Supervisory Control for Modal Specifications of Services

Philippe Darondeau, Jérémy Dubreil, Hervé Marchand

Preprints of WODES 2010
August 30th – September 1st, 2010
Berlin, Germany

Supervisory Control for Modal Specifications of Services

Philippe Darondeau * **Jérémy Dubreil** ** **Hervé Marchand** ***

* *INRIA-Rennes Bretagne Atlantique, Campus universitaire de Beaulieu, 35042 Rennes, France (email: philippe.darondeau@inria.fr)*
** *Laboratoire d'Informatique de l'Ecole Polytechnique (LIX), Route de Saclay, 91128 Palaiseau, France (email: jeremy.dubreil@inria.fr)*
*** *INRIA-Rennes Bretagne Atlantique, Campus universitaire de Beaulieu, 35042 Rennes, France (email: herve.marchand@inria.fr)*

Abstract: In the service oriented architecture framework, a modal specification, as defined by Larsen in [5], formalises how a service should interact with its environment. More precisely, a modal specification determines the events that the server may or must allow at each stage in an interactive session. In this paper, we investigate the adaptation of the supervisory control theory of Ramadge and Wonham to enforce a modal specification (with final states marking the ends of the sessions) on a system modelled by a finite LTS. We prove that there exists at most one most permissive solution to this control problem. We also prove that this solution is regular and we present an algorithm for the effective computation of the corresponding controller.

Keywords: discrete event systens, supervisory control, modal specifications, services, partial observation.

1. INTRODUCTION

In [11, 12, 13], Ramadge and Wonham laid the foundations of the theory of supervisory control for discrete event systems and proposed solutions to the Basic Supervisory Control Problem, or BSCP, that may be stated as follows. Let G be a finite automaton, with a subset of final (or marked) states, over a set of events Σ with two independent partitions $\Sigma = \Sigma_c \cup \Sigma_{uc}$ and $\Sigma = \Sigma_o \cup \Sigma_{uo}$. The automaton G represents a plant. The plant's events in Σ_o and Σ_c may be observed and controlled, respectively, from the environment of the plant. Let L_{max} be a regular language over Σ. BSCP is the problem whether there exists some proper controller C such that the language $\mathcal{L}(C/G)$ of the controlled system satisfies the relation $\mathcal{L}(C/G) \subseteq L_{max}$. The plant and the controller produce joint runs, in which the controller acts by disabling at each step a subset of events of the plant. The set of disabled events must be a function of the subsequence of events currently observed. A proper controller should be admissible (it never disables uncontrollable events) and non-blocking (it always leaves a possibility to reach some final state of the plant). Ramadge and Wonham's theory characterizes the existence of proper controllers and proposes algorithms for computing the maximally permissive controller C under the assumption $\Sigma_c \subseteq \Sigma_o$.

Ramadge and Wonham's theory aims chiefly at enforcing *safety* properties on autonomous or semi-autonomous systems, e.g. automated manufacturing systems, while taking special care of the property of nonblocking. Our goal in this paper is to adapt the theory and algorithms of supervisory control for enforcing on a plant G a different type of property, namely the *conformance to the modal specifications of a service*. We assume that the plant is described by a finite automaton G over Σ, with a set of final states Q_F, and the service expected from the plant is specified by a modal automaton $\mathcal{S}$ over Σ_o, with a set of final states S_F. We assume moreover that $\Sigma_c \subseteq \Sigma_o$. Such assumptions are quite natural in service oriented architectures where only inputs from or outputs to servers can be observed, and only the outputs can be controlled, even though nonblocking depends also upon internal actions in $\Sigma \backslash \Sigma_o$ that cannot be observed by the controller. We want to compute a proper and maximally permissive controller C under which the controlled system C/G conforms to $\mathcal{S}$, i.e., it provides the specified service. This means in particular that under the considered control C, both sets of final states Q_F and S_F stay forever *jointly* reachable so that any interactive session can terminate.

Modal specifications, also called modal transition systems, were introduced in [5] in the form of transition systems $(S, \Sigma, \rightarrow_\Box, \rightarrow_\Diamond, s_0)$, with two modal transition relations: the *must* transitions denoted $\rightarrow_\Box$ and the *may* transitions denoted $\rightarrow_\Diamond$, such that $\rightarrow_\Box \subseteq \rightarrow_\Diamond$. Every modal transition system MTS determines a corresponding family $\mathcal{M}$ of labelled transition system (LTS) which are called the models of MTS. Intuitively, an LTS is a model of MTS if there exists a relation $\models$ between their respective sets of states Q and S such that $\models$ holds between the initial states and whenever $q \models s$, all *must* transitions from s are simulated by transitions from q, all transitions from q are simulated by *may* transitions from s and $\models$ is preserved under simulation of transitions in both directions. We let $G \models MTS$ mean that G is an LTS model of MTS.

Example 1. The modal specification depicted in Figure 1, where the relations $\rightarrow_\Box$ and $\rightarrow_\Diamond$ are represented with plain arrows and dashed arrows respectively, expresses the fact

that the presence of the first transition a is mandatory while the second one is not, and that after any a the system must be able to trigger a b (the execution of this b transition is not mandatory, since the system may alternatively trigger a second a). Finally, the presence of a transition b returning to the initial state is optional.

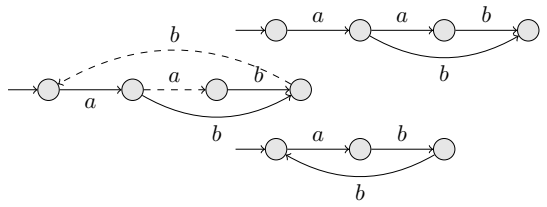

Figure 1. A modal specification MTS and some associated models

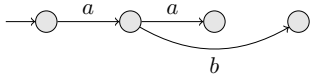

Figure 2. A labelled transition system that is not a model of MTS

The two LTSs on the right hand side of Figure 1 are models of the specification, whereas the one depicted in Figure 2 is not. Indeed, after the sequence aa, the specification requires a transition labelled by b which in not present in this LTS. $\diamond$

It is worthwhile noticing that a language generated by an LTS G which is a model of a specification MTS belongs to a language interval $[L_{min}, L_{max}]$ whose endpoints L_{min} and L_{max} are the languages of the labelled transition systems $(S, \Sigma, \rightarrow_{\Box}, s_0)$ and $(S, \Sigma, \rightarrow_{\diamond}, s_0)$, respectively. However, not all the languages in the interval $[L_{min}, L_{max}]$ can be generated by models of MTS. Consider for instance the modal transition system MTS depicted in Fig. 1. then $L_{min} = \{\epsilon, a, ab\}$ and L_{max} is the prefix-closure of $(a(ab+b)b)^*$. However, for $L = \{\epsilon, a, ab, aa\}$, there exists no LTS G generating L such that $G \models MTS$. This example shows that modal specification are strictly more expressive than tolerances considered in Ramadge and Wonham's theory.

The expressive power of modal specifications added over language specifications comes precisely from the possibility they offer to formulate service requirements in a conditional way, e.g. *"if the system serves request a, then the next b request will be served."* This feature makes modal specifications very convenient for describing the interface between a partially observed plant (service provider) and its environment (service requester). For instance, after a coin has been inserted into a coffee machine, the client should always get a cup of coffee or the coin back but there is no guarantee that coins can always be entered, which depends on unobservable phenomena inside the machine.

In [5], determinism was required neither from the labelled transition relations $\rightarrow_{\Box}$ and $\rightarrow_{\diamond}$ of modal transition systems, nor from the labelled transition relations $\rightarrow$ of their transition system models. In this paper, for simplicity, we shall limit ourselves to deterministic transition systems and to deterministic modal transition systems. However, we shall extend modal transition systems in another respect since we will introduce a satisfaction relation $\models$ between LTS's over Σ (modelling the plant) and MTS's over $\Sigma_o \subseteq \Sigma$ (specifying the expected service as observed from the environment of the plant). The proposed extension is based on the assumption that the plant interacts fairly with its environment, i.e. it does not refuse possible interactions forever. Consequently, infinite unobservable behaviours are considered impossible.

We shall also extend modal transition systems in a second direction by providing them with marked or final states, thus obtaining modal automata. In modal specifications of services, final states serve to represent the potential points of termination of a session from the service requester's perspective. In a similar way, we replace labelled transition systems with automata for modelling plants, where final states represent the potential points of termination of a session from the service provider's perspective. We end up thus with final states in G (the plant) and S (the specification), whose joint reachability is to be taken care of in the search for non-blocking controllers C such that $C/G \models MTS$. There, G is an automaton over Σ, S is a modal specification over $\Sigma_o \subseteq \Sigma$, and C is a labelled transition system over Σ_o that includes Σ_c (the set of controllable events).

An earlier adaptation of the theory of supervisory control to modal specifications was proposed in [3]. In that work, *total observation* was assumed and *nonblocking* was not considered. The contribution of the present paper is to lift these two limitations. The working assumptions of [3] are retrieved in the particular case where $\Sigma = \Sigma_o$ and all states of the plant automaton and the modal specification are final states. Partial observation is the prevailing situation in service oriented architectures, where systems interact with their environment by input and output actions. Hence, extending [3] in this respect is crucial for achieving the objectives of this paper.

The remaining sections of the paper are organized as follows. We recall in Section 2 some basic concepts and notations concerning automata, supervisory control and modal specifications of services. In Section 3, we investigate the supervisory control problem for modal specifications. We show that when there exists a solution to this problem, there exists a unique supremal controller enforcing the given modal specification and we show in a non-constructive way that this optimal supervisor is finite state. We present in Section 4.5 an algorithm to compute this supervisor. When there is no solution to the control problem, the algorithm produces an empty supervisor.

2. BACKGROUND

We recall in this section basic definitions and results about transition systems and supervisory control.

2.1 Transition Systems and Automata

A deterministic *labelled transition system* (or LTS) over Σ is a 4-tuple $LTS = (Q, \Sigma, \delta, q_0)$ where Q is a finite set of states, $q_0 \in Q$ is an *initial state*, and δ is a partial map from $Q \times \Sigma$ to Q, called the *labelled transition map*. This map is extended inductively to $\delta : Q \times \Sigma^* \rightarrow Q$ by setting $\delta(q, \varepsilon) =$

q (where ε is the empty word), and $\delta(q, wa) = \delta(\delta(q, w), a)$ for all $q \in Q$, $w \in \Sigma^*$ and $a \in \Sigma$. A state $q \in Q$ is *reachable* (from q_0) if $\delta(q_0, w) = q$ for some word $w \in \Sigma^*$. An LTS is *finite* if Q and Σ are finite; it is *reduced* if all states in Q are reachable and every event $a \in \Sigma$ is enabled at some state q, i.e. $\delta(q, a)$ is defined for the considered state q. In the sequel, we always consider reduced transition systems unless explicitly stated otherwise. The *language* of LTS is the set of words $\mathcal{L}(LTS) = \{w \in \Sigma^* \mid \delta(q_0, w) \text{ defined }\}$.

Given labelled transition systems $LTS = (Q, \Sigma, \delta, q_0)$ and $LTS' = (Q', \Sigma', \delta', q'_0)$, their *product* is the (reachable restriction of the) labelled transition system $LTS \times LTS' = (Q \times Q', \Sigma \cup \Sigma', \delta \times \delta', (q_0, q'_0))$ where $(\delta \times \delta')((q, q'), a)$ is defined as $(\delta(q, a), q')$ for $a \in \Sigma \setminus \Sigma'$, $(q, \delta'(q', a))$ for $a \in \Sigma' \setminus \Sigma$, and $(\delta(q, a), \delta'(q', a))$ for $a \in \Sigma \cap \Sigma'$.

A deterministic *automaton* over Σ is a labelled transition system with final states, i.e. $A = (Q, \Sigma, \delta, q_0, Q_F)$ with $Q_F \subseteq Q$. The *labelled transition system underlying* A is $\mathcal{U}(A) = (Q, \Sigma, \delta, q_0)$. The *language* of the automaton A is $\mathcal{L}(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in Q_F\}$. Thus, $\mathcal{L}(A) \subseteq \mathcal{L}(\mathcal{U}(A)) \subseteq \Sigma^*$. In the sequel, we sometimes consider also (for technical convenience) automata $A = (Q, \Sigma, \delta, Q_0, Q_F)$ with a set Q_0 of initial states.

Given a language $L \subseteq \Sigma^*$, its *prefix-closure* is $\overline{L} = \{u \in \Sigma^* \mid \exists v \in \Sigma^* : uv \in L\}$. A language L is prefix-closed if $L = \overline{L}$. A prefix-closed language L induces a (possibly infinite) transition system $\mathcal{LTS}(L) = (L, \Sigma, \delta, \varepsilon)$ where $\delta(u, a) = ua$ for every word $ua \in L$ with $a \in \Sigma$.

2.2 Supervisory Control

The presentation given here has been adapted from [2]. Let $\Sigma_o \subseteq \Sigma$ and $\Sigma_c \subseteq \Sigma$ be the sets of *observable* and *controllable* events, respectively.

A *run* of a *plant* automaton $G = (Q, \Sigma, \delta, q_0, Q_F)$ is an alternated sequence $\rho = q_0, a_1, q_1, a_2, \ldots, q_{n-1}, a_{n-1}, q_n$ such that $n \geq 0$ and $\delta(q_{i-1}, a_{i-1}) = q_i$ for all $i \leq n$. The run is *accepted* if $q_n \in Q_F$. The *trace* of the run is the (possibly empty) word $a_1 a_2 \ldots a_{n-1}$. The *observable trace* of the run (w.r.t. Σ_o) is the *natural projection* of $a_1 a_2 \ldots a_{n-1}$ on Σ_o^*, defined inductively with $\pi_o(\varepsilon) = \varepsilon$ and

- $\pi_o(a_1 a_2 \ldots a_i) = \pi_o(a_1 a_2 \ldots a_{i-1}) a_i$ if $a_i \in \Sigma_o$,
- $\pi_o(a_1 a_2 \ldots a_i) = \pi_o(a_1 a_2 \ldots a_{i-1})$ otherwise.

An admissible control w.r.t. $\Sigma_c \subseteq \Sigma$ is a map $f : \Sigma_o^* \to 2^\Sigma$ such that all values taken by this map are supersets of $\Sigma \setminus \Sigma_c$. Applying the control f to G means disabling after a trace w all events which do not belong to $f(\pi_o(w))$. The induced restrictions of $\mathcal{L}(G)$ and $\mathcal{L}(\mathcal{U}(G))$ under the control f are denoted $\mathcal{L}(f/G)$ and $\mathcal{L}(f/\mathcal{U}(G))$, respectively. The control f is *non-blocking* if $\mathcal{L}(f/\mathcal{U}(G))$ is equal to the prefix-closure of $\mathcal{L}(f/G)$, i.e. every run of G under the control f can be extended to an accepted run compatibly with f. An admissible and non-blocking control is said to be a proper control. Given a control f, let C be any (possibly infinite) LTS such that the language of C is equal to $\mathcal{L}(f/\mathcal{LTS}(\Sigma^*))$. Then $\mathcal{L}(f/\mathcal{U}(G)) = \mathcal{L}(\mathcal{U}(G) \times C)$ and $\mathcal{L}(f/G) = \mathcal{L}(G \times C)$ where the final states of $G \times C$ are all pairs (q, s) with q final in G. Moreover, if one considers exclusively the *maximal* control maps f such that $\mathcal{L}(f/G) = \mathcal{L}(G \times C)$, where $f \leq f'$ if $f(\omega) \subseteq f'(\omega)$ for all $\omega \in \Sigma_o^*$, then f and C determine each other up to the above equality relation. For this reason, C is called a *controller*, and $G \times C$ and $\mathcal{U}(G) \times C$ are rewritten C/G and $C/\mathcal{U}(G)$ to stress this view. In order to characterize the behaviours that may be enforced on a given plant by supervisory control, Ramadge and Wonham introduced two central concepts.

Definition 1. A prefix-closed language $K \subseteq \mathcal{L}(\mathcal{U}(G))$ is *controllable* w.r.t. Σ_c if $K \cdot (\Sigma \setminus \Sigma_c) \cap \mathcal{L}(\mathcal{U}(G)) \subseteq K$. $\diamond$

Definition 2. A prefix-closed language $K \subseteq \mathcal{L}(\mathcal{U}(G))$ is *observable* w.r.t. Σ_o and Σ_c if, for any $w, w' \in K$ with equal observations $\pi_o(w) = \pi_o(w')$ and for any controllable event $a \in \Sigma_c$, $(wa \in \mathcal{L}(\mathcal{U}(G)) \wedge w'a \in K) \Rightarrow wa \in K$. $\diamond$

A classical Ramadge and Wonham's theorem states that, given a prefix-closed sublanguage $K \subseteq \mathcal{L}(\mathcal{U}(G))$, $K = \mathcal{L}(f/\mathcal{U}(G))$ for some control f if and only if K is controllable and observable. Similarly, if K is a sublanguage of $\mathcal{L}(G)$, then $K = \mathcal{L}(f/G)$ for some non-blocking control f if and only if $K = \overline{K} \cap \mathcal{L}(G)$ and $\overline{K}$ is controllable and observable. In the particular case where every controllable event is observable, i.e. $\Sigma_c \subseteq \Sigma_o$, a prefix-closed language $K \subseteq \mathcal{L}(\mathcal{U}(G))$ is observable if and only if it is *normal* according to the following definition.

Definition 3. A prefix-closed language $K \subseteq \mathcal{L}(\mathcal{U}(G))$ is *normal* w.r.t. Σ_o if $\pi_o^{-1} \circ \pi_o(K) \cap \mathcal{L}(\mathcal{U}(G)) \subseteq K$. $\diamond$

Prefix-closedness, controllability and normality are preserved under arbitrary unions of languages. Therefore, if K is a prefix-closed sublanguage of $\mathcal{L}(\mathcal{U}(G))$, then there exists a supremal prefix-closed, controllable and normal sublanguage $K^\dagger$ of K. Similarly, if K is a sublanguage of $\mathcal{L}(G)$, then there exists a supremal sublanguage $K^\dagger$ of K such that $\overline{K}^\dagger$ is controllable and normal. In both situations, Ramadge and Wonham have shown that the induced control $f^\dagger$ such that $K^\dagger = \mathcal{L}(f^\dagger/\mathcal{U}(G))$ or $K^\dagger = \mathcal{L}(f^\dagger/G)$ is regular, i.e. $K^\dagger = \mathcal{L}(C/\mathcal{U}(G))$ or $K^\dagger = \mathcal{L}(C/G)$ for some finite state controller C, and they have given algorithms for computing C. In the particular case where $\Sigma_c \subseteq \Sigma_o$, such a controller is said to be maximally permissive because $\mathcal{L}(C'/G) \subseteq \mathcal{L}(C/G)$ for any other proper controller C' such that $\mathcal{L}(C'/G) \subseteq K$. Moreover in that case, one may assume w.l.o.g. that $\delta(r, a) = r$ in $C = (R, \Sigma, \delta, r_0)$ for all $r \in R$ and for all $a \in (\Sigma \setminus \Sigma_o)$, hence one may as well consider C as an LTS over Σ_o. This is the kind of supervisors we shall consider in the sequel. Note that when $\Sigma_c \subseteq \Sigma_o$, if a supervisor C is given as an LTS over Σ_o, then $\mathcal{L}(C/G)$ is normal by construction with respect to Σ_o and $\mathcal{L}(G)$ and thus it is observable.

2.3 Modal Specifications of Services

In this section, we propose to specify services using an extended form of Larsen's modal specifications [5]. The extension is twofold. On the one hand, service specifications express requirements about the behaviour of the service seen from the environment (service requester), hence they should abstract from the unobservable events of the service provider. Given modal specifications over a set of observable events Σ_o, we shall therefore consider as an associated class of models a family of LTS's over a larger

alphabet of events $\Sigma \supseteq \Sigma_o$ (drawing inspiration from Hüttel and Larsen's *observable refinements* which were introduced for the same purpose in [6]). On the other hand, we shall provide modal specifications with final states, and similarly for their LTS models (service providers), so that the ability to terminate interactive sessions can be taken into account as a main requirement in the definition of the satisfaction relation.

In the sequel, Σ and Σ_o are two fixed finite sets of events, with $\Sigma_o \subseteq \Sigma$, and we let $\Sigma_{uo} = \Sigma \backslash \Sigma_o$. Given an automaton $A = (Q, \Sigma, \delta, q_0, Q_F)$, we say that a subset of states $P \subseteq Q$ is *closed under unobservable transitions* if $\delta(P, \Sigma_{uo}^*) \subseteq P$.

Definition 4. A deterministic *modal transition system* (or MTS) over Σ_o is a 5-tuple $MTS = (S, \Sigma_o, \delta^\Box, \delta^\Diamond, s_0)$ where $\delta^\Box : S \times \Sigma_o \to S$ and $\delta^\Diamond : S \times \Sigma_o \to S$ are two partial maps, called the *strong* and *weak* labelled transition maps, respectively, subject to the constraint that $\delta^\Box$ is a restriction of $\delta^\Diamond$. A deterministic *modal automaton* or *modal specification* over Σ_o is a modal transition system with final states, i.e. $\mathcal{S} = (S, \Sigma_o, \delta^\Box, \delta^\Diamond, s_0, S_F)$ with $S_F \subseteq S$. The *plain transition system underlying* $\mathcal{S}$ is $\mathcal{U}(\mathcal{S}) = (S, \Sigma_o, \delta^\Diamond, s_0)$. ◇

An easy way to define a satisfaction relation between automata over Σ and modal specifications over Σ_o would be to set $A \models \mathcal{S}$ if $Det_{\Sigma_o}(A) \models \mathcal{S}$ where $Det_{\Sigma_o}(A)$ is the deterministic automaton over Σ_o produced by applying to A the classical subset construction. $P \models s$ could then be defined inductively for states P of $Det_{\Sigma_o}(A)$ and s of $\mathcal{S}$ as suggested in the introduction. Such a definition would unfortunately not ensure that, when $P \models s$ and $q \in P$, joint final states can be reached from (q, s) in $A \times \mathcal{U}(\mathcal{S})$, nor that a can be triggered from some state in the unobservable reach of q whenever $\delta^\Box(s, a)$ is defined. Therefore, unobservable actions must be considered explicitly in the definition of the satisfaction relation. In the sequel, we use $s \xrightarrow{a}_\Box s'$ and $s \xrightarrow{a}_\Diamond s'$ as abbreviations respectively for $\delta^\Box(s, a) = s'$ and $\delta^\Diamond(s, a) = s'$. This notation is extended inductively from letters $a \in \Sigma_o$ to words $\omega \in \Sigma_o^*$.

Definition 5. An automaton $A = (Q, \Sigma, \delta, q_0, Q_F)$ satisfies a modal specification $\mathcal{S} = (S, \Sigma_o, \delta^\Box, \delta^\Diamond, s_0, S_F)$ (noted $A \models \mathcal{S}$) if $Q_0 \models s_0$ where $Q_0 = \delta(q_0, \Sigma_{uo}^*)$ and $\models$ is the largest relation between subsets of states P closed under unobservable transitions and states of $\mathcal{S}$ such that $P \models s$ entails the following properties for all $a \in \Sigma_o$ and $q \in P$:

(1) $\delta(q, a)$ defined $\Rightarrow s \xrightarrow{a}_\Diamond s'$ and $P' \models s'$ for $P' = \delta(P, a\Sigma_{uo}^*)$,
(2) $s \xrightarrow{a}_\Box s' \Rightarrow \delta(q, \Sigma_{uo}^* a) \neq \emptyset$,
(3) $Q_F \times S_F$ can be reached from (q, s) in $A \times \mathcal{U}(\mathcal{S})$. ◇

According to the above definition, A satisfies $\mathcal{S}$ if the fair abstraction of A w.r.t. the unobservable transitions satisfies $\mathcal{S}$ with the definition given in [5] and moreover, any interactive session in which A is used as specified in $\mathcal{S}$ can be completed (the service provider and the service requester can always reach final states jointly). An illustration of the use of modal automata for specifying services is proposed below.

Example 2. Consider the modal specification $\mathcal{S}$ depicted in Figure 3. $\mathcal{S}$ defines the service offered by a (special) coffee-machine [1]. The initial state is s_0. The final states are s_0 and s_5. Initially, the user must be enabled to insert

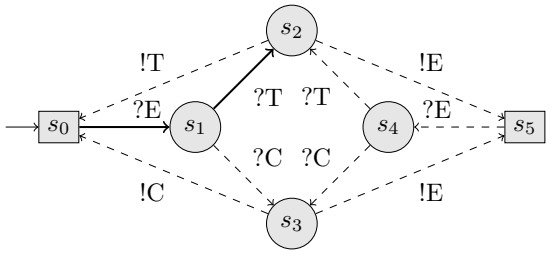

Figure 3. Specification $\mathcal{S}$ of a coffee-machine

a coin (?E) in the machine and further to order a tea (?T). He may also have the possibility to order a coffee (?C), but this is optional. Afterwards, the machine may deliver the requested beverage (!C or !T)) or return the coin (!E) and the service may then end (in state s_0 or s_5). If the coin is returned, there is no guarantee that the user can insert a new coin and order a coffee or a tea (all these transitions are optional). However, if the user succeeds to have his beverage, then he must be enabled to insert a new coin and order a tea.

The automaton A of Figure 4 represents a possible coffee-machine plant, where $\Sigma_{uo} = \{\text{NC, NT, R}\}$ (NC (resp. NT) means no coffee (resp. no tea) and R stands for Reset) and $Q_F = \{1, 2, 3\}$. It is easy to check that the coffee-machine

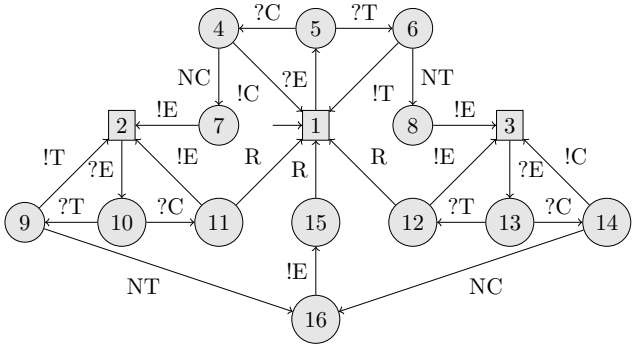

Figure 4. A possible implementation A of the coffee-machine

plant A fulfills properties *1.* and *2.* of Definition 5 w.r.t. $\mathcal{S}$ whereas requirements *3.* is not satisfied. Indeed, after the observation ?E?C!E?E?C, $\mathcal{S}$ is in state s_3 and A is either in states 11 or 1, i.e. $P = \{11, 1\}$ and in $A \times \mathcal{U}(\mathcal{S})$, the state $(1,s_3)$ is a deadlock state. So there is no possibility for A and $\mathcal{S}$ to reach final states jointly. Note that state 1 is not a blocking state of A considered alone, since it is a final state. ◇

Modal specifications where all states are final have the same expressive power as μ-calculus without least fixed points and disjunction [3]. Hence, unlike CTL, CTL* or μ-calculus, they cannot be used to describe branching behaviors. The addition of final states gives modal specifications the ability to capture nonblocking properties that cannot

[1] The transitions $\to_\Box$ and $\to_\Diamond$ are respectively represented with plain arrows and dashed arrows

be expressed in μ-calculus without least fixed points and disjunction.

Modal specifications of services diverge notably from the *operating guidelines for services* introduced in [8] and further studied in [9], although they have much in common. Operating guidelines abstract from unobservable events, and they take final states into account for guaranteeing the absence of deadlock in the closed system formed by the service provider and the service requester. However, operating guidelines do not guarantee that this closed system is free of dead-ends, i.e. global states which are not deadlocks but from which global final states cannot be reached. For the rest, operating guidelines are technically closer to acceptance automata [4] than to modal automata [5]. In the absence of final states, modal automata are less expressive than acceptance automata, see [10] for a comparison. Nevertheless, this weakness seems to be overcompensated by the provision of a satisfaction relation for modal specifications which guarantees the absence of deadends.

3. ANY SERVICE SPECIFICATION HAS AN OPTIMAL FINITE STATE SUPERVISOR

In the rest of the paper, $\Sigma_c \subseteq \Sigma_o \subseteq \Sigma$, where Σ_o and Σ_c are the subsets of observable and controllable events, respectively [2] . We let $\Sigma_{uo} = \Sigma \backslash \Sigma_o$, respectively $\Sigma_{uc} = \Sigma \backslash \Sigma_c$ denote the subset of unobservable, respectively uncontrollable events. We consider a *plant* $G = (Q, \Sigma, \delta_Q, q_0, Q_F)$, or service provider, and a modal specification of the expected service $\mathcal{S} = (S, \Sigma_o, \delta_S^{\Box}, \delta_S^{\Diamond}, s_0, S_F)$. The control problem, we want to solve is the following:

Problem 1. Given a plant G and an expected service as above, we search for the optimal (i.e. maximally permissive) supervisor $C = (R, \Sigma_o, \delta_R, r_0)$ such that $C/G \models \mathcal{S}$ and C is an admissible controller for G w.r.t. Σ_c.

Let us recall that C/G denotes the reachable restriction of the automaton $(Q \times R, \Sigma, \delta_{Q\times R}, (q_0, r_0), Q_F \times R)$ where $\delta_{Q\times R}((q,r),a) = (\delta_Q(q,a), \delta_R(r,a))$ if $a \in \Sigma_o$ and $\delta_{Q\times R}((q,r),a) = (\delta_Q(q,a), r)$ otherwise (assuming that the right members of these equations are defined).

By definition, in order that C be an admissible controller for G w.r.t. (Σ_o and) Σ_c, the following condition should hold for any state (q,r) of C/G and for any uncontrollable event $a \in \Sigma_{uc} \cap \Sigma_o$:

- $\delta_Q(q,a)$ defined $\Rightarrow \delta_R(r,a)$ defined.

Note that as previously mentioned, as C is defined over Σ_o, the language of C/G is automatically normal w.r.t. Σ_o and G.

In view of Definition 5, in order that $C/G \models \mathcal{S}$, it is moreover necessary that $(Q_0, r_0) \models s_0$ where $Q_0 = \delta_Q(q_0, \Sigma_{uo}^*)$ and $\models$ is the largest relation on $(2^Q \times R) \times S$ such that, whenever $(P,r) \models s$, the following properties hold for all $a \in \Sigma_o$ and $q \in P$:

(1) $\delta_Q(q,a)$ defined and $\delta_R(r,a)$ defined $\Rightarrow$ $s \xrightarrow{a}_{\Diamond} s'$ and $(P', r') \models s'$ for $P' = \delta_Q(P, a\Sigma_{uo}^*)$ and $r' = \delta_R(r,a)$,
(2) $s \xrightarrow{a}_{\Box} s' \Rightarrow \delta_Q(q, \Sigma_{uo}^* a) \neq \emptyset$ and $\delta_R(r,a)$ defined,
(3) $(Q_F \times R) \times S_F$ can be reached from $((q,r),s)$ in $(\mathcal{U}(G) \times C) \times \mathcal{U}(\mathcal{S})$.

It is important to note that the above characterization applies unchanged to $G' = (Q, \Sigma, \delta_Q, Q_0, Q_F)$, where $Q_0 = \delta_Q(q_0, \Sigma_{uo}^*)$, i.e. to the original plant automaton G in which the initial state q_0 has been replaced with its unobservable reach Q_0.

The state oriented and therefore co-inductive characterization of the admissible supervisors C such that $C/G \models \mathcal{S}$ which we have presented above is not always convenient. Alternatively, a non-inductive characterization of these supervisors may be given in terms of their languages $\mathcal{L}(C)$, as follows.

Given any non-empty prefix-closed language $K \subseteq \Sigma_o^*$, let $C = \mathcal{LTS}(K)$ and hence $K = \mathcal{L}(C)$, then C is an admissible supervisor for G if and only if

$$\forall a \in \Sigma_{uc} \cap \Sigma_o \, \forall \omega \in K \, \forall w \in \mathcal{L}(\mathcal{U}(G)) :$$
$$\omega = \pi_o(w) \wedge wa \in \mathcal{L}(\mathcal{U}(G)) \Rightarrow \omega a \in K$$

With the same definition as above, $C/G \models \mathcal{S}$ if and only if the following properties hold for all $a \in \Sigma_o$, for all $\omega \in K$, for all $s \in S$ such that $s_0 \xrightarrow{\omega}_{\Diamond} s$ in $\mathcal{U}(\mathcal{S})$, and for all $w \in \mathcal{L}(\mathcal{U}(G))$ such that $\omega = \pi_o(w)$:

(1) $wa \in \mathcal{L}(\mathcal{U}(G))$ and $\omega a \in K \Rightarrow s \xrightarrow{a}_{\Diamond} s'$ for some $s' \in S$,
(2) $s \xrightarrow{a}_{\Box} s' \Rightarrow \exists u \in \Sigma_{uo}^* : wua \in \mathcal{L}(\mathcal{U}(G))$ and $\omega a \in K$,
(3) $\exists v \in \Sigma^* \, \exists \nu \in \Sigma_o^* :$ $wv \in \mathcal{L}(G) \wedge \nu = \pi_o(v) \wedge \omega\nu \in K \wedge s \xrightarrow{\nu}_{\Diamond} s'$ for some $s' \in S_F$.

The two characterizations are equivalent since one may pass from the former to the latter by replacing q with $\delta_Q(q_0, w)$ and r by $\delta_R(r_0, \omega)$, and vice-versa. Therefore, in the latter characterization, after applying universal quantification over $w \in \mathcal{L}(\mathcal{U}(G))$ subject to $\pi_o(w) = \omega$, one gets three conditions that depend exclusively upon $P = \delta_Q(q_0, \pi_o^{-1}(\omega))$, $r = \delta_R(r_0, \omega)$ and s. From now on, let $\mathcal{K}$ denote the family of the non-empty prefix closed languages $K \subseteq \Sigma_o^*$ that satisfy the conditions stated in the second characterization, and let $K^\dagger = \cup\mathcal{K}$. It is straightforward to show that $K^\dagger$ belongs to $\mathcal{K}$. Therefore, if $\mathcal{K} \neq \emptyset$, there exists a unique solution to problem 1.

In the rest of the section we prove that, if there exists an admissible supervisor C such that $C/G \models \mathcal{S}$, then there exists such an optimal and moreover finite-state supervisor $C^\dagger$. Further, we show that $R = 2^Q \times S$ may be chosen as the set of states of $C^\dagger$.

Given any non-empty prefix-closed language $K \subseteq \Sigma_o^*$, let $C = \mathcal{LTS}(K)$. Then C is an admissible supervisor and $C/G \models \mathcal{S}$ if and only if $K \in \mathcal{K}$. As $K^\dagger$ is the supremal element of $\mathcal{K}$, it follows clearly that, unless $\mathcal{K} = \emptyset$, $\mathcal{LTS}(K^\dagger)$ is an optimal supervisor. However, this supervisor may have an infinite number of states. We prove below that $K^\dagger$ is a regular language, showing that $K^\dagger = \mathcal{L}(C^\dagger)$ for some finite state supervisor $C^\dagger$.

[2] Thus we set ourselves in a case where observability and normality coincide, entailing the existence of an optimal control for language specifications.

Proposition 1. $K^\dagger$ is a regular language.

Proof For any $\omega \in K^\dagger$, let $\rho(\omega) = \{\nu \in \Sigma_o^* \,|\, \omega\nu \in K^\dagger\}$ (thus $\rho(\omega)$ is a right derivative of $K^\dagger$), and let $\xi(\omega) = (P, s)$ with $P = \delta_Q(q_0, \pi_o^{-1}(\omega))$ and $s_o \xrightarrow{\omega}_\diamond s$. By Myhill and Nerode's theorem, in order to show that $K^\dagger$ is regular, it suffices to construct $\overline{\rho} : 2^Q \times S \to \mathcal{P}(\Sigma_o^*)$ such that $\rho(\omega) = \overline{\rho} \circ \xi(\omega)$ for all $\omega \in K^\dagger$. We show that this relation is satisfied with $\overline{\rho}\,(P, s) = \cup\{\rho(\omega) \,|\, \xi(\omega) = (P, s)\}$. Let $\omega, \omega' \in K^\dagger$ such that $\xi(\omega) = \xi(\omega')$. By Lemma 1 given below, $\rho(\omega) = \rho(\omega')$, and therefore $\rho(\omega) = \overline{\rho} \circ \xi(\omega)$. □

Lemma 1. For any $\omega \in K^\dagger$, let $K^\dagger \triangleright \omega = \{\nu \in \Sigma_o^* \,|\, \omega\nu \in K^\dagger\}$ $(= \rho(\omega))$, and let $\mathcal{S} \triangleright \omega = (S, \Sigma_o, \delta_S^\Box, \delta_S^\Diamond, s_o \triangleright \omega, S_F)$ and $G \triangleright \omega = (Q, \Sigma, \delta_Q, q_0 \triangleright \omega, Q_F)$ be defined with $s_0 \xrightarrow{\omega}_\diamond (s_o \triangleright \omega)$ and $q_0 \triangleright \omega = \delta_Q(q_o, \pi_o^{-1}(\omega))$. Then $K^\dagger \triangleright \omega$ is the language of an admissible controller enforcing the modal specification $\mathcal{S} \triangleright \omega$ on the plant $G \triangleright \omega$, and moreover it is the largest language with this property.

Proof When specialized to words $\omega\omega' \in K^\dagger$, the conditions specified in the characterization of $\mathcal{K}$ in order that $K^\dagger$ should be an admissible controller enforcing $\mathcal{S}$ on G are the same as the conditions required in order that $K^\dagger \triangleright \omega$ should be an admissible controller enforcing $\mathcal{S} \triangleright \omega$ on $G \triangleright \omega$. The second assertion may be established by contradiction. If $K^\dagger \triangleright \omega$ was not the largest solution to the derived control problem, then $K^\dagger \cup \omega(K^\dagger \triangleright \omega)$ would be a solution to the original control problem strictly larger than $K^\dagger$, which is impossible. □

4. AN ITERATIVE ALGORITHM FOR COMPUTING AN OPTIMAL FINITE STATE SUPERVISOR

With Proposition 1, we have obtained a non-constructive proof of the regularity of $K^\dagger$. In this section, we propose an algorithm for constructing from G and $\mathcal{S}$ an optimal finite state supervisor $C^\dagger$ or deciding that no supervisor can enforce $\mathcal{S}$ on G. It will be proved in Section 4.5 that the proposed decision and synthesis algorithm is correct: if it does not yield any supervisor, then $K^\dagger = \emptyset$, and in the converse case, $K^\dagger = \mathcal{L}(C^\dagger)$. Not surprisingly, the set of states of $C^\dagger$ is a subset of $2^Q \times S$ (Q and S are the respective sets of states of G and $\mathcal{S}$). As usual in supervisor synthesis, the algorithm starts with an expansion stage (A), in which an abstraction of the reachable part of $\mathcal{U}(G) \times \mathcal{U}(\mathcal{S})$ is built inductively, and it proceeds with reduction stages, performed in rounds until a fixpoint is reached. A particularity lies in the (non-strict) alternation between two types of reduction stages, on the one hand stages (B) that eliminate states inconsistent with modal transitions, and on the other hand stages (C) that eliminate states from which joint termination is not possible. The algorithm has the control pattern A;(B;C)*.

4.1 The expansion stage A

Given $\mathcal{U}(G) = (Q, \Sigma, \delta_Q, q_0)$ and $\mathcal{U}(\mathcal{S}) = (S, \Sigma_o, \delta_S^\Diamond, s_0)$, let $r_0 = (\delta_Q(q_0, \Sigma_{uo}^*), s_0)$ where $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ and let $C^0 = (R, \Sigma_o, \delta_R, r_0)$ be the LTS defined inductively as follows. $R \subseteq 2^Q \times S$ and $\delta_R : R \times \Sigma_o \to R$ are the least set and partial function, respectively, such that $r_0 \in R$ and for any $(P, s) \in R$ and $a \in \Sigma_o$, $\delta_R((P, s), a) = (P', s') \in R$ with $P' = \delta_Q(P, a\Sigma_{uo}^*)$ and $s' = \delta_S^\Diamond(s, a)$, unless $\delta_S^\Diamond(s, a)$ is undefined or $(\forall q \in P)\ \delta_Q(q, a) = \emptyset$ or $\delta_S^\Box(s, a)$ is defined and $(\exists q \in P)\ \delta_Q(q, \Sigma_{uo}^* a) = \emptyset$. As R is a subset of the finite set $2^Q \times S$, this inductive construction is finite.

4.2 The reduction stage B

Given $C^i = (R, \Sigma_o, \delta_R, r_0)$ where i is an even number, one computes C^{i+1} from C^i by removing iteratively from C^i all states and transitions which are found inconsistent with the requirements expressed by the modal transitions in $\mathcal{S}$. This is done by applying the classical Ramadge-Wonham algorithm for state based supervisory control, with $\Sigma_{uc} = \Sigma \setminus \Sigma_c$ as the set of uncontrollable events.

Declare *inconsistent w.r.t. controllability or modalities* any state $(P, s) \in R$ such that at least one of the following two properties hold:

- $\exists a \in \Sigma_{uc} \cap \Sigma_o\ \exists q \in P : \delta_Q(q, a)$ defined $\wedge$ $\delta_R((P, s), a)$ undefined,
- $\exists a \in \Sigma_o : \delta_S^\Box(s, a)$ defined $\wedge$ $\delta_R((P, s), a)$ undefined

Whenever some state (P, s) is found inconsistent, this state is removed from R and from the image of the partial function δ_R, which may lead to new inconsistencies w.r.t. controllability or modalities. Note that the inconsistencies w.r.t. $\delta_S^\Diamond$ have already been considered in the construction of C^0 and new inconsistency of this type may be introduced by restricting R and δ_R. As R is a finite set, this iterative cleaning procedure terminates (possibly with $R = \emptyset$). The result does not depend upon the order in which the states and transitions are removed.

4.3 The reduction stage C

Given $C^i = (R, \Sigma_o, \delta_R, r_0)$ where i is an odd number, one computes C^{i+1} from C^i by removing iteratively from C^i all states and transitions that cannot lead to joint termination w.r.t. the final states of G and $\mathcal{S}$.

Define $H^i = \mathcal{U}(G) \times C^i$, thus any state of H^i is of the form $(q, (P, s))$ with $q \in P$. Declare *inconsistent w.r.t. termination* any state $(P, s) \in R$ such that, for some $q \in P$, $q \notin Q_F \vee s \notin S_F$ and there is no path in H^i from $(q, (P, s))$ to any $(q', (P', s'))$ with $q' \in Q_F$ and $s' \in S_F$. Whenever some state (P, s) is found inconsistent, it is simply removed from R and from the image of the partial function δ_R.

Remark 1. In practice, H^i may be computed from H^{i-1} by just cancelling states whose second projection is not in C^i.

4.4 The halting condition

The algorithm executes according to the pattern A;(B;C)*. The iteration is stopped as soon as $C^i = C^{i+2}$, which must eventually occur since, at each step in the iteration, the set of states is decreased or left constant. When the fixpoint is reached, one declares that the control problem has no solution if C^i has an empty set of states, and one sets $C^\dagger = C^i$ otherwise.

Example 3. To illustrate the algorithm, let us come back to example 2. We assume that the set of controllable events is reduced to $\Sigma_c = \{?\text{E}, ?\text{T}\}$ (note that there

is however no direct relation between may/must and controllable/uncontrollable). Figure 5 represents the LTS C^0 computed from $\mathcal{S}$ (Figure 3) and A (Figure 4) according to Section 4.1. As remarked in Example 2, there iss no

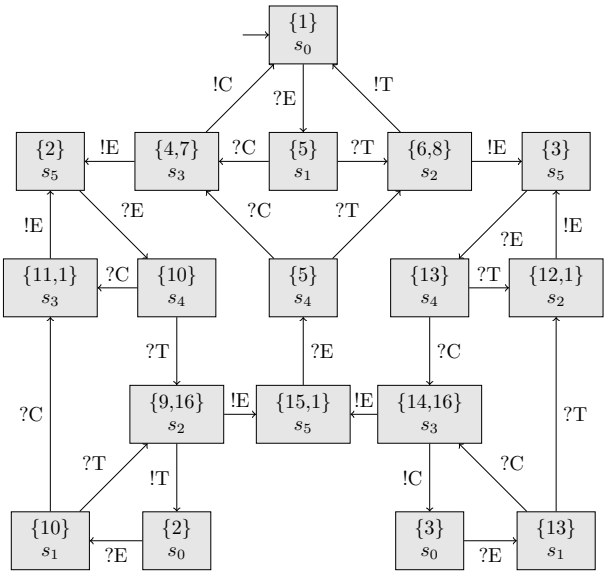

Figure 5. C^0

inconsistent state w.r.t. controllability or modality in C^0. Thus, Stage B of the algorithm does not remove any states in C^0 and $C^1 = C^0$. However, state $(\{11,1\}, s_3)$ as well as state $(\{12,1\}, s_2)$ are inconsistent w.r.t. termination and have to be removed from the state space. We thus obtain C^2. As $C^2 \neq C^0$, we need to iterate the process. In C^2, states $(\{10\}, s_4)$ and $(\{10\}, s_1)$ are now inconsistent w.r.t. controllability and are removed. Moreover, $(\{13\}, s_1)$ has also become inconsistent w.r.t. modalities since in $\mathcal{S}$, the transition ?T is mandatory from s_1. By removing this state, $(\{3\}, s_0)$ also becomes inconsistent w.r.t. modalities and the same applies in turns to $(\{14,16\}, s_3)$ and to $(\{13\}, s_4)$ due to the uncontrollability of !C and ?C. We thus obtain the LTS C^3 described in Figure 6.

Finally, in C^3, the sink states $(\{2\}, s_5)$, $(\{3\}, s_5)$ map to final states in $H^3 \times \mathcal{U}(\mathcal{S})$, so $C^3 = C^4$ and we have reached the fix-point. ◇

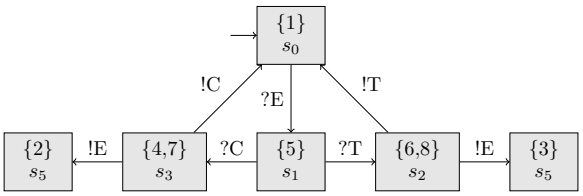

Figure 6. C^3

4.5 Correctness of the algorithm: $\mathcal{L}(C^\dagger) = K^\dagger$

In this section, we show that the finite state LTS $C^\dagger$ constructed in section 4 realizes $K^\dagger$, thus in particular $C^\dagger$ has an empty set of states if and only if $K^\dagger$ is an empty language.

For all $i \geq 0$, let $H^i = \mathcal{U}(G) \times C^i$. Three observations about the LTS's H^i and $H^i \times \mathcal{U}(\mathcal{S})$ are fundamental for the propositions established below. First, every state of H^i is of the form $(q, (P, s))$ with $q \in P$ and P closed under unobservable transitions in G. Second, every reachable state of $H^i \times \mathcal{U}(\mathcal{S})$ is of the form $((q, (P, s)), s)$, where the same state $s \in S$ occurs twice. Third, for $i \geq 1$, H^i is isomorphic, as an LTS over Σ, to the expansion $Exp(C^i)$ of C^i defined as follows.

Definition 6. Given $C^i = (R, \Sigma_o, \delta_R, r_0)$, let $Exp(C^i) = (Exp(R), \Sigma, \delta^i, E_0)$ where the set of expanded states is $Exp(R) = \{(q, (P, s)) \,|\, (P, s) \in R \wedge q \in P\}$, the initial state is $E_0 = (q_0, (\delta_Q(q_0, \Sigma_{uo}^*), s_0))$, and the transition function $\delta^i : Exp(R) \times \Sigma \to Exp(R)$ is defined as follows:

- for $a \in \Sigma_{uo}$, let $\delta^i((q, (P, s)), a) = (\delta_Q(q, a), (P, s))$,
- for $a \in \Sigma_o$, let
  $\delta^i((q, (P, s)), a) = (\delta_Q(q, a), (\delta_Q(P, a\Sigma_{uo}^*), \delta_S^\Diamond(s, a))$
  unless $\delta_Q(P, a) = \emptyset$. ◇

Remark 2. H^0 is not necessarily isomorphic to $Exp(C^0)$ because in this particular case there may exist $(P, s) \in R$, $q, q' \in P$ and $a \in \Sigma_o$ such that $\delta_S^\Box(s, a)$ defined, $\delta_Q(q, a)$ defined, and $\delta_Q(q', \Sigma_{uo}^* a) = \emptyset$, and then $\delta_R((P, s), a)$ is undefined in C^0 (see stage A of the synthesis algorithm).

Proposition 2. If $C^\dagger$ has a non-empty set of states, then $C^\dagger$ is an admissible controller and $C^\dagger / G \models \mathcal{S}$.

Proof Suppose for a contradiction that $C^\dagger$ is not an admissible controller, and let $C^\dagger = C^i$ with $i \geq 2$, thus $C^\dagger / G = H^i$. In view of the isomorphism between H^i and $Exp(C^i)$, the following situation is met for some $a \in \Sigma_{uc} \cap \Sigma_o$ and for some reachable state $(q, (P, s))$ of $Exp(C^i)$:

- $\delta_Q(q, a)$ is defined and $\delta^i((q, (P, s)), a)$ is undefined, but in this case,
  $\delta_Q(P, a) \neq \emptyset$ since $q \in P$, hence $\delta_S^\Diamond(s, a)$ must be undefined, and therefore (P, s) is an inconsistent state of C^i (w.r.t. controllability), which is impossible.

Now suppose for a contradiction that $C^\dagger / G$ does not satisfy the modal specification $\mathcal{S}$. In view of Definition 5 and the isomorphism between H^i and $Exp(C^i)$, at least one of the following situations is met for some $a \in \Sigma_o$ and for some reachable state $(q, (P, s))$ of $Exp(C^i)$:

- $\delta^i((q, (P, s)), a)$ is defined and $\delta_S^\Diamond(s, a)$ is undefined, but this is not possible because the first assumption implies that $\delta_R((P, s), a)$ is defined in C^i and hence in C^0, entailing that $\delta_S^\Diamond(s, a)$ is defined,
- $s \xrightarrow{a}_\Box s'$ and $\delta^i((q, (P, s)), \Sigma_{uo}^* a) = \emptyset$,
  then by definition of δ^i, one of the following two situations is met (recall that $q \in P$ and P is closed under unobservable transitions):
  - $\delta_Q(q, a\Sigma_{uo}^* a) = \emptyset$, but in this case, $\delta_R((P, s), a)$ would have been left undefined in C^0 (see Stage A of the synthesis algorithm), hence (P, s) would have been removed from R in C^1 at Stage B of the synthesis algorithm, showing a contradiction;
  - $\delta_Q(P, a) \neq \emptyset$ and $(\delta_Q(P, a\Sigma_{uo}^*), \delta_S^\Diamond(s, a))$ is not a state of C^i, but then (P, s) in an inconsistent

state of C^i (w.r.t. modalities), which is impossible.

- one cannot reach any state $(q', (P', s'))$ with $q' \in Q_F$ and $s' \in S_F$ from $(q, (P, s))$ in $Exp(C^i)$,
  but then (P, s) in an inconsistent state of C^i (w.r.t. termination), which is impossible.

As all cases have been examined, the proposition obtains. □

Proposition 3. If $K^\dagger \neq \emptyset$, then $\mathcal{L}(C^\dagger) = K^\dagger$.

Proof For any $\omega \in \Sigma_o^*$, let $\xi(\omega) = (\delta_Q(q_0, \pi_o^{-1}(\omega)), \delta_S^\Diamond(s_o, \omega))$. By Lemma 1, $\xi(\omega) = \xi(\omega') \Rightarrow K^\dagger \triangleright \omega = K^\dagger \triangleright \omega'$ and moreover, $K^\dagger \triangleright \omega$ is the language of an admissible controller enforcing $\mathcal{S} \triangleright \omega$ on $G \triangleright \omega$. Thus, if $\omega \in K^\dagger$ and $\xi(\omega) = (P, s)$, then for all $a \in \Sigma_o$ such that $\delta_S^\Box(s, a)$ is defined, $\delta_Q(q, \Sigma_{uo}^* a)$ is non-empty for all $q \in P$ $(= \delta_Q(q_o, \pi_o^{-1}(\omega)))$. Therefore, the prefix closed language $K^\dagger$ may be generated by a sub-system of C^0, that is to say, by the induced restriction of C^0 on specific subsets of states and transitions. In particular, $K^\dagger \subseteq \mathcal{L}(C^0)$.

Assume by induction on $i \geq 0$ that $K^\dagger \subseteq \mathcal{L}(C^i)$.

- If $C^i = C^{i+2}$ then $C^\dagger = C^i$ and by Proposition 2, $\mathcal{L}(C^\dagger) \subseteq K^\dagger \subseteq \mathcal{L}(C^\dagger)$, showing the result.
- In the converse case, C^i is not an admissible controller or C^i/G does not satisfy $\mathcal{S}$. By the proof of Proposition 2, C^i contains inconsistent states (P, s). Therefore, C^{i+2} has a strictly smaller set of states. We claim that no reachable sub-system of C^i containing a state missing in C^{i+2} can generate a language in $\mathcal{K}$. Therefore, necessarily $K^\dagger \subseteq \mathcal{L}(C^{i+2})$, and the proposition obtains by the induction on i.

  The claim may be established by an induction on the set of inconsistent states successively removed, using at each step the isomorphism between C/G and $Exp(C)$ for any reachable sub-system C of C^i for $i \geq 1$. □

5. CONCLUSION

We have investigated the application of the supervisory control theory to enforce the modal specification of a service on a given plant automaton. We have established that this control problem has at most one solution and that this solution can be represented as a finite state supervisor. Finally, we have shown how to compute this supervisor. The point made in the paper is that, under a mild adaptation (dealing jointly with controllability and modal consistency), Ramadge and Wonham's theory and algorithms of supervisory control under partial observation extend smoothly to specifications more expressive than language specifications. The control synthesis algorithm stays linear in the size of the modal specification (but exponential in the size of the plant due to partial observation). The control synthesis problem is EXP-TIME complete in the size of formulas for CTL and μ-calculus and doubly EXP-TIME complete for CTL* [7], but it is not clear that our algorithm does better in practice. In this paper, we have made the assumption that the set of events that are observed by the controller is the set of events involved in the modal specification of the service. In order to obtain a more general framework, it would be interesting to investigate the control problem for modal specifications in the different setting where the controller's view of the system is not directly related with the interactive user's view. In the context of computer security, modal specifications could also serve to express additional availability constraints. For that reason, it would be interesting to define a common extension of this work and the one in [2] and then to consider a wide range of security properties mixing integrity constraints given by safety properties, confidentiality properties given by opacity predicates and availability properties given by modal specifications. Modal specifications are a simple but powerful description of the functional service that a user expects from an implementation. Among the set of controlled systems satisfying the specification, some implementations may be preferred to others because, e.g., the average elapsed time between the requests and the answers is shorter. It would be interesting to add quantitative criteria to help a selection among the non necessarily maximally permissive solutions of the control problem for modal specifications.

REFERENCES

[1] J.W. Bryans, M. Koutny, L. Mazaré, P.Y.A. Ryan: Opacity Generalized to Transition Systems. *Int. Journal of Computer Security*, vol. 7(6), 2008, pp 421-435.

[2] J. Dubreil, P. Darondeau, H. Marchand: Supervisory Control for Opacity. *IEEE Trans. Automatic Control*, 55(5), 2010, pp 1089-1100.

[3] G. Feuillade, S. Pinchinat: Modal Specifications for the Control Theory of Discrete Event Systems. *Discrete Event Dyn Syst*, vol. 17, 2007, 211-232.

[4] M. Hennessy: Acceptance Trees. *J. ACM*, vol. 32, 1985, 896-928.

[5] K.G. Larsen: Modal Specifications. in: *Automatic Verification Methods for Finite State Systems*, Springer-Verlag, LNCS vol. 407, 1990, pp 232-246.

[6] H. Hüttel and K. G. Larsen. The use of static constructs in a modal process logic. In *Logic at Botik*, 1989, pp 163–180.

[7] S, Jiang, R. Kumar: Supervisory Control of Discrete Event Systems with CTL * Temporal Logic Specifications. *SIAM J. of Control and Optimization*, vol. 44(6), 2006, pp. 2079-2103.

[8] N. Lohmann, P. Massuthe, K. Wolf: Operating Guidelines for Finite-State Services. Proc. ICATPN, Springer-Verlag, LNCS vol. 4546, 2007, pp. 321-341.

[9] N. Lohmann, K. Wolf: Petrifying Operating Guidelines for Services. Proc. ACSD, IEEE Computer Society, 2009, pp. 80-88.

[10] J.B. Raclet: Residual for Component Specifications. *ENTCS* vol. 215, 2008, pp. 93-110.

[11] P.J. Ramadge, W.M. Wonham: Supervisory Control of a Class of Discrete Event Processes. *SIAM J. of Control and Optimization*, vol. 25, 1987, pp 206-230.

[12] P.J. Ramadge, W.M. Wonham: On the Supremal Controllable Language of a Given Language. *SIAM J. of Control and Optimization*, vol. 25, 1987, pp 637-659.

[13] P.J. Ramadge, W.M. Wonham: The Control of Discrete Event Systems. *Proc. of the IEEE, Special Issue on Dynamics of Discrete Event Systems*, vol. 77, 1989, pp 81-98.