Opacité des artefacts d'un système workflow

Eric Badouel, Mohamadou Lamine Diouf

Opacité des artefacts

d'un système workflow

Eric Badouel[a,c] et Mohamadou Lamine Diouf[b,c]

(a) Inria Rennes-Bretagne Atlantique, Campus Universitaire de Beaulieu,
35042 Rennes Cedex, France
(b) Université Cheikh Anta Diop (UCAD), Dakar, Sénégal
(c) LIRIMA, Yaoundé.
{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. Nous indiquons quelques techniques pour assurer l'opacité d'un système.

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. Some techniques for enforcing opacity are indicated.

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* [12, 5] 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 équivalente 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 nombre de propriétés $S_1, \cdots S_n \subseteq U$. Il s'agit de propriétés du système, appelées *secrets*, qui ne doivent pas être dévoilées à cet utilisateur. Typiquement certaines informations confidentielles doivent être accessibles à certains utilisateurs autorisés et être tenues secrètes pour les autres. 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 $S \subseteq U$ attachés 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. [5, 13]. Nous revenons sur ces travaux dans la conclusion lorsqu'il s'agira d'envisager des techniques pour assurer l'opacité d'un système en s'inspirant des techniques de monitoring [9] ou de contrôle [1] qui ont été développées dans ce contexte d'origine.

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* [7]). 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, et données. Les éléments de l'univers de référence 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 connues de 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.

La discussion précédente va nous guider dans le choix des structures grammaticales (statuant de la conformité d'un document) ainsi que sur les fonctions d'observations ϕ :

$U \to O$ utilisées pour l'extraction des informations à transmettre à un service donné. Un autre critère important sur le choix de ces deux paramêtres est de pouvoir aboutir à des algorithmes permettant de vérifier de manière effective si un système donné est opaque ou non. Un de nos principaux objectif est effectivement d'aboutir à une formalisation des notions de structure grammaticale des documents, de fonctions d'observation et de secrets qui garantisse la décision de l'opacité.

L'hypothèse sous-jacente à la notion d'opacité (via la fonction d'observation $\phi : U \to O$) est que l'observateur a une parfaite connaissance du système : il connait l'ensemble U c'est-à-dire la structure (donnée par une grammaire) des documents complets. Par contre il n'observe que l'information $\phi(d)$ qui lui est transmise pour un document $d \in U$ donné. Sa connaissance de la grammaire peut néanmoins lui permettre de déduire des informations complémentaires. Typiquement il déduira que le document vérifie la propriété P lorsque $\phi^{-1}(\phi(d)) \subseteq P$. On parle alors de *fuite d'information*, ce qui constitue une faille de sécurité dans le cas où la propriété qui a été déduite est un secret.

Dans la conclusion nous donnons quelques éléments sur la façon dont on peut modifier le système pour assurer son opacité. La première approche, qui s'inspire des techniques de contrôle des systèmes à événements discrets, permet de construire une nouvelle grammaire qui restreint les documents de la grammaire d'origine à ceux pour lesquels aucun secret n'est dévoilé. Cette technique peut avoir l'inconvénient d'écarter des documents que le concepteur du système aurait par ailleurs toutes les raisons de vouloir considérer comme parfaitement licites. Une méthode moins intrusive, qui s'inspire de techniques d'extensions conservatives de schémas XML, consiste au contraire à étendre la grammaire.

2. Conformité d'un artefact

Un artefact est une liste de documents. Ces derniers ont une structure d'arbre. Les informations attachées aux noeuds d'un arbre sont données par des paires attributs/valeurs. On suppose donnée une abstraction finie de ces informations sous la forme d'un alphabet Ω utilisé pour étiqueter les différents noeuds des documents. La structure arborescente reflète l'organisation logique du document : $d = \omega(d_1, \ldots, d_n)$ signifie que d est un document portant une information ω et dont $d_1, \ldots, d_n$ sont les constituents (sous documents). Comme une forêt $f = d_1 \ldots d_n$ peut-être encodée par l'expression $\sharp(d_1, \ldots, d_n)$ et quitte à rajouter le symbole $\sharp$ à l'ensemble Ω, nous pourrons supposer qu'un artefact est également donné par un arbre.

Un arbre peut-être évalué selon une algèbre qui sert à extraire (ou synthétiser) des informations du document. Evaluer un arbre selon une algèbre revient à exécuter une requète sur le document pour en extraire de l'information. Si cette information est une valeur de vérité (la valeur retournée par l'évaluation est booléenne) alors l'algèbre peut être vue comme une propriété des documents. En particulier un secret, vue comme une propriété des artefacts, sera identifié à une algèbre.

Définition 2.1 (Algèbre)**.** *Une Ω-algèbre de support D est une application $\mathcal{A} : \Omega \times D^* \to D$. On note $\omega^{\mathcal{A}} : D^* \to D$, appelée* ***fonction d'interprétation*** *de $\omega \in \Omega$, l'application définie par : $\omega^{\mathcal{A}}(x_1, \ldots, x_n) = \mathcal{A}(\omega, x_1, \ldots, x_n)$. L'ensemble $T(\Omega)$ des arbres étiquetés sur Ω est le plus petit ensemble tel que*

$$t_1, \ldots t_n \in T(\Omega) \quad \wedge \quad \omega \in \Omega \implies \omega(t_1, \ldots, t_n) \in T(\Omega)$$

La valeur $t^{\mathcal{A}}$ d'un arbre $t \in T(\Omega)$ selon l'algèbre $\mathcal{A}$ est définie inductivement par :

$$t = \omega(t_1, \ldots, t_n) \implies t^{\mathcal{A}} = \omega^{\mathcal{A}}(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}})$$

Si $\mathcal{A}$ est une algèbre et $x \in D$ un élément de son support on pose

$$L(\mathcal{A}, x) = \left\{ t \in T(\Omega) \;\middle|\; t^{\mathcal{A}} = x \right\}$$

NOTATIONS. — Notons $\mathcal{A}_{\omega,x} = \left(\omega^{\mathcal{A}}\right)^{-1} = \{u \in D^* \mid \mathcal{A}(\omega, u) = x\}$ pour $\omega \in \Omega$ et $x \in D$. Pour un symbole ω donné ces ensembles sont donc disjoints : $x \neq y \implies \mathcal{A}_{\omega,x} \cap \mathcal{A}_{\omega,y} = \emptyset$. On note $\mathcal{A}_\omega = \bigcup_{x \in D} \mathcal{A}_{\omega,x}$ leur union. De façon similaire nous associons à $x \in D$ l'ensemble $\mathcal{A}_x = \bigcup_{\omega \in \Omega} \mathcal{A}_{\omega,x}$.

REMARQUE. — **Algèbre non-déterministe** — Une application $\mathcal{A} : \Omega \times D^* \to \wp(D)$, appelée algèbre non-déterministe, est assimilée à l'algèbre $\mathcal{A}^d : \Omega \times \wp(D)^* \to \wp(D)$ définie sur $\wp(D)$ par $\mathcal{A}^d(X_1, \ldots, X_n) = \bigcup \{\mathcal{A}(x_1, \ldots, x_n) \mid \forall i \in \{1, \ldots n\} \;\; x_i \in X_i\}$. Ainsi $L(\mathcal{A}, x) = L(\mathcal{A}^d, x) = \left\{ t \in T(\Omega) \;\middle|\; x \in t^{\mathcal{A}} \right\}$. Si $\mathcal{A} : \Omega \times D^* \to \wp(D)$ est une algèbre non-déterministe, on pose pour $\omega \in \Omega$ et $x \in D$

$$\mathcal{A}_{\omega,x} = \bigcup_{x \in X} \mathcal{A}^d_{\omega,X} = \{u \in D^* \mid s \in \mathcal{A}(\omega, u)\}$$

La première propriété d'un artefact qu'il convient de vérifier est sa conformité à un certain schéma donné par une grammaire.

Définition 2.2 (Grammaire)**.** *Une grammaire $G = (\Omega, \Xi, \mathcal{L})$ sur Ω est la donnée d'un ensemble fini Ξ de **sortes** et d'un langage régulier $\mathcal{L}_{\omega,s} \subseteq \Xi^*$ associé à chaque $\omega \in \Omega$ et $s \in \Xi$. On note $G \vdash t :: s$ pour signifier que t est un arbre conforme à la grammaire G et est de sorte s. On note $L(G, s) = \{t \mid G \vdash t :: s\}$ cet ensemble. Ces ensembles sont définis inductivement comme suit :*

$$(\forall i \in \{1, \ldots, n\} \;\; G \vdash t_i :: s_i \;\; \wedge \;\; s_1 \cdots s_n \in \mathcal{L}_{\omega,s}) \;\; \Rightarrow \;\; G \vdash \omega(t_1, \ldots, t_n) :: s$$

Le langage régulier $\mathcal{L}_{\omega,s} \subseteq \Xi^*$ décrit les sortes possibles des constituants (noeuds fils) d'un noeud de sorte s. 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.

NOTATIONS. — **Grammaire sous forme de productions** — Une grammaire sera présentée syntaxiquement sous la forme d'un ensemble de **productions** $A \to \omega\langle E \rangle$ où E est une expression régulière telle que $L(E) = \mathcal{L}_{\omega,A}$ lorsque cet ensemble est non vide.

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*. Comme dans le cas des algèbres, une grammaire non-déterministe peut être transformée en une grammaire déterministe équivalente (c'est-à-dire ayant le même langage).

Définition 2.3 (Déterminisation d'une grammaire)**.** *La déterminisation d'une grammaire $G = (\Omega, \Xi, \mathcal{L})$ est la grammaire déterministe $G^d = (\Omega, \wp(\Xi) \setminus \{\emptyset\}, \mathcal{L}^d)$ pour laquelle $\mathcal{L}^d_{\omega,X} = \bigcup_{s \in X} \mathcal{L}_{\omega,s}$.*

La remarque suivante montre que les grammaires ne sont rien d'autres que les algèbres régulières, c'est-à-dire les algèbres dont les langages $\mathcal{A}_{\omega,x}$ sont réguliers. Par défaut néanmoins une algèbre est déterministe alors qu'une grammaire est par défaut non-déterministe.

REMARQUE. — **Grammaires vs algèbres** – Une grammaire G peut-être assimilée à l'algèbre non-déterministe $\mathcal{A}$ de domaine Ξ telle que $\mathcal{A}(\omega, u) = \{s \in \Xi \mid u \in \mathcal{L}_{\omega,s}\}$ pour $\omega \in \Omega$, $s \in \Xi$ et $u \in \Xi^*$, et donc $\mathcal{A}_{\omega,s} = \mathcal{L}_{\omega,s}$. Une grammaire déterministe G peut être assimilée à l'algèbre de domaine $D = \Xi \cup \{\top\}$ où $\top$ est un symbole supplémentaire ($\top \notin \Xi$) et

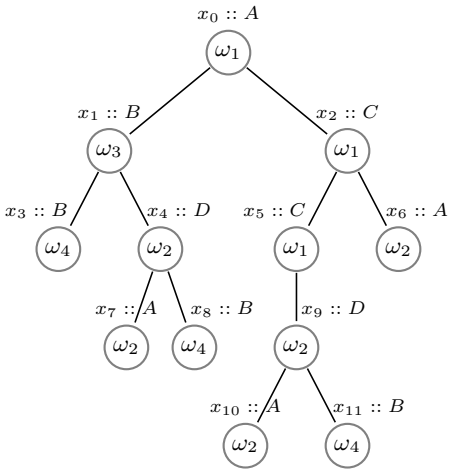
$$\omega^G(s_1, \ldots, s_n) \quad = \quad \begin{cases} s & \textbf{si} \quad s_1 \cdots s_n \in \mathcal{L}_{\omega,s} \\ \top & \textbf{sinon} \end{cases}$$

Les opérations respectives de déterminisation des grammaires et des algèbres se correspondent, c'est-à-dire que G^d décrite dans la définition 2.3 est la déterminisation de G vue comme algèbre non-déterministe : l'élément additionnel $\perp$ correspond à la partie vide qu'on avait pris soin de ne pas introduire dans la définition de G^d. La correspondance entre une grammaire déterministe et l'algèbre qui lui est associée est donnée par l'identité $L(G, s) = \left\{t \in T(\Omega) \;\middle|\; t^G = s\right\}$ qui découle immédiatement des définitions 2.1 et 2.2. Un arbre est ainsi conforme à la grammaire (ou bien formé) si et seulement si $t^G \neq \top$. Remarquons que comme $\mathcal{L}_{\omega,s} \subseteq \Xi^*$ et $\top \notin \Xi$ il vient que $t = \omega(t_1, \ldots, t_n)$ est non conforme (c'est-à-dire $\omega^G(t_1^G, \ldots, t_n^G) = \top$) dès qu'un de ses sous-arbres est non conforme.

NOTE. — **Langage d'une grammaire** – Dans la suite nous ferons l'hypothèse suivante sur les grammaires. L'ensemble des sortes contient un symbole spécifique $\mathbf{ax} \in \Xi$, appelé **axiome** de la grammaire, pour lequel il est associé une unique production $\mathbf{ax} \to \sharp\langle E_0 \rangle$ telle que le symbole $\sharp \in \Omega$ (appelé racine) n'apparait dans aucune des autres productions de la grammaire. $t^G = s$ pour $t \in T(\Omega \backslash \{\sharp\})$ et $s \in \Xi \backslash \{\top\}$ signifie que t est bien formé de sorte s, et on pose $L(G) = \left\{t_1 \ldots t_n \in T(\Omega \setminus \{\sharp\})^* \;\middle|\; t_1^G, \ldots, t_n^G \in L(E_0)\right\}$ le langage de la grammaire G, i.e., $t_1 \ldots t_n \in L(G)$ si, et seulement si $(\sharp(t_1, \ldots, t_n))^G = \mathbf{ax}$.

En utilisant les résultats classiques sur les langages reconnaissable d'arbres [4, 8] on peut montrer que les langages des grammaires sont clos de manière effective par les opérations booléennes et qu'on peut décider de la vacuité et donc de l'inclusion de ces langages, ainsi la relation $G \leq G' \iff L(G) \subseteq L(G')$ est décidable.

Comme illustré à la figure 1, un arbre peut être décrit par un système d'équations E de la forme $x = \omega(x_1, \ldots, x_n)$ dans laquelle la variable x est dite *définie* et les variables x_i ($1 \leq i \leq n$) sont dites *utilisées*. Les variables correspondent bijectivement aux noeuds de l'arbre. Chacune d'elles doit être définie par exactement une équation et ne doit être utilisée qu'au plus une fois. Enfin la relation de dépendance, qui est donnée par : $x \ll y$ ssi "*y est utilisée dans l'équation qui définit x*", doit être acyclique. L'évaluation de l'arbre dans une algèbre revient à associer une valeur $x^{\mathcal{A}}$ pour chaque variable de sorte que $x = \omega(x_1, \ldots, x_n) \implies x^{\mathcal{A}} = \omega^{\mathcal{A}}(x_1^{\mathcal{A}}, \ldots, x_n^{\mathcal{A}})$. Autrement dit le vecteur $\vec{x}^{\mathcal{A}}$ est solution du système $E_{\mathcal{A}}$, dit *interprétation de E dans $\mathcal{A}$*, formé des équations $x = \omega^{\mathcal{A}}(x_1, \ldots, x_n)$, c.-à-d. $\vec{x}^{\mathcal{A}} = E_{\mathcal{A}}(\vec{x}^{\mathcal{A}})$. Les composantes de ce vecteur $\vec{x}^{\mathcal{A}} : X \to D$ peuvent être calculées suivant un tri topologique du graphe de dépendance (on évalue un noeud après évaluation des noeuds dont il dépend). L'avantage de cette présentation équationnelle de



$$
\begin{array}{ll}
x_0 = \omega_1(x_1, x_2) & x_6 = \omega_2() \\
x_1 = \omega_3(x_3, x_4) & x_7 = \omega_2() \\
x_2 = \omega_1(x_5, x_6) & x_8 = \omega_4() \\
x_3 = \omega_4() & x_9 = \omega_2(x_{10}, x_{11}) \\
x_4 = \omega_2(x_7, x_8) & x_{10} = \omega_2() \\
x_5 = \omega_1(x_9) & x_{11} = \omega_4()
\end{array}
$$

$$
\begin{array}{l}
A \quad \textbf{where} \\
A \to \omega_1\langle BC\rangle \ + \ \omega_2\langle\varepsilon\rangle \\
B \to \omega_3\langle BD\rangle \ + \ \omega_4\langle\varepsilon\rangle \\
C \to \omega_1\langle CA\rangle \ + \ \omega_4\langle CCB\rangle \ + \ \omega_1\langle D\rangle \\
D \to \omega_2\langle AB\rangle
\end{array}
$$

Figure 1. *Un arbre vu comme un système d'équations dont la résolution dans une algèbre (par exemple une grammaire déterministe) donne la valeur de l'arbre. Le préambule "L $\mathbf{where}$ ..." à la liste des productions de la grammaire représente la production $\mathbf{ax} \to \sharp\langle L\rangle$ associée à son axiome (qui n'est, ainsi que le symbole en racine $\sharp$, pas représenté explicitement).*

l'évaluation d'un document (par opposition à une présentation plus opérationnelle à l'aide de machines à piles) est de pouvoir s'appliquer sans modification au cas où on souhaite pouvoir faire du partage de documents : l'hypothèse qu'une variable n'est jamais utilisée de façon multiple sert à caractériser les arbres mais n'a aucune utilité dans la résolution des systèmes d'équations correspondants.

On peut également lever l'hypothèse que le graphe de dépendance est acyclique en résolvant le système d'équation $\vec{x}^{\mathcal{A}} = E_{\mathcal{A}}(\vec{x}^{\mathcal{A}})$ de manière itérative. Il faut pour cela supposer que le domaine de l'algèbre est un treillis complet (dont le plus petit élément $\bot$, la valeur indéfinie, est la valeur partagée par tout objet et le plus grand élément $\top$, la valeur contradictoire, est une valeur qui ne peut-être prise par aucun objet) et que les fonctions d'interprétations sont continues (c'est-à-dire commutent aux bornes supérieurs). Dans ce cas le plus petit point fixe de l'équation $\vec{x} = E_{\mathcal{A}}(\vec{x})$, noté $E^{\dagger}_{\mathcal{A}}$ est la borne supérieure de la suite obtenue en itérant le constructeur $E_{\mathcal{A}}$ à partir du vecteur totalement indéfini $\vec{\bot}$ (dont toutes les composantes sont égalent à $\bot$), c'est-à-dire que $E^{\dagger}_{\mathcal{A}} \bigvee E^{n}_{\mathcal{A}}(\vec{\bot})$. Certaines hypothèses sur le support de l'algèbre (comme le fait qu'il soit de hauteur bornée) garantissent que la suite croissante $\langle E^{n}_{\mathcal{A}}(\vec{\bot})\rangle_{n\in\mathbb{N}}$ est stationaire, c'est-à-dire se stabilise toujours à partir d'un rang m pour lequel $E^{\dagger}_{\mathcal{A}} = E^{m}_{\mathcal{A}}(\vec{\bot})$.

Un cycle dans la relation de dépendance semble dire qu'un document est une composante de lui-même, ce qui ne fait pas de sens. Néanmoins dans certains cas la relation de dépendance ne correspondra pas uniquement à la relation de contenance (ce qu'on appelera la structure physique sous-jacente du document) mais aussi à des liens symboliques qui peuvent être utilisés pour rechercher des informations. C'est-à-dire qu'on trouvera un chemin dans le graphe de dépendance de x vers y si le calcul d'un certain attribut du noeud x peut dépendre de la valeur de certains attributs du noeud y. Même si on ne doit pas trouver de cycle dans les dépendances entre les attributs on pourra en touver entre les noeuds qui portent ces attributs dès que la valeur d'un attribut peut dépendre

d'informations provenant du contexte de ce noeud (et non seulement de ses sous-arbres). Par exemple il est montré dans [3] que les règles sémantiques d'une grammaire attribuée peuvent se décrire par une algèbre du premier ordre sur une signature étendue dans laquelle aux liens d'un père vers ses fils on rajoute un lien supplémentaire de ce noeud vers son propre père afin d'accéder à son contexte. Pour des raisons d'effectivité on doit se restreindre à des algèbres dont le support est fini et donc considérer des abstractions des informations attachées aux noeuds et des règles sémantiques qui permettent de les calculer. Ce sera plus facile à obtenir si nous partons d'une algèbre du premier ordre, d'où l'intérêt ici de pouvoir manipuler des liens symboliques permettant d'exprimer des propriétés des documents (en particulier des secrets) qui ne seront pas nécessairement définis par induction sur leur structure physique mais pourront aussi dépendre d'informations contextuelles.

Dans ce travail préliminaire nous ne nous intéressons pas à la forme des règles sémantiques, en particulier nous ne cherchons pas à identifier une syntaxe pour exprimer les secrets sur les documents. Par ailleurs pour simplifier la présentation on se limitera à la manipulation de termes (arbres finis conformes à une grammaire). On gardera néanmoins en mémoire les observations précédentes.

REMARQUE. — **Grammaires vs signatures multi-sortes** – Les grammaires sont une généralisation des signatures multisortes qui peuvent-être assimilées aux grammaires dans lesquelles pour chaque symbole $\omega \in \Omega$ les langages $\mathcal{L}_{\omega,s}$ sont tous vides sauf un qui est réduit à un seul mot. C'est-à-dire que chaque symbole a une arité et une sorte qui sont fixées : on note $\omega :: s_1 \dots s_n \to s$ lorsque $\mathcal{L}_{\omega,s} = \{s_1 \cdots s_n\}$ (et donc $s \neq s' \implies \mathcal{L}_{\omega,s'} = \emptyset$). Inversement, à toute grammaire déterministe on peut associer une signature multisorte dont les opérateurs sont les paires $\langle \omega, u \rangle$ pour $u = s_1 \dots s_n \in \Xi^*$ telles que $G(\omega, u) = s \neq \top$ il s'agit alors d'un opérateur d'arité u et de sorte s, c'est-à-dire $\langle \omega, u \rangle :: s_1 \dots s_n \to s$. Rappelons qu'une algèbre $\mathcal{A}$ pour une signature multi-sorte est donnée par un ensemble non vide $\mathcal{A}_s$ associé à chaque sorte $s \in \Xi$ et une fonction d'interprétation $\omega^{\mathcal{A}} : \mathcal{A}_{s_1} \dots \mathcal{A}_{s_n} \to \mathcal{A}_s$ associée à chaque opérateur $\omega :: s_1 \dots s_n \to s$. Les algèbres pour la signature multisorte associée à une grammaire déterministe G sont les Ω-algèbres $\mathcal{A} : \Omega \times D^* \to D$ pour lesquelles il existe une application surjective $\varphi : D \twoheadrightarrow \Xi$ telle que $\mathcal{L}_{\omega,\varphi(x)} = \{\varphi(x_1) \dots \varphi(x_n) \mid \mathcal{A}(\omega, x_1 \cdots x_n) = x\}$. Autrement dit $G \circ \langle id_\Omega, \varphi^* \rangle = \varphi \circ \mathcal{A}$. Nous dirons par analogie que ces algèbres sont les **algèbres de la grammaire**.

Un artefact (liste de documents) est conforme s'il appartient au langage de la grammaire et en particulier chacun des documents qui le constitue est bien formé. Une fois que la conformité de l'artefact est acquise on peut en extraire de l'information en l'évaluant selon des algèbres de la grammaire.

3. Abstraction d'un artefact

Dans la section précédente nous nous sommes intéressé à la structure logique d'un artefact (donnée par une grammaire) et à son évaluation selon une algèbre. Nous allons maintenant nous intéresser aux données qu'il contient. Les informations attachées à un noeud correspondent à son étiquette $\omega \in \Omega$. Nous pouvons voir un artefact comme des boîtes imbriquées avec pour chacune d'entre elles les informations qui lui sont attachées. 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 cor-

respondent. L'effet de cette transformation sur la structure du document peut se décrire comme suit.

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 pour $t = \omega(t_1, \ldots, t_n)$ par :*

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

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

Définition 3.2. *Si $\pi = p_{\Xi'}$ est la projection associée à $\Xi' \subseteq \Xi$, on pose $\mathcal{A} = G/\pi$ l'algèbre non-déterministe définie par $\mathcal{A}_{\omega,s} = \{v \in \Xi'^* \mid \exists u \in \mathcal{L}_{\omega,s} \quad u \vdash^* v\}$ où $\vdash^*$ est la fermeture réflexive et transitive de la relation*

$$\vdash \; = \{(u \cdot s \cdot u', u \cdot v \cdot u' \mid s \in \Xi \setminus \Xi' \;\; \textit{et} \;\; v \in \mathcal{L}_s\}$$

Ainsi $\mathcal{A}^d(\omega, V) = \{s \in \Xi' \mid \exists u \in \mathcal{L}_{\omega,s} \; \exists v \in V \quad u \vdash^* v\}$.

Théorème 3.3. $L(G/\pi) = \pi(L(G))$.

Afin d'établir ce théorème nous introduisons sur les arbres une relation, indexée par les contextes, qui correspond à la relation de dérivation $\vdash$ sur les sortes.

Définition 3.4 (Contextes)**.** *Les opérateurs dérivés, ou **contextes**, $C \in \Omega^{@}$ sont les arbres construits sur la signature $\Omega \cup \{[\,]\}$ obtenue à partir de Ω en ajoutant un symbole spécial $[\,]$ figurant la position d'un argument du contexte. Les contextes sont ainsi définis par la syntaxe BNF suivante :*

$$C \;\; := \;\; [\,] \;\mid\; \omega(C_1, \ldots, C_n)$$

*L'**arité** d'un contexte, correspondant à son nombre d'arguments, est défini inductivement par $\mathbf{ar}([\,]) = 1$ et $\mathbf{ar}(\omega(C_1, \ldots, C_n)) = \mathbf{ar}(C_1) + \cdots + \mathbf{ar}(C_n)$ (avec par convention $\mathbf{ar}(\omega()) = 0$ correspondant au cas où $n = 0$). On note $\Omega_n^{@}$ l'ensemble des contextes d'arité n et $\omega_n \in \Omega_n^{@}$ le contexte d'arité n associé à l'opérateur ω, il s'agit du contexte $\omega_n = \omega([\,], \ldots, [\,])$ contenant exactement n arguments. Un contexte $C \in \Omega_n^{@}$ d'arité n s'interprète selon une algèbre $\mathcal{A} : \Omega \times D^* \to D$ comme l'opérateur $C^{\mathcal{A}} : D^n \to D$ donné par :*

(1) $[\,]^{\mathcal{A}}(x) = x$

(2) $(\omega(C_1, \ldots, C_n))^{\mathcal{A}}(v) = \omega^{\mathcal{A}}(C_1^{\mathcal{A}}(v_1), \ldots, C_n^{\mathcal{A}}(v_n))$
où $|v_i| = \mathbf{ar}(C_i)$ et $v_1 \cdots v_n = v$.

*En particulier on assimilera un contexte avec son interprétation dans l'algèbre libre $F : \Omega \times T(\Omega)^* \to T(\Omega)$ (pour laquelle $\omega^F(t_1, \ldots, t_n) = \omega(t_1, \ldots, t_n)$), c'est-à-dire que l'expression $C(t_1, \ldots, t_n)$ sera définie si, et seulement si, ω est d'arité n, et dans ce cas elle correspond à l'arbre obtenu en substituant les arbres $t_1, \ldots, t_n$ aux arguments du contexte pris dans leur ordre d'apparition. L'écriture $t = C(t_1, \ldots, t_n)$ correspond à une **décomposition** d'un arbre en un contexte et une suite d'arbres associés aux arguments du contexte.*

Nous allons décomposer un arbre de sorte visible $t \in T(\Omega)_{\Xi'} = \left\{t \in T(\Omega) \;\middle|\; t^G \in \Xi'\right\}$ sous la forme $t = C(t_1, \ldots, t_n)$ où C est le plus petit contexte non trivial (i.e., $C \neq [\,]$) pour lequel $t_i^G \in \Xi'$. En particulier si tous les sous arbres stricts t' de t sont tels que $t' \notin \Xi'$, on aura $t = C$ (et donc le contexte ne contiendra aucun argument). Cette décomposition est donnée par la relation $t \gg_\omega t_1 \ldots t_n$ définie ci-dessous.

Définition 3.5 (Décomposition d'un arbre)**.** *La relation* $\vdash \subseteq T(\Omega) \times \Omega^{@} \times T(\Omega)^*$*, où* $(t, C, t_1 \ldots, t_n) \in \vdash$ *est noté* $t \vdash_C t_1 \ldots t_n$*, est la plus petite relation telle que*

(1) $t \vdash_{[]} t$ *si* $t^G \in \Xi'$

(2) $t \vdash_{\omega_n} t_1 \ldots t_n$ *si* $t = \omega(t_1, \ldots, t_n)$ *et* $t^G \notin \Xi'$

(3) *Si* $t \vdash_{\omega_n} t_1 \ldots t_n$ *et* $t_i \vdash_{C_i} v_i$ *pour tout* $1 \leq i \leq n$ *alors* $t \vdash_{\omega_n(C_1,\ldots,C_n)} v$ *où* $v = v_1 \cdots v_n$ *est la concaténation des* v_i*.*

Pour $t = \omega(t_1, \ldots, t_n) \in T(\Omega)_{\Xi'}$ *on pose* $t \gg_\omega v$ *lorsque* $t_i \vdash_{C_i} v_i$ *où* $v_i \in (T(\Omega)_{\Xi'})^*$ *et* $v = v_1 \cdots v_n$ *est la concaténation des suites d'arbres* $v_1, \ldots, v_n$*.*

La correspondance entre cette décomposition d'un arbre et la relation de dérivation sur les sortes est donnée par les quatre lemmes suivants.

Lemme 3.6. $t \vdash_C t_1 \cdots t_n \implies t^G \vdash^* t_1^G \cdots t_n^G$

Démonstration. La démonstration s'effectue par récurrence sur la preuve de la relation $t \vdash_C t_1 \cdots t_n$. Le cas de base se décline suivant la valeur de t^G :

$t^G \in \Xi'$**.** On a d'une part $t \vdash_{[]} t$ et d'autre part $t^G \vdash^* t^G$ par reflexivité de cette relation.

$t^G \notin \Xi'$**.** Si t est de la forme $t = \omega(t_1, \ldots, t_n)$ on déduit que $t \vdash_{\omega_n} t_1 \ldots t_n$. D'autre part $t_1^G \ldots t_n^G \in \mathcal{L}_{\omega,t^G}$ et donc $t_i^G \vdash t_1^G \ldots t_n^G$.

Pour le cas général supposons $t \in \omega(t_1, \ldots, t_n)$ et $t_i \vdash_{C_i} v_i$ pour tout $1 \leq i \leq n$. On a dans ce cas $t \vdash_{\omega_n(C_1,\ldots,C_n)} v_1 \ldots v_n$. Par hypothèse de récurrence $t_i^G \vdash^* v_i^G$ en convenant que $(t'_1 \ldots t'_k)^G = (t'_1)^G \ldots (t'_k)^G$. Par ailleurs de $t_1^G \ldots t_n^G \in \mathcal{L}_{\omega,t^G}$ il vient $t_i^G \vdash t_1^G \ldots t_n^G$, et donc $t^G \vdash^* v_1^G \ldots v_n^G$. □

Lemme 3.7. $s \vdash^* t_1^G \cdots t_n^G \implies (\exists t \in T(\Omega))\ \big(\exists C \in \Omega_n^{@}\big) \quad t \vdash_C t_1 \cdots t_n$ *et* $t^G = s$

Démonstration. La preuve est par récurrence sur la longueur de la dérivation de $\vdash^*$.

Dérivations de longueur nulle : ceci correspond au cas où $s \in \Xi'$, et alors $n = 1$ et $s = t_1^G$. Le résultat découle de $t_1 \vdash_{[]} t_1$.

Cas général : la dérivation $s \vdash^* t_1^G \cdots t_k^G$ peut se décomposer en $s \vdash s_1 \ldots s_n$ –ce qui signifie que $s_1 \ldots s_n \in \mathcal{L}_{\omega,s}$ pour un certain opérateur $\omega \in \Omega$– et, pour tout $1 \leq i \leq n$, $s_i \vdash v_i$ tels que $v_1 \cdots v_n = t_1^G \cdots t_k^G$=. Par hypothèse de récurrence il existe un arbre $t_i \in T(\Omega)$ et un contexte $C_i \in \Omega^{@}$ tels que $t_i^G = s_i$ et $t_i \vdash_C v_i$. L'arbre $t = \omega(t_1, \ldots, v_n)$ vérifie $t^G = s$ et $t \vdash_{\omega_n(C_1,\ldots,C_n)} t_1 \ldots t_n$

□

Lemme 3.8. $t \gg_\omega t'_1 \cdots t'_k \implies \exists s_1 \ldots s_n \in \mathcal{L}_{\omega,t^G} \quad s_1 \ldots s_n \vdash^* (t'_1)^G \ldots (t'_k)^G$

Démonstration. $t \gg_\omega t'_k \cdots t'_k$ si, et seulement si, $t = \omega(t_1, \ldots, t_n)$ tels que $t_1^G \cdots t_1^G \in \mathcal{L}_{\omega,t^G}$ et pour tout $1 \leq i \leq n \quad t_i \vdash_{C_i} v_i$ tels que $v_1 \ldots v_n = t'_1 \ldots t'_k$. Par le lemme 3.6 $t_i^G \vdash^* v_i^G$ et donc $t_1^G \ldots t_n^G \vdash^* (t'_1)^G \ldots (t'_k)^G$. □

Lemme 3.9. $\big(\exists s_1 \ldots s_n \in \mathcal{L}_{\omega,s}$ *t.q.* $s \in \Xi'$ *et* $s_1 \ldots s_n \vdash^* (t'_1)^G \ldots (t'_k)^G\big) \implies$ $\big(\exists t \in T(\Omega)$ *t.q.* $t \gg_\omega t'_1 \ldots t'_k$ *et* $t^G = s\big)$

Démonstration. Supposons $s_1 \ldots s_n \vdash^* (t'_1)^G \ldots (t'_k)^G$ pour $s \in \Xi'$ et $s_1 \ldots s_n \in \mathcal{L}_{\omega,s}$. Par le lemme 3.7 on déduit l'existence d'arbres t_i et de contextes C_i tels que $t_i \vdash_{C_i} v_i$, $v_1 \ldots v_n = t'_1 \ldots t'_k$ et $t_i^G = s_i$. Posons $t = \omega(t_1, \ldots, t_n)$. De $t_i \vdash_{C_i} v_i$, $v_1 \ldots v_n = t'_1 \ldots t'_k$ on déduit $t \gg_\omega t'_1 \ldots t'_k$. Par ailleurs de $s_1 \ldots s_n \in \mathcal{L}_{\omega,s}$ et $t_i^G = s_i$ on déduit $t^G = s$. □

A tout arbre $t \in T(\Omega)_{\Xi'}$ est associé un unique opérateur $\omega \in \Omega$ et une unique liste d'arbres $t'_k \cdots t'_k \in (T(\Omega)_{\Xi'})^*$ tels que $t \gg_\omega t'_k \cdots t'_k$. Autrement dit la relation $\gg = \bigcup_{\omega\in\Omega} \gg_\omega$ détermine une application $\gg$: $T(\Omega)_{\Xi'} \to (T(\Omega)_{\Xi'})^*$. Par ailleurs $T(\Omega)_{\Xi'}$ est le plus petit ensemble E tel que $t'_1, \ldots, t'_k \in E$ et $t \gg_\omega t'_1 \ldots t'_k \implies t \in E$. On peut ainsi raisonner par récurrence sur l'ensemble des arbres dont la sorte est visible en utilisant la décomposition associée à la relation $\gg$. On note par ailleurs que $t \gg_\omega t'_1 \ldots t'_k \implies \pi(t) = \omega(\pi(t'_1), \ldots, \pi(t'_k))$. Nous disposons maintenant de tous les ingrédients nous permettant d'établir le théorème 3.3.

Démonstration. **[du théorème 3.3]**
Nous montrons que $L(G/\pi, s) = \pi(L(G, s))$ pour tout $s' \in \Xi'$. C'est-à-dire que les deux ensembles suivants coïncident $\{\pi(t) \mid t^G = s\} = \{t \mid s \in t^{G/\pi}\}$. Pour cela nous montrons successivement que *(i)* $t^G \in (\pi(t))^{G/\pi}$ and *(ii)* $s \in t'^{G/\pi} \implies (\exists t$ t.q. $t^G = s$ et $\pi(t) = t')$.

(i) On raisonne par récurrence sur la décomposition de t donnée par la relation $\gg$. Supposons $t \gg t'_1 \ldots t'_k$. par hypothèse de récurrence $t_i^G \in (\pi(t_i))^{G/\pi}$. Du lemme 3.8 il vient que $\exists s_1 \ldots s_n \in \mathcal{L}_{\omega,t^G}$ t.q $s_1 \ldots s_n \vdash^* (t'_1)^G \cdots (t'_k)^G$. On déduit que $\pi(t)^{G/\pi} = \omega^{G/\pi}(\pi(t_1)^{G/\pi}, \ldots, \pi(t_n)^{G/\pi})$ contient t^G du fait que $s_1 \ldots s_n \in \mathcal{L}_{\omega,t^G}$, $t_i^G \in (\pi(t_i))^{G/\pi}$ et $s_1 \ldots s_n \vdash^* (t'_1)^G \cdots (t'_k)^G$.

(ii) On raisonne par récurrence sur la structure de l'arbre t'. Supposons $t' = \omega(t'_1 \ldots t'_k)$ et $s \in (t')^{G/\pi}$. En utilisant la définition 3.2 on en déduit l'existence de $s_1 \ldots s_n \in \mathcal{L}_{\omega,s}$ et de $s'_i \in (t'_i)^{G/\pi}$ pour tout $1 \leq i \leq k$ tels que $s_1 \ldots s_n \vdash^* s'_1 \ldots s'_k$. Par hypothèse de récurrence il existe des arbres t_i tels que $t_i^G = s'_1$ et $\pi(t_i) = t'_i$. De $s_i \ldots s_n \in \mathcal{L}_{\omega,s}$ et $s_1 \ldots s_n \vdash^* t_1^G \ldots t_k^G$ on en déduit par le lemme **??** l'existence d'un arbre $t \in T(\Omega)_{\Xi'}$ tel que $t \gg_\omega t_1 \ldots t_k$ et $t^G = s$. De $t \gg_\omega t_1 \ldots t_k$ et $\pi(t_i) = t'_i$ on déduit $\pi(t) = \omega(\pi(t_1), \ldots, \pi(t_k)) = \omega(t'_1 \ldots t'_k) = t'$.

Comme $\sharp \in \Xi'$ le résultat découle de

$$\begin{aligned}\pi(L(G)) &= \{\pi(t_1) \ldots \pi(t_n) \mid \sharp(t_1, \ldots, t_n) \in L(G, \mathbf{ax})\} \\ &= \{\pi(t_1) \ldots \pi(t_n) \mid \sharp(\pi(t_1), \ldots, \pi(t_n)) \in L(G/\pi, \mathbf{ax})\} \\ &= L(G/\pi)\end{aligned}$$

Car $\pi : L(G, s) \to L(G/\pi, s)$ est une surjection pour tout $s \in \Xi'$. □

4. Discussion

Nous nous intéressons aux deux problèmes suivants.

PROBLÈME. — **Invocation d'un service** – Supposons qu'on veuille développer un système, manipulant des documents conformes à une grammaire $G = (\Omega, \Xi, \mathcal{L})$, en utilisant des services dont l'interface est également donnée par une grammaire G' qui décrit les documents que le service est disposer à recevoir. Un sous ensemble $\Xi' \subseteq \Xi$ de sortes visibles peut représenter les parties du document qu'on souhaite transmettre au service pour traitement. On doit pouvoir vérifier que $p_{\Xi'}(L(G)) \subseteq L(G')$ pour s'assurer que les documents transmis par le système au service seront toujours reconnus conformes par ce dernier.

PROBLÈME. — **Opacité** – 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 $p_{\Xi'}(S \cap L(G)) \leq p_{\Xi'}(L(G) \setminus S)$.

Le langage $\mathcal{A}_{\omega,s} = \{v \in \Xi'^* \mid \exists u \in \mathcal{L}_{\omega,s} \quad u \vdash^* v\}$ associé à l'algèbre $\mathcal{A} = (G/p_{\Xi'})$ qui caractérise les projections des documents conformes à la grammaire G, i.e., $L(\mathcal{A}) = p_{\Xi'}(L(G))$, est un langage algébrique. Ces langages sont rationnels si la grammaire est non récursive : on ne peut trouver de cycles dans la relation de dépendance $s > s'$ correspondant au fait que s' apparait en partie droite d'une production associée à s. Dans ce cas l'algèbre $\mathcal{A}$ est une grammaire. Les grammaires sont closes de manière effective par les opérations booléennes. La complémentation et donc l'union ne préservent pas la non-circularité. Néanmoins les deux opérations qui nous intéressent, à savoir l'intersection et la différence, préservent cette propriété. Enfin on sait décider de la vacuité et donc de l'inclusion des langages de telles grammaires. Les deux problèmes ci-dessus peuvent donc être résolus de manière effective dans ce cas. En particulier on peut construire la grammaire $\pi_{\Xi'}(S \cap G) \setminus p_{\Xi'}(G \setminus S)$ qui reconnait exactement les cas de violation du secret. Dans le cadre d'un outil interactif cette grammaire peut être utilisée pour fournir des contre-exemples lorsque l'opacité n'est pas assurée. Cette information peut être utilisée par le concepteur du système pour l'aider à modifier la description du workflow en vue d'en assurer l'opacité.

Il est assez naturel de considérer comme nous l'avons fait des grammaires dans lesquelles les parties droites des productions sont des langages réguliers (et non un ensemble fini de mots) car on aura souvent besoin de considérer qu'un noeud d'un artefact puisse contenir une liste de sous documents d'une certaine sorte sans que la quantité de tels documents puissent fixée *a priori*. En revanche on aura rarement besoin de considérer des grammaires récursives. Ainsi du point de vue des artefacts l'hypothèse de non circularité est tout à fait raisonnable. Néanmoins comme nous l'avons signalé plus haut dans le texte cette hypothèse restreindra la classe des secrets que nous pourront prendre en compte puisqu'on ne pourra pas exprimer des propriétés qui dépendent d'information contextuelles.

La solution alternative que nous considérons dans la section suivante est d'interpréter les documents modulo permutations des sous arbres d'un noeud. Nous restons de cette façon dans le cadre rationnel grâce au thérème de Parikh qui nous dit l'image commutative d'un langage algébrique est rationnel.

5. Vérifier l'opacité d'un système

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 $\mathcal{L}_{\omega,A}$ 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. Pour cette raison, plutot considérer la relation de "conformité modulo permutations" définie comme suit.

Définition 5.1 (Conformité modulo permutations)**.** *La relation de conformité modulo permutations est la plus petite relation telle que :*

$$(\forall i \in \{1, \ldots, n\} \quad G \vdash_c t_i :: s_i \quad \wedge \quad s_1 \cdots s_n \in [\mathcal{L}_{\omega,s}]) \quad \Rightarrow \quad G \vdash_c \omega(t_1, \ldots, t_n) :: s$$

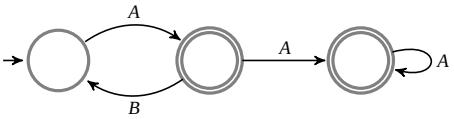
où $[L] = \{u \mid \exists v \in L \quad u \approx v\}$ *est la cloture de* L *modulo commutations. Le* ***langage modulo commutations*** *d'une grammaire est donné par :*

$$\begin{aligned} L_c(G, s) &= \{t \in T(\Omega) \mid G \vdash_c t :: s\} \\ L_c(G) &= \{t_1 \dots t_n \in T(\Omega) \mid G \vdash_c \sharp(t_1, \dots, t_n) :: \mathbf{ax}\} \end{aligned}$$

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

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 5.2. *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). On peut utiliser l'automate associé au langage régulier* $\mathcal{L}$



comme un générateur *de* $[\mathcal{L}]$*. Pour cela on interprète l'expression rationnelle* $(AB)^*A^*$ *dans* $\mathbb{N}^2$ *en identifiant les lettres* A *et* B *avec les deux générateurs* $(1, 0)$ *et* $(0, 1)$ *respectivement. Au niveau de l'automate cela revient à cumuler les valeurs des étiquettes rencontrées le long d'un chemin reconnaissant de l'automate. Cet automate procure ainsi un mécanisme qui permet d'engendrer* $[\mathcal{L}] = \{(x, y) \in \mathbb{N}^2 \mid x \geq y\}$*. Cet ensemble 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.*

RAPPEL. — 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 $\mathbb{N}^2$ sont les unions finies de "grilles rectangulaires" . Ces dernières désignent des ensembles de la forme $\{(x_0 + n \cdot x_1, y_0 + m \cdot y_1) \mid n, m \in \mathbb{N}\}$ où x_0, x_1, y_0 et y_1 sont des constantes entières et on peut vérifier que le quadrant $\{(x, y) \in \mathbb{N}^2 \mid x \geq y\}$ ne peut s'écrire comme une union finie de telles grilles (il s'agit d'une partie rationnelle non reconnaissable).

Les ensembles qui nous intéressent sont donc les images commutatives des langages rationnels, c'est-à-dire les parties rationnelles de $\mathbb{N}^k$ (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 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 $\mathbb{N}^k$ coïncident avec les parties semi-linéaires.

RAPPEL. — 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.

Ces ensembles semi-linéaires coïncident par ailleurs avec les ensembles définissables par des formules de Presburger, i.e., de la forme $\{(x_1, \cdots, x_n) \mid F(x_1, \cdots, x_n)\}$ où F est une formule de Presburger.

RAPPEL. — 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 $(\exists z) \cdot \ y = x + z$.

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 [10, 11, 14, 6]). 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 $\mathcal{L}_\omega = \cup_{s \in \Xi} \mathcal{L}_{\omega,s}$.

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 plus haut, nous pouvons décider si le langage d'une grammaire est à commutations près contenu dans celui d'une autre, i.e., la relation $G \leq_c G' \iff L_c(G) \subseteq L_c(G')$ est décidable.

La définition suivante est l'adaptation de la définition 3.2 pour les grammaires à commutations près.

Définition 5.3. *Si $\Xi' \subseteq \Xi$ est un sous-ensemble de l'alphabet des sortes d'une grammaire $G = (\Omega, \Xi, \mathcal{L})$ et $s \in \Xi \setminus \Xi'$, on pose $L(G, \Xi', s) = \{u \in \Xi'^* \mid s \vdash^* u\}$. La projection de G à Ξ' est la grammaire $p_{\Xi'}(G) = (\Omega, \Xi', \mathcal{L}')$ où $\mathcal{L}'_{\omega,s'} = \mathcal{L}_{\omega,s'}[[L(G, \Xi', s)]/s \ ; \ s \in \Xi \setminus \Xi']$, c'est-à-dire qu'on substitue l'image commutative du langage algébrique $L(G, \Xi', s)$ (qui est donc un langage rationnel par le théorème de Parikh) à la variable non visible $s \in \Xi \setminus \Xi'$ dans la partie droite $\mathcal{L}_{\omega,s'}$ de la règle, dans la grammaire d'origine, associée au symbole visible $s' \in \Xi'$.*

REMARQUE. — Dans la pratique nous n'introduisons pas explicitement le langage régulier $[L(G, \Xi', s)]$ 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 équation $X = A(X) \cdot X + T$ dans laquelle la variable X n'apparait pas dans T est à commutations près donnée par l'expression régulière $A(T)^* \cdot T$

Exemple 5.4 (voir figure 2)**.**
La projection de la grammaire

$$G = \left(\begin{array}{l} A \quad \textbf{where} \\ \quad A \to \omega_1\langle BC\rangle \ + \ \omega_2\langle\varepsilon\rangle \\ \quad B \to \omega_3\langle BD\rangle \ + \ \omega_4\langle\varepsilon\rangle \\ \quad C \to \omega_1\langle CA\rangle \ + \ \omega_4\langle CCB\rangle \ + \ \omega_1\langle D\rangle \\ \quad D \to \omega_2\langle AB\rangle \end{array} \right)$$

sur le sous-alphabet $\Xi' = \{A, B\}$ *est donnée par*

$$p_{\{A,B\}}(G) = \left(\begin{array}{lll} A & \textbf{where} & \\ \quad A & \to & \omega_1\langle BC\rangle \ + \ \omega_2\langle\varepsilon\rangle \\ \quad B & \to & \omega_3\langle BD\rangle \ + \ \omega_4\langle\varepsilon\rangle \\ \quad C & =_c & CA + CCB + D \\ \quad D & =_c & AB \end{array} \right)$$

En utilisant $CA + CCB + D =_c (A + CB)C + D$ *on obtient* $C =_c (A + DB)^* D = (A + ABB)^* AB$ *et donc*

$$p_{\{A,B\}}(G) = \left(\begin{array}{lll} A & \textbf{where} & \\ \quad A & \to & \omega_1\langle B(A + ABB)^* AB\rangle \ + \ \omega_2\langle\varepsilon\rangle \\ \quad B & \to & \omega_3\langle BAB\rangle \ + \ \omega_4\langle\varepsilon\rangle \end{array} \right)$$

L'arbre de la figure 1 (qu'on retrouve à gauche de la figure 2) se présente sous la forme

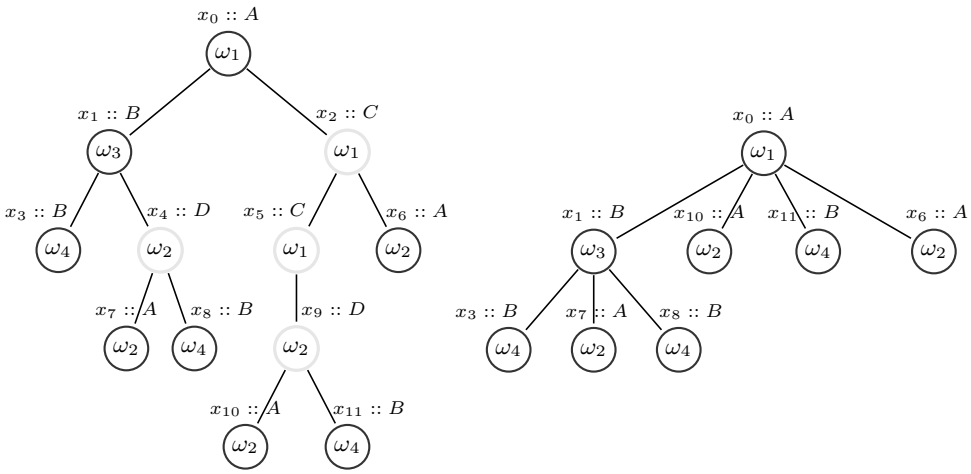


Figure 2. *A gauche, l'arbre* t *conforme à la grammaire* G *provenant de la figure 2 :* $G \vdash_c t :: A$*. A droite, la projection de* t *sur le sous-alphabet de sortes visibles* $\Xi' = \{A, B\}$*. On vérifie que* $p_{\{A,B\}}(G) \vdash_c p_{\{A,B\}}(t) :: A$*.*

du système d'équations suivant :

$$\begin{array}{llll} x_0 = \omega_1(x_1, x_2) & x_3 = \omega_4() & x_6 = \omega_2() & x_9 = \omega_2(x_{10}, x_{11}) \\ x_1 = \omega_3(x_3, x_4) & x_4 = \omega_2(x_7, x_8) & x_7 = \omega_2() & x_{10} = \omega_2() \\ x_2 = \omega_1(x_5, x_6) & x_5 = \omega_1(x_9) & x_8 = \omega_4() & x_{11} = \omega_4() \end{array}$$

On peut étiqueter chaque variable par la sorte qui lui est associée lorsqu'on résout ce système dans l'algèbre associée à la grammaire G

$$\begin{array}{llll} x_0^A = \omega_1(x_1^B, x_2^C) & x_3^B = \omega_4() & x_6^A = \omega_2() & x_9^D = \omega_2(x_{10}^A, x_{11}^B) \\ x_1^B = \omega_3(x_3^B, x_4^D) & x_4^D = \omega_2(x_7^A, x_8^B) & x_7^A = \omega_2() & x_{10}^A = \omega_2() \\ x_2^C = \omega_1(x_5^C, x_6^A) & x_5^C = \omega_1(x_9^D) & x_8^B = \omega_4() & x_{11}^B = \omega_4() \end{array}$$

Le système d'équation associé à la projection de l'arbre est obtenu en "effaçant" le symbole d'opérateur en partie droite d'une équation dont la variable en partie gauche est d'une sorte non visible :

$$\begin{array}{llll} x_0^A = \omega_1(x_1^B, x_2^C) & x_3^B = \omega_4() & x_6^A = \omega_2() & x_9^D = x_{10}^A, x_{11}^B \\ x_1^B = \omega_3(x_3^B, x_4^D) & x_4^D = x_7^A, x_8^B & x_7^A = \omega_2() & x_{10}^A = \omega_2() \\ x_2^C = x_5^C, x_6^A & x_5^C = x_9^D & x_8^B = \omega_4() & x_{11}^B = \omega_4() \end{array}$$

On simplifie ensuite ce système en supprimant toute équation de la forme $x^s = x_1^{s_1} \dots x_n^{s_n}$ associée à une sorte non visible $s \in \Xi \setminus \Xi'$ après avoir substitué la définition de x_s, i.e., $x_1^{s_1} \dots x_n^{s_n}$, à chacune de ses occurrences d'utilisation (en fait ici on ne peut avoir qu'une seule telle occurrence car ce système d'équation provient d'un arbre) :

$$\begin{array}{llll} x_0^A = \omega_1(x_1^B, x_{10}^A, x_{11}^B, x_6^A) & x_3^B = \omega_4() & x_7^A = \omega_2() & x_{10}^A = \omega_2() \\ x_1^B = \omega_3(x_3^B, x_7^A, x_8^B) & x_6^A = \omega_2() & x_8^B = \omega_4() & x_{11}^B = \omega_4() \end{array}$$

6. conclusion

Dans ce papier nous avons proposé une formalisation de l'opacité des artefacts dans les systèmes workflow centrés sur l'échange de données structurées. Nous avons identifié deux contextes dans lesquels non seulement l'opacité est décidable mais l'algorithme de décision peut générer des contre exemples à l'opacité lorsque cette propriété n'est pas satisfaite : lorsque les grammaires sont non récursives ou lorsque les documents ne sont considérś qu'à la permutation près de sous arbres.

Le retour d'information produit par l'algorithme de décision peut être utilisé par le concepteur du système afin qu'il puisse faire les modifications nécessaires au rétablissement de l'opacité. On peut souhaiter que le système aille un peu plus loin en effectuant lui-même ces modifications ou tout du moins en proposant à l'utilisateur des modifications de nature à résoudre les éventuels problèmes de fuites d'informations.

De telles techniques, dite de contrôle, ont été étudiées dans le contexte des systèmes à événement discrets, cadre dans lequel l'opacité a été étudiée jusque là. Rappelons brièvement ce qui a été fait pour assurer l'opacité dans les systèmes à événement discrets. Dans ce cas l'univers U est donné par un langage régulier dont les éléments (des mots) sont toutes les exécutions du système, c'est-à-dire les suites d'événements (action interne ou une interaction avec un utilisateur) qui peuvent 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 $\phi : U \to O$ est la projection qui efface dans toutes les exécutions toutes les occurrences d'événements non observables. Comme *(i)* l'ensemble des langages réguliers est clos de manière effective par les opérations boolénnes, *(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.

Le langage régulier $L_{\varphi,S} = \phi^{-1}\left(\phi(S) \setminus \phi(U \setminus S)\right)$ donne toutes les exécutions qui témoignent de la non-opacité de P vis-à-vis du secret S. De façon plus précise le langage $L = U \setminus \left(\bigcup \left\{\phi^{-1}\left(\phi(S) \setminus \phi(U \setminus S)\right) \mid S \text{ secret pour } \phi\right\}\right)$ est le plus grand langage opaque contenu dans U. Ainsi on peut rendre le système opaque en le synchronisant avec un contrôleur, c'est-à-dire qu'on autorise le système à exécuter une séquence que lorsque celle-ci est de façon simultanée autorisée, c'est-à-dire exécutable, par le contrôleur. Le langage du système contrôlé est donc l'intersection du langage du système d'origine avec celui du contrôleur. Ce contrôleur est optimal en ce sens qu'il n'interdit que les comportements qui violent l'opacité du système. Qui plus est [1] ce contrôle est distribué, c'est-à-dire qu'on peut de façon équivalente associer à chaque utilisateur ϕ un contrôleur $K_\phi = (\bigcup \{\phi(S) \setminus \phi(U \setminus S) \mid S \text{ secret pour } \phi\})^c$ qui restreint les interactions autorisées de cet utilisateur avec le système.

Nous disposons ici de la même caractérisation logique du contrôleur quitte à remplacer des automates de mots par des automates d'arbres (non ordonnés et à arité variables). Par ailleurs nous savons [2] que la synchronisation de vues partielles d'un même document peut s'obtenir via le produit des automates d'arbres associés à chacune de ces vues. Les techniques de synthèse et de distribution d'un contrôleur devraient pouvoir s'étendre sans difficultés majeures aux artefacts d'un workflow.

La solution obtenue par contrôle produit une grammaire qui génère exactement les documents de U pour lesquels aucun secret n'est dévoilé. Le résultat peut néanmoins être peu satisfaisant d'un point de vue pratique si cela force à exclure trop de documents, et en particulier des documents que le concepteur du système souhaiterait conserver dans son modèle. Une solution moins intrusive, qui s'inspire de techniques d'extensions conservatives de schémas XML, est à l'étude. Cette méthode consiste à n'invalider aucun document mais au contraire à étendre la grammaire : l'introduction de nouveaux documents licites constitue du "bruit" qui peut empêcher la détection de secrets. Plus précisémment on doit pouvoir trouver un document $d' \in \psi^{-1}(\psi(d)) \setminus S$ de la grammaire étendue ($d' \in U' = L(G')$) pour chaque document $d \in U = L(G)$ de la grammaire G d'origine qui dévoilait ce secret ($\phi^{-1}(\phi(d)) \subseteq S$) où $\psi : U' \to O$ est l'extension de $\phi : U \to O$. Des telles méthodes sont à l'étude.

Une étape suivante de ce travail consiste à préciser la forme des règles sémantiques qui régissent le calcul des attributs attachés aux noeuds des documents. Cela déterminera par la même occasion la syntaxe pour les algèbres et donc en particulier pour les requêtes faites sur un document et pour l'expression des secrets. Nous comptons également implémenter les algorithmes de décision et de contrôle de l'opacité en utilisant les bibliothèques existantes de manipulation des ensembles semi-linéaires.

7. 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] ERIC BADOUEL, MAURICE TCHOUPÉ TCHENDJI « Merging Hierarchically-Structured Documents in Workflow Systems », *Electronic Notes in Theoretical Computer Science*, vol. 203, n° 5, 2008 :3-24.

[3] ERIC BADOUEL, RODRIGUE TCHOUGONG , CÉLESTIN NKUIMI-JUGNIA, BERNARD FOTSING « Attribute grammars as tree transducers over cyclic representations of infinite trees and

their descriptional composition », *Theoretical Computer Science*, vol. 480, n° 0, 2013 :1–25.

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

[5] 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.

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

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

[8] 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.

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

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

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

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

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

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