Complexité de l'algorithme de l'opacité dans les systèmes Workflows centrés sur les documents

Mohamadou Lamine Diouf
Université Cheikh Anta Diop de Dakar
Département de Mathématique et Informatique
Laboratoire d'Algèbre de Cryptologie et de Géométrie Algébrique et Applications
Email: mohamadoulamine@gmail.com

Sophie Pinchinat
Université de Rennes 1, France
Email: sophie.pinchinat@irisa.fr

***Abstract*—Une propriété d'un objet est dit opaque pour un observateur si celui-ci ne peut déduire que la propriété est satisfaite sur la base de l'observation qu'il a de cet objet. Dans ce papier, on se propose d'étudier la complexité du problème de l'opacité des artefacts d'un système à flots de tâches(système workflow). Notre propos est par conséquent de donner une formalisation optimale du problème de l'opacité dans ces systèmes afin que ce problème soit décidable. Nous donnons ensuite la complexité de l'algorithme qui a été proposé pour ces systèmes dans cet article.**

1. Introduction

La sécurité dans les systèmes à flots de tâches et les services web occupent une place de plus en plus importante dans le dévelopement d'application, car manipulant le plus souvent des informations confidentielles. Quelques exemples de ces processus métiers qu'on peut citer sont les bases de données médicales, les systémes d'administration electronique, les systémes de commerce electronique. Dans ce contexte, le développement de techniques assurant la confidentialité des informations traitées est primordial. Afin d'étudier de tels algorithmes, la propriété de sécurité centrale qui nous intéresse ici est l'opacité, pour laquelle on étudie sa complexité dans ce papier. C'est une propriété qui caractérise l'absence de flux d'information confidentielle d'un système vers un observateur de ce système qui peut être un service web en l'occurence.
Les systèmes considérés ici sont des systèmes distribués asynchrones dans lesquels la coordination des activités s'effectue par la transmission de documents structurés (à la XML) combinant structure logique, et données. Ces systèmes sont aussi appelés système à flots de tâches centrés sur les artefacts car ils mettent plus l'accent sur les documents échangés, appelés artefacts que sur l'enchainement des procédures ou des tâches. Les artefacts sont des documents qui combinent à la fois des données et des procédés. Ces procédés peuvent être des appels de services web comme dans le modèle ActiveXML [6].
Pour qu'il y'ait communication entre le système workflows centrés sur les artefacts et les services web, il faudrait publier des informations telle qu'une adresse de messagerie, la description du service web, $\cdots$. Il faudrait également envoyer de l'information extraite du système workflow vers le service web pour plusieurs raisons.
D'une part nous voulons éviter de surcharger un service par des informations qui ne lui sont pas utiles à la réalisation de la tâche qui lui incombe. D'autre part pour des raisons de confidentialité, et c'est ce point qui nous intéresse ici, certaines informations sensibles ne doivent pas être connues de tous. Et on doit pour cela pouvoir garantir qu'il n'y a pas de fuite d'information. Cette situation nécessite le développement d'algorithmes capables de détecter ces failles de sécurité.
La notion d'opacité introduite pour la première fois dans [2] est une formalisation de l'abilité du système à garder secrète certaines informations vis à vis de l'observateur. Elle a été longtemps étudié dans le contexte des systèmes à événements discrets. Elle a servi à modéliser des problèmes de sécurité et/ou de confidentialité pour des systèmes informatiques ou des protocoles, voir e.g. [7], [8]. Ce problème a été étudié dans le contexte des systèmes à flots de tâches centrés sur les artefacts pour la première fois dans [5].
L'hypothèse sous-jacente à la notion d'opacité est que l'observateur a une parfaite connaissance du système. Ceci pour des raisons de sûreté. On considére que l'artefact est donné par une suite de documents conformes à une grammaire d'arbres modélisant le système centré sur les données, gardant secret certaines informations confidentielles contre un service. On supposera aussi que l'observateur a une vue partielle de l'artefact. Cette vue partielle est donnée par une fonction d'observation définie de l'ensemble des documents vers un sous ensemble de documents observables par effacement de sous-documents non visibles appelé sorte.
Considérant une propriété P du système, le secret S consiste à l'ensemble des documents qui vérifient P. Et une propriété P est dit opaque pour le système si chaque fois qu'un

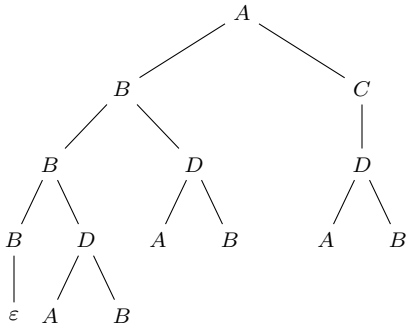
document d satisfaisant P est observable, alors il existe un autre document d' qui aura la même observation du point de vue de l'observateur et qui ne satisfait pas P. Et donc il ne pourra pas déduire que le document d satisfait la propriété P.

Dans ce papier, nous commencons par donner les notions basiques qui nous permettent d'introduire la notion d'artefact et la notion de conformité d'un artefact par rapport à une grammaire. Cette notion de grammaire est introduit dans la même section. La section 4 introduit la notion d'observation d'un artefact. Dans la section 5, on y définit la notion d'information confidentielle et on donne un modèle d'observateur du système. Ensuite la notion d'opacité est introduite dans la section 6 et l'algorithme qui permet de vérifier l'opacité d'un système est donné dans la section 7. On termine cette section 7 par la compléxité de l'algorithme proposé. La dernière section est réservé à la conclusion.

2. Notions basiques

2.1. Arbres à arité variable

L'abstraction la plus simple pour un artefact(suite de documents) est une suite d'arbres avec des informations associées à chaque nœud représenté par des ensembles attributs/valeurs décrivant sa nature ou catégorie syntaxique. Nous utiliserons le terme de forêt pour désigner une suite d'arbres et un arbre comme une forêt ayant une seule racine. Les nœuds d'une forêt ou d'un arbre sont étiquetés par des symboles d'un alphabet fini Ω. Le symbole ε désignera la chaine vide. l'exemple suivant illustre un arbre à arité variable.



dans ce schéma, l'alphabet fini Ω est donné, $\Omega = \{A, B, C, D\}$.
Dans un arbre à arité variable, le nombre de successeur d'un nœud n'est pas défini par son étiquette. L'arbre ci-dessus est composé d'un symbole racine A et d'une forêt dont les symboles racines sont B et C, comme sous-arbres.

2.1.1. Relation de préfixe. Soit $\mathcal{N}^*$ l'ensemble de toutes les suites d'entiers positifs non nuls.
La relation de préfixe dans $\mathcal{N}^*$, noté $\preceq$, est définie comme suit: $u \preceq v$ si $uw = v$ pour un $w \in \mathcal{N}^*$. Étant donné un sous-ensemble fini $D \subseteq \mathcal{N}^*$, nous disons que D est fermé pour l'ordre $\preceq$ si pour tous u et v de $\mathcal{N}^*$ tels que $u \preceq v$, $v \in D$ implique $u \in D$.

2.1.2. Arbres. L'arbre t étiqueté par Ω est une application:

$$t : D(t) \to \Omega$$

où $D(t) \subseteq \mathcal{N}^*$ est un ensemble non vide et fermé pour $\preceq$. qui satisfait:

$$(\forall j \geq 0)(uj \in D(t)) \Rightarrow (\forall i)(0 \leq i \leq j \Rightarrow ui \in D(t))$$

L'ensemble D(t) est aussi appelé l'ensemble des positions de t. La position de la racine t est représentée par ε. Nous écrivons $t(v) = \omega$, pour chaque $v \in D(t)$, pour indiquer que le symbole $\omega \in \Omega$ est associé au nœud dans la position v. Un arbre vide est tel que $D(t) = \emptyset$.Un arbre t sera dit fini si D(t) est fini.

Définition 2.1. Les ensembles d'arbres à arité variable et de forêts sur Ω sont les plus petits ensembles $T(\Omega)$ et $T(\Omega)^*$, respectifs, tels que:

- Toute suite d'arbre dans $T(\Omega)$ est dans $T(\Omega)^*$, incluant la forêt vide ε, et
- Pour tout $\omega \in \Omega$, et $t_1, \cdots, t_n \in T(\Omega)^*$, $\omega(t_1, \ldots, t_n) \in T(\Omega)$

Remarque 2.2. Lorsqu'on supprime la racine de l'arbre, on obtient une forêt. Et inversement, l'ajout d'une racine à une forêt donne un arbre. Une forêt $t_1 \cdots t_n$ peut être réécrite en $\sharp(t_1 \cdots t_n)$ en rajoutant le symbole $\sharp$ à l'ensemble Ω. On pourra supposer par la suite qu'une forêt est donnée par un arbre.

Lorsque l'ordre des successeurs d'un nœud donné n'est pas considéré, nous dirons que le nœud est non ordonné. Dans la suite nous ne considérons que des arbres non ordonnés à arité variable. La raison en est que dans la pratique, on rencontre souvent ces exigences dans les formalismes tels que RELAX NG [11].

Remarque 2.3. Les arbres définis ci-dessus doivent être reconnus conforme par des grammaires d'arbres régulières, i.e la partie droite des règles est décrite par des expressions régulières. La section suivante donne un rappel des notions d'expressions régulières et de langages réguliers.

2.2. Langages réguliers, expressions régulières

2.2.1. Définition. [9] Une expression régulière E sur un alphabet Ω et son langage associé L(E) sont récursivement définis de la manière suivante:

1. Les constantes $\emptyset$ et ε sont des expressions régulières et elles dénotent les langages $\{\varepsilon\}$ et $\emptyset$, respectivement, c'est-à-dire $L(\varepsilon) = \{\varepsilon\}$ et $L(\emptyset) = \emptyset$.
2. Si $a \in \Omega$, alors a est une expression régulière qui dénote le langage $L(a) = \{a\}$.
3. Si E et F sont des expressions régulières, alors E+F est une expression régulière(notée aussi par E—F) qui dénote l'union de L(E) et L(F), i.e., L(E +F) = L(E) + L(F).
4. Si E et F sont des expressions régulières, alors EF est une expression régulière qui dénote la concaténation

de L(E) et L(F), i.e., L(EF)=L(E)L(F). L'opérateur . peut être utilisé pour dénoter la concaténation.

5 Si E est une expression régulière, alors E^* est une expression régulière qui dénote la fermeture de L(E)(étoile de kleene), i.e.,$L(E^*) = (L(E))^*$.

6 Si E est une expression régulière, alors E^+ est une expression régulière qui dénote la fermeture positive de L(E), i.e., $L(E^+) = L(E)L(E^*)$

7 Si E est une expression régulière, alors E? est une expression régulière qui dénote le langage L(E) qui de plus, accepte le mot vide, i.e., $L(E?) = L(E) \cup \{\varepsilon\}$.

L'expression $(AB)^*D^*$ est régulière alors qu'il n'existe pas d'expression régulière qui décrit le langage de tous les mots de la forme $(AB)^nD^n$ où n est un entier.
Dans la section suivante, Nous allons présenter les grammaires d'arbres qui modélisent les systèmes workflows centrés sur les données.

2.2.2. Grammaire d'arbres régulière et langages. [1]

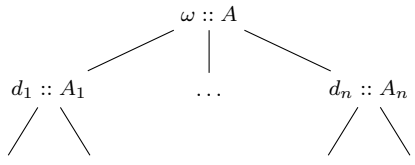
Définition 2.4. Une grammaire G est la donnée d'un triplet $(\Omega, \Xi, \mathcal{L})$ où:

- Ω: l'ensemble des symboles étiquettant les noeuds de l'arbre.
- Ξ : un ensemble fini de sortes ou types.
- $\mathcal{L} \subseteq \Xi^*$: un langage régulier associé à chaque symbole $\omega \in \Omega$ et sorte $A \in \Xi$ sur l'ensemble des sortes Ξ, noté $\mathcal{L}_{\omega,A}$

3. Conformité d'un artefact

La première propriété d'un artefact qu'il convient de vérifier est sa conformité à une certaine grammaire.
Un document d, représenté comme un arbre sera dit conforme à une grammaire G et est de sorte A, noté $G \vdash d :: A$ si chacun de ses successeurs d_i est de sorte A_i et la suite $(A_n)_n$ appartient à $\mathcal{L}_{\omega,A}$. L'ensemble des documents conformes à la grammaire est défini inductivement comme le plus petit ensemble tel que :

$$\Big(\forall i \in \{1,\dots,n\} \;\; G \vdash d_i :: A_i \;\; \wedge \;\; A_1 \cdots A_n \in \mathcal{L}_{\omega,A}\Big) \Rightarrow \;\; G \vdash \omega(d_1,\dots,d_n) :: A$$



$$(\omega, A) \longmapsto \mathcal{L}_{\omega,A} \in Rat(\Xi^*)$$

que l'on note $L(G) = \bigcup_{A\in\Xi} L(G,A)$ où $L(G,A) = \{t \mid G \vdash t :: A\}$. L'ensemble $L(G)$ est un langage d'arbre, définie comme une partie de $T(\Omega)$.

La vérification de la conformité d'un arbre peut-être implémentée par un automate à pile. Cet automate à pile sera déterministe si $s \neq s' \Rightarrow \mathcal{L}_{\omega,s} \cap \mathcal{L}_{\omega,s'} = \emptyset$, c'est-à-dire que la sorte d'un noeud est caractérisée par son étiquette et la sorte de ses successeurs. Lorsque cette propriété est vérifiée la grammaire est dite *déterministe*.

4. Abstraction d'un artefact

Dans la section précédente nous nous sommes intéressés à la structure logique d'un artefact (donnée par une grammaire). Nous allons maintenant nous intéresser aux données qu'il contient. On a vu que les artefacts ont un structure arborescente avec des informations attachées à chaque nœud. Ces informations attachées à un noeud correspondent à son étiquette $\omega \in \Omega$ et sont données par des paires attributs/valeurs. Les différents noeuds des documents sont étiquetés par l'alphabet Ω. Une abstraction (ou observation) consiste à supprimer toutes les informations associées à des sortes ou nœuds "non visibles" en supprimant par la même occasion les données qui leur correspondent. L'effet de cette transformation sur la structure du document peut se décrire comme suit.

Définition 4.1. La fonction de projection $p_{\Xi'} : T(\Omega) \to T(\Omega)^*$ associée à un sous-ensemble $\Xi' \subseteq \Xi$ (les sortes visibles) est donnée pour $t = \omega(t_1,\dots,t_n)$ par :

$$p_{\Xi'}(t) = \begin{cases} \omega(p_{\Xi'}(t_1)\dots p_{\Xi'}(t_n)) & \text{si} \quad t^G \in \Xi' \\ p_{\Xi'}(t_1)\cdots p_{\Xi'}(t_n) & \text{si} \quad t^G \notin \Xi' \end{cases}$$

qu'on peut interpréter de la manière suivante:

- Si la sorte d'un arbre est visible alors sa projection est un arbre.
- Sinon c'est une suite d'arbres, appelés aussi forêts.

Les sortes visibles contiennent toujours le symbole en racine : $\sharp \in \Xi'$.

On a le théorème fondamental suivant qui donne la relation qui existe entre le langage de la grammaire de départ et celle observée.

Théorème 4.2. ([5]) $L(G/\pi) = \pi(L(G))$.

Les sections qui suivent abordent les notions fondamentales pour traiter le problème de l'opacité.

5. Notion d'information confidentielle et modèle d'observateur

Définition 5.1. Une propriété secrète P sur les arbres de notre système $G = (\Omega, \Xi, \mathcal{L})$ est l'ensemble des arbres tels que s'ils vérifient la propriété alors constituent les informations confidentielles qu'on souhaite préserver d'un observateur.

On dit qu'un arbre t vérifie la propriété P lorsque t appartient à P. On note $t \in P$, et $t \notin P$ si l'objet ne vérifie pas la propriété. Donc P est un sous ensemble de G, noté $P \subseteq G$. Soit un observateur représenté par sa fonction d'observation $\phi : G \to O$ dans laquelle O est l'ensemble des observations possibles et $\phi(d)$ est l'information que l'observateur a de l'arbre t. Le but de cet observateur est d'inférer de l'information sur t, sachant que l'observateur a une parfaite connaissance du système : il connait la grammaire G décrivant les arbres, il sait par ailleurs comment sa fonction d'observation ϕ est construite et il est en mesure de déterminer l'ensemble $\phi^{-1}(o)$ des arbres qui donnent lieu à une observation donnée $o \in O$. De cette façon, la connaissance qu'il a d'un arbre $t \in G$ est donnée par l'ensemble $\phi^{-1}(\phi(t))$ des arbres donnant lieu à la même observation que celui-ci. Ainsi il sera capable de déterminer qu'un arbre t vérifie la propriété P lorsque l'ensemble $\phi^{-1}(\phi(t))$ est complétement inclus dans P. Ce que traduit la formule suivante:

$$\phi^{-1}(\phi(t)) \subseteq P$$

Lorsque cette relation est vérifiée par un certain arbre, on dit qu'il y a fuite d'information, ce qui constitue une faille de sécurité dans le cas où la propriété qui a été déduite est une information sensible qu'on souhaitait garder secrète. L'élément t vérifiant la relation ci-dessus est appelé un témoin de la fuite de l'information P.

6. Notion d'opacité

Définition 6.1. Une propriété secrète est opaque sur G pour un observateur si pour tout observation d'un arbre de G satisfaisant la propriété P, noté $t \models P$, il existe un autre arbre t', ayant la même observation. On peut l'exprimer formellement

$$\forall t \in P \; \exists t' \notin P \; \phi(t) = \phi(t')$$

On peut reformuler la formule ci-dessus de façon ensembliste.

Définition 6.2. La propriété P est opaque vis-à-vis de ϕ si et seulement si $\phi(P) \subseteq \phi(U \backslash P)$
Ces deux définitions indiquent que l'observateur possède une information partielle sur ce qui se passe réellement quand un arbre est produite par la grammaire G.

On peut introduire la relation d'équivalence θ sur les arbres pour modéliser cette notion d'information partielle de l'observateur. Sa sémantique est donnée par :
si un arbre engendré par la grammaire G vérifie P, alors il existe un autre arbre t' ne vérifiant pas P tel que $t \simeq_\theta t'$, alors l'observateur ne saurait discerner t et t' mais sait qu'au moins un arbre de la classe de $[t]_\theta$ est observable. Dans ce contexte l'observateur connaissant $[t]_\theta$, souhaiterait inférer un arbre t satisfaisant P, noté $t \models P$. En particulier il est sûr que $t \models P$ lorsque tout arbre de $[t]_\theta$ satisfait P.

Remarque 6.3. On simplifie dans ce papier la notion de secret en le considérant comme propriété des arbres. Dans ce travail préliminaire on ne cherche pas à identifier une syntaxe pour exprimer les secrets sur les documents.

7. Opacité dans les systèmes workflows

Etant donné l'ensemble C des arbres reconnnus par la grammaire $G = (\Omega, \Xi, \mathcal{L})$. Soit $\Xi' \subseteq \Xi$ un sous-ensemble de sortes visibles. Les observations partielles sont données par la fonction d'observation $\phi_{\Xi'}$, et pour chaque observateur, une propriété des arbres représentée par un automate d'arbres A_S qui est appelée secret et est notée S. Un automate d'arbre est une représentation des grammaires d'arbres. On s'interroge sur la possibilité d'un observateur d'inférer qu'un arbre du système se trouve dans le secret, à partir de la vue partielle qu'il a de cet arbre.

La figure suivante illustre les propos ci-dessus:

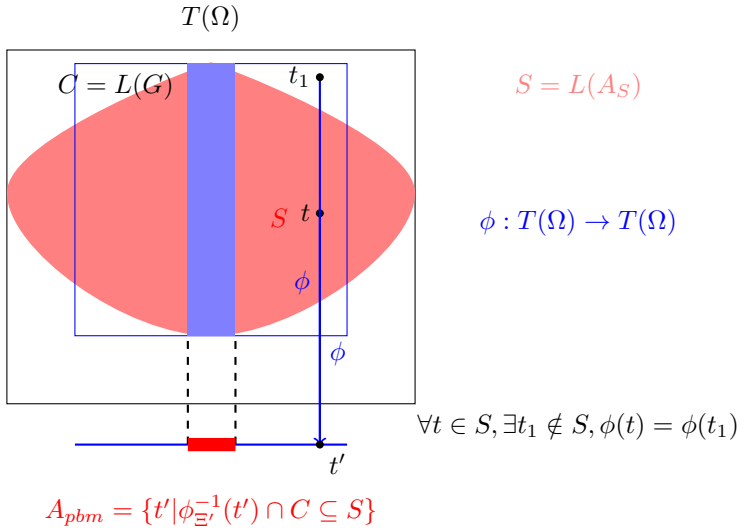


Figure 1. Opacité dans les systèmes workflows

L'ellipse rouge décrit l'ensemble des secrets $S = L(A_S)$, le carré bleu décrit notre système, $C = L(G)$ et le grand carré noir représente l'ensemble des arbres construit sur l'alphabet Ω. La bande bleue représente les arbres pour lesquels le système n'est pas opaque. Le trait bleu horizontal représente les arbres projetés, le petit rectangle rouge sur cette bande représente A_{pbm}.
A_{pbm} décrit la projection des arbres contre-exemples, i.e des arbres pour lesquels le système n'est pas opaque. Le secret S est opaque vis-à -vis de ϕ si et seulement si A_{pbm} est vide, c'est à dire que $\phi_{\Xi'}(S \cap C) \leq_c \phi_{\Xi'}(C \setminus S)$. L'algorithme du paragraphe suivant est proposé pour vérifier l'opacité d'un système donné en entrée.

7.1. Vérification de l'opacité

Dans cette section on donne un algorithme de vérification de l'opacité d'un système workflows centré sur

les données. Les entrées de cet algorithme sont le système workflows centrés sur les données reprsésentées par un automate d'arbres, la propriété secrète représentée aussi par un automate d'arbres et la fonction d'observation ϕ. En sortie, l'algorithme donne une réponse sur l'opacité du système donné en entrée. La compléxité de cet algorithme est étudiée par la suite.

Algorithme de vérification de l'opacité

Entrées: C, S : automate d'arbres finis déterministes, ϕ

1 Calculer $C \cap S$
2 Calculer $C \backslash S$
3 Résoudre le système d'équation pour les symboles de $\Xi \backslash \Xi'$ pour la grammaire associée à $C \cap S$. Soit $\phi_{\Xi'}(G)$.
4 Résoudre le système d'équation pour les symboles de $\Xi \backslash \Xi'$ pour la grammaire associée à $C \backslash S$. Soit $\phi_{\Xi'}(G^{'})$.
5 Calculer $F = \phi_{\Xi'}(G) \backslash \phi_{\Xi'}(G^{'})$ et vérifier la vacuité.

Théorème 7.1. L'algorithme proposé ci-dessus est EXPTIME-complet.

On va d'abord énoncer le résultat suivant qui nous facilitera la preuve du théorème ci-dessus.

Proposition 7.2. [4] Les arbres à arité variable non ordonnées sont clos par opération booléennes.

Sachant qu'également que la projection d'une grammaire d'arbres régulière selon ϕ est une grammaire régulière, on peut donner la preuve du théorème:

Démonstration 7.3.

1 Les automates d'arbres C et S étant déterministes alors la compléxité des calculs de la différence $C \backslash S$ and $C \cap S$ est polynomial en temps.
2 Le calcul de $\phi_{\Xi'}(X)$ est linéaire en la taille des symboles non-visibles $\Xi \backslash \Xi'$ donc polynomial aussi.
3 Tester la vacuité de l'automate $\phi_{\Xi'}(S \cap C) \backslash \phi_{\Xi'}(C \backslash S)$ est équivalent au problème de l'inclusion des automates d'arbres non déterministes $\phi_{\Xi'}(S \cap C) \subseteq \phi_{\Xi'}(C \backslash S)$ qui est exponentielle.

Pour la complétude, on utilisera le résultat suivant:

Théorème 7.4. ([10]) L'intersection non vide d'automates d'arbres et d'automates d'arbres déterministes est EXPTIME-complet.

L'idée est de montrer qu'on peut réduire le problème de l'opacité en temps polynomial au problème d'intersection non vide d'automates d'arbres qui est un problème EXPTIME-complet.

Soit $T(\Omega)^{\sharp}$ l'ensemble des arbres construits au dessus de $\Omega \cup \{\sharp\}$, où $\sharp$ est un nouveau symbole. Si $L \subseteq T(\Omega)$, on lui associe le langage $L^{\sharp} = \{\sharp(t) \mid t \in L\}$ appartenant à $T(\Omega)^{\sharp}$. Soit $\mathcal{A}_1$ un automate d'arbres déterministes et $\mathcal{A}_2$ un automate d'arbres non-déterministes. $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \emptyset$ ssi $L(\mathcal{A}_1)^{\sharp} \subseteq L(\mathcal{A}_2)^{\sharp c}$.

On considère le problème d'opacité associé à $C = L(\mathcal{A}_1)^{\sharp}$ et $S = L(\mathcal{A}_2)^{\sharp c}$ pour lequel l'ensemble des symboles visibles $\Xi' = \{\sharp\}$ est réduit au seul symbole $\sharp$.

Alors pour tout arbre $t \in T(\Omega)^{\sharp}$, on a $\phi^{-1}(t) = T(\Omega)^{\sharp}$. Par conséquent le système est opaque ssi $C \subseteq S$ i.e. $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \emptyset$.

8. Conclusion

Dans ce papier on a pu montrer que le problème de l'opacité des artefacts d'un système à flots de tâches (système workflow) est EXPTIME-complet. Dans la suite, nous souhaiterons intégrer l'algorithme proposé, dans l'outil Xduce développé par Hosoya et al [3] afin d'assurer la confidentialité de certaines informations contenus dans ces systèmes.

References

[1] A. Bruggemann-Klein and M. Murata and D. Wood, *Regular tree, regular hedge languages over unranked alphabets*, HKUST-TCSC-2001-05, 2001.

[2] D. Hughes and V. Shmatikov, *Hiding, Anonymity and Privacy: a Modular Approach*, Journal of Computer Security, 12, 1, 3-36, 2004.

[3] H. Hosoya and J. Vouillon and B. C. Pierce, *Regular expression types for XML*, ACM Trans. Program. Lang. Syst., 27(1), 46–90, 2005.

[4] H. Comon and M. Dauchet and R. Gilleron and F. Jacquemard and D. Lugiez and C. Loding and S. Tison and M. Tommasi, *Tree Automata Techniques and Applications*, http://tata.gforge.inria.fr, 2008.

[5] E. Badouel and M.L. Diouf, *Opacité des artefacts d'un système Workflow*, Revue ARIMA, 17, 177–196, 2014.

[6] S. Abiteboul and P. Bourhis and A. Galland and B. Marinoiu, *The AXML Artifact Model*, In Proc. 16th Intl. Symp. on Temporal Representation and Reasoning (TIME), 2009.

[7] F. Lin, *Opacity of discrete event systems and its applications*, Automatica, 47, 3, 2011, 496-500.

[8] J. Bryans and M. Koutny and L. Mazaré and P. Y. A. Ryan, *Opacity generalised to transition systems*, Int. J. Inf. Sec., 7, 6, 2008, 421-435.

[9] J. E. Hopcroft and R. Motwani and J. D. Ullman, *Introduction to Automata Theory Languages, and Computation*. Addison-Wesley Publishing Company, 2nd edition, 2001.

[10] Margus Veanes, *On Computational Complexity of Basic Decision Problems of Finite Tree Automata*, Uppsala University, Computing Science Department, January, 133, UPMAIL Technical Report, http://research.microsoft.com/apps/pubs/default.aspx?id=78382, 1997.

[11] J. Clark and M. Murata, *RELAX NG Specification*, http://www.oasis-open.org/committees/relax-ng/spec-20011203.html, 2001.