Opacité dans les systèmes workflows

Eric Badouel, Lamine Diouf

Opacité dans les systèmes workflows

Eric Badouel et Mohamadou Lamine Diouf

Inria Rennes-Bretagne Atlantique, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France
Université Cheikh Anta Diop (UCAD), Dakar, Sénégal
Equipe ALOCO du LIRIMA.
{eric.badouel,mohamadou.diouf}@inria.fr

RÉSUMÉ. Une propriété d'un objet est dite *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. Supposons qu'un certain de nombre de propriétés (appelées *secrets*) soient attachées à chaque intervenant d'un système, nous dirons alors que le système lui-même est opaque si chaque secret d'un observateur lui est opaque: il ne peut percer aucun des secrets qui lui ont été attachés. L'opacité a été étudiée préalablement dans le contexte des systèmes à événements discrets où différents jeux d'hypothèses ont pu être identifiés pour lesquels on pouvait d'une part décider de l'opacité d'un système et d'autre part développer des techniques pour diagnostiquer et/ou forcer l'opacité. Ce papier constitue, à notre connaissance, la première contribution au problème de l'opacité des artefacts d'un système à flots de tâches (système workflow). Notre propos est par conséquent de formaliser ce problème en dégageant les hypothèses qui doivent être posées sur ces systèmes pour que l'opacité soit décidable. Les techniques pour forcer l'opacité seront l'objet de travaux futurs.

ABSTRACT. A property (of an object) is opaque to an observer when he or she cannot deduce the property from its set of observations. If each observer is attached to a given set of properties (the so-called secrets), then the system is said to be opaque if each secret is opaque to the corresponding observer. Opacity has been studied in the context of discrete event dynamic systems where technique of control theory were designed to enforce opacity. To the best of our knowledge, this paper is the first attempt to formalize opacity of artifacts in data-centric workflow systems. We motivate this problem and give some assumptions that guarantee the decidability of opacity. The techniques for enforcing opacity are left as a direction for further work.

MOTS-CLÉS : Opacité, système à flots de tâches, artefact, documents structurés.

KEYWORDS : Opacity, data-centric workflow systems, artifact, structured documents.

1. Introduction

La propriété centrale qui nous intéresse dans ce travail est celle d'*opacité* [10, 3] dont nous donnons, dans le contexte de ce papier, la présentation suivante. Une propriété d'un objet est dite *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. Etant donné un ensemble U (pour univers) dont les éléments sont les objets en question, on peut représenter un observateur par sa fonction (d'observation) $\phi : U \rightarrow O$ et une propriété par l'ensemble $P \subseteq U$ des objets qui la vérifient. P est alors opaque vis-à-vis de ϕ si on a

$$\forall x \in P \;\; \exists y \notin P \quad \phi(x) = \phi(y)$$

c'est-à-dire qu'il n'est pas possible de déduire qu'un objet vérifie la propriété ($x \in P$) sur la base de son observation $\phi(x)$ puisqu'il serait toujours possible de trouver un autre élément y donnant lieu à la même observation ($\phi(x) = \phi(y)$) et qui néanmoins ne vérifie pas cette propriété ($y \notin P$). L'opacité de P vis-à-vis de ϕ peut s'exprimer de façon plus synthétique par : $\phi(P) \subseteq \phi(U \setminus P)$. Supposons qu'à chaque intervenant (identifié à sa fonction d'observation $\phi : U \rightarrow O$) soit attaché un certain de nombre de propriétés $S_1, \cdots S_n \subseteq U$ (appelées *secrets*). Nous exigeons alors qu'aucun secret ne puisse être dévoilé : le système est *opaque* si pour chaque observateur $\phi : U \rightarrow O$ et pour chacun des secrets $S \subseteq U$ attaché à cet observateur on a $\phi(S) \subseteq \phi(U \setminus S)$.

Le problème de l'opacité a été jusqu'à présent étudié dans le contexte des systèmes à événements discrets et a servi à modéliser des problèmes de sécurité et/ou de confidentialité pour des systèmes informatiques ou des protocoles, voir e.g. [3, 11]. Nous reviendrons dans la conclusion sur ces travaux pour envisager les suites au travail présenté ici en s'inspirant des techniques de monitoring [7] ou de contrôle [1] qui ont été développées pour assurer l'opacité.

Notre travail est la première tentative, à notre connaissance, d'appliquer ce concept d'opacité à des systèmes à flots de tâches centrés sur les données (*data-centric workflow systems* [5]). Il s'agit de systèmes distribués asynchrones dans lesquels la coordination des activités s'effectue par la transmission de documents structurés (à la XML) combinant structure logique, données et parfois des parties procédurales : il s'agit de formulaires contenant accessoirement des parties actives. Ces dernières peuvent s'exprimer par des règles sémantiques attachées à des noeuds qui permettent de dériver la valeur de certains attributs à partir de données contextuelles, à la manière d'un tableur, ou bien qui contraignent les interactions avec l'utilisateur pour le guider dans le renseignement de certains champs. Les éléments de notre univers U sont donc des documents structurés, appelés *artefacts*, qui servent de support à la réalisation d'une tâche : il s'agira par exemple du dossier médical d'un patient, d'un dossier administratif, d'un dossier pour le suivi d'une commande ... Au cours de ce traitement nous avons besoin d'invoquer un certain nombre de services auxquels nous serons ainsi amené à transmettre des informations extraites de ce dossier. Il y a plusieurs raisons qui conduisent à ne transmettre qu'une partie des informations contenues dans l'artefact. D'une part nous voulons éviter de surcharger un service par des informations qui ne 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 accédées par tous. Enfin les services invoqués ont généralement été définis antérieurement et de façon par conséquent indépendante du workflow qui les invoque ; celui-ci doit donc extraire de l'artefact un document conforme à ce que le service est préparé à recevoir. Ceci à des incidences

sur les structures grammaticales (statuant de la conformité d'un document) ainsi que sur les fonctions d'observations $\phi : U \rightarrow O$ qui décrivent le mécanisme d'extraction des informations transmises à un service donné.

L'objectif de ce travail préliminaire est d'introduire progressivement une formalisation des concepts précédents (structure grammaticale des documents, fonctions d'observation, secrets) qui assure que les différentes opérations nécessaires à la vérification de l'opacité puissent être implémentées de manière effective. Dans la conclusion nous esquissons des pistes pour la poursuite de ce travail.

2. Conformité d'un artefact

Les artefacts ont une structure arborescente qui reflète l'organisation logique des données. Chaque noeud est étiqueté par un élément $\omega \in \Omega$ qui précise sa nature (*catégorie syntaxique*). On dispose par ailleurs d'un ensemble de *sortes* Ξ. A chaque catégorie syntaxique $\omega \in \Omega$ et sorte $A \in \Xi$ on associe un langage régulier $\mathcal{L}_{\omega,A} \subseteq \Xi^*$ qui décrit l'ensemble des constituants (sortes des noeuds fils) associé à un tel noeud. Par exemple $\mathcal{L}_{\omega,A} = B^* + C$ exprime le fait qu'un noeud ω sera de sorte A si ses successeurs immédiats sont soit une liste (éventuellement vide) de noeuds de sorte B soit un unique noeud de sorte C. Remarquons que ces noeuds ont une arité variable : l'étiquette d'un noeud ne caractérise pas le nombre de ses successeurs.

Définition 2.1 *Une grammaire $G = (\Omega, \Xi, \mathcal{L})$ est la donnée de deux ensembles finis Ω et Ξ, et d'un langage régulier $\mathcal{L}_{\omega,A} \subseteq \Xi^*$ associé à chaque $\omega \in \Omega$ et $A \in \Xi$. On note $G \vdash t :: A$ pour signifier que t est un arbre conforme à la grammaire G et est de sorte A. On note $L(G, A) = \{t \mid G \vdash t :: A\}$ cet ensemble et $L(G) = \bigcup_{A \in \Xi} L(G, A)$ représente l'ensemble des arbres conformes à G. L'ensemble des arbres conformes à la grammaire est défini inductivement comme le plus petit ensemble tel que*

$$(\forall i \in \{1, \ldots, n\} \quad G \vdash t_i :: A_i \quad \wedge \quad A_1 \cdots A_n \in \mathcal{L}_{\omega,A}) \quad \Rightarrow \quad G \vdash \omega(t_1, \ldots, t_n) :: A$$

L'ensemble $T(\Omega)$ des arbres sur l'alphabet Ω est donné par $T(\Omega) = L(G_\Omega)$ où G_Ω désigne la grammaire sur Ω avec une unique sorte $\Xi = \{-\}$ et pour laquelle $\mathcal{L}_{\omega,*} = (-)^*$ pour tout $\omega \in \Omega$.

La vérification de la conformité d'un arbre peut être implémentée par un automate à pile ; [1] celui-ci sera déterministe si $A \neq A' \Rightarrow \mathcal{L}_{\omega,A} \cap \mathcal{L}_{\omega,A'} = \emptyset$, i.e. 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*. [2]

[1] Une grammaire sera présentée syntaxiquement sous la forme d'un ensemble de *productions* $A \rightarrow \omega\langle E \rangle$ où E est une expression régulière telle que $L(E) = \mathcal{L}_{\omega,A}$. Une grammaire est dite *locale* si $\Omega = \Xi$ et $\mathcal{L}_{\omega,A} = \emptyset$ si $\omega \neq A$. Une grammaire locale n'est donc rien d'autre qu'une grammaire algébrique étendue (les parties droites des productions sont des expressions régulières) sans symboles terminaux ; dans ce cas nous noterons simplement $A \rightarrow E$ la production associée à A.

[2] Un ensemble $R \subseteq T(\Omega)$ est dit *reconnaissable* s'il existe un *automate d'arbres*, constitué d'une grammaire $G = (\Omega, \Xi, \mathcal{L})$ et d'un ensemble de sortes, dites *finales*, $F \subseteq \Xi$, tel que $R = \cup_{A \in F} L(G, A)$. Tout ensemble reconnaissable peut être reconnu par un automate dont la grammaire sous-jacente est déterministe et l'ensemble des langages reconnaissables est clos de manière effective par les opérations booléennes [2, 6]. Dans ces diverses constructions l'alphabet Ω est fixé

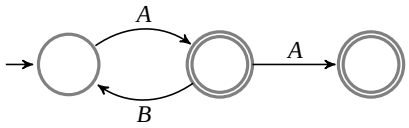
REMARQUE. — La relation d'ordre $G \leq G'$ définie par $\Xi \subseteq \Xi'$ et $\forall A \in \Xi \quad \forall \omega \in \Omega$ $\mathcal{L}_{\omega,A} \subseteq \mathcal{L}'_{\omega,A}$ (ce qui entraîne $L(G,A) \subseteq L(G',A)$ pour tout $A \in \Xi$) est une relation décidable. [3]

Nous considérons que l'ordre dans lequel les successeurs d'un noeud apparaissent n'est pas significatif. Pour prendre en compte cette contrainte on pourrait décider de se restreindre aux grammaires pour lesquelles les langages $\mathcal{L}_\omega$ sont clos par commutations (la plus petite congruence $\approx$ pour laquelle $\omega \cdot \omega' \approx \omega' \cdot \omega$). Mais ceci peut être beaucoup trop restrictif. Nous allons, pour cette raison, plutot considérer la relation de "conformité modulo commutations" définie comme suit : [4]

$$(\forall i \in \{1,\ldots,n\} \quad G \vdash_c t_i :: A_i \quad \wedge \quad A_1 \cdots A_n \in [\mathcal{L}_{\omega,A}]) \quad \Rightarrow \quad G \vdash_c \omega(t_1,\ldots,t_n) :: A$$

où $[L] = \{u \mid \exists v \in L \quad u \approx v\}$ est la cloture de L modulo commutations. Dans la mesure où la cloture par commutations d'un langage rationnel n'est généralement pas un langage rationnel, cette seconde approche est plus générale.

Exemple 2.2 *La cloture par permutation de $\mathcal{L} = (AB)^* \cdot A^*$ est l'ensemble des mots dont le nombre d'occurrences de A est supérieur ou égal au nombre d'occurrences de B. Ce dernier n'est pas rationnel (il possède un ensemble infini de résidus). Si on interprète l'expression rationnelle $(AB)^*A^*$ dans N^2 en identifiant les lettres A et B avec les deux générateurs $(1,0)$ et $(0,1)$,*
alors en cummulant les valeurs des étiquettes rencontrées le long d'un chemin de l'automate (associé à l'expression rationnelle)



celui-ci devient une machine qui permet d'engendrer l'ensemble $\{(x,y) \in N^2 \mid x \geq y\}$. Ce dernier est associé à la formule $F(x,y) \equiv (\exists z) \cdot x = y + z$. Néanmoins cet automate ne peut être utilisé comme un reconnaisseur. [5]

Les ensembles qui nous intéressent sont donc les images commutatives des langages rationnels, c'est-à-dire les parties rationnelles de N^k (où k est la taille de l'alphabet). Le fait que ces ensembles ne soient pas reconnaissables pourrait sembler constituer un obstacle pour la vérification de la conformité d'un document (modulo commutations). En fait il

---

mais les ensembles de sortes varient. On peut aussi observer que l'ensemble des arbres conformes à une grammaire est un ensemble reconnaissable. Ces ensembles sont les ensembles reconnaissables clos par sous-expressions (tout sous-arbre d'un arbre bien formé est lui-même bien formé).

[3] car l'inclusion des langages rationnels est décidable : $L \subseteq L' \Leftrightarrow L \cap \neg L' = \emptyset$ et les langages réguliers sont clos de manière effective par les opérations booléennes et la vacuité d'un langage rationnel est décidable (il suffit de vérifier s'il existe un état final accessible à partir de l'état initial dans un automate correspondant).

[4] Pour que la procédure de reconnaissance soit déterministe nous supposerons, non seulement que les différents langages $\mathcal{L}_{\omega,A}$ –lorsque A varie– sont disjoints, mais que leurs clôtures commutatives le sont : $A \neq A' \Rightarrow [\mathcal{L}_{\omega,A}] \cap [\mathcal{L}_{\omega,A'}] = \emptyset$.

[5] On peut définir, de façon générale, ce qu'est une partie reconnaissable d'un monoïde. Les parties rationnelles et reconnaissables coïncident dans le cas des monoïdes libres mais pas en général. Les parties reconnaissables d'un monoïde finiement engendré sont rationnelles. A titre d'exemple les parties reconnaissables de N^2 sont les unions finies de "grilles rectangulaires" de la forme $\{(x_0 + n \cdot x_1, y_0 + m \cdot y_1) \mid n,m \in N\}$ pour x_0, x_1, y_0 et y_1 quatre constantes entières et on peut vérifier que le quadrant $\{(x,y) \in N^2 \mid x \geq y\}$ ne peut s'écrire comme une union finie de telles grilles (il s'agit d'une partie rationnelle non reconnaissable).

n'en est rien, bien que ce processus va devoir suivre une procédure un peu plus compliquée que dans le cas de base. Les parties rationnelles de N^k coïncident avec les parties semi-linéaires [6] qui elle-mêmes sont les ensembles $\{(x_1, \cdots, x_n) \mid F(x_1, \cdots, x_n)\}$ définissables par des formules de Presburger. [7] Ces différentes correspondances sont effectives : on peut passer entre expressions régulières, ensembles semi-linéaires et formules de Presburger en utilisant des algorithmes bien établis (voir [8, 9, 12, 4]). L'intérêt de la logique de Presburger est sa décidabilité (au contraire de l'arithmétique dans son ensemble, qui est indécidable). Comme pour la décision de la logique du premier ordre, la méthode consiste à construire (par induction sur la structure de la formule) un automate qui reconnait l'ensemble des vecteurs vérifiant la formule (pour un codage particulier des vecteurs en des mots sur un certain alphabet). La validité de la formule se réduit en la non vacuité de l'automate correspondant. Nous utilisons cette construction pour vérifier comme suit la validité d'un arbre. Considérant un noeud étiqueté par un opérateur ω, on suppose inductivement que chacun de ses sous-arbes immédiats ait été jugé conforme avec une sorte déterminée, on collecte ces différentes sortes pour former un vecteur dont on vérifie qu'il satisfait la formule de Presburger associée à l'image commutative de $\mathcal{L}_\omega = \cup_{A \in \Xi} \mathcal{L}_{\omega,A}$.

Les ensembles semi-linéaires sont par ailleurs clos de manière effective par les opérations booléennes, ce qui fait que, de façon analogue à ce que nous avons exprimé dans la Remarque 2, nous pouvons décider si le langage d'une grammaire est à commutations près contenu dans celui d'une autre, i.e. la relation suivante est décidable

$$G \leq_c G' \quad \Leftrightarrow \quad \Big(\Xi \subseteq \Xi' \ \wedge \ (\forall A \in \Xi \ \ \forall \omega \in \Omega) \ \ [\mathcal{L}_{\omega,A}] \subseteq [\mathcal{L}'_{\omega,A}]\Big)$$

REMARQUE. — $G \leq_c G' \ \Rightarrow \ (\forall \omega \in \Omega) \ L_c(G, A) = \{t \mid G \vdash_c t :: A\} \subseteq L_c(G', A)$

3. Abstraction d'un artefact

Dans la section précédente nous avons décrit la structure logique d'un artefact, nous allons maintenant nous intéresser aux données qu'il contient. Les données attachées à un noeud sont présentées sous la forme d'une liste attributs/valeurs déterminée par la sorte du noeud. Nous ne nous intéresserons pas ici à la façon dont la valeur d'un attribut est calculée (via une règle sémantique qui invoque la valeur d'autres attributs ou le résultat d'une interaction avec un utilisateur). Nous pouvons voir un artefact comme des boîtes imbriquées avec pour chacune d'entre elles la liste attributs/valeurs qui lui est attachée. Une abstraction (ou observation) consiste à supprimer toutes les boîtes associées à des sortes "non visibles" en supprimant par la même occasion les données qui leur correspondent. L'effet de cette transformation sur la structure arborescente de l'artefact peut se décrire comme suit.

---

[6] Une partie linéaire est l'ensemble des vecteurs de la forme $v_0 + n_1 \cdot v_1 + \cdots + n_\ell \cdot v_\ell$ (une expression affine dans laquelle les vecteurs v_0 –l'origine– et $v_1, \ldots v_\ell$ –les vecteurs de base– sont fixés et les variables n_i représentent des entiers arbitraires). Une partie semi-linéaire est une union finie de parties linéaires.

[7] La logique de Presburger est le fragment de l'arithmétique dans lequel on exclut la multiplication, c'est-à-dire qu'il s'agit du calcul propositionnel avec quantification sur les entiers et avec l'addition. On peut bien sûr écrire la multiplication par une constante puisque par ex. $3x = x + x + x$ de la même manière on peut écrire des inéquations : par ex. $x \leq y$ est une abréviation de $(\exists z) \cdot \ y = x + z$.

Définition 3.1 *La fonction de projection $p_{\Xi'} : T(\Omega) \to T(\Omega)^*$ associée à un sous-ensemble $\Xi' \subseteq \Xi$ (les sortes visibles) est donnée par :*

$$p_{\Xi'}(\omega(t_1,\ldots,t_n)) = \begin{cases} \omega(p_{\Xi'}(t_1),\ldots,p_{\Xi'}(t_n)) & \textit{si} \quad \omega(t_1,\ldots,t_n) :: A \in \Xi \\ p_{\Xi'}(t_1)\cdots p_{\Xi'}(t_n) & \textit{sinon} \end{cases}$$

Remarquons que si la sorte d'un arbre est visible sa projection est un arbre sinon il s'agit d'une foret (une suite d'arbres). On souhaite trouver une grammaire sur Ω et Ξ' permettant de caractériser les arbres obtenus par projection. Pour cela nous scindons la grammaire d'origine en deux parties associées respectivement aux symboles visibles et aux symboles non visibles. Nous devons dans chacune de ces sous-grammaires faire la distinction entre les *symboles définis* (qui apparaissent en partie gauche des règles $A \to \omega\langle E_{\omega,A}\rangle$) et les *paramètres* (symboles qui n'apparaissent qu'en partie droite). Nous avons besoin pour cela d'introduire des grammaires avec paramètres.

Définition 3.2 *Une grammaire avec paramètres $G(\Omega,\mathcal{N},\mathcal{T},\mathcal{L})$ est constituée d'un ensemble fini $\mathcal{N}$ de symboles non terminaux, d'un ensemble fini $\mathcal{T}$ de paramètres, ou symboles terminaux (disjoints de $\mathcal{N}$), et d'une application $\mathcal{L}$ qui associe à tout $\omega \in \Omega$ et $X \in \mathcal{N}$ un langage régulier $\mathcal{L}_{\omega,X}$ sur $\Xi = \mathcal{N} \cup \mathcal{T}$ (ensemble des symboles de la grammaire).*

On associe une variable à chaque symbole grammatical en posant $\mathcal{X} = \{x_A \mid A \in \Xi\}$, on désigne par $T(\mathcal{N},\mathcal{T})$ l'ensemble des arbres $T(\mathcal{N} \cup \mathcal{X}_\mathcal{T})$ sur l'alphabet $\mathcal{N} \cup \mathcal{X}_\mathcal{T}$ où $\mathcal{X}_\mathcal{T} = \{x_A \mid A \in \Xi\}$ est l'ensemble des variables associées aux symboles terminaux. La relation de conformité, donnée inductivement par les règles $G \vdash_c x_A :: A$ pour $A \in \Xi$ et

$$(\forall i \in \{1,\ldots,n\} \quad G \vdash_c t_i :: A_i \quad \wedge \quad A_1\cdots A_n \in [\mathcal{L}_{\omega,A}]) \quad \Rightarrow \quad G \vdash_c \omega(t_1,\ldots,t_n) :: A$$

pour chaque $A \in \mathcal{N}$, stipule que $\{x_A = g^\dagger(x_A) \mid A \in \mathcal{N}\}$, où $g^\dagger : \mathcal{X}_\mathcal{N} \to \wp(T(\mathcal{N},\mathcal{T}))$ est définie par $g^\dagger(x_A) = L_c(G,A) = \{t \mid G \vdash_c t :: A\}$, est *la plus petite solution* du système d'équations $g = \{x_A = \{\omega(x_{A_1},\ldots,x_{A_n}) \mid A_1\cdots A_n \in [\mathcal{L}_{\omega,A}]\} \mid A \in \mathcal{N}\}$. Les éléments de $L_c(G,A)$ sont les arbres, équivalents par permutations de sous-arbres, à des arbres de dérivation –de sorte A– de la grammaire ; leurs *feuillages* sont définis comme l'image commutative $\mathcal{F}(G,A) = [\mathcal{L}(G,A)]$ de l'ensemble des mots constitués de symboles terminaux $\mathcal{L}(G,A) = \{u \in \mathcal{T}^* \mid A \to^* u\}$ dérivables dans G à partir du symbole non terminal $A \in \mathcal{N}$.

La restriction d'une grammaire $G = (\Omega,\Xi,\mathcal{L})$ à un sous alphabet $\Xi' \subseteq \Xi$ est la grammaire avec paramètres $G_{\Xi'} = (\Omega,\Xi',\Xi \setminus \Xi',\mathcal{L}')$ où $\mathcal{L}'$ est la restriction de $\mathcal{L}$ à Ξ'. On peut résoudre le système d'équations g (sans paramètres) associé à G par la méthode de substitutions en considérant les systèmes g_1 et g_2 associés respectivement à ses deux restrictions $G_1 = G_{\Xi'}$ et $G_2 = G_{\Xi\setminus\Xi'}$. Le principe de Bekič stipule que *(i)* pour $A' \in \Xi'$ on a $g^\dagger(A') = (g/g_2)^\dagger(A')$ où g/g_2 est le système résiduel formé des équations $x_{A'} = g_1(x_A)[g_2^\dagger(A)/A \in \Xi \setminus \Xi']$ obtenues à partir du système g_1 en substituant à chacune des variables $A \in \Xi \setminus \Xi'$ leur solution dans g_2, et *(ii)* $g^\dagger(A) = g_2^\dagger(A)[g^\dagger(A')/A' \in \Xi']$ pour $A' \in \Xi \setminus \Xi'$, c'est-à-dire qu'on substitue les solutions précédemment calculées (pour $A' \in \Xi \setminus \Xi'$) dans la solution du premier système.

La définition suivante introduit la grammaire projetée $p_{\Xi'}(G)$ dont le système d'équations va correspondre, à effacement près des symboles non visibles de $\Xi \setminus \Xi'$, au système résiduel d'équations g/g_2.

Définition 3.3 *La projection d'une grammaire $G = (\Omega,\Xi,\mathcal{L})$ à un sous alphabet $\Xi' \subseteq \Xi$ est la grammaire $p_{\Xi'}(G) = (\Omega,\Xi',\mathcal{L}')$ où $\mathcal{L}'(\omega,A') = \mathcal{L}(\omega,A')[\mathcal{F}(G_{\Xi\setminus\Xi'},A)/A \in \Xi \setminus$*

$\Xi']$, *c'est-à-dire qu'on substitue l'image commutative du langage algébrique* $\mathcal{L}(G_{\Xi\setminus\Xi'}, A)$ *(qui est donc un langage rationnel par le théorème de Parikh) à la variable non visible* $A \in \Xi \setminus \Xi'$ *dans la partie droite* $\mathcal{L}(\omega, A')$ *de la règle, dans la grammaire d'origine, associée au symbole visible* $A' \in \Xi'$.

REMARQUE. — Dans la pratique nous n'introduisons pas explicitement le langage régulier $\mathcal{F}(G_{\Xi\setminus\Xi'}, A)$ mais on exhibe une expression régulière qui le caractérise à commutations près. Pour cela on utilise le fait que la plus petite solution d'une equation $X = A(X) \cdot X + T$ dans laquelle la variable X n'apparait pas dans T est à commutations près donnée par l'expression régulière $A(T)^* \cdot T$

Exemple 3.4 *(grammaire locale) En utilisant* $CA + CCB + D \approx (A + CB)C + D$ *on obtient :*

$$p_{\{A,B\}} \left(\begin{array}{lcl} A & \rightarrow & BC + \varepsilon \\ B & \rightarrow & BD \\ C & \rightarrow & CA + CCB + D \\ D & \rightarrow & AB \end{array} \right) = \left(\begin{array}{lcl} A & \rightarrow & B(A + ABB)^* AB + \varepsilon \\ B & \rightarrow & BAB \end{array} \right)$$

Théorème 3.5 *Les arbres conformes à la grammaire projetée sont les projections des arbres conformes à la grammaire d'origine :* $G \vdash_c t :: A \;\Rightarrow\; p_{\Xi'}(G) \vdash_c p_{\Xi'}(t) :: A$ *et*

$$p_{\Xi'}(G) \vdash_c t' :: A \;\Rightarrow\; \exists t \in T(\Omega) \;\; (t' = p_{\Xi'}(t) \;\wedge\; G \vdash_c t :: A)$$

Les deux remarques suivantes montrent que le cadre ainsi élaboré est bien adapté à la modélisation et à la vérification de l'opacité dans les workflows.
REMARQUE. — Si Ξ, utilisé comme interface, est un alphabet commun à un système, manipulant des documents conformes à une grammaire G, et à un service (associée à une grammaire G') on peut décider si les documents transmis par le système au service seront reconnus conformes par ce dernier en vérifiant si $p_\Xi(G) \leq_c G'$.

REMARQUE. — Si un ensemble reconnaissable d'arbres $S \subseteq T(\Omega)$ représente un secret vis-à-vis de l'observateur associé à la fonction d'abstraction $\phi = p_{\Xi'}$ –opérant sur les arbres conformes à une grammaire $G = (\Omega, \Xi, \mathcal{L})$ avec $\Xi' \subseteq \Xi$– alors S est opaque vis-à-vis de ϕ si et seulement si $p_{\Xi'}(S \cap G) \leq_c p_{\Xi'}(G \setminus S)$. Cette relation est donc décidable. Mieux, on peut étendre la projection p_Ξ des grammaires aux automates d'arbres (en posant $F_{p_\Xi(G)} = F_G \cap \Xi$), et l'opacité équivaut à la vacuité du langage de l'automate $A_{G,\Xi',S} = p_{\Xi'}(S \cap G) \setminus p_{\Xi'}(G \setminus S)$.

Tout arbre reconnu par cet automate réprésente un cas de violation du secret. Dans le cadre d'un outil interactif cet automate, en fournissant des contre-exemples, peut permettre d'affiner la description du workflow en vue d'en assurer l'opacité.

4. Conclusion

Revenons, pour une comparaison, au cadre des systèmes à événements discrets dans lequel cette notion d'opacité a été largement étudiée. Dans ce cas l'univers U est donné par un langage régulier dont les éléments (des mots) sont toutes les exécutions possibles du système : suites d'événements, action interne ou une interaction avec un utilisateur, qui peut advenir dans le système. Chaque observateur ne voit que les actions avec lesquelles il interagit avec le système (un sous alphabet de l'ensemble E des événements).

La fonction d'observation qui lui est associée $\phi : U \rightarrow O$ est la projection qui efface dans toutes les exécutions toutes les occurrences d'événements non observable. Comme *(i)* l'ensemble des langages réguliers est clos de manière effective par les opérations booléennes, *(ii)* l'image d'un langage régulier par un morphisme de monoïdes est (de manière effective) un langage régulier, *(iii)* on peut décider de la vacuité d'un langage régulier (et donc aussi de l'inclusion de deux langages réguliers) on en déduit qu'on peut décider si $\phi(S) \subseteq \phi(U \setminus S)$, c'est-à-dire l'opacité du système. L'automate associé à $\phi(S) \setminus \phi(U \setminus S)$ reconnait toutes les exécutions qui témoignent de la non-opacité de P vis-à-vis de ϕ. Ces automates peuvent être utilisés par le concepteur du système afin d'analyser les sources potentielles de fuite d'information et les modifications peuvent être alors apportées du système afin d'en rétablir l'opacité.

Dans certain cas il est possible [1] de synthésiser un contrôleur qui restreint le comportement du système de façon minimale afin d'en restaurer l'opacité. De façon similaire nous souhaiterions disposer de méthodes pour restructurer la grammaire de telle sorte qu'on préserve au mieux les fonctionnalités de celle-ci tout en assurant l'opacité. Cette recherche constitue la suite logique de ce travail.

5. Bibliographie

[1] ERIC BADOUEL, MAREK A. BEDNARCZYK, , ANDRZEJ M. BORZYSZKOWSKI, BENOÎT CAILLAUD , PHILIPPE DARONDEAU, « Concurrent Secrets », *Discrete Event Dynamic Systems*, vol. 17, n° 4, 2007 :425-446.

[2] ANNE BRUGGEMANN-KLEIN, A. MAKOTO MURATA, DERICK WOOD, « Regular Tree and Regular Hedge Languages over Unranked Alphabets », *HKUST-TCSC-2001-05*, 2001.

[3] JEREMY BRYANS, MACIEJ KOUTNY, LAURENT MAZARÉ, PETER Y. A. RYAN, « Opacity generalised to transition systems », *Int. J. Inf. Sec.*, vol. 7,n° 6, 2008 :421-435.

[4] FABRICE CHEVALIER, JÉRÉMIE CHALOPIN, « Représentation et algorithmique des ensembles semi-linéaires », *Rapport de stage, LSV - ENS Cachan*, 2001.

[5] DAVID COHN, RICHARD HULL, « Bussiness Artifacts : A Data-centric Approach to Modeling Business Operations and Processes », *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2009.

[6] HUBERT COMON, MAX DAUCHET, RÉMI GILLERON, FLORENT JACQUEMARD, DENIS LUGIEZ, CHRISTOF LÖDING, SOPHIE TISON, MARC TOMMASI, « Tree Automata Techniques and Applications », `http ://tata.gforge.inria.fr`, 2008.

[7] JÉRÉMY DUBREIL, THIERRY JÉRON, HERVÉ MARCHAND, « Monitoring information flow by diagnosis techniques », *European Control Conference, ECC'09*, 2009.

[8] SEYMOUR GINSBURGH, « The Mathematical Theory of Context-Free Languages », *Mc Graw-Hill*, 1966.

[9] SEYMOUR GINSBURGH, E.H. SPANIER, « Semigroups, Presburger formulas, and languages », *Pacific Journal of Mathematics*, vol. 16, n° 2, 1966 :285-296. .

[10] DOMINIC HUGHES, VITALY SHMATIKOV, « Information Hiding, Anonymity and Privacy : a Modular Approach », *Journal of Computer Security*, vol. 12, n° 1, 2004 :3-36.

[11] FENG LIN, « Opacity of discrete event systems and its applications », *Automatica*, vol. 47, n° 3, 2011 :496-500.

[12] CHRISTOPHE REUTENAUER, « Aspects mathématiques des réseaux de Petri », *Masson, Paris*, 1988.