Discrete Event Dyn Syst (2007) 17:425–446
DOI 10.1007/s10626-007-0020-5

Concurrent Secrets

E. Badouel · M. Bednarczyk · A. Borzyszkowski · B. Caillaud · P. Darondeau

Received: 17 October 2006 / Accepted: 24 July 2007 /
Published online: 12 September 2007

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.

Keywords Supervisory control · Computer security · Opacity · Concurrency · Automata

The biographies and photos of the authors are not available.

E. Badouel · B. Caillaud · P. Darondeau (✉)
IRISA, campus de Beaulieu, 35042 Rennes Cedex, France
e-mail: darondeau@irisa.fr

E. Badouel
e-mail: ebadouel@irisa.fr

B. Caillaud
e-mail: bcaillau@irisa.fr

M. Bednarczyk
IPIPAN, Abrahama 18, 81-825 Sopot, Poland
e-mail: m.bednarczyk@ipipan.gda.pl

A. Borzyszkowski
Inst. of Math., Gdańsk University, Wita Stwosza 57, 80-952, Gdańsk, Poland
e-mail: a.borzyszkowski@math.univ.gda.pl

1 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 Ramadge and Wonham (1987a,b) yields a finite, nonblocking, and maximal permissive control of the plant enforcing this behaviour (when unobservable events are uncontrollable). Controller synthesis is a desirable complement to model checking, for it may cure 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 that can 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 represented as 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 information 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 Mazaré (2004) and studied further in Bryans et al. (2006). 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 Bryans et al. (2006) that anonymity problems and noninterference 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 Bryans et al. (2006) 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 Bryans et al. (2006), 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 smallest 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.

As a motivating example, consider a computer system that provides services to n clients $C1, \ldots, Cn$ with disjoint alphabets $\Sigma'_1, \ldots, \Sigma'_n$. Let $L \subseteq \Sigma^*$ be the language of the system, where $\Sigma'_i \subseteq \Sigma$ for all i. One wants to give every client the guarantee that no coalition of other users can ever be sure that he has started interacting with the system. For each i, let $S_i = L \cap \Sigma^* \Sigma'_i \Sigma^*$ and $\Sigma_i = \cup_{j \neq i} \Sigma'_j$. The problem is to compute the optimal control $K \subseteq L$ enforcing the opacity of the concurrent secret $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_n, S_n)\}$, which means that for any $w \in K$ and for any $i \in \{1, \ldots, n\}$, the projection $\pi_i(w)$ of w on Σ_i^* coincides with the projection $\pi_i(w')$ of some word $w' \in K \setminus S_i$. A finite automaton realizing the optimal control K may be computed for this case. Moreover, one can construct an equivalent decentralized controller.

The rest of the paper is organized as follows. The notation and the problem are introduced in Section 2. Section 3 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 4; conditions on these proof trees entailing the regularity of the optimal control are also stated there. Section 5 produces closely connected conditions on concurrent secrets. A complete example is presented in Section 6. Decentralized control is discussed in Section 7. Directions for further work are suggested in a short conclusion.

2 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^*). Throughout the paper, L is a nonempty prefix-closed language in $Rat(\Sigma^*)$ and for all $i \in \{1, \cdots, n\}$, $\Sigma_i \subseteq \Sigma$, $S_i \in Rat(\Sigma^*)$, and $S_i \subseteq L$. For $u, v \in L$ and $i \in \{1, \cdots, n\}$, let $u \simeq_i v$ if $\pi_i(u) = \pi_i(v)$ (thus $\simeq_i$ is an equivalence on 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 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. An event $\sigma \in \Sigma_i \cap \Sigma_j$ represents 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 in the subset $S_i \subseteq L$ is intended to be kept *secret* from user i. Therefore, we refer to a subset S_i as a *secret* and to a n-tuple of secrets $(S_1, \ldots, S_n)$ as a *concurrent secret*. In the terminology of Mazaré (2004) and Bryans et al. (2006), each 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 $\overline{S_i}$, nonopacity 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 $(S_1, \ldots, S_n)$ is *concurrently opaque* (w.r.t. L) if for all i, S_i is opaque w.r.t. π_i.

Checking concurrent opacity reduces obviously to 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* $(S_1, \ldots, S_n)$ *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 by an a or by 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 $(S_1, \ldots, S_n)$ is concurrently opaque. A control is any nonempty 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 $(S'_1, \ldots, S'_n)$ 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.

3 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 monotone 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 Tarski (1955) 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 Knaster–Tarski's fixpoint theorem. □

Remark 1 The condition $L' \subseteq SupK(L, \mathcal{S})$ is necessary *but not sufficient* for some nonempty 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 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 case under consideration $SupK(L, \mathcal{S})$ is regular. The rest of the section illustrates the converse situation.

3.1 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 $\Sigma_1 = \{c, f\}$, $S_1 = \Sigma^* afc\,(\Sigma \setminus \{c\})^* \cap L$ (this secret is opaque w.r.t. π_1 and L 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 $\Sigma_2 = \{b, e\}$, $S_2 = \Sigma^* deb\ (\Sigma \setminus \{b\})^* \cap L$ (this secret is opaque 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 in that it may be inferred from the projection fc on Σ_1^* that the run is in 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.

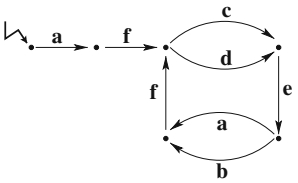
Let us now extend the concurrent secret into $\mathcal{S} = \{(\Sigma_1, S_1), (\Sigma_2, S_2), (\Sigma_3, S_3)\}$ with (Σ_1, S_1) and (Σ_2, S_2) as above, $\Sigma_3 = \emptyset$ and $S_3 = L \setminus (\Sigma^* c \Sigma^*)$. Then, the closure ordinal of $K(\bullet, \mathcal{S})$ increases from ω to $\omega + 1$. To see this observe that, since Σ_3 is empty, the secret S_3 is opaque w.r.t. any language $L' \subseteq L$ containing at least one word which contains 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 from 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.

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

Fig. 1 An automaton



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 nonregular language $\cup_{n\in\mathbb{N}}\ (ax)^n\,(\varepsilon + ab)\,(yb)^n$. A detailed proof of this fact may be found in Appendix.

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.

4 Control enabling and ω-trees

This section serves as a bridge between the general problem and the practical solutions that we shall propose in specific cases. From now on, $S_i\Sigma^* \cap L \subseteq S_i$ is imposed on all sets S_i in $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_n, S_n)\}$. The assumption that secrets are suffix-closed is motivated by its convenience (if not its necessity) for enforcing opacity with finite control. Although this assumption was not satisfied in the examples from Sections 3.1 and 3.2, 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 5.

Definition 5 (modified form of 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 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 Definition 3 and Definition 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 *control 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 control 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 as above 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 $q \in Q$.

Remark 2 The notation $q \cdot u$ used above for the next state function $\delta(q, u)$ is customary in the algebraic theory of (deterministic and complete) automata, where symbols τ are interpreted as mappings $M_\tau : Q \to Q$. These mappings generate a semigroup, in which $(M_u\, M_\tau)(q) = M_\tau(M_u(q)) = \delta(\delta(q, u), \tau)$. This notation, found e.g. in Berstel (1978); Eilenberg (1974); Ginzburg (1968), will be used frequently in the sequel.

The idea behind Definition 8 is that proof trees contain bounded information up to the choice of a bounded number of words in L.

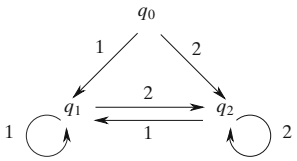


Fig. 2 A finite pattern for proofs of control enabledness

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 $w \in SupK(L, \mathcal{S})$.

The existence of proof patterns (Definition 8) does not always entail the existence of a regular control, which is our ultimate goal. One reason is that, given L, $\mathcal{S}$ and $(Q, \{1, \ldots, n\}, q_0)$, the set of 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 Berstel (1978) 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). Another reason 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 Definition 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.

Note that in both Examples 2 and 3, $SupK(L, \mathcal{S}) = \Sigma^*$, and proofs of control enabledness may be obtained for all $w \in \Sigma^*$ by labelling the finite proof pattern shown in Fig. 4. In view of the above, one may concentrate on restricted proof patterns as follows.

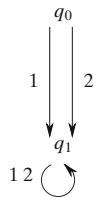


Fig. 3 A proof pattern for Example 3

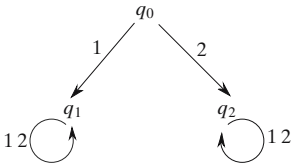


Fig. 4 A proof pattern for Examples 2 and 3

Definition 9 A *type* (of proof of control 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 Definition 8).

The set T in Definition 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 control 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 control 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 Arnold and Nivat (1982) 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 $\mathcal{A} = (\mathcal{Q}, \mathcal{V}, \vec{q}_0)$ defined as follows:

- The set of states $\mathcal{Q}$ is a set of T-vectors of states of automata $\mathcal{A}_\tau$ $(\tau \in T)$,
- For each $\tau \in T$, $\vec{q}_0(\tau)$ is the initial state of $\mathcal{A}_\tau$,
- For all $\vec{q} \in \mathcal{Q}$ and $\tau \in T$, $\vec{q}(\tau)$ is a state of $\mathcal{A}_\tau$,
- For all $\vec{q} \in \mathcal{Q}$, $\vec{v} \in \mathcal{V}$ 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 (see hereafter). □

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 $\tau j \in T$, $(\vec{v}_k(\tau) \in \Sigma_j \vee \vec{v}_k(\tau j) \in \Sigma_j) \Rightarrow \vec{v}_k(\tau) = \vec{v}_k(\tau j)$.*

Proof Let $\mathcal{E}$ be the set of all pairs (τ, i) such that $\tau \in T$ and $0 < i \leq |\vec{w}(\tau)|$. Let $<$ be the partial order on $\mathcal{E}$ defined with $(\tau, i) < (\tau', i')$ if $\tau = \tau'$ and $i < i'$. For each $j \in \{1, \ldots, n\}$, let $(\tau, i) \Vdash_j (\tau j, k)$ if $\pi_j(\vec{w}(\tau)[i])) = \pi_j(\vec{w}(\tau j)[k]))$, $\vec{w}(\tau)(i) = \vec{w}(\tau j)(k)$, and this letter is in Σ_j. Let $\equiv$ denote the equivalence on $\mathcal{E}$ generated from the union of the relations $\Vdash_j$. We claim that this equivalence does not intersect and is compatible with the partial order $<$. Let us establish this double claim.

1. Suppose for a contradiction that $(\tau, i) < (\tau, i')$ and $(\tau, i) \equiv (\tau, i')$. Then, by definition of $\equiv$ and the relations $\Vdash_j$, the words $\vec{w}(\tau)[i]$ and $\vec{w}(\tau)[i']$ end with a common letter $\vec{w}(\tau)(i) = \vec{w}(\tau)(i')$, and this letter occurs the same number of times in both words. As $i < i'$, this is clearly not possible.
2. Suppose for a contradiction that $(\tau, i) < (\tau, i')$ and $(\tau', j) < (\tau', j')$ while $(\tau, i) \equiv (\tau', j')$ and $(\tau, i') \equiv (\tau', j)$. Then, by definition of $\equiv$ and the relations $\Vdash_j$, $\vec{w}(\tau)(i) = \vec{w}(\tau')(j')$ and this letter σ occurs the same number of times in both words $\vec{w}(\tau)[i]$ and $\vec{w}(\tau')[j']$. In the same way, $\vec{w}(\tau)(i') = \vec{w}(\tau')(j)$ and this letter σ' occurs the same number of times in both words $\vec{w}(\tau)[i']$ and $\vec{w}(\tau')[j]$. Since $i < i'$ and $j < j'$, it follows that σ and σ' are different letters (see Fig. 5).

Now let $\tau = \rho x_1 \ldots x_k$ and $\tau' = \rho y_1 \ldots y_l$ where ρ is the longest common prefix of τ and τ' and $x_h, y_h \in \{1, \ldots, n\}$. Then by definition of $\equiv$ and the relations $\Vdash_j$, $(\tau, i) \equiv (\tau', j')$ and $(\tau, i') \equiv (\tau', j)$ entail that σ and σ' belong jointly to all the alphabets Σ_{x_h} $(1 \leq h \leq k)$ and Σ_{y_h} $(1 \leq h \leq l)$. On the other hand, by the part 1 of the proof, $(\tau, i) \equiv (\tau', j')$ entails that necessarily $\vec{w}(\tau)[i] \Vdash_{x_k}^{-1} \circ \ldots \circ \Vdash_{x_1}^{-1} \circ \Vdash_{y_1} \circ \ldots \circ \Vdash_{y_l} \vec{w}(\tau')[j']$. Therefore the words $\vec{w}(\tau)[i]$ and $\vec{w}(\tau')[j']$ must have the same number of occurrences of the letter σ', which is obviously not possible.

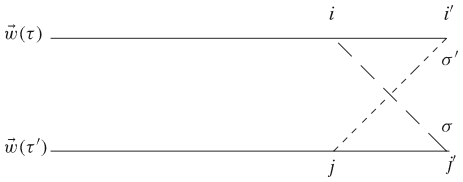


Fig. 5 Since $i < i'$ and $j < j'$, it follows that σ and σ' are different letters

Let $\mathcal{C} = (\mathcal{E}/\equiv)$. Since $<$ is compatible and does not intersect with $\equiv$, the binary relation $(< \cup \equiv)^*/\equiv$ is a strict partial order on $\mathcal{C}$. Let $C_1 \dots C_m$ be an enumeration of $\mathcal{C}$ compatible with this order. Each equivalence class $C \in \mathcal{C}$ induces naturally a vector $\vec{v} \in \mathcal{V}$, *viz.* $\vec{v}(\tau) = \vec{w}(\tau)(i)$ if $(\tau, i) \in C$ for some i, or ε otherwise. Let $\vec{v}_1 \dots \vec{v}_m$ be the vectors associated with $C_1 \dots C_m$, respectively. Then for any $\tau \in T$, $\vec{w}(\tau) = \vec{v}_1(\tau) \dots \vec{v}_m(\tau)$ as desired. □

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 5.

5 Concurrent secrets with a regular opacity control

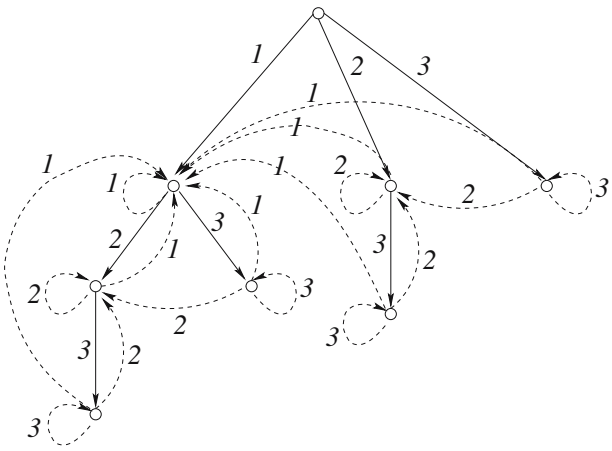
First, let us recall a definition.

Definition 10 Let $\simeq$ be an equivalence relation on a set X and let S be a subset of X. S is *saturated for* $\simeq$ if $x \in S \wedge x \simeq x' \Rightarrow x' \in S$ for any $x' \in X$.

A saturated set is thus a union of equivalence classes. In case where $X \subseteq \Sigma^*$ and $\simeq$ is $\simeq_j$, $S \subseteq X$ is saturated for $\simeq$ if and only if, for any word $x \in X$, one can infer whether $x \in S$ from its projection $\pi_j(x)$ on Σ_j^*.

We propose here conditions on concurrent secrets $\mathcal{S} = \{(\Sigma_1, S_1), \dots, (\Sigma_n, S_n)\}$ ensuring that the maximal permissive opacity control $SupK(L, \mathcal{S})$ is the language of a finite automaton, that may be effectively 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 $i \neq j$. We consider finally the combinations of the three cases for the different pairs (i, j).

Fig. 6 $\mathcal{T}_1$ for $n = 3$



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

Proof Given $\Sigma_1 \subseteq \Sigma_2 \subseteq \ldots \subseteq \Sigma_n$, 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. 6), $Q = T$ 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. 6). As $(\simeq_j \circ \simeq_i) \subseteq \simeq_i$ for $i \leq j$, $\mathcal{T}_1$ conforms to Definition 9. Finally, for any $w \in SupK(L, \mathcal{S})$, by Proposition 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 control enabledness of all trajectories $w \in SupK(L, \mathcal{S})$ may be shown with a single type of proofs $\mathcal{T}_2$.*

Proof Given $S_1 \subseteq S_2 \subseteq \ldots \subseteq S_n$, 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. 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). 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 Definition 9, and the desired conclusion follows from Proposition 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 control 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

Fig. 7 $\mathcal{T}_2$ for $n = 3$

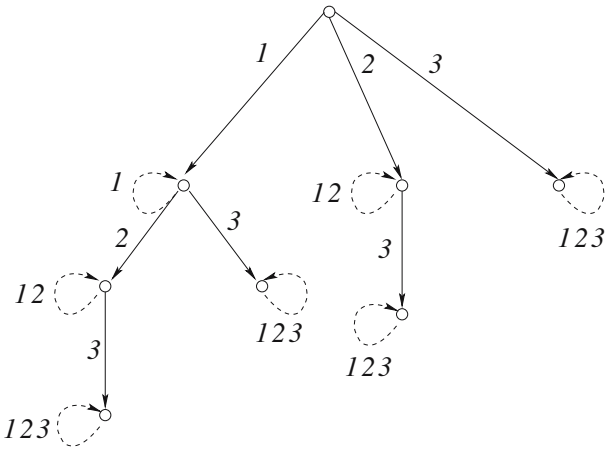
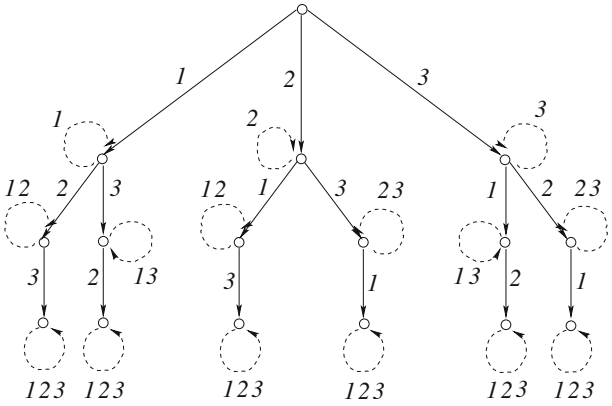


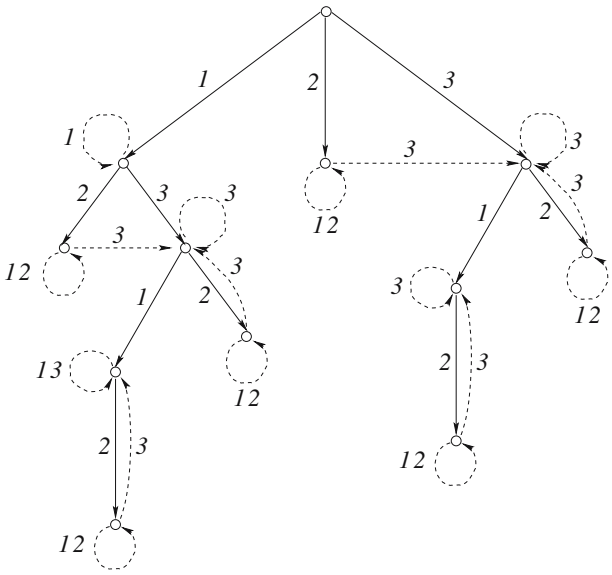
Fig. 8 $\mathcal{T}_3$ for $n = 3$



solid arcs in Fig. 8), $Q = T$ 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. 8). 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 Definition 9, and the desired conclusion follows from Proposition 1. □

Example 4 Proposition 7 applies to the concurrent secret $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_n, S_n)\}$ from Section 1, where $S_i = L \cap \Sigma^* \Sigma_i' \Sigma^*$ and $w \simeq_i w'$ if w and w' have the same projection on $(\cup_{j \neq i} \Sigma_j')^*$. Indeed, for $i \neq j$, $w \simeq_j w'$ entails that w and w' have the

Fig. 9 $\mathcal{T}_4$



same projection on $(\Sigma_i')^*$. One can construct for this case a finite automaton that accepts $SupK(L, \mathcal{S})$.

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 control enabledness of all $w \in SupK(L, \mathcal{S})$ may be proved using the type $\mathcal{T}_4$ (see Fig. 9).

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:

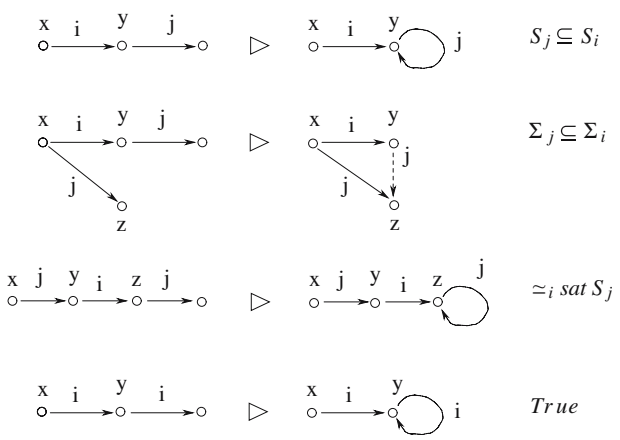
- $S_1 \subseteq S_2$, $\Sigma_2 \subseteq \Sigma_3$, and $\simeq_1$ saturates S_3,
- $S_1 \subseteq S_2$, $\simeq_2$ saturates S_3, and $\Sigma_3 \subseteq \Sigma_1$,
- $\simeq_1$ saturates S_2, $\simeq_2$ saturates S_3, and $\simeq_3$ saturates S_1.

The best we can do is therefore to propose an algorithm that constructs a unique type for all proofs of control 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. 10 (where $i \neq j$ and *sat* is an abbreviation for "saturates").

Proposition 8 *Given $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_n, S_n)\}$, let $\mathcal{R}$ be the set of 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 control 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 Definition 9 it is enough to show, for each graph G on the right hand side of a rewrite rule (see Fig. 10), 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

Fig. 10 Four rules



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 Proposition 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 As a preliminary remark, note that the rewrite rules in $\mathcal{R}$ are not necessarily confluent (i.e. $A \rhd B$ and $A \rhd C$ do not always entail that $B \rhd^* D$ and $C \rhd^* D$ for some D), hence the finite graph we compute is just one among several possible types of proofs of control enabledness.

Let $I = \{1, \ldots, n\}$ and let $F \subseteq I^*$ be the set of all words ii, or ij, or iji such that $True$, or $(S_j \subseteq S_i \vee \Sigma_j \subseteq \Sigma_i)$, or $\simeq_i sat S_j$, respectively. If the words in F are considered as forbidden factors for words in I^*, the remaining words form a regular language $T = I^* \setminus (I^* F I^*)$.

If T is infinite, the rewrite system $\mathcal{R}$ cannot terminate on the complete n-ary tree and it cannot produce any finite graph. If T is finite, let $(Q, \{1, \ldots, n\}, q_0)$ be the partial automaton defined with $Q = T$, $q_0 = \varepsilon$, and $\tau \cdot i = \tau i$ for τi in T.

To obtain a type of proof of control enabledness $(Q, \{1, \ldots, n\}, q_0, T)$ conforming Definition 9, it now suffices to complete the partial automaton $(Q, \{1, \ldots, n\}, q_0)$ as follows: for all words τ in T, and by increasing lengths of words τ,

- set $\tau i \cdot j = \tau i$ if $\tau i \cdot j$ is undefined and $S_j \subseteq S_i$ or $\simeq_i sat\, S_j$ and $\tau = \tau' \cdot j$,
- set $\tau i \cdot j = \tau \cdot j$ if $\tau i \cdot j$ is still undefined and $\Sigma_j \subseteq \Sigma_i$. □

6 A complete example

Let $\Sigma = \{a, b, c\}$. For every letter $\sigma \in \Sigma$ and for every word $w \in \Sigma^*$, let $\#\sigma(w)$ count the occurrences of σ in w. Let L be the set of all words $w \in \Sigma^*$ such that $0 \leq \#c(v) - \#a(v) - \#b(v) \leq 3$ for every prefix v of w. In shorter form, $L = \Downarrow (0 \leq \#c - \#a - \#b \leq 3)$. This language is prefix closed and regular. Let the concurrent secret $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_3, S_3)\}$ where $\Sigma_1 = \{a, c\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{b\}$ and the secret sets S_i are the respective differences $L \setminus \complement S_i$ defined with

$$\complement S_1 = L \cap \Downarrow (\#c - \#a - \#b \leq 2),$$
$$\complement S_2 = L \cap \Downarrow (\#c - \#a \leq 2),$$
$$\complement S_3 = L \cap \Downarrow (\#a = 0).$$

Thus $S_i = S_i \Sigma^* \cap L$ for all i. Intuitively, (Σ_1, S_1) means that one should not be able to infer from the projection of the system's trajectory on Σ_1^* that $\#c(v) - \#a(v) - \#b(v) = 3$ for some prefix v of this trajectory. Similarly, secret S_2 is that $\#c(v) - \#a(v)$ has reached value 3, and secret S_3 is that a has been performed. Clearly, $S_1 \subseteq S_2$, $\Sigma_3 \subseteq \Sigma_2$, and $\simeq_1$ saturates S_3. Applying the construction sketched in the proof

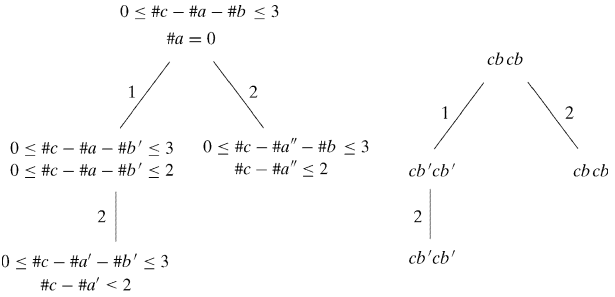


Fig. 11 Linear inequalities in the *left part* entail $\#b \le 2$. The tree in the *right part* proves that $cb\,cb \in SupK(L, \mathcal{S}) \setminus S_3$

of Proposition 9, one obtains therefore the type $\mathcal{T}_4$ and the spanning tree T displayed in Fig. 9.

It may be observed that each of the languages L, $\complement S_1$, $\complement S_2$, or $\complement S_3$ is specified by a set of linear inequalities, that must hold on the firing counts of letters a, b, and c for every word or prefix of word in the considered language. Using this observation, one may easily derive from the spanning tree T a nondeterministic Petri net generator for the optimal control $SupK(L, \mathcal{S})$. First, one attaches to each node of the spanning tree T a Petri net (with transitions a, b, c) as follows. The Petri net at the root of T generates the language L. For each path $\tau\, i$ in T, the Petri net at the target node of this path generates the language $\complement S_i$. Second, for each path $\tau\, i$, one merges the transitions named x in the nets respectively attached to paths τ and $\tau\, i$ if and only if $x \in \Sigma_i$. Last, one replaces all names of transitions by ε, except for merged transitions that include some transition of the net at the root node. The resulting net is large, some of its subnets are redundant (as we shall see), and it has the drawback to be nondeterministic.

To alleviate things, one may try logical reasoning instead of automated synthesis and proceed as follows. First, one attaches to each node of the spanning tree T the system of linear inequalities over $\#a$, $\#b$ and $\#c$ characteristic of the language L (for the root node) or $\complement S_i$ (for the target node of a path $\tau\, i$), using new names for variables $\#a$, $\#b$ and $\#c$ at each node. Second, one identifies names x_τ and $x_{\tau\, i}$ whenever $x \in \Sigma_i$. Third, one tries to eliminate all variables except those occurring at the root node. To illustrate the method, we compute $SupK(L, \mathcal{S})$ by stages following the structure of T. We proceed bottom up, beginning with the subtree of T reached by path 13 or 3. Decorating this subtree by linear inequalities yields the linear system in the left part of Fig. 11. A word $w \in \{a, b, c\}^*$ belongs to $SupK(L, \mathcal{S}) \setminus S_3$ if and only if this linear system (in the variables $\#a'$, $\#b'$, $\#a''$) is feasible for all parameters $\#a = \#a(v)$, $\#b = \#b(v)$, and $\#c = \#c(v)$ such that v is a prefix of w. Now, we do not need to

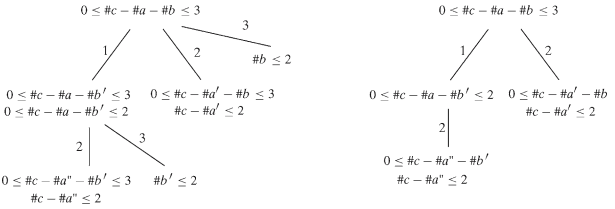


Fig. 12 Eliminating redundant equations from the *left part* yields the system in the *right part*

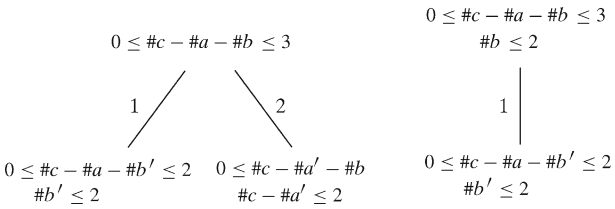


Fig. 13 Two other equivalent systems

compute $SupK(L, \mathcal{S}) \setminus S_3$, but just its projection on $\Sigma_3^* = b^*$. The linear inequalities in the left part of Fig. 11 entail $\#b \le 2$. The tree in the right part of Fig. 11 proves that $cbcb \in SupK(L, \mathcal{S}) \setminus S_3$. Therefore, the projection under consideration is the set $\Downarrow (\#b \le 2) = \{\varepsilon, b, bb\}$.

One may now replace the two subtrees (of T) at the end of paths $\tau = 13$ or 3 by inequalities $\#b_\tau \le 2$ and label the remaining nodes of T with linear inequalities. The result is shown in the left part of Fig. 12. Eliminating redundant equations yields the system in the right part of Fig. 12. We will show that the language defined by this system of linear constraints (on firing counts of letters) is equal to the language defined by the linear systems shown in Fig. 13.

The first stage is to show that for any word w in $\Downarrow (0 \le \#c - \#a - \#b' \le 2 \wedge \#b' \le 2)$, $w \simeq_2 w'$ for some word w' in $\Downarrow (\#c - 2 \le \#a'' \le \#c - \#b')$. Let u and v be two prefixes of w, with u shorter than v. Then $\#c(u) - 2 \le \#c(u) - \#b'(u) \le \#a(u) \le \#a(v) \le \#c(v) - \#b'(v)$. Therefore, one can define an increasing function $f(u)$ on prefixes u of w such that $\#c(u) - 2 \le f(u) \le \#c(u) - \#b'(u)$ for all u. A word w' may be produced by a transducer that takes w as input, erases all occurrences of a, and inserts occurrences of a'' whenever the value of f increases as desired. The second

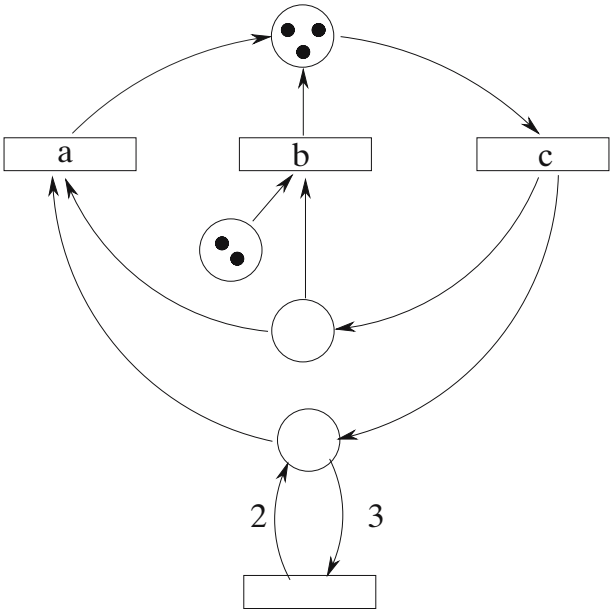


Fig. 14 $SupK(L, \mathcal{S})$ may be realized by a Petri net with transitions a, b, c plus one silent transition with high priority

stage is to show that both systems in Fig. 13 define the same language. This can be done by reproducing the above reasoning.

One can unfortunately not further simplify the rightmost system in Fig. 13. In fact, let w be a word in $\Downarrow (0 \leq \#c - \#a - \#b \leq 3 \ \wedge \ \#b \leq 2)$, then $w \simeq_1 w'$ for some word w' in $\Downarrow (0 \leq \#c - \#a - \#b' \leq 2 \ \wedge \ \#b' \leq 2)$ if and only if, for any prefixes u and v of w with u shorter than v, $(\#c(u) - \#a(u)) - (\#c(v) - \#a(v)) \leq 2$. This condition is clearly necessary. When it holds, $\#c(u) - \#a(u) - 2 \leq \#c(v) - \#a(v)$, hence one can define an increasing function $f(u)$ such that $\#c(u) - \#a(u) - 2 \leq f(u) \leq \#c(u) - \#a(u)$ for all u. This function may be used to monitor the insertion of occurrences of b' in w after all occurrences of b have been erased.

Summarizing the above, $SupK(L, \mathcal{S})$ is the set of words w such that, for any two prefixes u and v with u shorter than v, $0 \leq \#c(u) - \#a(u) - \#b(u) \leq 3$, $\#b(u) \leq 2$, and $\#c(u) - \#a(u) - 2 \leq \#c(v) - \#a(v)$. The last condition is not satisfied e.g. in the words $w = cccaaa$ or $w = cccbcaacaa$. This is not the language of any Petri net with set of transitions $\{a, b, c\}$. $SupK(L, \mathcal{S})$ may however be realized by a Petri net with transitions a, b, c plus one silent transition with high priority, as shown in Fig. 14.

7 Decentralized control

We did not strive until now to obtain decentralized solutions of the concurrent opacity control problem. Fortunately, such solutions come for free from the results established in Sections 4 and 5, as the following theorem states.

Theorem 2 *Let $\mathcal{S} = \{(\Sigma_1, S_1), \ldots, (\Sigma_n, S_n)\}$ be a concurrent secret upon $L \subseteq \Sigma^*$, then there exists prefix closed languages $K_i \subseteq \Sigma_i^*$ (for $i = 1 \ldots n$) such that $SupK(L, \mathcal{S}) = (\cap_i \pi_i^{-1}(K_i)) \cap L$. Moreover, if $SupK(L, \mathcal{S})$ is regular, then the K_i are regular.*

Proof By Proposition 4, $w \in SupK(L, \mathcal{S})$ if and only if there exists a proof of the control enabledness of w. In view of Definition 6 and again by Proposition 4, this holds if and only if, for all $i \in \{1, \ldots, n\}$, $w \simeq_i w_i$ for some $w_i \in SupK(L, \mathcal{S}) \setminus S_i$. Therefore, $SupK(L, \mathcal{S}) = (\cap_i \pi_i^{-1}(K_i)) \cap L$ where for all i, $K_i = \pi_i(SupK(L, \mathcal{S}) \setminus S_i)$. The second statement in the theorem is obvious. □

Theorem 2 states that concurrent opacity may always be enforced by decentralized control. This may help the users of distributed computer facilities to accept more readily the idea of opacity control. A user U_i mistrusting the other users U_j might be more inclined to trust a local controller K_i than a global controller, directly connected to all users.

8 Conclusion

The work we have presented is an attempt to approach the problem of concurrent opacity control from automata based Ramadge–Wonham framework. We have proposed sufficient conditions under which the optimal control (that always exists if some control exists) may be enforced by a finite automaton. We have also observed that the optimal control can always be enforced by a decentralized controller. These

results may be found encouraging, but the topic of concurrent opacity control is in an early stage and much is left to be done.

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 4, we worked with open secrets, i.e. secrets S_i such that $S_i\Sigma^* \subseteq S_i$. The goal was to make Definition 3 equivalent to the simpler definition Definition 5. In fact, a weaker condition on secrets S_i suffices to obtain this equivalence, namely the following condition, where $\leq$ is the prefix order:

$$(\forall w \in L \setminus S_i)\, \pi_i(w) = u\sigma \Rightarrow (\exists v \in L \setminus S_i)\, (v \leq w \ \wedge\ \pi_i(v) = u)$$

Such secrets may e.g. carry the information that some system process *is* in a critical section.

As regards the control objective, we focused 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 controllability. On the side of the observation functions, we have restricted our attention to projections on subalphabets, but it would be better to accommodate also all alphabetic morphisms. As regards control, we dealt with all events as controllable events, but it would be more realistic to accommodate also uncontrollable events.

Acknowledgements We would like to thank the reviewers for their help to improve an earlier version of this paper. This research was supported by CATALYSIS, a program within CNRS/PAN cooperation framework.

Appendix

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

In this appendix we give a detailed proof of the fact that the set $SupK(L, \mathcal{S})$ is not regular where language L and secret $\mathcal{S}$ are given as follows (recalled from Section 3.2). Let $\Sigma = \{a, b, x, y\}$ and L be the set of prefixes of words in $(ax)^*\,(\varepsilon + ab)\,(yb)^*$. Define $\mathcal{S} = \{(\Sigma_i, S_i) \mid 1 \leq i \leq 3\}$ where:

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. $\Sigma_3 = \{a, b, x, y\}$, $\complement S_3 = \varepsilon + a\Sigma^*$

In order to establish the claim, we will show that $SupK(L, \mathcal{S})$ is equal to the set L' of all prefixes of words in the nonregular language $\cup_{n\in\mathbb{N}}\ (ax)^n\,(\varepsilon + ab)\,(yb)^n$. Recalling that the subset of maximal words in a regular language is a regular language, it is then clear that $SupK(L, \mathcal{S})$ is not regular. We show that $SupK(L, \mathcal{S}) = L'$ by proving the reciprocal inclusion of the two sets.

For $i \in \{1, 2, 3\}$ and for all $w \in L$, define $w(\!|i|\!) = \left\{w' \in L' \mid w \simeq_i w' \ \wedge\ w' \in \complement S_i\right\}$. Since $SupK(L, \mathcal{S})$ is the maximal permissive control enforcing the opacity of $\mathcal{S}$, it

suffices to show that $\mathcal{S}$ is opaque w.r.t. L' in order to establish $L' \subseteq SupK(L, \mathcal{S})$. Therefore, as $L' \subseteq L$, it suffices to prove the assertion:

(1) For all $w \in L'$ and $i \in \{1, 2, 3\}$, the set $w(\!|i|\!)$ is nonempty.

The relation $SupK(L, \mathcal{S}) \subseteq L'$ may be reduced to a similar assertion. For $\tau \in \{1, 2, 3\}^*$ and for all $w \in L$, let $w(\!|\tau|\!)$ be inductively defined with $w(\!|\varepsilon|\!) = \{w\}$ and $w(\!|\tau i|\!) = \cup\{w'(\!|i|\!) \mid w' \in w(\!|\tau|\!)\}$. Whenever, for some $w \in L$, $w(\!|\tau|\!) = \emptyset$ for some τ, $w \notin SupK(L, \mathcal{S})$. Actually, if τ has length j, then w does not belong to the j^{th} language L_j encountered in the greatest fixpoint iteration for $SupK(L, \mathcal{S})$, *i.e.* $L_0 = L$, $L_1 = K(L_0, \mathcal{S})$, and so on. Therefore, $SupK(L, \mathcal{S}) \subseteq L'$ follows from the assertion:

(2) For all $w \in L \setminus L'$, the set $w(\!|\tau|\!)$ is empty for some $\tau \in \{1, 2, 3\}^*$.

Note that for any $w \in L$, if $w \in \varepsilon + a\Sigma^*$ then $w(\!|3|\!) = \{w\}$ else $w(\!|3|\!) = \emptyset$, hence $SupK(L, \mathcal{S})$ or any stronger opacity enforcing control cannot contain any word starting with a letter different from a (this was the point in introducing the secret S_3).

Let us now prove (1) and (2). For this purpose, one may classify the words $w \in L$ into categories as follows (singleton sets of words are represented as words).

1. $w = (ax)^n\, ab\, (yb)^m$:
   $w \in w(\!|1|\!)$, $w(\!|2|\!) = (ax)^n\, (yb)^m$, $w(\!|3|\!) = w$
2. $w = (ax)^n\, (yb)^m$:
   (a) $n, m \geq 1 \Rightarrow w(\!|1|\!) = (ax)^{n-1}\, ab\, (yb)^{m-1}$, $w \in w(\!|2|\!)$, $w(\!|3|\!) = w$
   (b) $m = 0 \Rightarrow w \in w(\!|1|\!) \cap w(\!|2|\!)$, $w(\!|3|\!) = w$
   (c) $n = 0, m \geq 1 \Rightarrow w(\!|1|\!) = \emptyset$, $w \in w(\!|2|\!)$, $w(\!|3|\!) = \emptyset$
3. $w = (ax)^n\, ab\, (yb)^m\, y$:
   $w(\!|2|\!) = (ax)^n\, (yb)^{m+1}$, $(ax)^n\, ab\, (yb)^m \in w(\!|1|\!)$, $w(\!|3|\!) = w$
4. $w = (ax)^n\, a$:
   $w \in w(\!|1|\!)$, $w(\!|2|\!) = (ax)^n$, $w(\!|3|\!) = w$
5. $w = (ax)^n\, (yb)^m\, y$:
   (a) $n, m \geq 1 \Rightarrow w(\!|1|\!) = (ax)^{n-1} ab\, (yb)^{m-1}$, $(ax)^n (yb)^{m+1} \in w(\!|2|\!)$, $w(\!|3|\!) = w$
   (b) $n \geq 1, m = 0 \Rightarrow w \in w(\!|1|\!)$, $(ax)^n\, yb \in w(\!|2|\!)$, $w(\!|3|\!) = w$
   (c) $n = 0, m \geq 1 \Rightarrow w(\!|1|\!) = \emptyset$, $(yb)^{m+1} \in w(\!|2|\!)$, $w(\!|3|\!) = \emptyset$
   (d) $n, m = 0 \Rightarrow \varepsilon \in w(\!|1|\!)$, $yb \in w(\!|2|\!)$, $w(\!|3|\!) = \emptyset$

The words $w' \in L'$ may be classified similarly into five categories as follows:

1. $w' = (ax)^n\, ab\, (yb)^m$ with $n \geq m$,
2. $w' = (ax)^n\, (yb)^m$ with $n \geq m$,
3. $w' = (ax)^n\, ab\, (yb)^m\, y$ with $n \geq m + 1$,
4. $w' = (ax)^n\, a$, or
5. $w' = (ax)^n\, (yb)^m\, y$ with $n \geq m + 1$.

It results from a comparison of the two classifications that the property (1) holds. Finally, the property (2) also holds, since all words $w \in L \setminus L'$ necessarily meet one the five cases below.

1. $w = (ax)^n\, ab\, (yb)^m$ with $n < m$: then $w(\!|(2 \cdot 1)^{n+1}|\!) = \emptyset$
2. $w = (ax)^n\, (yb)^m$ with $n < m$: then $w(\!|(1 \cdot 2)^n \cdot 1|\!) = \emptyset$
3. $w = (ax)^n\, ab\, (yb)^m\, y$ with $n \leq m$: then $w(\!|1 \cdot (2 \cdot 1)^{n+1}|\!) = \emptyset$
4. $w = (ax)^n\, (yb)^m\, y$ with $0 < n \leq m$: then $w(\!|(1 \cdot 2)^n \cdot 1|\!) = \emptyset$
5. $w = (yb)^m\, y$: then $w(\!|3|\!) = \emptyset$

We have thus proved that $SupK(L, \mathcal{S})$ is not a regular language.

References

Arnold A, Nivat M (1982) Comportements de processus. In: Actes du Colloque AFCET "Les mathématiques de l'informatique", pp 35–68

Berstel J (1978) Transductions and context-free languages. Teubner

Bryans JW, Koutny M, Mazaré L, Ryan PYA (2006) Opacity generalised to transition systems. In: revised selected papers of the 3rd international workshop on formal aspects in security and trust (FAST 2005). LNCS, vol 3866. Newcastle upon Tyne, UK, Springer Verlag, pp 81–95

Eilenberg S (1974) Automata, languages and machines, vol A. Academic, New York

Ginzburg A (1968) Algebraic theory of automata. Academic, New York

Mazaré L (2004) Using unification for opacity properties. In: Proc. of the 4th IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'04), Barcelona, Spain, pp 165–176

Ramadge PJ, Wonham WM (1987a) Supervisory control of a class of discrete event processes. SIAM J Control Optim 25:206–230

Ramadge PJ, Wonham WM (1987b) On the supremal controllable language of a given language. SIAM J Control Optim 25:637–659

Tarski A (1955) A lattice-theoretical fixpoint theorem and its applications. Pacific J Math 5:285–309