Concurrent Secrets

Eric Badouel*, Marek Bednarczyk**, Andrzej Borzyszkowski***, Benoît Caillaud*, and Philippe Darondeau*

***Abstract*— Given a finite state system with partial observers and for each observer, a regular set of trajectories which we call a secret, we consider the question whether the observers can ever find out that a trajectory of the system belongs to some secret. We search for a regular control on the system, enforcing the specified secrets on the observers, even though they have full knowledge of this control. We show that an optimal control always exists although it is generally not regular. We state sufficient conditions for computing a finite and optimal control of the system enforcing the concurrent secret as desired.**

I. INTRODUCTION

This work is an attempt to import supervisory control into the area of computer security. Given an automaton, or plant, and given specifications of the desired behaviour of the plant, Ramadge and Wonham's theory presented in [1] [2] yields a finite, non blocking, and maximal permissive control of the plant enforcing this behaviour (in the normal case or when unobservable events are uncontrolable). Controller synthesis is a desirable complement to model checking, for it allows curing the problems that model checkers can reveal. Supervisory control has found applications in manufacturing systems, in embedded systems, and more generally in safety critical systems. We feel it could find applications as well in computer security, and we shall strive to support this thesis.

With the above goal in mind, we have searched for a class of security problems likely to be dealt with as control problems. We model an interactive computer system and its users as a closed entity in which the users observe their own interactions with the system. The closed entity is represented with a finite automaton over an alphabet Σ. The synchronous interactions between each user i and the system are figured by the elements of a corresponding sub-alphabet $\Sigma_i \subseteq \Sigma$ (users may synchronize when their sub-alphabets intersect). Usually in supervisory control, the control objective is a predicate on the runs of the plant, specifying some combination of safety and liveness properties, and the observers act as sensors, *i.e.* they supply informations on the status of the plant, used by the controller to produce an adequate feedback enabling or disabling events in the plant. Here, the game is different: the observers are not on the side of the controller but they are opponents. As for the control objective, there are still predicates (S_i) on the runs of the system, but the interpretation is again different: an observer i should never find out that the actual trajectory of the system belongs to the secret (S_i) he has been assigned.

One reason why we believe the model sketched above is worth investigating is that, in the case of a single observer, it has already been introduced independently in [3] and studied further in [4]. What we call *secrets* here was called there *opaque predicates*, albeit with larger families of predicates (sets of runs) and observation functions. It was shown in [4] that anonymity problems and non-interference problems may be reduced to opacity problems, using suitable observation functions. It was shown *ibidem* that model-checking a system for opacity is undecidable in the general case where an opaque predicate may refer to the visited states or may be any recursive predicate on sequences of event labels. Nonetheless, techniques based on abstract interpretation were proposed in [4] for checking opacity in unbounded Petri nets.

In this paper, we limit ourselves to deal with finite state systems and with regular predicates defined on sequences of transition labels. We have thus all cards in hands to decide opacity, even though several pairs *(observer, secret)* are taken into simultaneous account. Now differing from [4], we want to be able to *enforce* opacity by supervisory control when the result of the decision is negative. In other terms, we want to disable the least possible family of trajectories such that no observer can ever find out that the system's actual trajectory belongs to some secret. At first sight, this looks like a simple problem, all the more when it is assumed that all events are controllable as we do in this paper (we leave the uncontrollable events to further consideration). The problem is in fact not that simple, for the observers have full knowledge of the system, hence any control device that may be added to the system is known to them. We will nevertheless show that there exists always an optimal control for enforcing the concurrent secrets on opponents, fully aware of this control. We will also provide techniques for computing this optimal control under assumptions that fit at least with some applications.

The rest of the paper is organized as follows. The notation and the problem are introduced in section II. Section III shows that a unique optimal solution always exists, but it is generally not regular. Using the fixpoint characterization of the optimal control, proofs of control enabledness of trajectories are presented as infinite trees in section IV; conditions on proof trees entailing the regularity of the optimal control are also stated there. Section V produces closely connected conditions on concurrent secrets. An application is sketched in section VI, where directions for further work are also suggested.

* E. Badouel, B. Caillaud, and P. Darondeau are with INRIA-Rennes
e-mail: ebadouel@irisa.fr bcaillau@irisa.fr darondeau@irisa.fr
** M. Bednarczyk is with IPIPAN-Gdańsk
e-mail: m.bednarczyk@ipipan.gda.pl
*** A. Borzyszkowski is with the University of Gdańsk and IPIPAN
e-mail: a.borzyszkowski@math.univ.gda.pl

II. SECRETS, CONCURRENT SECRETS, AND THE CONTROL PROBLEM

To begin with, let us fix the notation. Σ is a finite alphabet, Σ^* is the free monoid generated by Σ, and $Rat(\Sigma^*)$ is the family of rational subsets of Σ^* i.e. the family of regular languages over Σ. Let uv denote the concatenation product of the words u and v, thus u is a prefix of uv and the empty word ε is a prefix of every word. The length of u is denoted by $|u|$. For $l \leq |u|$, $u[l]$ denotes the prefix of u with the length l, and for $0 < l \leq |u|$, $u(l)$ denotes the l^{th} letter occurring in u. For any sub-alphabet $\Sigma_i \subseteq \Sigma$, let $\pi_i : \Sigma^* \to \Sigma_i^*$ be the unique monoid morphism extending the map $\pi_i(\sigma) = \sigma$ if $\sigma \in \Sigma_i$ else ε (letters $\sigma \in \Sigma$ are mapped to words by the usual embedding of Σ into Σ^*). For $u, v \in \Sigma^*$, let $u \simeq_i v$ if $\pi_i(u) = \pi_i(v)$. Throughout the paper, L is a non-empty prefix-closed language in $Rat(\Sigma^*)$ and for all $i \in \{1, \cdots, n\}$, $\Sigma_i \subseteq \Sigma$, $S_i \in Rat(\Sigma^*)$, and $S_i \subseteq L$.

The language L represents the behaviour of a system with n users. For $i \in \{1, \cdots, n\}$, the sub-alphabet Σ_i represents the set of the interactions that may take place between the system and the user i. Users observe the system by interacting with it. If the system's trajectory is represented by $w \in L$, then the induced observation for the user i is $\pi_i(w)$. Two users can communicate only by jointly interacting with the system, *e.g.* $\sigma \in \Sigma_i \cap \Sigma_j$ is an interaction of the system with the users i and j.

For each $i \in \{1, \cdots, n\}$, the membership of the actual system's trajectory to the subset $S_i \subseteq L$ is intended to be kept *secret* from the user i. In the terminology of [3] and [4], the predicate S_i should be *opaque* w.r.t. the observation function π_i and the language L.

Definition 1: S_i is *opaque* w.r.t. π_i (and L) if $(\forall w \in S_i)\,(\exists w' \in L \setminus S_i)\; w \simeq_i w'$

When the predicate S_i coincides with its prefix closure $\overline{S_i}$, non-opacity is the same as normality which may be expressed as $\forall w \in \overline{S_i}\ \ \forall w' \in L\ \ w \simeq_i w' \ \Rightarrow\ w' \in \overline{S_i}$. However, opacity is not the opposite of normality, as the following example shows. Given $L = (ab)^* + (ab)^* a$ let $\Sigma_i = \{b\}$ and $S_i = (ab)^* a$ then S_i is both opaque and normal.

As we explained in the introduction, we use here a strongly restricted form of the original definition of opacity where the observation functions may be state and history dependent. On the other hand, we consider a concurrent version of opacity.

Definition 2: $(S_i)_i$ is *concurrently opaque* (w.r.t. L) if for all i, S_i is opaque w.r.t. π_i.

Dealing with concurrent opacity does not make a big change for checking opacity, which is easy in our case (although not necessarily computationally simple) since we consider exclusively regular systems and secrets.

Proposition 1: It is decidable whether $(S_i)_i$ is concurrently opaque.

Proof: By definition, it suffices to decide for each $i \in \{1, \ldots, n\}$ whether S_i is opaque w.r.t. π_i. The considered property holds if and only if $\pi_i(S_i) \subseteq \pi_i(L \setminus S_i)$. As L and S_i are regular, $L \setminus S_i$ is regular, and since morphic images of regular languages are regular, this relation can be decided. ■

Example 1: Let $\Sigma = \{a, b, c\}$ and L be the set of prefixes of words in $(a+b)c$. Let $\Sigma_1 = \Sigma_2 = \{c\}$, and let S_1 and S_2 be the intersections of L with $\Sigma^* a \Sigma^*$ and $\Sigma^* b \Sigma^*$, respectively. The concurrent secret (S_1, S_2) is opaque. From the observation of the event c, one is indeed unable to infer whether it was preceded with an a or with a b.

In the sequel, $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_n, S_n)\}$ denotes a concurrent secret upon a fixed language $L \subseteq \Sigma^*$ ($\Sigma_i \subseteq \Sigma$ and $S_i \subseteq L \subseteq \Sigma^*$ for all i). We say that $\mathcal{S}$ is opaque if $(S_i)_i$ is concurrently opaque. A control is any non-empty prefix-closed language $L' \subseteq L$ (we assume here that all events $\sigma \in \Sigma$ are controllable). We say that $\mathcal{S}$ is opaque under the control $L' \subseteq L$ if the induced secret $(S_i')_i$ defined with $S_i' = S_i \cap L'$ is concurrently opaque w.r.t. L'.

Our purpose is to solve the concurrent opacity control problem stated as follows.

Problem 1: Show that the set of controls enforcing the opacity of $\mathcal{S}$ either is empty or has a greatest element, and compute this maximal permissive control.

Enforcing concurrent opacity ($n > 1$) requires, as we shall see, significantly more efforts than enforcing opacity.

III. A FIXPOINT CHARACTERIZATION OF THE MAXIMAL PERMISSIVE CONTROL ENFORCING CONCURRENT OPACITY

In this section, we show that the concurrent opacity control problem has a unique maximal solution that we characterize as a greatest fixpoint. We propose two counter-examples in which this maximal permissive control either is not regular or cannot be computed within a finite number of fixpoint iterations.

Definition 3: For any prefix-closed subset L' of L, the *safe kernel* of L' w.r.t. the secret $\mathcal{S}$, notation $K(L', \mathcal{S})$, is the subset of all words $w \in L'$ such that $w = uv \Rightarrow (\forall i)(\exists u' \in L' \setminus S_i)\; u \simeq_i u'$.

Thus, $\mathcal{S}$ is opaque under the control $L' \subseteq L$ if and only if $L' = K(L', \mathcal{S})$, *i.e.* L' is a fixpoint of $K(\bullet, \mathcal{S})$. It is immediately observed that $K(L', \mathcal{S})$ is continuous in the first argument (w.r.t. set inclusion). As the prefix-closed subsets of L form a complete sub-lattice of $\mathcal{P}(\Sigma^*)$, it follows from Knaster-Tarski's theorem [5] that $K(\bullet, \mathcal{S})$ has a greatest fixpoint in this sub-lattice.

Definition 4: Let $SupK(L, \mathcal{S})$ be the greatest fixed point of the operator $K(\bullet, \mathcal{S})$.

Proposition 2: $SupK(L, \mathcal{S})$ is the union of all controls enforcing the opacity of $\mathcal{S}$. If $SupK(L, \mathcal{S}) \neq \emptyset$, then it is the maximal permissive control enforcing the opacity of $\mathcal{S}$, otherwise no such control can exist.

Proof: This is a direct application of the Knaster-Tarski's fixpoint theorem. ■

Remark 1: The condition $L' \subseteq SupK(L, \mathcal{S})$ is necessary *but not sufficient* for some non-empty control L' to enforce the opacity of $\mathcal{S}$. For instance, in Example 1, $SupK(L, \mathcal{S}) = L$, but the secret S_1 is not opaque w.r.t. $L' = \varepsilon + a + ac$.

The fixpoint characterization of the optimal control enforcing opacity does not show that $SupK(L, \mathcal{S})$ can be computed, nor that the control can be implemented with a

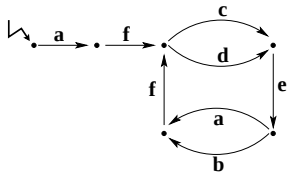

Fig. 1. An automaton

finite device. When $n = 1$, *i.e.* when $\mathcal{S} = \{(\Sigma_1, S_1)\}$, this is not a problem because in this particular case, $SupK(L,\mathcal{S})$ is equal to $K(L,\mathcal{S})$ and it may be shown that $K(L,\mathcal{S})$ is the set of words with all prefixes in $L \cap \pi_1^{-1}(L \setminus S_1)$. Therefore, $SupK(L,\mathcal{S}) = \Sigma^* \setminus ((\Sigma^* \setminus (L \cap \pi_1^{-1}(L \setminus S_1))\Sigma^*)$ which is regular. When $n > 1$, two situations contrast. The nice situation is when $SupK(L,\mathcal{S})$ can be computed from L by a finite number of iterated applications of the operator $K(\bullet,\mathcal{S})$. Actually, when L' is a regular subset of L, the same holds for $K(L',\mathcal{S})$, hence in the considered case $SupK(L,\mathcal{S})$ is regular. The rest of the section illustrates the converse situation.

A. A case where the closure ordinal of $K(\bullet,\mathcal{S})$ is transfinite

Let $\Sigma = \{a,b,c,d,e,f\}$ and let L be the prefix-closed language accepted by the finite automaton of Fig.1 (where all states are accepting states). Define $\mathcal{S} = \{(\Sigma_1,S_1),(\Sigma_2,S_2)\}$ with $\Sigma_1 = \{c, f\}$, $S_1 = \Sigma^* afc(\Sigma \setminus \{c\})^*$ (this secret is safe if, by observing only c and f, one cannot find out in any run that the last occurrence of c was preceded by af), and $\Sigma_2 = \{b, e\}$, $S_2 = \Sigma^* deb(\Sigma \setminus \{b\})^*$ (this secret is safe if, by observing only b and e, one cannot find out in any run that the last occurrence of b was preceded by de). Let $L_1 = K(L,\mathcal{S})$ be the first language encountered in the greatest fixpoint iteration converging to $SupK(L,\mathcal{S})$, then $L_1 = L \setminus afc\Sigma^*$ (the run afc reveals the secret S_1 and the runs in $afd\Sigma^*$ reveal nothing). The second item $L_2 = K(L_1,\mathcal{S})$ is the language $L_1 \setminus afdeb\Sigma^*$ (relatively to L_1, the run $afdeb$ reveals the secret S_2, and the runs in $afdea\Sigma^*$ reveal nothing). After afc and $afdeb$ have been eliminated, the initial situation reproduces up to the prefix $afdea$. Therefore, the fixpoint iteration produces a strictly decreasing and infinite sequence of languages L_j. The limit $SupK(L,\mathcal{S})$ of this decreasing chain is the set of all prefixes of words in the regular set $L_\omega = (afde)^*$, hence it is regular. The optimal control enforcing the opacity of $\mathcal{S}$ may be implemented by any finite automaton recognizing L_ω.

Let us now extend the concurrent secret into $\mathcal{S} = \{(\Sigma_1,S_1),(\Sigma_2,S_2),(\Sigma_3,S_3)\}$ with (Σ_1,S_1) and (Σ_2,S_2) as above, $\Sigma_3 = \emptyset$ and $S_3 = \Sigma^* \setminus (\Sigma^* c \Sigma^*)$. Then, the closure ordinal of $K(\bullet,\mathcal{S})$ increases from ω to $\omega+1$. To see this observe that, since Σ_3 is empty, the secret S_3 is safe relatively to any language $L' \subseteq L$ containing at least one word containing at least one occurrence of c. The greatest fixpoint iteration for $SupK(L,\mathcal{S})$ starts with the same decreasing sequence L_j as before, but $K(L_\omega,\mathcal{S})$ differs now from L_ω because L_ω contains no word containing c (differing in that form all L_j). In fact, $L_{\omega+1} = K(L_\omega,\mathcal{S}) = \emptyset$ and this is a fixpoint. Opacity can therefore not be enforced.

B. A case where $SupK(L,\mathcal{S})$ is not regular

Let $\Sigma = \{a,b,x,y\}$ and L be the set of prefixes of words in $(ax)^*(\varepsilon + ab)(yb)^*$. Define $\mathcal{S} = \{(\Sigma_i,S_i) \mid 1 \le i \le 3\}$ as follows (letting $\complement L' = L \setminus L'$ for $L' \subseteq L$):

1) $\Sigma_1 = \{a,b\}$, $\complement S_1 = \varepsilon + (ax)^* ab(yb)^* + (\Sigma \setminus \{b\})^*$
2) $\Sigma_2 = \{x,y\}$, $\complement S_2 = (ax)^*(yb)^*$
3) $\Sigma_3 = \{a,b,x,y\}$, $\complement S_3 = \varepsilon + a\Sigma^*$

We claim that $SupK(L,\mathcal{S})$ is not a regular language and worse, the family of regular controls enforcing the opacity of $\mathcal{S}$ has no largest element. Recall that the subset of maximal words in a regular language is regular. In order to establish the first part of the claim, one can show that $SupK(L,\mathcal{S})$ is equal to the set L' of all prefixes of words in the non regular language $\cup_{n\in\mathbb{N}}\ (ax)^n(\varepsilon+ab)(yb)^n$. A detailed proof of this fact may be found in [6]. To show that the family of regular controls enforcing the opacity of $\mathcal{S}$ has no largest element, one assumes the opposite. Let R be the largest prefix-closed regular subset of L such that $\mathcal{S}$ is opaque w.r.t. R. Necessarily, $(ax)^n(yb)^n \notin R$ for some n. If it were otherwise, because $(ax)^{n-1}(yb)^{n-1}$ is the sole word $w' \in L' \setminus S_1$ such that $w \simeq_1 w'$ for $w = (ax)^n(yb)^n$, R would coincide with L', which is not possible (L' is not regular). Let n be the least integer such that $(ax)^n(yb)^n \notin R$, and let R' be R augmented with all prefixes of words in $\{(ax)^n(yb)^n, (ax)^{n-1}ab(yb)^{n-1}\}$ not already in R. The language R' is prefix-closed and regular, and one can verify that $\mathcal{S}$ is opaque w.r.t. R'. Thus, a contradiction has been reached.

IV. CONTROL ENABLING AND ω-TREES

Warning 1: From now on, $S_i\Sigma^* \subseteq S_i$ is imposed on all sets S_i in $\mathcal{S} = \{(\Sigma_1,S_1),\ldots,(\Sigma_n,S_n)\}$.

This section serves as a bridge between the general problem and the practical solutions that we shall propose in specific cases. The working assumption that secrets are suffix-closed is motivated by its convenience (if not its necessity) for enforcing opacity with finite control. Although this working assumption was not satisfied in the examples from sections III-A and III-B, it is quite natural since it amounts to strengthening the secrecy requirement as follows: an observer i should never have the knowledge that the trajectory of the system is in S_i *or was* in S_i at some instant in the past. We give below a simpler definition of the operator $K(\bullet,\mathcal{S})$, which is equivalent to the earlier definition when secrets are suffix-closed. Then we consider ω-trees that may be seen as proofs of control enabledness of trajectories. Finally, we propose conditions on sets of proof trees entailing the regularity of $SupK(L,\mathcal{S})$, thus paving the way for section V.

Definition 5 (modified form of Def. 3): For any prefix-closed subset L' of L, the *safe kernel* of L' w.r.t. the secret $\mathcal{S}$, notation $K(L',\mathcal{S})$, is the largest subset of L' such that $w \in K(L',\mathcal{S}) \Rightarrow (\forall i)(\exists w' \in L' \setminus S_i)\ w \simeq_i w'$.

Proposition 3: Definitions 3 and 5 are equivalent.

Proof: For the duration of this proof, let $K(\bullet,\mathcal{S})$ and $K'(\bullet,\mathcal{S})$ be the two operators from Def. 3 and Def. 5, respectively. Clearly, $K(L',\mathcal{S}) \subseteq K'(L',\mathcal{S})$ for any L'. We show the converse relation. Consider any word $w \in K'(L',\mathcal{S})$ and let $w = uv$ be any decomposition of this word into two factors. We should prove that for all $i \in \{1,\ldots,n\}$, $u \simeq_i u'$ for some $u' \in L' \setminus S_i$. As $w \in K'(L',\mathcal{S})$ and by definition, $w \simeq_i w'$ for some $w' \in L' \setminus S_i$. Now $w' \simeq_i uv$, hence there exists at least one decomposition $w' = u'v'$ such that $u \simeq_i u'$. Finally, $u' \in L'$ by prefix-closedness of L', and $u' \notin S_i$ by suffix-closedness of S_i. Therefore, $w \in K(L',\mathcal{S})$. ■

Definition 6: Given $w \in L$, a *proof* of *enabledness* of w is a map $f : \{1,\ldots,n\}^* \to L$ such that $f(\varepsilon) = w$ and for all $\tau \in \{1,\ldots,n\}^*$ and $j \in \{1,\ldots,n\}$, $f(\tau) \simeq_j f(\tau j)$ and $f(\tau j) \notin S_j$.

The map f in the above definition is just a complete n-ary ordered tree labelled on nodes, thus in particular it is an infinite tree. The next proposition follows immediately from the co-inductive definition of $SupK(L,\mathcal{S})$.

Proposition 4: For any $w \in L$, $w \in SupK(L,\mathcal{S})$ if and only if there exists a proof of the control enabledness of w.

A nice situation is when the control enabledness of a trajectory may be proved with a regular tree. Let us recall the definition.

Definition 7: Let $f : \{1,\ldots,n\}^* \to L$ be a (complete n-ary ordered labelled) tree. For any $\tau \in \{1,\ldots,n\}^*$, the sub-tree of f rooted at τ, in notation f/τ, is the (complete n-ary ordered labelled) tree defined with $(f/\tau)(\tau') = f(\tau\tau')$ for all $\tau' \in \{1,\ldots,n\}^*$. The tree f is regular if it has a finite number of sub-trees f/τ.

Any regular tree f may be folded to a finite rooted graph. When the control enabledness of the (good) trajectories may be proved using regular trees exclusively, this predicate is therefore recursively enumerable. This condition is necessary and sufficient for being able to enforce control, but not efficiently. In the rest of the section, we search for additional conditions entailing the regularity of the control $SupK(L,\mathcal{S})$.

A first attempt towards this goal is to impose an upper bound on the number of (different) subtrees of a regular proof tree. Equivalently, one may require that all proof trees conform to a finite collection of finite patterns as follows.

Definition 8: A *finite pattern for proofs* (of enabledness of trajectories) is a finite, deterministic and complete automaton $(Q,\{1,\ldots,n\},q_0)$ (thus $q_0 \in Q$ and any $i \in \{1,\ldots,n\}$ maps Q to itself). A proof tree $f : \{1,\ldots,n\}^* \to L$ *conforms to* a finite pattern if there exists a labelling map $\lambda : Q \to L$ such that $f(\tau) = \lambda(q_0 \cdot \tau)$ for all $\tau \in \{1,\ldots,n\}^*$ letting $q \cdot \tau$ be defined inductively with $q \cdot \varepsilon = q$ and $q \cdot (\tau_1\tau_2) = (q \cdot \tau_1) \cdot \tau_2$ for all $q \in Q$.

The idea behind this definition is that proof trees contain bounded information up to the choice of a bounded number of words in L.

Example 2: Let $\Sigma = \{a,b\}$ and $L = \Sigma^*$. Let $\mathcal{S} = \{(\Sigma_1,S_1),(\Sigma_2,S_2)\}$ with $\Sigma_1 = \{a\}$, $\complement S_1 = b^*a^*$ and $\Sigma_2 = \{b\}$, $\complement S_2 = a^*b^*$. The finite pattern shown on Fig. 2 supplies proofs of control enabledness for all trajectories. For any word w with n occurrences of a and m occurrences of b, the labelling map defined with $\lambda(q_0) = w$, $\lambda(q_1) = b^m a^n$, and $\lambda(q_2) = a^n b^m$ induces in fact an ω-tree witnessing that $w \in SupK(L,\mathcal{S})$.

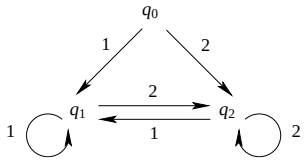

Fig. 2.

There are two sources of problems with the proof patterns from Def. 8. The first difficulty is that, given L, $\mathcal{S}$ and $(Q,\{1,\ldots,n\},q_0)$, the set of the labelling maps $\lambda : Q \to L$ considered in this definition is generally not regular, *i.e.* it cannot be defined with a finite automaton on $(\Sigma^*)^{|Q|}$. For instance, if the labelling maps considered in example 2 did form a regular set, then the set of all pairs $(b^m a^n, a^n b^m)$ would be regular, but the iteration lemma for rational sets [7] entails the opposite (if the set is regular, for some $N > 1$ and for large enough n and m, $(b^m a^n, a^n b^m)$ could be written as $(x,x')(y,y')(z,z')$ where $0 < |y| + |y'|$, $|x| + |x'| + |y| + |y'| \leq N$, and $(x,x')(y,y')^*(z,z')$ is included in the set). The second difficulty is that, given L, $\mathcal{S}$ and $(Q,\{1,\ldots,n\},q_0)$, the set of values taken at $q = q_0$ by the labelling maps from Def. 8 is sometimes not regular. An example is shown hereafter.

Example 3: Let $\Sigma = \{a,b\}$ and $L = \Sigma^*$. Let $\mathcal{S} = \{(\Sigma_1,S_1),(\Sigma_2,S_2)\}$ where $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, and $\complement S_1 = \complement S_2 = (\varepsilon + b)(ab)^*(\varepsilon + a)$. Consider the set of all maps labelling adequately the finite proof pattern from Fig. 3. The set of values taken by these maps at $q = q_0$ is the set of all words in which the numbers of occurrences a and b differ by at most one, hence it is not regular.

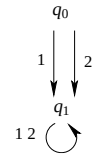

Fig. 3.

Note that in both examples 2 and 3, $SupK(L,\mathcal{S}) = \Sigma^*$, and proofs of enabledness may be obtained for all $w \in \Sigma^*$ by labelling the finite proof pattern shown in Fig. 4.

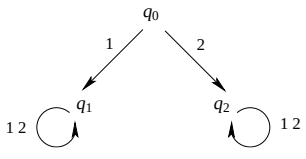

Fig. 4.

In order to dodge the problems, one may concentrate on restricted proof patterns as follows.

Definition 9: A *type* (of proof of enabledness) is a finite pattern $\mathcal{T} = (Q,\{1,\ldots,n\},q_0)$ with a prefix-closed subset $T \subseteq \{1,\ldots,n\}^*$ such that $(\forall q \in Q)\,(\exists!\,\tau \in T)\,(q = q_0 \cdot \tau)$ and

for any map $\lambda : Q \to L$,
$(\forall \tau)(\forall j)\ \ (\tau j \in T \wedge \lambda(q_0 \cdot \tau) \simeq_j \lambda(q_0 \cdot \tau j) \wedge \lambda(q_0 \cdot \tau j) \notin S_j)$
$\Rightarrow\ (\forall q)(\forall j)\ \ (\lambda(q) \simeq_j \lambda(q \cdot j) \wedge \lambda(q \cdot j) \notin S_j)$
where τ and j range over $\{1,\ldots,n\}^*$ resp. over $\{1,\ldots,n\}$. A proof tree $f : \{1,\ldots,n\}^* \to L$ has type $\mathcal{T}$ if it conforms to this pattern (see Def. 8).

The set T in Def. 9 induces a (finite) tree, rooted at q_0, that spans the automaton $(Q, \{1,\ldots,n\}, q_0)$. The point is that for any map $\lambda : Q \to L$, if $(\lambda(q) \simeq_j \lambda(q \cdot j) \wedge \lambda(q \cdot j) \notin S_j)$ for all arcs $(q, q \cdot j)$ in the spanning tree, then it holds also for all chords, *i.e.* for all remaining edges of (the underlying graph of) $(Q, \{1,\ldots,n\}, q_0)$.

Theorem 1: If there exists a finite number of types of proofs of enabledness for all trajectories $w \in SupK(L, \mathcal{S})$, then $SupK(L, \mathcal{S})$ is a regular language.

Proof: It suffices to show that when type $\mathcal{T} = (Q, \{1,\ldots,n\}, q_0, T)$ has been fixed, the set of trajectories $w \in L$ with proofs of enabledness of type $\mathcal{T}$ is regular. In view of the definitions 8 and 9, a word w belongs to the considered set if and only if $\lambda(q_0) = w$ for some map $\lambda : Q \to L$ satisfying $\lambda(q_0 \cdot \tau) \simeq_j \lambda(q_0 \cdot \tau j)$ and $\lambda(q_0 \cdot \tau j) \notin S_j$ whenever $\tau j \in T$ and $j \in \{1,\ldots,n\}$. In order to show that this is a regular set, we construct the Arnold-Nivat product [8] of a family of automata $\mathcal{A}_\tau$ indexed with $\tau \in T$, as follows. Let $\mathcal{A}_\varepsilon$ be a (finite deterministic) partial automaton recognizing L, and for each sequence τj in T with $j \in \{1,\ldots,n\}$, let $\mathcal{A}_{\tau j}$ be a (finite deterministic) partial automaton recognizing $L \setminus S_j$. This defines the components of the product. As for the synchronizations, let $\mathcal{V}$ be the set of T-vectors $\vec{v} : T \to (\Sigma \cup \{\varepsilon\})$ such that $(\vec{v}(\tau) \in \Sigma_j \vee \vec{v}(\tau j) \in \Sigma_j) \Rightarrow \vec{v}(\tau) = \vec{v}(\tau j)$ whenever τj in T and $j \in \{1,\ldots,n\}$. The induced product is a (finite deterministic) partial automaton $\mathcal{A} = (Q, \mathcal{V}, \vec{q}_0)$ defined as follows:
- the set of states Q is a set of T-vectors,
- for each $\tau \in T$, $\vec{q}_0(\tau)$ is the initial state of $\mathcal{A}_\tau$,
- for all $\vec{q} \in Q$ and $\tau \in T$, $\vec{q}(\tau)$ is a state of $\mathcal{A}_\tau$,
- for all $\vec{q} \in Q$, $\vec{v} \in \mathcal{V}$ and $\tau \in T$, $(\vec{q} \cdot \vec{v})(\tau) = \vec{q}(\tau) \cdot \vec{v}(\tau)$.

Therefore, $\vec{q} \cdot \vec{v}$ is defined if and only if $\vec{q}(\tau) \cdot \vec{v}(\tau) = \vec{q}'(\tau)$ is defined for all τ.

Let $\vec{v}_1 \ldots \vec{v}_m$ be a word over $\mathcal{V}$ accepted by $\mathcal{A}$. An associated T-vector $\vec{w} : T \to L$ may be defined by setting $\vec{w}(\tau) = \vec{v}_1(\tau) \ldots \vec{v}_m(\tau)$ for all $\tau \in T$. It follows directly from the construction that the map $\lambda : Q \to L$ such that $\lambda(q_0 \cdot \tau) = \vec{w}(\tau)$ for all $\tau \in T$ satisfies $\lambda(q_0 \cdot \tau) \simeq_j \lambda(q_0 \cdot \tau j)$ and $\lambda(q_0 \cdot \tau j) \notin S_j$ for $\tau j \in T$ and $j \in \{1,\ldots,n\}$, hence $\vec{w}(\varepsilon) \in SupK(L, \mathcal{S})$.

As $\mathcal{A}$ is a finite automaton, the projection of the language of $\mathcal{A}$ along ε is a regular language. In order to complete the proof, it suffices therefore to show that for any map $\lambda : Q \to L$ satisfying $\lambda(q_0 \cdot \tau) \simeq_j \lambda(q_0 \cdot \tau j)$ and $\lambda(q_0 \cdot \tau j) \notin S_j$ for all $\tau j \in T$, the vector $\vec{w} : T \to L$ defined with $\vec{w}(\tau) = \lambda(q_0 \cdot \tau)$ for all $\tau \in T$ may be written as a word $\vec{v}_1 \ldots \vec{v}_m$ recognized by $\mathcal{A}$. Given the construction of this automaton, it suffices to exhibit a sequence $\vec{v}_1 \ldots \vec{v}_m \in \mathcal{V}^*$ such that $\vec{w}(\tau) = \vec{v}_1(\tau) \ldots \vec{v}_m(\tau)$ for all $\tau \in T$. This is the contribution of the lemma 1 (proved in [6]). ■

Lemma 1: Let $\vec{w} : T \to \Sigma^*$ where T is a prefix-closed subset of $\{1,\ldots,n\}^*$ and $\vec{w}(\tau) \simeq_j \vec{w}(\tau j)$ for all $\tau j \in T$ with $j \in \{1,\ldots,n\}$. Then $\vec{w}(\tau) = \vec{v}_1(\tau) \ldots \vec{v}_m(\tau)$ for all $\tau \in T$ for some sequence of vectors $\vec{v}_k : T \to \Sigma \cup \{\varepsilon\}$ such that for all $\tau j \in T$, $(\vec{v}_k(\tau) \in \Sigma_j \vee \vec{v}_k(\tau j) \in \Sigma_j) \Rightarrow \vec{v}_k(\tau) = \vec{v}_k(\tau j)$.

Theorem 1 opens the way to the practical synthesis of supervisory control for concurrent opacity. The conditions for its application are examined further in section V.

V. CONCURRENT SECRETS WITH A REGULAR OPACITY CONTROL

We propose here conditions on concurrent secrets $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_n, S_n)\}$ ensuring that the maximal permissive opacity control $SupK(L, \mathcal{S})$ is the language of a finite automaton, that may effectively be constructed from finite automata accepting the language L and the secrets S_i. We examine first the case where the alphabets Σ_i form a chain for the inclusion, second the case where the secrets S_i form a chain for the inclusion, third the case where every secret S_i is saturated by any equivalence $\simeq_j$ such that $i \neq j$ (a set is *saturated* by an equivalence if it is a union of equivalence classes). We consider finally the combinations of the three cases for the different pairs (i, j).

Proposition 5: If the alphabets Σ_i form a chain for the inclusion, then the enabledness of all trajectories $w \in SupK(L, \mathcal{S})$ may be shown with a single type of proofs $\mathcal{T}_1$.

Proof: Given the chain $\Sigma_1 \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n$, we construct a type $\mathcal{T}_1 = (Q, \{1,\ldots,n\}, q_0, T)$ as follows. T is the set of strictly increasing sequences of numbers in $\{1,\ldots,n\}$ (T is drawn with solid arcs in Fig. 5), $Q = T$ and $q_0 = \varepsilon$. For any τ in T and $i \in \{1,\ldots,n\}$, $\tau \cdot i = \tau' i$ where τ' is the largest prefix of τ formed of integers strictly smaller than i (see again Fig. 5). As $(\simeq_j \circ \simeq_i) \subseteq \simeq_i$ for $i \leq j$, $\mathcal{T}_1$ conforms to Def. 9. Finally, for any $w \in SupK(L, \mathcal{S})$, by Prop. 1, there must exist a map $\lambda : Q \to L$, *i.e.* $\lambda : T \to L$, such that $\lambda(\varepsilon) = w$ and for all $\tau j \in T$, $\lambda(\tau) \simeq_j \lambda(\tau j) \wedge \lambda(\tau j) \notin S_j$. ■

Proposition 6: If the secrets S_i form a chain for the inclusion, then the enabledness of all trajectories $w \in SupK(L, \mathcal{S})$ may be shown with a single type of proofs $\mathcal{T}_2$.

Proof: Given the chain $S_1 \subseteq S_2 \subseteq \ldots \subseteq S_n$, we construct a type $\mathcal{T}_2 = (Q, \{1,\ldots,n\}, q_0, T)$ as follows. T is the set of strictly increasing sequences of numbers in $\{1,\ldots,n\}$ (T is drawn with solid arcs in Fig. 6), $Q = T$ and $q_0 = \varepsilon$. For any τ in T and $i \in \{1,\ldots,n\}$, $\tau \cdot i = \tau i$ if $\tau i \in T$ and $\tau \cdot i = \tau$ otherwise (see again Fig. 6). If $i \leq j$, then for any τj in T ($= Q$), and for any map $\lambda : Q \to L$, $\lambda(\tau j) \notin S_j \Rightarrow \lambda(\tau j \cdot i) \notin S_i$ since $S_i \subseteq S_j$. Therefore, $\mathcal{T}_2$ conforms to Def. 9, and the desired conclusion follows from Prop. 1. ■

Proposition 7: If for all distinct $i, j \in \{1,\ldots,n\}$, the secret S_i is saturated by the equivalence relation $\simeq_j$, then the enabledness of all trajectories $w \in SupK(L, \mathcal{S})$ may be shown with a type of proofs $\mathcal{T}_3$.

Proof: We construct a type $\mathcal{T}_3 = (Q, \{1,\ldots,n\}, q_0, T)$ as follows. T is the set of sequences in $\{1,\ldots,n\}^*$ with at most one occurrence of each number (T is drawn with solid arcs in Fig. 7), $Q = T$ and $q_0 = \varepsilon$. For any τ in T and $i \in \{1,\ldots,n\}$, $\tau \cdot i = \tau i$ if $\tau i \in T$ and $\tau \cdot i = \tau$ otherwise (see again Fig. 7). Let $\lambda : Q \to L$ be any map such that $\lambda(\tau) \simeq_j \lambda(\tau j) \wedge \lambda(\tau j) \notin S_j)$ whenever $\tau j \in T$. One may show by

induction on τ that $\lambda(\tau) \notin S_i$ for any $i \in \{1,\ldots,n\}$ occurring in τ. Indeed, if this property holds for τ, it must hold for τj because $\lambda(\tau) \simeq_j \lambda(\tau j)$ and $\simeq_j$ saturates S_i and $L \setminus S_i$ for all i occurring in τ. Therefore, $\mathcal{T}_3$ conforms to Def. 9, and the desired conclusion follows from Prop. 1. ■

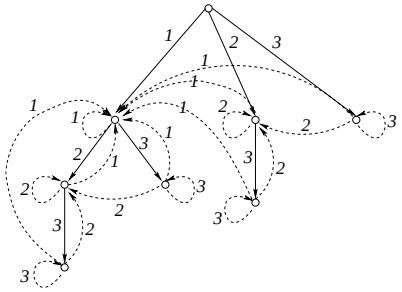

Fig. 5. $\mathcal{T}_1$ for $n = 3$

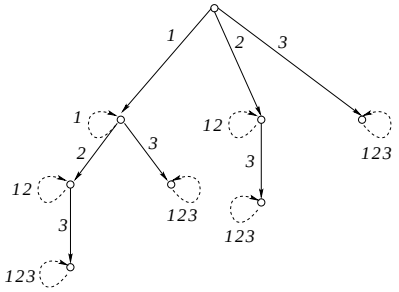

Fig. 6. $\mathcal{T}_2$ for $n = 3$

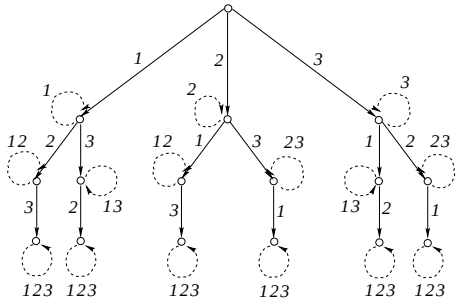

Fig. 7. $\mathcal{T}_3$ for $n = 3$

One can deal similarly with many other situations where $\Sigma_i \subseteq \Sigma_j$ or $S_i \subseteq S_j$ or $\simeq_i$ saturates S_j, or conversely with i and j, for all distinct $i, j \in \{1,\ldots,n\}$. For instance, let $n = 3$, and suppose $S_1 \subseteq S_2$, $\Sigma_3 \subseteq \Sigma_2$, and $\simeq_1$ saturates S_3. Then the enabledness of all $w \in SupK(L,\mathcal{S})$ may be proved using the type $\mathcal{T}_4$ (see Fig. 8).

Unfortunately, we cannot extend propositions 5,6, and 7 into a general proposition, for we do not know whether $SupK(L,\mathcal{S})$ is regular in three particular cases:
- $S_1 \subseteq S_2$, $\Sigma_2 \subseteq \Sigma_3$, and $\simeq_1$ saturates S_3,
- $S_1 \subseteq S_2$, $\simeq_2$ saturates S_3, and $\Sigma_3 \subseteq \Sigma_1$,
- $\simeq_1$ saturates S_2, $\simeq_2$ saturates S_3, and $\simeq_3$ saturates S_1.

The best we can do is therefore to propose an algorithm

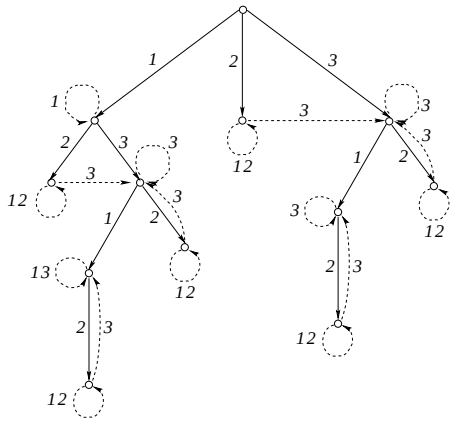

Fig. 8. $\mathcal{T}_4$

that constructs a unique type for all proofs of enabledness in all cases where this is possible. In this perspective, we introduce rewrite rules on labelled graphs. In each rule, one vertex of the left member is dropped and the edges that were incident to this vertex are redirected to other vertices. The vertices and edges present on both sides of a rule serve as an application context (indicated by the labels put on the concerned vertices). The rewrite rules are displayed in Fig. 9 (where $i \neq j$ and *sat* is an abbreviation for "saturates").

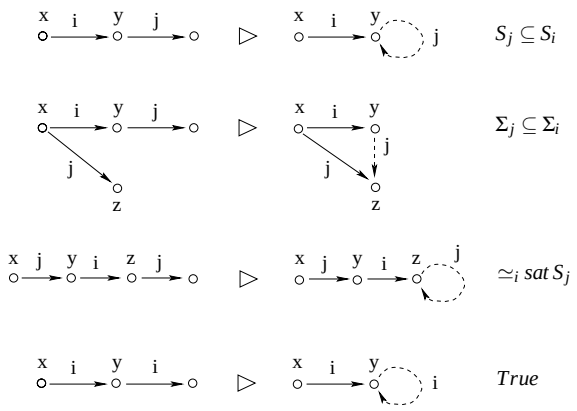

Fig. 9. Four rules

Proposition 8: Given $\mathcal{S} = \{(\Sigma_1,S_1),\ldots,(\Sigma_n,S_n)\}$, let $\mathcal{R}$ be the set of the rewrite rules that correspond to predicates true in $\mathcal{S}$. Whenever the complete n-ary tree rewrites to some finite graph, any such graph yields a uniform type $\mathcal{T}$ for proving the enabledness of all trajectories. The spanning tree of $\mathcal{T}$ is the subset of edges of the complete n-ary tree that have been preserved by the rewriting.

Proof: In view of Def. 9 it is enough to show, for each graph G on the right hand side of a rewrite rule (see Fig. 9), that any map $\lambda : \{x,y\} \to L$ or $\lambda : \{x,y,z\} \to L$ compatible with the rigid edges of G is compatible also with the dashed edge of G, where λ is compatible with $x \xrightarrow{i} y$ if $\lambda(x) \simeq_i \lambda(y)$ and $\lambda(y) \notin S_i$. Considering the predicates defining the application conditions of the rewrite rules, this verification is immediate. ■

When proposition 8 can be applied, the construction proposed in the proof of proposition 1 may be used to produce a finite automaton realizing the maximal permissive opacity control, but Prop. 8 is not immediately effective. We remedy now this deficiency.

Proposition 9: It is decidable whether some finite graph may be derived from the complete n-ary tree using the rules in $\mathcal{R}$ and such graphs can be computed when they exist.

Proof: Let $I = \{1,\dots,n\}$ and let $F \subseteq I^*$ be the set of all words ii, or ij, or jij such that $True$, or $(S_j \subseteq S_i \vee \Sigma_j \subseteq \Sigma_i)$, or $\simeq_i satS_j$, respectively. Define $T = I^* \setminus (I^*FI^*)$, then $\mathcal{R}$ produces finite graphs from the complete n-ary tree if and only if T is finite. The detailed argumentation and the construction of a type $\mathcal{T}$ with the spanning tree T may be found in [6]. ■

Example 4: Let $\Sigma = \{a,b,c\}$ and let L be a prefix-closed regular language over Σ. Define $\mathcal{S} = \{(\Sigma_1,S_1),\dots,(\Sigma_3,S_3)\}$ such that $\Sigma_1 = \{a,c\}$, $\Sigma_2 = \{b,c\}$, $\Sigma_3 = \{b\}$, and $S_i = S_i\Sigma^*$ for all $i \in \{1,\dots,3\}$. The construction sketched in the proof of proposition 9 yields the type $\mathcal{T}_4$ and the spanning tree T displayed in Fig. 8.

$SupK(L,\mathcal{S})$ may be computed by stages following the structure of T. One computes first $SupK(L,\mathcal{S}) \setminus S_3$, using the type that appears in $\mathcal{T}_4$ at the end of both paths 13 and 3. Next, one computes $SupK(L,\mathcal{S}) \setminus S_1$ from $SupK(L,\mathcal{S}) \setminus S_3$, using the type at the end of the path 1 in $\mathcal{T}_4$. Finally, one computes $SupK(L,\mathcal{S})$ from $SupK(L,\mathcal{S}) \setminus S_1$ and $SupK(L,\mathcal{S}) \setminus S_3$.

VI. CONCLUSION

We shall try first in this section to illustrate the possible applications of the work we have presented. Consider a computer system that provides services to n users $U1,\dots,Un$ with disjoint alphabets $\Sigma_1,\dots,\Sigma_n$. Let $L \subseteq \Sigma^*$ be the language of the system, where $\Sigma \supseteq \Sigma_i$ for all i. One wants to give every user Ui the guarantee that no coalition of other users can ever be sure that he has started working. The problem is therefore to enforce the opacity of the concurrent secret $\mathcal{S} = \{(\Sigma'_1,S_1),\dots,(\Sigma'_n,S_n)\}$ where for each i, $S_i = L \cap \Sigma^*\Sigma_i\Sigma^*$ and $\Sigma'_i = \cup_{j\neq i}\Sigma_j$. As $\simeq_j$ saturates S_i for $j \neq i$, one can construct a finite automaton accepting $SupK(L,\mathcal{S})$. We feel this example is typical of many practical security problems.

Some limitations of this work are voluntary, *e.g.* we restricted ourselves on purpose to regular languages and to regular control, but some other limitations could hopefully be lifted in continuations of this work. A list follows.

From the beginning of section IV, we worked with open secrets, *i.e.* secrets S_i such that $S_i\Sigma^* \subseteq S_i$. The goal was to make Def. 3 equivalent to the simpler definition Def. 5. Another way to obtain this equivalence is to impose on each secret S_i the following condition, where $\leq$ is the order prefix: $(\forall w \in L \setminus S_i)\ \pi_i(w) = u\sigma \Rightarrow (\exists v \in L \setminus S_i)\ (v \leq w \wedge \pi_i(v) = u)$. Such secrets may *e.g.* carry the information that some system process *is* in a critical section.

As regards the control objective, we focussed our efforts on opacity, but we did not take the deadlock freeness or the liveness of the controlled system into consideration and this is a shortcoming. Another valuable extension would be to work with boolean combinations of opacity predicates, *e.g.* if S_1 is opaque w.r.t. Σ_1 then S_2 is not opaque w.r.t. Σ_2.

We end with a few words on observability and controlability. On the side of the observation functions, we have restricted our attention to projections on subalphabets, but it would be more adequate to accomodate also all alphabetic morphisms. As regards control, we dealt with all events as controlable events, but it would be more realistic to accomodate also uncontrolable events.

REFERENCES

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

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

[3] L. Mazaré, Using unification for opacity properties, *in Proc. of the Workshop on Issues in the Theory of Security (WITS'04)*, 2004.

[4] J.W. Bryans, M. Koutny, L. Mazaré and P.Y.A. Ryan, Opacity Generalised to Transition Systems, *in Proc. of the Workshop on Formal Aspects in Security and Trust (FAST 2005)*, 2005.

[5] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, *Pacific Journal of Mathematics*, vol. 5, 1955, pp. 285-309.

[6] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, Concurrent Secrets (full version), INRIA-RR 5771 (available from http://www.inria.fr/rrrt), 2005.

[7] J. Berstel, *Transductions and Context-Free Languages*, Teubner Verlag, 1978.

[8] A. Arnold and M. Nivat, Comportements de processus, *in Actes du Colloque AFCET "Les mathématiques de l'informatique"*, 1982, pp.35-68.

This research was supported by CATALYSIS, a program within CNRS/PAN cooperation framework