# Supervisory Control Theory in Epistemic Temporal Logic

Guillaume Aucher

# Supervisory Control Theory in Epistemic Temporal Logic

Guillaume Aucher
University of Rennes 1
INRIA
Rennes, France
guillaume.aucher@irisa.fr

## ABSTRACT

Supervisory control theory deals with problems related to the existence and the synthesis of supervisors. The role of a supervisor in a system is to control and restrict the behavior of this system in order to realize a specific behavior. When there are multiple supervisors, such systems are in fact multi-agent systems. The results of supervisory control theory are usually expressed in terms of operations like intersection and inclusion between formal languages. We reformulate them in terms of model checking problems in an epistemic temporal logic. Our reformulations are very close to natural language expressions and highlight their underlying intuitions. From an applied perspective, they pave the way for applying model checking techniques developed for epistemic temporal logics to the problems of supervisory control theory.

## Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Multiagent systems

## Keywords

Supervisory control theory, epistemic temporal logic, model checking, synthesis

## 1. INTRODUCTION

Supervisory control theory was developed in the 1980s to deal with so-called Discrete Event Systems (DES), which arise in a number of applications like manufacturing, vehicular traffic, logistics (conveyance and storage of goods) and computer networks [4]. DESs require some control if we want them to comply to a goal behavior. To this aim, supervisory control theory deals with the following issues: does there exist a supervisor that can control a given DES and, if it does, how can we construct it? The role of a supervisor is to control the behavior of a DES in order to realize a goal behavior. Various assumptions are usually considered, such as partial controllability, partial observability and multiplicity of supervisors. This work is relevant for multi-agent systems because, in the case of multiple supervisors, a DES can, in fact, be viewed as a multi-agent system: the agents are the supervisors and the system is the DES itself.

**Appears in:** *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*


A DES is usually simply modeled by an automaton $A$ called the *plant*, and the goal behavior by another automaton $O$. Among other results, supervisory control theory provides conditions deciding under various assumptions whether or not the goal behavior can be *fully realized* by a supervisor $f$, meaning that the behavior of the plant controlled by the supervisor $f$ (denoted $\mathcal{L}(A, f)$) is equal to the goal behavior: $\mathcal{L}(A, f) = \mathcal{L}(O)$. In general, these conditions (called controllability, observability and coobservability) are expressed in terms of intersection, union, complementation, concatenation and projection of the formal languages $\mathcal{L}(A)$ and $\mathcal{L}(O)$. Expressing conditions in these terms does not lend itself easily to a reading in natural language, and the underlying intuitions are often lost or hidden in the formalization. Moreover, there is no generic and automatic procedure for deriving algorithms to check an arbitrary condition.

We reformulate the main results of supervisory control theory in terms of model checking problems expressed in an epistemic temporal logic. As a result of our new approach, we provide not only original solutions to the problems of supervisory control theory, but also new results about decentralized supervision with multiple supervisors. Moreover, because we represent supervisors in terms of model checking problems, this makes it possible to lazily compute them online. Overall, our approach paves the way for applying model checking techniques to supervisory control theory.

The paper is organized as follows. It begins in Section 2 with some formal preliminaries. In Section 3, we recall the basics of supervisory control theory. Our epistemic temporal logic $\mathsf{CTL}^*\mathsf{K}_n^{\mathsf{D}}$ is defined in Section 4, which also contains a complexity analysis of its model checking problem. Then, in Section 5, we provide conditions for the existence of supervisors. When supervisors fully realizing a goal behavior do exist, we provide methods for synthesizing all of them. In Section 6, we provide similar results when the goal behavior cannot be fully realized. We conclude in Section 7 with a discussion of related works and with some remarks.

*Note.* All the proofs of this article can be found in [2].

## 2. PRELIMINARIES

Throughout this article, $E$ is a finite set of *events*. An event of $E$ is generally denoted $\sigma$. $E^*$ is the set of all finite strings of events of $E$, including the empty string denoted $\epsilon$, and $E^\omega$ is the set of infinite strings of events of $E$. Elements of $E^* \cup E^\omega$ are called *words*. A string of events $w = \sigma_1\sigma_2\sigma_3 \ldots \in E^* \cup E^\omega$ represents a sequence of events. If $\mathcal{J}, \mathcal{K} \subseteq E^* \cup E^\omega$, then the *concatenation of* $\mathcal{J}$ *and* $\mathcal{K}$ is the set $\mathcal{J}\mathcal{K} := \{wv : w \in \mathcal{J} \cup \{\epsilon\}, v \in \mathcal{K} \cup \{\epsilon\}\}$. A *regular lan-*

*guage* is a subset of $E^*$ obtained by applying a finite number of times the operations of union, concatenation and Kleene star $*$ over a finite set of events. A *formal language* $\mathcal{K}$ is a subset of $E^* \cup E^\omega$. If $w \in E^* \cup E^\omega$, then a *prefix of* $w$ is an element $\overline{w} \in E^*$ such that $w = \overline{w}v$ for some $v \in E^* \cup E^\omega$; in that case, we write $\overline{w} \leq w$. If $\mathcal{K}$ is a formal language, then the *prefix-closure of* $\mathcal{K}$ is the formal language defined as follows: $\overline{\mathcal{K}} := \{\overline{w} : \overline{w} \leq w \text{ for some } w \in \mathcal{K}\}$.

The set $\mathbb{P}$ denotes the set of *propositional letters* $\mathbb{P} := \{p_\mathcal{K}, p_{\mathcal{L}_m}\} \cup \{p_\sigma : \sigma \in E\}$ and $\mathbb{P}_0$ denotes the subset of propositional letters $\mathbb{P}_0 := \{p_\mathcal{K}, p_{\mathcal{L}_m}\} \subseteq \mathbb{P}$. The set $\mathbb{A}$ denotes the set of *agents*, $\mathbb{A} := \{1, \ldots, n\}$, where $n \in \mathbb{N}^*$.

The behavior of a (discrete event) system is modeled by a transition system. A (deterministic) *transition system* is a tuple $\mathcal{E} = (E, S, \delta, s_0, Lab)$ where $S$ is a set of *states*; $\delta : E \times S \to S$ is a function called the *transition function*; $s_0$ is the *initial state*; and $Lab : S \to 2^{\mathbb{P}_0}$ is a *valuation function*. The transition function $\delta$ is extended to a function on $E^* \times S$ as follows: for all $w \in E^*$, $\delta(w, s) := s$ if $w = \epsilon$, and $\delta(w, s) := \delta(\sigma, \delta(v, s))$ if $w = v\sigma$. The *size* of a transition system $\mathcal{E}$, written $|\mathcal{E}|$, is the cardinality of its states (possibly infinite). A (deterministic) *automaton* or *plant* is a transition system without a valuation function $Lab$ but instead a set of *final states* $F$, which is a subset of the set of states $S$.

The following definitions hold not only for transition system but also for automata (plants). We denote by $\mathsf{Trace}(\mathcal{E}, s)^\omega \subseteq s(E \times S)^\omega$ the set of *infinite traces* $s_0\sigma_0 s_1 \ldots s_n\sigma_n s_{n+1} \ldots$ *of* $\mathcal{E}$ starting in $s_0 = s$ such that for all $i \in \mathbb{N}$, $\delta(\sigma_i, s_i) = s_{i+1}$, and similarly $\mathsf{Trace}(\mathcal{E}, s)^* \subseteq sS^*$ denotes the set of *finite traces* $s_0\sigma_0 s_1 \ldots s_n\sigma_n s_{n+1}$ *of* $\mathcal{E}$ starting in $s_0 = s$ such that for all $i \in \{0, \ldots, n\}$, $\delta(\sigma_i, s_i) = s_{i+1}$, for some $n \in \mathbb{N}$. We denote $\mathsf{Trace}(\mathcal{E})^\omega := \bigcup_{s \in S} \mathsf{Trace}(\mathcal{E}, s)^\omega$, $\mathsf{Trace}(\mathcal{E})^* := \bigcup_{s \in S} \mathsf{Trace}(\mathcal{E}, s)^*$ and $\mathsf{Trace}(\mathcal{E}) := \mathsf{Trace}(\mathcal{E})^\omega \cup \mathsf{Trace}(\mathcal{E})^*$. For a trace $\rho = s_0\sigma_0 s_1 \ldots s_n\sigma_n s_{n+1} \ldots \in \mathsf{Trace}(\mathcal{E})$ and $i, j \in \mathbb{N}$, $\rho(i) := s_i$ and $\rho(i, j) := s_i\sigma_i \ldots \sigma_{j-1}s_j \in \mathsf{Trace}(\mathcal{E})$. If $\rho \in \mathsf{Trace}(\mathcal{E})$, $|\rho| \in \mathbb{N} \cup \{\omega\}$ denotes the *length* of $\rho$. The *word associated to a trace* $\rho = s_0\sigma_0 s_1 \ldots s_n\sigma_n s_{n+1} \ldots$ is the unique word denoted $w(\rho)$ and defined by $w(\rho) := \sigma_0\sigma_1 \ldots \sigma_n\sigma_{n+1} \ldots$. Likewise, the *trace associated to a word* $w = \sigma_0\sigma_1 \ldots \sigma_n\sigma_{n+1} \ldots$ is the unique trace denoted $\rho(w)$ and defined by $\rho(w) := s_0\sigma_0 s_1 \ldots s_n\sigma_n s_{n+1} \ldots$ This trace is unique because we assume without loss of generality and, as is done in supervisory control theory, that $\delta$ is deterministic (*i.e.* it is a function). Dually, we define $\mathcal{L}(\mathcal{E})^\omega := \{w(\rho) : \rho \in \mathsf{Trace}(\mathcal{E})^\omega\} \subseteq E^\omega$ and $\mathcal{L}(\mathcal{E}) := \{w(\rho) : \rho \in \mathsf{Trace}(s_0)^*\} \subseteq E^*$. Finally, if $\mathcal{E}$ is an automaton (plant), then $\mathcal{L}_m(\mathcal{E}) := \{w \in \mathcal{L}(\mathcal{E}) : \delta(w, s_0) \in F\}$. These formal languages characterize the behavior of the plant/automaton/transition system. For each $i \in \mathbb{A}$, let $E_{o,i} \subseteq E$ be a set of *observable events for agent* $i$. The *observation function* $P_i : \mathcal{L}(\mathcal{E}) \cup \mathcal{L}(\mathcal{E})^\omega \to E_{o,i}^* \cup E_{o,i}^\omega$ is defined inductively as follows. First, for all $\sigma \in E$, $P_i(\sigma) := \sigma$ if $\sigma \in E_{o,i}$, and $P_i(\sigma) := \epsilon$ otherwise. Then, for all words $\sigma_0\sigma_1 \ldots \sigma_n\sigma_{n+1} \ldots \in \mathcal{L}(\mathcal{E}) \cup \mathcal{L}(\mathcal{E})^\omega$, $P_i(\sigma_0\sigma_1 \ldots \sigma_n\sigma_{n+1} \ldots) := P_i(\sigma_0)P_i(\sigma_1 \ldots \sigma_n\sigma_{n+1} \ldots)$. If there is a single agent $i$ (*i.e.* the set $\mathbb{A}$ is a singleton), then we omit the subscript $i$ in all the above notations.

**Notation 1.** Throughout this article, the automaton $A = (E, Q, \delta, q_0, F)$ denotes the *plant* and the automaton $O = (E, Q_o, \delta_o, q_{0,o}, F_o)$ denotes an objective or *goal automaton*. We define the *goal behaviors* by $\mathcal{K} = \mathcal{L}_m(O)$ and $\overline{\mathcal{K}} = \mathcal{L}(O)$.

Intuitively, the plant represents the (unrestricted) behavior of the system under consideration and the goal automaton represents the behavior that the supervisor should realize by controlling and restricting the behavior of the plant.

# 3. SUPERVISORY CONTROL THEORY

In this section, we recall the basics of supervisory control theory. We split our exposition in three parts: control with perfect information (Section 3.1), control with imperfect information (Section 3.2) and the special case of decentralized control with mutliple supervisors (Section 3.3). For more details on this theory, see [12, 4].

## 3.1 Control with Perfect Information

Throughout this article, $E_c$ is a subset of $E$. This set $E_c$ represents the *controllable* events, that is, the events that the supervisor can disable at any time. Dually, $E_{uc} = E - E_c$ represents the *uncontrollable* events, that is, the events over which the supervisor has no control, and therefore cannot disable. Formally, we represent the supervisor as a function $f$ that, given any string of events $w$, yields the set of events $f(w)$ that the supervisor will *not* disable after the occurrence of this sequence of events.

**Definition 1** (Supervisor)**.** A *supervisor* is a function $f : \mathcal{L}(A) \to \mathcal{P}(E)$ such that for all $w \in \mathcal{L}(A)$, it holds that $E_{uc} \subseteq f(w)$. ⊣

The supervisor controls the behavior of the plant. This means that after the occurrence of a sequence of events $w$, the supervisor will disable the occurrence of the events $E - f(w)$. The supervisor can only restrict the behavior of the plant, it cannot force the plant to generate events. The control action is allowed to change instantaneously after the execution of an observable event by the plant. This control restrains the strings of events that can possibly occur. It induces the controlled behavior of the plant, denoted $\mathcal{L}(A, f)$. A sequence of events $w\sigma$ belongs to the controlled behavior $\mathcal{L}(A, f)$ of the plant when $w$ already belongs to the controlled behavior ($w \in \mathcal{L}(A, f)$) and after the occurrence of $w$, the event $\sigma$ is not disabled by the supervisor ($\sigma \in f(w)$) and can indeed occur according to the original behavior of the plant ($w\sigma \in \mathcal{L}(A)$).

**Definition 2** (Supervised behavior of a plant)**.** The *behavior of* $A$ *supervised by* $f$, written $\mathcal{L}(A, f)$, is the formal language defined inductively as follows: for all $w \in E^*$, all $\sigma \in E$,

1. $\epsilon \in \mathcal{L}(A, f)$;
2. $w\sigma \in \mathcal{L}(A, f)$ if, and only if, $w \in \mathcal{L}(A, f)$, $\sigma \in f(w)$, and $w\sigma \in \mathcal{L}(A)$.

The *language recognized by* $(A, f)$ is defined by $\mathcal{L}_m(A, f) := \mathcal{L}(A, f) \cap \mathcal{L}_m(A)$. ⊣

One of the central problems in supervisory control theory is to decide whether or not there exists a supervisor $f$ that can control the behavior $\mathcal{L}(A)$ of the plant so as to realize the behavior specified by the *goal behavior* $\overline{\mathcal{K}} := \mathcal{L}(O)$.

If the supervisor could control any event, the answer to this problem would obviously be "yes". The difficulty comes from the fact that the supervisor cannot control all the events. A supervisor realizing the behavior $\mathcal{K}$ exists only if the occurrence of uncontrollable events will not lead to illegal states. More precisely, this requires that, at any time,

if the system is already in a state accepted by the specified behavior $\overline{\mathcal{K}}$, then the occurrence of any uncontrollable event $\sigma$ (permitted by the plant) will result in a state which is still accepted by the specified behavior $\overline{\mathcal{K}}$. This fact is formalized by the condition of *controllability*. Note the discrepancy between the intuitive formulation of this property and its formulation in formal terms.

**Definition 3** (Controllability)**.** $\mathcal{K}$ is *controllable w.r.t.* $\mathcal{L}(A)$ *and* $E_c$ when it holds that $\overline{\mathcal{K}}E_{uc} \cap \mathcal{L}(A) \subseteq \overline{\mathcal{K}}$. ⊣

**Theorem 1.** *[12, 4] There exists a supervisor* $f$ *such that* $\mathcal{L}(A, f) = \overline{\mathcal{K}}$ *if, and only if,* $\mathcal{K}$ *is controllable w.r.t.* $\mathcal{L}(A)$ *and* $E_c$*.*

But what if the plant cannot be controlled by a supervisor $f$ so that $\mathcal{L}(A, f) = \overline{\mathcal{K}}$? In other words, what if $\mathcal{K}$ is not controllable? In that case, we would still want the plant to be controlled by a supervisor such that the controlled behavior remains in the specified behavior $\overline{\mathcal{K}}$ and realizes as much as possible of $\overline{\mathcal{K}}$. This leads us to define the notions of *safe* and *maximally safe* supervisor.

**Definition 4** (Safe and maximally safe supervisor)**.** A supervisor $f$ is *safe for* $\mathcal{K}$ when $\mathcal{L}(A, f) \subseteq \overline{\mathcal{K}}$. A supervisor $f$ is *maximally safe* for $\mathcal{K}$ when for any supervisor $f'$ which is also safe for $\mathcal{K}$, it is not the case that $\mathcal{L}(A, f) \subset \mathcal{L}(A, f')$. ⊣

Now if $\mathcal{K}$ is not controllable, then there exists a maximal sublanguage of $\mathcal{K}$ (w.r.t. set inclusion) which is controllable, called the *supremal controllable sublanguage*. Moreover, the following theorem states that any maximally safe supervisor realizes this maximal sublanguage of $\mathcal{K}$.

**Definition 5** (Supremal controllable sublanguage)**.** The *supremal controllable sublanguage of* $\mathcal{K}$ is the maximal controllable sublanguage of $\mathcal{K}$ for set inclusion: $\mathcal{K}^{\uparrow C} := \bigcup\{\mathcal{J} \subseteq \mathcal{K} : \mathcal{J}$ is controllable w.r.t. $\mathcal{L}(A)$ and $E_c\}$. ⊣

Note that $\mathcal{K}^{\uparrow C} = \overline{\mathcal{K}^{\uparrow C}} \cap \mathcal{K}$ and if $\overline{\mathcal{K}} = \mathcal{K}$, then $\mathcal{K}^{\uparrow C} = \overline{\mathcal{K}^{\uparrow C}}$.

**Proposition 2.** *[12, 4] A supervisor* $f$ *is maximally safe for* $\overline{\mathcal{K}}$ *if, and only if,* $\mathcal{L}(A, f) = \overline{\mathcal{K}^{\uparrow C}}$*.*

The supervisor restrains the behavior $\mathcal{L}(A)$ of the plant. We do not want, however, the supervisor to *block* the plant, in the sense that it may be impossible at some point to terminate the execution of the task at hand and eventually reach a final state. This leads us to define the notion of *non-blocking* supervisor.

**Definition 6** (Non-blocking supervisor and $\mathcal{L}_m(A)$-closure)**.** A supervisor $f$ is *non-blocking* when $\overline{\mathcal{L}_m(A, f)} = \mathcal{L}(A, f)$. $\mathcal{K}$ is said to be $\mathcal{L}_m(A)$*-closed* when $\mathcal{K} = \overline{\mathcal{K}} \cap \mathcal{L}_m(A)$. ⊣

**Theorem 3.** *[12, 4] There exists a non-blocking supervisor* $f$ *such that* $\mathcal{L}_m(A, f) = \mathcal{K}$ *and* $\mathcal{L}(A, f) = \overline{\mathcal{K}}$ *if, and only if,* $\mathcal{K}$ *is* $\mathcal{L}_m(A)$*-closed and controllable w.r.t.* $\mathcal{L}(A)$ *and* $E_c$*.*

Here are the counterparts of Definition 5 and Proposition 2 in the context of non-blocking control.

**Definition 7** (Safe and maximally safe non-blocking supervisor)**.** A non-blocking supervisor $f$ is *safe* for $\mathcal{K}$ when $\mathcal{L}_m(A, f) \subseteq \mathcal{K}$. A non-blocking supervisor $f$ is *maximally safe* for $\mathcal{K}$ when for any non-blocking supervisor $f'$ which is also safe for $\mathcal{K}$, $\mathcal{L}_m(A, f) \subset \mathcal{L}_m(A, f')$ is not the case. ⊣

**Theorem 4.** *[12, 4] A non-blocking supervisor* $f$ *is maximally safe for* $\mathcal{K}$ *if, and only if,* $\mathcal{L}_m(A, f) = \mathcal{K}^{\uparrow C}$*.*

## 3.2 Control with Imperfect Information

In practical applications, the supervisor sometimes cannot observe the occurrence of some events. The set of *observable* events that the supervisor can observe is a subset of $E$, denoted $E_o$. The set $E_{uo} = E - E_o$ denotes the set of *unobservable* events.

Given that the supervisor can only observe some of the events, its decision to disable the occurrence of an event depends only on what it has observed. Moreover, we assume that its control action is immediate and is executed right after the occurrence of the last observable event and until the occurrence of the next observable event. This leads us to define the notion of $P$*-supervisor*.

**Definition 8** ($P$-supervisor)**.** A $P$*-supervisor* is a supervisor $f : \mathcal{L}(A) \to \mathcal{P}(E)$ such that for all $w, v \in \mathcal{L}(A)$ such that $P(w) = P(v)$, it holds that $f(w) = f(v)$. Moreover, $f$ is said to be *non-blocking* when $\overline{\mathcal{L}_m(A, f)} = \mathcal{L}(A, f)$. ⊣

Under the assumption of partial observability, we need an extra condition to ensure the existence of a $P$-supervisor controlling the plant. This condition is called *observability*.

**Definition 9** (Observability)**.** $\mathcal{K}$ is *observable w.r.t.* $\mathcal{L}(A), E_o$ *and* $E_c$ when for all $w \in \overline{\mathcal{K}}$, all $\sigma \in E_c$, if $w\sigma \notin \overline{\mathcal{K}}$ and $w\sigma \in \mathcal{L}(A)$ then $P^{-1}[P(w)]\sigma \cap \overline{\mathcal{K}} = \emptyset$. ⊣

**Theorem 5.** *[12, 4]There exists a non-blocking* $P$*-supervisor* $f$ *such that* $\mathcal{L}(A, f) = \overline{\mathcal{K}}$ *(and* $\mathcal{L}_m(A, f) = \mathcal{K}$*) if, and only if,* $\mathcal{K}$ *is controllable, observable (and, respectively,* $\mathcal{L}_m(A)$*-closed) w.r.t.* $\mathcal{L}(A)$*,* $E_c$ *and* $E_o$*.*

## 3.3 Decentralized Control

Now we assume that we have multiple supervisors that control and observe only some of the events. For each supervisor $i \in \mathbb{A}$, we denote by $E_{c,i} \subseteq E$ the set of events *controllable* by supervisor $i$, and we denote by $E_{o,i} \subseteq E$ the set of events *observable* by supervisor $i$. Moreover, we assume that $E_c := \bigcup_{i \in \mathbb{A}} E_{c,i}$ and that $E_o := \bigcup_{i \in \mathbb{A}} E_{o,i}$. Also, for all $i \in \mathbb{A}$, we denote by $E_{uc,i} := E - E_{c,i}$ the set of events *uncontrollable* by supervisor $i$, and we denote by $E_{uo,i} := E - E_{o,i}$ the set of events *unobservable* by supervisor $i$. If $\sigma \in \mathbb{A}_c$, we denote by $\mathbb{A}_c(\sigma)$ the subset of supervisors that controls $\sigma$: $\mathbb{A}_c(\sigma) := \{i \in \mathbb{A} : \sigma \in E_{c,i}\}$.

At each tick of the clock in the system, each supervisor observes an event (or not) and then sends a recommendation to a central supervisor who then decides which event to disable following a specific fusion rule (conjunctive, disjunctive,...) that takes into account the recommendation of each supervisor.

**Definition 10** (Decentralized supervisor)**.** A *conjunctive (disjunctive) decentralized supervisor* is a $P$-supervisor such that there exists a family of $P_i$-supervisors $\mathcal{F} = \{f_i : i \in \mathbb{A}\}$ such that for all $w \in \mathcal{L}(A)$, $f(w) := \bigcap_{i \in \mathbb{A}} f_i(w)$ (resp. $f(w) := \bigcup_{i \in \mathbb{A}} f_i(w)$). $\mathcal{F}$ is called the *family of* $P_i$*-supervisors associated to* $f$*.* ⊣

**Definition 11** (Conjunctive and disjunctive coobservability)**.** $\mathcal{K}$ is said to be *CP-coobservable w.r.t.* $\mathcal{L}(A), E_{o,i}$ *and* $E_{c,i}$ when for all $w \in \overline{\mathcal{K}}$, all $\sigma \in E_c$, if $w\sigma \notin \overline{\mathcal{K}}$ and $w\sigma \in \mathcal{L}(A)$ then there is $i \in \{1, \ldots, n\}$ such that $P_i^{-1}[P_i(w)]\sigma \cap \overline{\mathcal{K}} = \emptyset$ and $\sigma \in E_{c,i}$.

$\mathcal{K}$ is said to be *DA-coobservable w.r.t.* $\mathcal{L}(A), E_{o,i}$ *and* $E_{c,i}$ when for all $w \in \overline{\mathcal{K}}$, all $\sigma \in E_c$, if $w\sigma \in \overline{\mathcal{K}}$ then there is $i \in \{1,\dots,n\}$ such that $\left(P_i^{-1}[P_i(w)] \cap \overline{\mathcal{K}}\right)\sigma \cap \mathcal{L}(A) \subseteq \overline{\mathcal{K}}$ and $\sigma \in E_{c,i}$. ⊣

**Theorem 6.** *[12, 4] There exists a conjunctive (disjunctive) decentralized supervisor $f$ such that $\mathcal{L}(A,f) = \overline{\mathcal{K}}$ (and $\mathcal{L}_m(A,f) = \mathcal{K}$) if, and only if, $\mathcal{K}$ is controllable (and, respectively, $\mathcal{L}_m(A)$-closed) w.r.t. $\mathcal{L}(A)$ and $E_{uc}$ and CP-coobservable (DA-coobservable) w.r.t. $\mathcal{L}(A)$, $E_{o,i}$ and $E_{c,i}$.*

# 4. EPISTEMIC TEMPORAL LOGIC

In this Section, we introduce our epistemic temporal logic $\mathsf{CTL}^*\mathsf{K}_n^{\mathsf{D}}$. It has an asynchronous semantics with perfect recall and branching time. It is one of the 96 epistemic temporal logics introduced and studied in depth by Halpern & al. [8, 6, 15].

## 4.1 Syntax and Semantics

To verify that a given property (such as the absence of a deadlock) holds in a software or hardware system, we model check the corresponding formula of temporal logic in the transition system modeling the behavior of this system [3]. The main difference with software verification is that, in our case, what we need to take into account is not only the system under consideration, but also its goal behavior. Hence, our transition system that we call an *environment* is a combination of the plant $A$ and the objective $O$.

**Definition 12** (Environment)**.** The *environment associated to $A$ and $O$* is the transition system $\mathcal{E} := (E, S, \delta, s_0, Lab)$ defined as follows:

- $S := (Q \sqcup \{q_{dead}\}) \times (Q_o \sqcup \{q_{dead,o}\})$;
- $\delta : E \times S \to S$ is the transition function defined as follows: for all $\sigma \in E$, $\delta(\sigma,(q,q_o))$ is defined by
  - $(\delta(\sigma,q), \delta_o(\sigma,q_o))$ if $\delta(\sigma,q)$ and $\delta_o(\sigma,q_o)$ are defined;
  - $(q_{dead}, \delta_o(\sigma,q_o))$ if only $\delta_o(\sigma,q_o)$ is defined;
  - $(\delta(\sigma,q), q_{dead,o})$ if only $\delta(\sigma,q)$ is defined;
  - $(q_{dead}, q_{dead,o})$ otherwise;
- $s_0 := (q_0, q_{0,o})$;
- $Lab : S \to 2^{\mathbb{P}_0}$ is the valuation function defined as follows: $p_{\mathcal{L}_m} \in Lab(q,q_o)$ iff $q \in F$, $p_{\mathcal{K}} \in Lab(q,q_o)$ iff $q_o \in F_o$.

We abuse notation and write $s \in \mathcal{E}$ for $s \in S$. ⊣

**Definition 13** (Interpreted system and perfect recall)**.** The *interpreted system associated to $A$ and $O$* is the interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ defined as follows:

- $\mathcal{R}$ is a *system*, *i.e.* a set of *runs*, where each run $r \in \mathcal{R}$ is a function $r : \mathbb{N} \to E^{n+1}$ associated with an infinite word $w \in \mathcal{L}(\mathcal{E})^\omega$. The run associated with an infinite word $w = \sigma_0\sigma_1\dots\sigma_n\sigma_{n+1}\dots \in \mathcal{L}(\mathcal{E})^\omega$ is defined inductively as follows: for all $m \geq 0$,
  - $r(0) := (nil,\dots,nil,nil)$ where $nil$ is a new symbol;
  - $r(m+1) := \left(\sigma^1_{m+1},\dots,\sigma^n_{m+1},\sigma_m\right)$, where for all $i \in \{1,\dots,n\}$,
    $$\sigma^i_{m+1} := \begin{cases} \sigma_m & \text{if } \sigma_m \in E_{o,i} \\ \sigma^i_m & \text{otherwise} \end{cases}$$
    We assume that agents have *perfect recall* in system $\mathcal{R}$.
- $\pi : \mathcal{R} \times \mathbb{N} \to 2^{\mathbb{P}}$ is a valuation function defined as follows: for all $r \in \mathcal{R}$ and all $m \in \mathbb{N}$, $\pi(r,m) := Lab(\delta(\sigma_0\dots\sigma_m, s_0)) \cup \{p_{\sigma_m}\}$.

The *run associated with the word $w$* is denoted $r(w)$. If $w$ is a *finite* word, then we also define with the same induction the run associated with the word $w$, also denoted $r(w)$, as a function $r(w) : \{1,\dots,|w|\} \to E^{n+1}$. Reciprocally, the *word associated with a run $r$*, denoted $w(r)$, is the word (possibly infinite) defined by $w(r) := r_{n+1}(1)r_{n+1}(2)\dots r_{n+1}(m)r_{n+1}(m+1)\dots$ We say that two points $(r,m)$ and $(r',m')$ are *indistinguishable* to agent $i$, written $(r,m) \sim_i (r',m')$, when $r_i(m) = r'_i(m')$. Finally, $r_i(0,m)$ denotes the sequence of local states of agent $i$ in the run $r$ from time 0 to $m$.

*Agent $i$'s local-state sequence at the point* $(r,m)$ is the sequence $l_0,\dots,l_k$ of local states that agent $i$ takes on in run $r$ up to and including time $m$, with consecutive repetitions omitted. We say that *agent $i$ has perfect recall* (alternatively, *agent $i$ does not forget*) in system $\mathcal{R}$ if at all points $(r,m)$ and $(r',m')$ in $\mathcal{R}$, if $(r,m) \sim_i (r',m')$, then $r$ has the same local-state sequence at both $(r,m)$ and $(r',m')$. ⊣

**Notation 2.** In the rest of this article, $\mathcal{I}$ is the interpreted system associated to $A$ and $O$, and $\mathcal{E}$ is the environment associated to $A$ and $O$.

One can easily show that agent $i$ has perfect recall if, and only if, for all points $(r,m)$ and $(r',m')$ of $\mathcal{R}$, it holds that $(r,m) \sim_i (r',m')$ if, and only if, $P_i(w(0,m)) = P_i(w'(0,m'))$, where $r, r'$ are the runs associated to some words $w, w'$ respectively.

**Definition 14** (Language $\mathcal{L}_{\mathsf{CTL}^*\mathsf{K}_n^{\mathsf{D}}}$)**.** The syntax of the epistemic temporal language $\mathcal{L}_{\mathsf{CTL}^*\mathsf{K}_n^{\mathsf{D}}}$ is defined inductively as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\mathcal{U}\psi \mid \exists\varphi \mid K_i\varphi \mid D_I\varphi$$

where $p$ ranges over $\mathbb{P}$, $i$ over $\mathbb{A}$ and $I$ over subsets of $\mathbb{A}$. ⊣

**Definition 15** (Truth conditions)**.** Let $\mathcal{I} = (\mathcal{R}, \pi)$ be the interpreted system associated to $A$ and $O$ where agents have perfect recall. Let $r \in \mathcal{R}$, let $m \in \mathbb{N}$ and let $\varphi \in \mathcal{L}_{\mathsf{CTL}^*\mathsf{K}_n^{\mathsf{D}}}$. The *satisfaction relation* $\mathcal{I}, r, m \models \varphi$ is defined inductively as follows:

| | | |
|---|---|---|
| $\mathcal{I}, r, m \models p$ | iff | $p \in \pi(r(m))$ |
| $\mathcal{I}, r, m \models \neg\varphi$ | iff | not $\mathcal{I}, r, m \models \varphi$ |
| $\mathcal{I}, r, m \models \varphi \wedge \psi$ | iff | $\mathcal{I}, r, m \models \varphi$ and $\mathcal{I}, r, m \models \psi$ |
| $\mathcal{I}, r, m \models \bigcirc\varphi$ | iff | $\mathcal{I}, r, m+1 \models \varphi$ |
| $\mathcal{I}, r, m \models \varphi\mathcal{U}\psi$ | iff | there exists $m' \geq m$ such that $\mathcal{I}, r, m' \models \psi$ and for all $m''$ with $m \leq m'' < m'$, we have $\mathcal{I}, r, m'' \models \varphi$ |
| $\mathcal{I}, r, m \models \exists\varphi$ | iff | there exists $r' \in \mathcal{R}$ such that $r'(0,m) = r(0,m)$ and $\mathcal{I}, r', m \models \varphi$ |
| $\mathcal{I}, r, m \models K_i\varphi$ | iff | $\mathcal{I}, r', m' \models \varphi$ for all $r' \in \mathcal{R}$ and $m'$ such that $(r,m) \sim_i (r',m')$ |
| $\mathcal{I}, r, m \models D_I\varphi$ | iff | $\mathcal{I}, r', m' \models \varphi$ for all $r' \in \mathcal{R}$ and $m'$ such that for all $i \in I$, $(r,m) \sim_i (r',m')$ |

Also, we introduce the following notation: If $w \in \mathcal{L}(\mathcal{E})^\omega$ is an infinite word, then we write $\mathcal{I}, w \models \varphi$ iff $\mathcal{I}, r(w), 0 \models \varphi$. If $w \in \mathcal{L}(\mathcal{E})$ is a finite word and $m \leq |w|$, then we write

$$\mathcal{I}, w, m \models \varphi \quad \text{iff} \quad \mathcal{I}, r, m \models \varphi \text{ for all runs } r \in \mathcal{R} \text{ extending } r(w).$$

and we write $\mathcal{I}, w \models \varphi$ iff $\mathcal{I}, w, |w| \models \varphi$. If $\rho \in \mathsf{Trace}(\mathcal{E})$ is a trace, then we write $\mathcal{I}, \rho \models \varphi$ iff $\mathcal{I}, w(\rho) \models \varphi$. If $s \in S$ is a state, then we write:

$$\mathcal{I}, s \models \varphi \quad \text{iff} \quad \mathcal{I}, \rho \models \varphi \text{ for all } \rho \in \mathsf{Trace}(\mathcal{E}, s)^\omega.$$

Finally, we say that $\varphi$ is *true in* $\mathcal{I}$, written $\mathcal{I} \models \varphi$, when for all $r \in \mathcal{R}$ and all $m$, it holds that $\mathcal{I}, r, m \models \varphi$. Also, we say that $\varphi$ is *valid*, written $\models \varphi$, when $\mathcal{I} \models \varphi$ for all interpreted systems $\mathcal{I}$. ⊣

**Proposition 7.** *Let $w \in \mathcal{L}(\mathcal{E})$. Then, the following holds:*

$$w \in \mathcal{L}(A) \quad \textit{iff} \quad \mathcal{I}, w \models \exists\Diamond p_{\mathcal{L}_m}$$
$$w \in \overline{\mathcal{K}} \quad \textit{iff} \quad \mathcal{I}, w \models \exists\Diamond p_{\mathcal{K}}$$

**Notation 3.** From now on, we use the following notation: $\mathsf{safe} := \exists\Diamond p_{\mathcal{K}}$, $\mathsf{safe}_{p_{\mathcal{L}_m}} := \exists\Diamond p_{\mathcal{L}_m}$, $\Diamond\varphi := \top\mathcal{U}\varphi$, $[\sigma]\varphi := \bigcirc\left(\left(p_\sigma \wedge \mathsf{safe}_{p_{\mathcal{L}_m}}\right) \rightarrow \varphi\right)$, $\langle\sigma\rangle\varphi := \neg[\sigma]\neg\varphi$, $\varphi\,[\mathcal{U}]\,\psi := \neg(\varphi\mathcal{U}\neg\psi)$, $[\mathsf{uc}]\mathsf{safe} := (\mathsf{safe}_{p_{\mathcal{L}_m}} \wedge \bigvee_{\sigma\in E_{uc}} p_\sigma)\,[\mathcal{U}]\,((\mathsf{safe}_{p_{\mathcal{L}_m}} \wedge \bigvee_{\sigma\in E_{uc}} p_\sigma) \rightarrow \mathsf{safe})$. The intuitive readings of the formulas and notations introduced are as follows. $\mathsf{safe}$: 'the system is in a safe state'; $\mathsf{safe}_{p_{\mathcal{L}_m}}$: 'the current state is allowed by the plant'; $\Diamond\varphi$: 'eventually in the future, $\varphi$ holds'; $\exists\varphi$: 'there exists an extension of the current run for which $\varphi$ holds'; $K_i\varphi$: 'agent/supervisor $i$ knows that $\varphi$ holds'; $D_I\varphi$: 'it is distributed knowledge among the set of agents/supervisors $I$ that $\varphi$ holds', which means that if all the agents/supervisors of $I$ pooled their knowledge, then they would know that $\varphi$ holds; $[\sigma]\varphi$: 'if the execution of event $\sigma$ is possible in the plant, then after its execution $\varphi$ holds'; $\langle\sigma\rangle\varphi$: 'it is possible to execute event $\sigma$ in the plant and after its execution, $\varphi$ holds'; $\varphi\,[\mathcal{U}]\,\psi$: 'while $\varphi$ holds, $\psi$ also holds'; $[\mathsf{uc}]\mathsf{safe}$: 'the current state is safe and the occurrence of any finite sequence of uncontrollable events in this state still leads to a safe state'. The modalities $[\sigma]\varphi$ and $\langle\sigma\rangle\varphi$ correspond to the standard modalities of Propositional Dynamic Logic [9].

## 4.2 Model Checking

The *model checking problem* consists of answering the question: given a *finite* transition system $\mathcal{I}$, a state $s \in \mathcal{E}$ and $\varphi \in \mathcal{L}_{\mathsf{CTL}^*\mathsf{K}_n^\mathsf{D}}$, is it the case that $\mathcal{I}, s \models \varphi$?

The *alternation depth* of a formula $\varphi \in \mathcal{L}_{\mathsf{CTL}^*\mathsf{K}_n^\mathsf{D}}$, written $ad(\varphi)$, is the number of alternations of distinct $K_i$'s in $\varphi$; temporal and branching operators do not count. Theorem 8 below is also an instance of a more general result of [11].

**Theorem 8.** *The model checking problem is decidable. If $ad(\varphi) = 0$ then it is in PSPACE, and if $ad(\varphi) > 0$ then it is in $ad(\varphi)$-EXPTIME.*

*Proof sketch.* We adapt the classic powerset construction of [13] to our setting. Then, we incrementally reduce the degree of the formula $\varphi$ by eliminating the knowledge operators step by step. Each elimination reduces the alternating depth of $\varphi$ by one and increases the size of the environment exponentially. As in [11], our model checking algorithm is adapted from the model checking algorithm of CTL [3]. □

# 5. CONTROL WITH FULL REALIZATION

In this section, we will reformulate in $\mathsf{CTL}^*\mathsf{K}_n^\mathsf{D}$ the different conditions for fully realizing a goal behavior under different assumptions: partial controllability (Section 5.1), partial observation (Section 5.2) and decentralized supervision (Section 5.3). Moreover, we will show that supervisors can be represented in terms of model checking problems. More precisely, we will show that for any ($P$-)supervisor $f$, any $w \in \mathcal{L}(A)$ and any $\sigma \in E_c$, there is a formula $F(\sigma)$ of $\mathcal{L}_{\mathsf{CTL}^*\mathsf{K}_n^\mathsf{D}}$ such that $\sigma \in f(w)$ if, and only if, $\mathcal{I}, w \models F(\sigma)$.

## 5.1 Control with Perfect Information

We first reformulate in our epistemic temporal logic $\mathsf{CTL}^*\mathsf{K}_n^\mathsf{D}$ the condition of controllability. Proposition 9 states that the system is controllable if, and only if, at any time, if the system is in a safe state then the system remains in a safe state after the occurrence of any finite sequence of uncontrollable events.

**Proposition 9.** *The language $\mathcal{K}$ is controllable w.r.t. $\mathcal{L}(A)$ and $E_c$ if, and only if,*

$$\mathcal{I} \models \mathsf{safe} \rightarrow [\mathsf{uc}]\mathsf{safe}. \tag{1}$$

The following theorem shows how the supervisor realizing a goal behavior can be represented in terms of model checking problems.

**Theorem 10** (Synthesis of supervisor)**.** *Let $f : \mathcal{L}(A) \rightarrow \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f(w) = E_{uc} \cup \{\sigma \in E_c : \mathcal{I}, w \models [\sigma]\mathsf{safe}\}. \tag{2}$$

*If $\mathcal{K}$ is controllable w.r.t. $\mathcal{L}(A)$ and $E_c$, then $f$ is a supervisor and $\mathcal{L}(A, f) = \overline{\mathcal{K}}$.*

Proposition 11 and Theorem 12 below provide results similar to Proposition 9 and Theorem 10 in the context of *non-blocking* control.

**Proposition 11.** *The language $\mathcal{K}$ is $\mathcal{L}_m(A)$-closed if, and only if, $\mathcal{I} \models \mathsf{safe} \rightarrow (p_{\mathcal{K}} \leftrightarrow p_{\mathcal{L}_m})$.*

In other words, Proposition 11 states that a language is $\mathcal{L}_m(A)$-closed if this language is the restriction of its prefix closure to $\mathcal{L}_m(A)$. So, this reformulation is close to the initial formulation of $\mathcal{L}_m(A)$-closure in Theorem 3. Here is the counterpart of this theorem:

**Theorem 12** (Synthesis of non-blocking supervisor)**.** *Let $f : \mathcal{L}(A) \rightarrow \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f(w) = E_{uc} \cup \{\sigma \in E_c : \mathcal{I}, w \models [\sigma]\mathsf{safe}\}. \tag{3}$$

*If $\mathcal{K}$ is controllable and $\mathcal{L}_m(A)$-closed w.r.t. $\mathcal{L}(A)$ and $E_c$, then $f$ is a non-blocking supervisor, $\mathcal{L}_m(A, f) = \mathcal{K}$ and $\mathcal{L}(A, f) = \overline{\mathcal{K}}$.*

## 5.2 Control with Imperfect Information

The condition of observability is expressed in our epistemic temporal logic by the Expression (5). It reads as "if the system is in a safe state and the occurrence of the event $\sigma$ will lead to an unsafe state, then the supervisor knows it".

**Proposition 13.** *The language $\mathcal{K}$ is observable w.r.t. $\mathcal{L}(A)$, $E_o$ and $E_c$ if, and only if, for all $\sigma \in E_c$,*

$$\mathcal{I} \models \langle\sigma\rangle\mathsf{safe} \rightarrow K(\mathsf{safe} \rightarrow [\sigma]\mathsf{safe}) \tag{4}$$
$$\textit{iff} \quad \mathcal{I} \models (\mathsf{safe} \wedge \langle\sigma\rangle\neg\mathsf{safe}) \rightarrow K[\sigma]\neg\mathsf{safe} \tag{5}$$

Now, we show how $P$-supervisors can be represented in our logical language.

**Theorem 14** (Synthesis of $P$-supervisor)**.** *Let $f : \mathcal{L}(A) \to \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f(w) = E_{uc} \cup \{\sigma \in E_c : \mathcal{I}, w \models K(\mathsf{safe} \to [\sigma]\mathsf{safe})\}. \quad (6)$$

*If $\mathcal{K}$ is controllable, observable and $\mathcal{L}_m(A)$-closed w.r.t. $\mathcal{L}(A)$, $E_o$ and $E_c$, then $f$ is a non-blocking $P$-supervisor such that $\mathcal{L}_m(A, f) = \mathcal{K}$ and $\mathcal{L}(A, f) = \overline{\mathcal{K}}$.*

In other words, Expression (6) states that the supervisor will not disable an event $\sigma$ when it *knows* that, if the system is already in a safe state, the occurrence of this event $\sigma$ will lead to another safe state.

## 5.3 Decentralized Control

In this section, we assume that there are multiple supervisors and that each of them has a partial observation and a partial control over the events that occur in the system. Thus, we consider a specific kind of multi-agent system. We are going to reformulate the conditions of coobservability of the goal behavior in terms of model checking problems in our epistemic temporal logic. We recall that they provide necessary and sufficient conditions for the existence of a family of supervisors that fully realizes a given goal behavior.

### 5.3.1 Existence and Synthesis of Decentralized Supervisors

Expression (7) reads as "if the system is in a safe state and the occurrence of the event $\sigma$ will lead us to an unsafe state, then at least one supervisor knows it".

**Proposition 15.** *The language $\mathcal{K}$ is CP-coobservable with respect to $\mathcal{L}(A)$, $E_{o,i}$ and $E_{c,i}$ if, and only if, for all $\sigma \in E_c$,*

$$\mathcal{I} \models (\mathsf{safe} \wedge \langle\sigma\rangle\neg\mathsf{safe}) \to \bigvee_{i \in \mathbb{A}_c(\sigma)} K_i[\sigma]\neg\mathsf{safe} \quad (7)$$

$$\mathcal{I} \models \left( \bigwedge_{i \in \mathbb{A}_c(\sigma)} \langle K_i\rangle\langle\sigma\rangle\mathsf{safe} \right) \to D_{\mathbb{A}_c(\sigma)}(\mathsf{safe} \to [\sigma]\mathsf{safe}). \quad (8)$$

Expression (9) reads as "if the occurrence of the event $\sigma$ will lead to a safe state, then at least one supervisor knows that the occurrence of $\sigma$ will keep the system in a safe state".

**Proposition 16.** *The language $\mathcal{K}$ is DA-coobservable with respect to $\mathcal{L}(A)$, $E_{o,i}$ and $E_{c,i}$ if, and only if, for all $\sigma \in E_c$,*

$$\mathcal{I} \models \langle\sigma\rangle\mathsf{safe} \to \bigvee_{i \in \mathbb{A}_c(\sigma)} K_i(\mathsf{safe} \to [\sigma]\mathsf{safe}). \quad (9)$$

Somehow, the intuitive reading of Expression (7) is coherent and meaningful. In a conjunctive architecture, an event is disabled when at least one supervisor recommends to disable it. A supervisor recommends to disable an event when it *knows* that this event will lead to an unsafe state. If we assume that the condition of CP-coobservability holds, that is, if we assume that at least one of the supervisors knows when an event will lead to an unsafe state, then, intuitively, we guarantee this way that the family of supervisors will be able to fully realize the goal behavior, in a conjunctive architecture. Likewise for Expression (9). In a disjunctive architecture, an event is enabled when at least one supervisor recommends enablement. A supervisor recommends the enablement of an event when it *knows* that this event will keep the system in a safe state. If we assume that the condition of DA-coobservability holds, that is, if we assume that at least one of the supervisors knows when an event will lead to a safe state, then, intuitively, we guarantee this way that the family of supervisors will be able to fully realize the goal behavior, in a disjunctive architecture. This explains informally why Theorem 6 holds.

**Theorem 17** (Synthesis of conjunctive supervisor)**.** *For all $i \in \mathbb{A}$, let $f_i : \mathcal{L}(A) \to \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f_i(w) = E_{uc,i} \cup \{\sigma \in E_{c,i} : \mathcal{I}, w \models \langle K_i\rangle\langle\sigma\rangle\mathsf{safe}\}. \quad (10)$$

*If $\mathcal{K}$ is controllable, $\mathcal{L}_m(A)$-closed and CP-coobservable w.r.t. $\mathcal{L}(A), E_{o,i}$ and $E_{c,i}$, then for all $i \in \mathbb{A}$, $f_i$ is a $P_i$-supervisor and the conjunctive decentralized supervisor $f$ associated to $\{f_i : i \in \mathbb{A}\}$ is non-blocking and satisfies $\mathcal{L}_m(A, f) = \mathcal{K}$ and $\mathcal{L}(A, f) = \overline{\mathcal{K}}$.*

*Moreover, the conjunctive decentralized supervisor $f$ also satisfies the following condition: for all $w \in \mathcal{L}(A, f)$, all $\sigma \in E_c$ such that $w\sigma \in \mathcal{L}(A)$,*

$$\sigma \in f(w) \quad \textit{iff} \quad \mathcal{I}, w \models D_{\mathbb{A}_c(\sigma)}(\mathsf{safe} \to [\sigma]\mathsf{safe}). \quad (11)$$

An event $\sigma$ is disabled by the family of supervisors $\{f_i : i \in \mathbb{A}\}$ defined by Expression (10) (after the sequence of events $w$) if, and only if, at least one of the supervisors *knows* that $\sigma$ will lead to an unsafe state: $\mathcal{I}, w \models K_i[\sigma]\neg\mathsf{safe}$. Expression (11) explains that an event $\sigma$ is enabled (after a sequence of events $w$) if, and only if, it is *distributed knowledge* among the supervisors that control $\sigma$ that this event $\sigma$ will keep the system in a safe state: $\mathcal{I}, w \models D_{\mathbb{A}_c(\sigma)}(\mathsf{safe} \to [\sigma]\mathsf{safe})$.

**Theorem 18** (Synthesis of disjunctive supervisor)**.** *For all $i \in \mathbb{A}$, let $f_i : \mathcal{L}(A) \to \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f_i(w) = E_{uc} \cup \{\sigma \in E_{c,i} : \mathcal{I}, w \models K_i(\mathsf{safe} \to [\sigma]\mathsf{safe})\} \quad (12)$$

*If $\mathcal{K}$ is controllable, $\mathcal{L}_m(A)$-closed and CP-coobservable w.r.t. $\mathcal{L}(A), E_{o,i}$ and $E_{c,i}$, then for all $i \in \mathbb{A}$, $f_i$ is a $P_i$-supervisor and the disjunctive supervisor $f$ associated to $\{f_i : i \in \mathbb{A}\}$ is non-blocking and satisfies $\mathcal{L}_m(A, f) = \mathcal{K}$ and $\mathcal{L}(A, f) = \overline{\mathcal{K}}$.*

*Moreover, the disjunctive decentralized supervisor $f$ also satisfies the following condition: for all $w \in \mathcal{L}(A, f)$, all $\sigma \in E_c$ such that $w\sigma \in \mathcal{L}(A)$,*

$$\sigma \in f(w) \quad \textit{iff} \quad \mathcal{I}, w \models D_{\mathbb{A}_c(\sigma)}(\mathsf{safe} \to [\sigma]\mathsf{safe}). \quad (13)$$

An event $\sigma$ is enabled by the family of supervisors $\{f_i : i \in \mathbb{A}\}$ defined above (after the sequence of events $w$) if, and only if, at least one of the supervisors *knows* that $\sigma$ will keep the system in a safe state: $\mathcal{I}, w \models K_i(\mathsf{safe} \to [\sigma]\mathsf{safe})$. Expression (13) explains that an event $\sigma$ is enabled (after a sequence of events $w$) if, and only if, it is *distributed knowledge* among the supervisors that control $\sigma$ that this event $\sigma$ will keep the system in a safe state: $\mathcal{I}, w \models D_{\mathbb{A}_c(\sigma)}(\mathsf{safe} \to [\sigma]\mathsf{safe})$.

### 5.3.2 Coobservability and Observability: their Relationship

In this section, we study the relationship between the notions of coobservability and observability. One can show that when there is a single supervisor, then these notions coincide. Moreover, the following proposition shows that the notion of coobservability is a more demanding notion than the notion of observability, and refines it.

**Proposition 19.** *If $\mathcal{K}$ is CP-coobservable (or DA-coobservable) with respect to $\mathcal{L}(A)$, $E_{o,i}$ and $E_{c,i}$, then $\mathcal{K}$ is observable w.r.t. $\mathcal{L}(A)$, $E_o$ and $E_c$.*

If the goal behavior is only observable and not coobservable, we can nevertheless find a $P$-supervisor that will fully realize the goal behavior. Note that it is the same as the $P$-supervisor defined in Expressions (11) and (13).

**Theorem 20.** *Let $f : \mathcal{L}(A) \to \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f(w) = E_{uc} \cup \{\sigma \in E_c : \mathcal{I}, w \models D_{\mathbb{A}}(\mathsf{safe} \to [\sigma]\mathsf{safe})\}.$$

*If $\mathcal{K}$ is controllable, $\mathcal{L}_m(A)$-closed and observable w.r.t. $\mathcal{L}(A)$, $E_o$ and $E_c$, then $f$ is a non-blocking $P$-supervisor, $\mathcal{L}_m(A, f) = \mathcal{K}$ and $\mathcal{L}(A, f) = \overline{\mathcal{K}}$.*

# 6. CONTROL WITH PARTIAL REALIZATION

So far we have investigated conditions for fully realizing a goal behavior. These conditions are called observability, controllability and coobservability. However, they are not very often met in actual applications. The problem then becomes to realize as much as possible of the goal behavior. In that case, we want the plant to be controlled by a supervisor $f$ so that $\mathcal{L}(A, f) \subseteq \overline{\mathcal{K}}$ instead of the more demanding condition $\mathcal{L}(A, f) = \overline{\mathcal{K}}$. The problem to find such a supervisor is called the "Basic Supervisory Control Problem" in control theory [4, p. 156].

**Definition 16** (Safe and maximally safe supervisors)**.** A supervisor $f$ is *safe* for $\overline{\mathcal{K}}$ when $\mathcal{L}(A, f) \subseteq \overline{\mathcal{K}}$. A supervisor $f$ is *maximally safe* for $\overline{\mathcal{K}}$ when for any supervisor $f'$ which is also safe for $\overline{\mathcal{K}}$, $\mathcal{L}(A, f') \subseteq \mathcal{L}(A, f)$. Safe and maximally safe *P-supervisors* are defined similarly. ⊣

In this section, we address the basic supervisory control problem under different assumptions.

## 6.1 Control with Perfect Information

The definition of a maximally safe supervisor in Expression (14) below is a generalization of the definition of the safe supervisor of Expression (2), in the sense that after the occurrence of $\sigma$, the system should not only be in a safe state but also remain in a safe state after the occurrence of any finite sequence of uncontrollable events.

**Theorem 21** (Synthesis of maximally safe supervisor)**.** *Assume that $\mathcal{I}, s_0 \models [\mathsf{uc}]\mathsf{safe}$. Let $f : \mathcal{L}(A) \to \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f(w) = E_{uc} \cup \{\sigma \in E_c : \mathcal{I}, w \models [\sigma][\mathsf{uc}]\mathsf{safe}\}. \qquad (14)$$

*Then, $f$ is a supervisor maximally safe for $\overline{\mathcal{K}}$. Moreover, if $\mathcal{K} \subseteq \mathcal{L}_m(A)$, then $f$ is a* non-blocking *maximally safe P-supervisor. Finally, $f$ satisfies the following condition:*

$$\mathcal{L}(A, f) = \{w \in \mathcal{L}(A) : \mathcal{I}, \overline{w} \models [\mathsf{uc}]\mathsf{safe} \text{ for all } \overline{w} \leq w\} = \overline{\mathcal{K}^{\uparrow C}}.$$

**Theorem 22** (Existence of maximally safe supervisor)**.** *Assume that $E_c \subseteq E_o$. Then, the following statements are equivalent:*

1. *$\mathcal{I}, s_0 \models [\mathsf{uc}]\mathsf{safe}$;*
2. *there exists a supervisor $f$ safe for $\overline{\mathcal{K}}$;*
3. *there exists a supervisor $f$* maximally *safe for $\overline{\mathcal{K}}$ (given by Expression (14)).*

*Moreover, if we assume that $\mathcal{K} \subseteq \mathcal{L}_m(A)$, then supervisors are also* non-blocking.

## 6.2 Control with Imperfect Information

As is often the case in supervisory control theory (and also in game theory), we will assume in this section and in the rest of the article that $E_c \subseteq E_o$ holds, *i.e.* all the controllable events are observable.

In a setting with imperfect information, the definition of a maximally safe supervisor in Expression (15) below is a generalization of the definition of the safe supervisor of Expression (14), in the sense that after the occurrence of $\sigma$, the system should not only remain in a safe state after the occurrence of any finite sequence of uncontrollable events, but also the supervisor should know this.

**Theorem 23** (Synthesis of maximally safe $P$-supervisor)**.** *Assume that $E_c \subseteq E_o$ and that $\mathcal{I}, s_0 \models K[\mathsf{uc}]\mathsf{safe}$. Let $f : \mathcal{L}(A) \to \mathcal{P}(E)$ be the function defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f(w) = E_{uc} \cup \{\sigma \in E_c : \mathcal{I}, w \models K[\sigma][\mathsf{uc}]\mathsf{safe}\} \qquad (15)$$

*Then, $f$ is a $P$-supervisor maximally safe for $\overline{\mathcal{K}}$. Moreover, if $\mathcal{K} \subseteq \mathcal{L}_m(A)$, then $f$ is a* non-blocking *maximally safe P-supervisor. Finally, $f$ satisfies the following condition:*

$$\mathcal{L}(A, f) = \{w \in \mathcal{L}(A) : \mathcal{I}, \overline{w} \models K[\mathsf{uc}]\mathsf{safe} \text{ for all } \overline{w} \leq w\}$$

Intuitively, the definition of $\mathcal{L}(A, f)$ means that the set of runs induced by the maximally safe supervisor $f$ of Expression (15) is characterized by the fact that, at any time, the supervisor *knows* that it is in a safe state and that the occurrence of any sequence of uncontrollable event will still lead to a safe state.

**Theorem 24** (Existence of safe $P$-supervisor)**.** *Assume that $E_c \subseteq E_o$. Then the following statements are equivalent:*

1. *$\mathcal{I}, s_0 \models K[\mathsf{uc}]\mathsf{safe}$;*
2. *there exists a $P$-supervisor $f$ safe for $\overline{\mathcal{K}}$;*
3. *there exists a $P$-supervisor $f$* maximally *safe for $\overline{\mathcal{K}}$ (given by Expression (15)).*

*Moreover, if we assume that $\mathcal{K} \subseteq \mathcal{L}_m(A)$, then $P$-supervisors are also* non-blocking.

## 6.3 Decentralized Control

In this section, we consider the control problem with imperfect information when there are multiple supervisors. We first elicit a necessary condition for the existence of a decentralized supervisor.

**Proposition 25.** *Assume that $E_c \subseteq E_o$. There exists a decentralized supervisor (either conjunctive or disjunctive) safe for $\overline{\mathcal{K}}$ only if $\mathcal{I}, s_0 \models D_{\mathbb{A}}[\mathsf{uc}]\mathsf{safe}$.*

The following proposition defines a family of $P_i$-supervisors whose associated decentralized supervisor (either conjunctive or disjunctive) is safe for the goal behavior. It also provides sufficient conditions under which this family of $P_i$-supervisors is *maximally* safe for the goal behavior.

**Proposition 26.** *Assume that $E_c \subseteq E_o$ and that $\mathcal{I}, s_0 \models D_{\mathbb{A}}[\mathsf{uc}]\mathsf{safe}$. For all $i \in \mathbb{A}$, let $f_i$ be the $P_i$-supervisor defined as follows: for all $w \in \mathcal{L}(A)$,*

$$f_i(w) = E_{uc} \cup \{\sigma \in E_{c,i} : \mathcal{I}, w \models K_i[\sigma][\mathsf{uc}]\mathsf{safe}\}. \quad (16)$$

*Then, the (conjunctive or disjunctive) decentralized supervisor associated to $\{f_i : i \in \mathbb{A}\}$ is a $P$-supervisor safe for $\overline{\mathcal{K}}$.*

*The $P$-supervisor $f$ is* maximally *safe for $\overline{\mathcal{K}}$ if, and only if, for all $v \in \overline{\mathcal{K}^{\uparrow CN}}$, all $\sigma \in E_c$ such that $v\sigma \in \mathcal{L}(A)$,*

$$\mathcal{I}, v \models D_{\mathbb{A}}[\sigma][\mathsf{uc}]\mathsf{safe} \rightarrow \bigwedge_{i \in \mathbb{A}_c(\sigma)} K_i[\sigma][\mathsf{uc}]\mathsf{safe} \quad \text{(Conjunctive)}$$

$$\mathcal{I}, v \models D_{\mathbb{A}}[\sigma][\mathsf{uc}]\mathsf{safe} \rightarrow \bigvee_{i \in \mathbb{A}_c(\sigma)} K_i[\sigma][\mathsf{uc}]\mathsf{safe}. \quad \text{(Disjunctive)}$$

The results above can be given an intuitive interpretation. For the conjunctive case, they state that the *conjunctive* supervisor defined by Expression (16) is, in fact, a maximally safe $P$-supervisor if, and only if, each time the supervisors can pool their knowledge to conclude that the occurrence of an event is safe ($D_{\mathbb{A}}[\sigma][\mathsf{uc}]\mathsf{safe}$), then in fact they already *all* know that the occurrence of this event is safe ($\bigwedge_{i \in \mathbb{A}_c(\sigma)} K_i[\sigma][\mathsf{uc}]\mathsf{safe}$). Likewise for the disjunctive case.

# 7. CONCLUSION

## 7.1 Related Work

In supervisory control theory, the work of Ricker & Rudie [14] is the closest to our work. They reformulate the decentralized control problem and other (new) notions of supervisory control theory in epistemic logic, but without action nor temporal modalities. The lack of expressiveness of their logic renders impossible the expression of properties such as coobservability in terms of model checking problems.

Our approach for synthesizing supervisors by reformulating them in terms of model checking problems resembles what is called in supervisory control theory "online supervision" [7]. The idea of this approach is first to compute a supervisor $f$ for the system under full observation and then to derive from this first supervisor a second supervisor $f'$ defined as follows: $f'(w) := \bigcap_{w' \in P^{-1}(P(w))} f(w')$. This definition is in a certain sense equivalent to the definition of $f$ in Expression (6).

Dima & al. [5] introduced a variant of ATL with distributed knowledge operators based on a perfect recall but *synchronous* semantics. The multi-agent situations that they deal with are in fact very similar to the situations considered in supervisory control theory. We also mention Huang [10] who follows a similar methodology by reformulating the notions of diagnosability studied in DES [4] as model checking problems in a probabilistic temporal logic.

Finally, as the reader might have noticed, issues and concepts of supervisory control theory are very similar to those of game theory. See Arnold & al. [1] for more details.

## 7.2 Concluding Remarks

From a conceptual point of view, our results highlight the underlying intuitions of supervisory control theory, as the numerous discussions and comments about our theorems and propositions show this (especially those of Section 5.3).

From an applied perspective, an interesting line of research would be to investigate whether our reformulation of supervisory control problems makes the implementation of the solutions of these problems more efficient. Our theorems make it possible to lazily compute the supervisors online. Given a $(P)$-supervisor $f$ as defined by Expression (2),(3),(6),(10),(12),(14),(15) or (16), an event $\sigma \in E$ and a word $w \in \mathcal{L}(A)$, the computational complexity of deciding whether $\sigma \in f(w)$ is in EXPTIME, according to Theorem 8. But in fact, there exist dedicated algorithms that run in time $O(m^2 \cdot n)$ for any of these $(P)$-supervisors (where $m = |A| \times |O| \times |E|$ and $n = |P(w)|$). Hence, these results and the symbolic methods developed for epistemic temporal logics sustain the possibility of applying model checkers (such as MCMAS or MCK) to supervisory control problems.

# 8. REFERENCES


[1] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theor. Comput. Sci.*, 1(303):7–34, 2003.

[2] G. Aucher. Infinite Games in Epistemic Temporal Logic via Supervisory Control Theory. Research Report RR-8374, INRIA, Sept. 2013.

[3] C. Baier and J. Katoen. *Principles of model checking*. MIT press, 2008.

[4] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, 2008.

[5] C. Dima, C. Enea, and D. P. Guelev. Model-checking an alternating-time temporal logic with knowledge, imperfect information, perfect recall and communicating coalitions. In A. Montanari, M. Napoli, and M. Parente, editors, *GANDALF*, volume 25 of *EPTCS*, pages 103–117, 2010.

[6] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. MIT Press, 1995.

[7] N. Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dynamic Systems*, 6(4):379–427, 1996.

[8] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time. i. lower bounds. *J. Comput. Syst. Sci.*, 38(1):195–237, 1989.

[9] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[10] X. Huang. Diagnosability in concurrent probabilistic systems. In M. L. Gini, O. Shehory, T. Ito, and C. M. Jonker, editors, *AAMAS*, pages 853–860, 2013.

[11] B. Maubert. *Logical Foundations of Imperfect Information Games: Uniform Strategies*. PhD thesis, University of Rennes 1, 2014.

[12] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[13] J. H. Reif. The complexity of two-player games of incomplete information. *J. Comput. Syst. Sci.*, 29(2):274–301, 1984.

[14] S. L. Ricker and K. Rudie. Know means no: Incorporating knowledge into discrete-event control systems. *IEEE Transactions on Automatic Control*, 45(9):1656–1668, 2000.

[15] R. van der Meyden and K.-S. Wong. Complete axiomatizations for reasoning about knowledge and branching time. *Studia Logica*, 75(1):93–123, 2003.