced by submachine H_o corresponds to the one with lower worst-case cost, knowing that G could evolve through unobservable trajectories with the worst possible cost. However the way we compute the cost of trajectories reduces the uncertainty. In general, there will exists several optimal submachines for an FSM.

As in the case of total observation [14], the following lemma is stated to note that optimal solutions lie within the class of controllable submachines.

Lemma 4 *Let $H \in \mathcal{M}(G, x)$. If $C_{Sup}(H) < \infty$ then H is controllable.*

Proof: $C_{Sup}(H) < \infty \Leftrightarrow \sup_{s \in \mathcal{L}_m(H)}(C_O(x, H, s)) < \infty$. Therefore, for all possible trajectories of H, starting in x, the second term of $C_O(x_i, H, s)$ (Eq. 12) is finite (where x_i denotes the state visited on the way). It entails that no uncontrollable event or conditionally controllable event has been disabled. Hence, H is controllable. $\diamond$

From Lemma 4, uncontrollable submachines are not candidates for optimality since the cost for restricting the system to those submachines is infinite (it is impossible to restrict the system to such a behavior). The following theorem gives necessary and sufficient conditions for the existence of optimal submachines:

Theorem 1 *An optimal submachine of G'_c exists if and only if there exists a submachine H of G'_c such that H is trim, controllable, with no cycles.*

Proof: If there exists a submachine H of G'_c such that H is trim, controllable, and if there does not exist any cycle in H, then it is obvious to see that $C_{Sup}(H) < \infty$. Therefore H is a good candidate for the optimal submachine. On the other hand, let H_o be an optimal submachine of G'_c. Then H_o is trim by hypothesis (Definition 9). Now, according to Definition 9, $C_{Sup}(H_o) < \infty$, which implies that H_o is controllable (Lemma 4). Moreover, there does not exist any cycle, otherwise there would exist trajectories in $\mathcal{L}_m(H_o)$ of the form ts^*t' with s such that there exists $x \in X$ with $f_H(s, x) = x$ and $C(x, H_o, f_H(s, y)) > 0$. Therefore, the cost of these trajectories would be equal to ∞ (the event costs are strictly positive). $\diamond$

Intuitively, this theorem states that an optimal solution exists when there are controllable submachines of G'_c in which there does not exist cycles. The controllability assumption ensures that the cycles can be broken using controllable events alone.

Theorem 2 [14] *Consider the set of optimal submachines starting at $x \in X$. Then, this set is closed under the "merge operation" (see Definition 2). In other words, the merge of two optimal submachines of $\mathcal{M}(G'_c, x)$ is still an optimal submachine of $\mathcal{M}(G'_c, x)$.* $\diamond$

The submachine that includes all the other optimal submachines will be called the maximal optimal submachine and will be denoted by $H^{\uparrow}{}_o$. Formally, we have :

$$H^{\uparrow}{}_o = \bigcup_{\substack{H_o \in \mathcal{M}(G_o, x) \\ H_o \text{ optimal}}} H_o \tag{15}$$

4.3.2 The DP-optimal submachines of G'_c

In general, the solution to the Optimal Supervisory Control Problem is not unique. Moreover, all the optimal solutions do not structurally have optimal sub-solutions, which means that they do not satisfy the principle of Dynamic Programming (see e.g. [1]). In fact, in the previous section, optimality is obtained only regarding the paths between the initial and final state and never the postfix paths between any state of the corresponding FSM and the final state. In this section, we will show that whenever an optimal solution exists, a solution having optimal sub-structure also exists. We call this latter type a DP-optimal solution (DP stands for Dynamical Programming) and define it as follows :

Definition 10 *A submachine H_{DO} of G'_c is DP-Optimal if it is optimal and $\forall x' \in X_{H_{DO}}$, $M(H_{DO}, x')$ is an optimal submachine in $\mathcal{M}(G'_c, x')$.* •

We have already seen that optimality actually exists when the worst-case cost from the initial state x_0 is to x_m is finite once minimized. DP-Optimality is obtained when any terminal path from any state of a submachine to the goal state x_m is optimal in the previous sense.

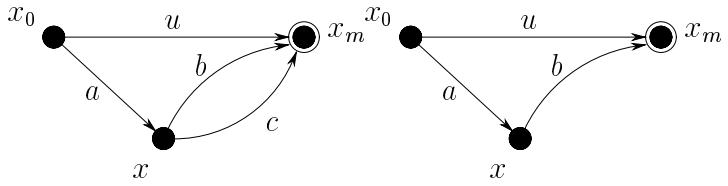
If a particular DP-Optimal FSM includes all other DP-Optimal FSMs as submachines of itself, then we call it the *maximal DP-Optimal submachine.* The maximal DP-Optimal submachine of a machine G'_c at q w.r.t. x_m will be denoted by $M^o_D(G'_c, x)$. Note that all DP-Optimal submachines are acyclic. The existence of a DP-Optimal submachine of G'_c is given by the following theorem (the proof can be found in [14]).

Theorem 3 *If an optimal submachine of G'_c exists, then the unique maximal DP-Optimal submachine $M^o_D(G'_c, x_0)$ of G w.r.t. x_m also exists.* ◇

We now give a simple example (Figure 5) that allows to grasp the notion of DP-Optimality versus that of optimality.

Example 3 *For simplicity, we have assumed that the states of G'_c were reduced to singleton. In this way, we exactly retrieve the definition of controllability as defined in [12] and therefore we are exactly in the framework of [14]. Moreover, we have assumed that the control cost function was reduced to 0 for controllable events and to ∞ for uncontrollable events.*

We can observe that, in both submachines, the cost for going from x_0 to x_m is optimal since there exist no other controllable path with a lower cost. However, in the first submachine, the worst-case cost to go from state x to state x_m is 3 (through event c); whereas the worst-case cost in the second submachine is optimized to be reduced to 1. Consequently, we can say that the second submachine is DP-Optimal (thereby optimal), but the first submachine is not DP-Optimal, but is optimal (since the optimality is obtained only regarding the paths between the initial and final states and never the postfix paths between any state of the corresponding FSM and the final state).



Event	c_e	Remarks
a,b	1	Controllable
c	3	Controllable
u	5	Uncontrollable

Figure 5: A simple example giving the difference between Optimality and DP-Optimality

4.3.3 The cyclic DP-Optimal algorithm

Consider a FSM $G = \langle \Sigma, Q, q_0, q_m, \delta \rangle$ with a unique initial state q_0, and a unique marked state q_m and its corresponding transformed C-observer $G'_c = \langle \Sigma'_o, X, x_0, x_m, f \rangle$.

Then there exists an algorithm [14], named **DP-Opt**, with a worst-case complexity $\mathcal{O}(|X|^2|\Sigma_o|log(|\Sigma_o|) + |X|^3|\Sigma_o|)$ (Theorem 6.10 of [14])[3], that constructs the desired maximal DP-Optimal submachine $M^o_D(G'_c, x_0)$ of the FSM G'_c w.r.t. x_0 and x_m. The algorithm also returns the worst inevitable cost $c^g_{sup}(M^o_D(G'_c, x_0))$. A simplified version of this algorithm can be found in [8] (when the control cost function is reduced to the null function for controllable events). We refer the reader to [14] for a complete description of **DP-Opt**. Instead, we present an overview of this algorithm inspired by the discussion in [13].

The solution of the algorithm can be viewed as a generalization of Dijkstra's algorithm. The algorithm is based on backward recursive dynamic programming algorithms from the final state to the initial state of the C-observer, with "local" optimization problems for each new state that can be reached after a new backward recursion step. It is obvious that all cycles have to be removed and since we are only interested in behaviors that terminate in marked states, all the transitions going out of x_m must be disabled. The state x_m is put in a closed list, say $\mathcal{C}$. This closed list is expanded by using dynamic programming on all "parent" states (by one backward step computation) of $\mathcal{C}$, that we denote by $P_f(\mathcal{C})$. Now, for any $x \in P_f(\mathcal{C})$, it is only necessary to decide which of the immediate children of x lies in the maximal DP-optimal solution starting at x. Since the "child" states are all in C, the maximal DP-Optimal solutions at these states have already been computed and by the principle of Dynamical Programming, they constitute the rest of the maximal DP-Optimal solution starting at x. Hence, it is somehow only necessary to compute the maximal DP-Optimal "one-step subgraph" of G'_c starting at x. This is computed as follows.

For all $y \in P_f(\mathcal{C})$ remove all transitions other than those from y into $\mathcal{C}$. This is a one-step subgraph. Order the transitions of this one-step subgraph by the worst case cost incurred if the plant executes the transition. Let the set be ordered by increasing cost. This defines a sequence of one-step subgraphs each of which consists of the first i elements of the set where i varies from one to the cardinality of the set. Find the lowest cost one-set subgraph of this sequence and denote this subgraph for $y \in P_f(\mathcal{C})$ by A_y.

Find x such that A_x has the minimum cost out of the collection of all A_y. The one step

[3]Note that we have $|\Sigma_o|$ and not $|\Sigma'_o|$ in this expression, it is because we only have to take into account the number of admissible events in a state x and this number is bounded by $|\Sigma_o|$.

graph A_x represents the maximal one-step DP-Optimal subgraph starting at x, which can be added to the closed list $\mathcal{C}$. We repeat the process until x_0 is added to $\mathcal{C}$.

4.4 Formalization and computation of the supervisor

The supervisor computation consists of different steps. Once the C-observer G_c derived from the initial FSM G is computed, we first have to transform it into G'_c by attaching the cost induced by the unobservable trajectories to the events in order to fit within the framework of [14] (see Section 4.1). From this machine, using the algorithm of [14], we compute (if it exists) the DP-Optimal solution $M_D^o(G'_c, x_0)$ of G'_c. At this point, we disable in G_c the corresponding sets of events in Σ'_o and for all $x \in X$, we retrieve $\Sigma_d(G_c, x)$, the set of disabled event at state x for the system to remain in submachine $M_D^o(G_c, x_0)$ of G_c. Call f_c the new transition function. It is formally given by :

$$\begin{array}{rcl} f_c : X \times \Sigma_o & \rightarrow & X \\ (x, \sigma) & \mapsto & \left\{ \begin{array}{l} f(x,\sigma) \text{ if it is defined and if } \sigma \notin \Sigma_d(G_c, x) \\ \text{undefined otherwise} \end{array} \right. \end{array}$$

Now, a supervisor S of G_c can be classically derived from $M_D^o(G_c, x_0)$ by attaching to this FSM an output function O from the set of states X to the set of observable events Σ_o :

$$\begin{array}{rcl} O : X & \rightarrow & \Sigma_o \\ x & \mapsto & \Sigma_d(G_c, x) \end{array}$$

The supervisor $S = \langle \Sigma_o, X, x_o, x_m, f_c, O \rangle$ will in fact be used for two purposes. It first plays the role of an observer that is able to rebuild part of the state in which the system has evolved. Based on this information, S sends back to the system the set of events that have to be disabled in order to force the closed loop system to eventually reach the marked state q_m by minimizing the global cost of the trajectory.

Remark 6 *Note that the DP-optimal solution does not, in general, lie in the set of submachines of G. Indeed, if one wants to use $M_D^o(G_c, x_0)$ in order to directly disable events in the initial system G, an intuition would be to disable in the initial system G the events that have been disabled in $M_D^o(G_c, x_0)$. Indeed for all macro-states $x \in X$, and for all micro-states (q, c_q), it is easy to compute the new active event set $\delta_c(q)$ of q by removing from its initial one the event of $\Sigma_d(G_c, x)$. However, doing that could possibly induce a deadlock in G. To illustrate this point, assume the state q has a and b as active event set. Let us suppose now that q belongs to the macro-state x_1 which, once controlled, disables event a and that q belongs also to the macro-state x_2 which, once controlled, disables event b. In this case, if the controlled system reaches the state q, it will be in deadlock!*

4.5 Example

To illustrate this part, consider the example described in Section 2. In the following, we will assume that the costs attached to each event (either observable, or unobservable) is the following

:

event	c_e	c_c
a, b, d	$7, 1, 3$	$1, 2, 1$
c	2	∞
$\sigma_1,\ \sigma_2,\ \sigma_3$	$1,\ 2,\ 3$	∞

Figure 6 illustrates the C-observer G_c with these new costs.

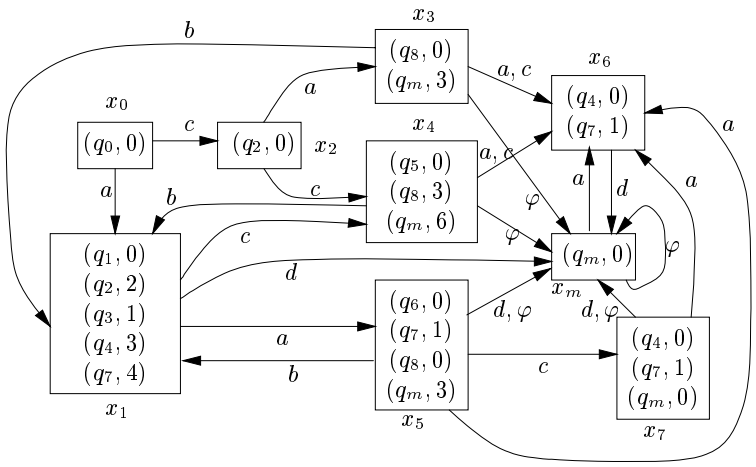


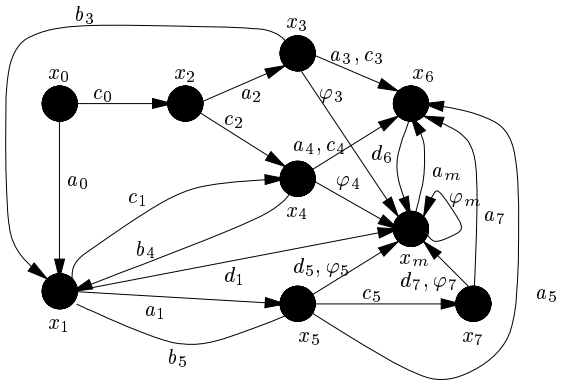
Figure 6: The resulting abstract FSM G_c

Following the transformation described in Section 4.1, we obtain the following C-observer G'_c with its new alphabet Σ'_o (see Figure 7). Note that, according to the new definition of controllability, some events that were assumed to be controllable become uncontrollable. Hence $d_1,\ d_5,\ d_6,\ d_7$ could not be disabled, otherwise the real system G could be in deadlock (in q_7). The DP-Optimal submachine of G'_c obtained by FSM the DP-Opt algorithm is shown in Figure 8. To simplify the notations in the figure, we note $C_{sup}(x_i)$ the worst case behavior that is possible in the DP-Optimal submachine submachine $M^o_D(G'_c, x_i)$ of G'_c (instead of $C_{Sup}(M^o_D(G'_c, x_i))$).

According to Section 4.4, the supervisor can be easily derived from this FSM. First, we only have to consider the accessible part of this FSM. Then, we associate to each state of this FSM, the set of events that have been disabled in the C-observer G_c in order to remain in $M^o_D(G'_c, x_o)$. It is presented in Figure 9 (note that we also rename the events).

5 Conclusion

In this paper, we have introduced a new type of optimal control for Discrete Event Systems. Previous works in optimal control deal with numerical performances in Supervisory Control theory when the system to be controlled is under full observation. In contrast, our aim, in this paper, was to account for partial observability, while controlling the system. The system to be controlled is represented by an FSM G with a unique marked state representing the state of interest (the achievement of a task for example) and some unobservable events. The first step is the derivation of an observer for the partially unobservable FSM, called a C-observer



event	c_e	c_c
$a_0,\ c_0$	7, 2	1, ∞
$a_1,\ c_1,\ d_1$	9, 4, 7	1, ∞, ∞
$a_2,\ c_2$	7, 2	1, ∞
$a_3,\ b_3,\ c_3,\ \varphi_3$	10, 1, 2, 3	1, 2, ∞, 0
$a_4,\ c_4,\ \varphi_4$	13, 5, 6	1, ∞, 0
$a_5,\ c_5,\ d_5,\ \varphi_5$	10, 2, 4, 3	1, ∞, ∞, 0
d_6	4	∞
$a_7,\ d_7,\ \varphi_7$	7, 4, 0	1, ∞, 0
$a_m,\ \varphi_m$	7, 0	1, 0

Figure 7: The resulting abstract FSM G'_c and the costs associated to Σ'_o

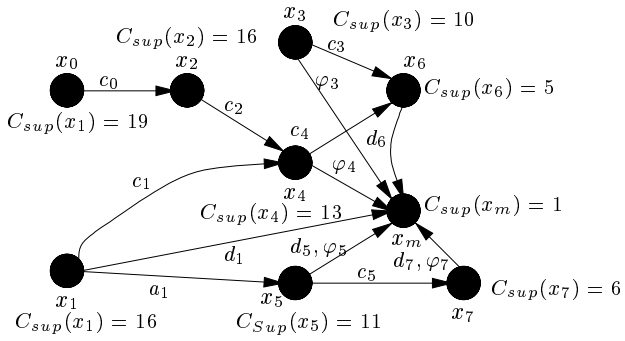


Figure 8: The resulting DP-Optimal Submachine of G'_c

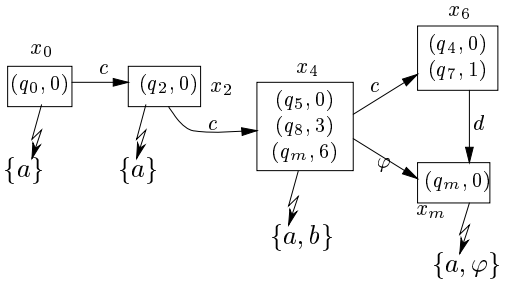


Figure 9: The corresponding Supervisor

G_c, which allows us to remember an approximation of the unobservable trajectory costs. We then presented a new definition of controllability derived from the classical one introduced by Ramadge & Wonham [12], that allows us to avoid the blocking of G without observing it. We then define the performance measure on this observer rather than on the FSM itself. In the second step, we first transform G_c into G'_c by shifting the cost of the macro-state to the events that can be executed in this macro-state. We then use the algorithm presented in [14] to synthesize an optimal submachine of the C-observer, which leads to the desired supervisor for the system. The behavior of the obtained controlled system is optimal with respect to Σ_o, in the sense that G_c carries on the best approximation of the unobservable trajectories between to observable events (without observing the whole system, it is not possible to have more informations dealing with these trajectories). Moreover it is optimal for G'_c and therefore for G_c (as explained in Remark 4). This optimality status is due to [14].

References

[1] R. Bellman. Dynamic programming. Princeton University, Press, NJ, 1957.

[2] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham. Formulas for calculating supremal and normal sublanguages. *Systems and Control Letters*, 15(8):111–117, 1990.

[3] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[4] H. Cho and S. J. Marcus. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Mathematics of Control, Signals and Systems*, 2(2):47–69, 1989.

[5] K. Inan. Nondeterministic supervision under partial observations. In G. Cohen and J. P. Quadrat, editors, *11th Int. Conf. on Analysis and Optimization of Systems, Discrete Event Systems*, volume 199 of *LNCS*, pages 39–48, Berlin, Germany, June 1994. Springer-Verlag.

[6] R. Kumar and V. K. Garg. Optimal control of discrete event dynamical systems using network flow techniques. In *Proc. of 29th Allerton Conf. on Communication, Control and Computing*, Champaign, IL, USA, October 1991.

[7] F. Lin and W. M. Wonham. On observability of discrete–event systems. *Information Sciences*, 44(3):173–198, 1988.

[8] H Marchand, O. Boivineau, and Lafortune S. On the synthesis of optimal schedulers in discrete event control problems with multiple goals. Technical Report CGR-98-10, Control Group, College of Engineering, Univeristy of Michigan, USA, July 1998.

[9] H. Marchand and M. Le Borgne. On the optimal control of polynomial dynamical systems over z/pz. In *4th International Workshop on Discrete Event Systems*, pages 385–390, Cagliari, Italy, August 1998.

[10] K. M. Passino and P. J. Antsaklis. On the optimal control of discrete event systems. In *Proc. of 28th Conf. Decision and Control*, pages 2713–2718, Tampa, Floride, December 1989.

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

[12] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems*, 77(1):81–98, 1989.

[13] R. Sengupta and S. Lafortune. Extensions to the theory of optimal control of discrete event systems. In S. Balemi, P. Kozák, and R. Smedinga, editors, *Discrete Event Systems: Modeling and Control*, volume 13 of *Progress in Systems and Control Theory*, pages 153–160. Birkhäuser Verlag , Basel, Switzerland, 1993. (Proc. of WODES'92, Aout 1992, Prague, Czechoslovakia).

[14] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM Journal on Control and Optimization*, 36(2), March 1998.

Contents