Assister l'utilisateur à expliciter un modèle de trace avec l'analyse de concepts formels

Béatrice Fuchs

Univ Lyon, Université Lyon 3, CNRS , LIRIS, UMR5205, F-69372, France. beatrice.fuchs@liris.cnrs.fr

Résumé : Nous proposons d'analyser des jeux de données représentant des traces afin de découvrir le modèle sous-jacent et assister leur intégration dans un système à base de traces dédié au stockage et à la manipulation de traces numériques. Ce principe est mis en oeuvre dans CSV2KTBS, un prototype interactif qui s'appuie sur l'analyse de concepts formels pour générer un modèle de la trace et assister un utilisateur dans le processus d'intégration dans un système à base de traces. L'utilisateur est sollicité pour interpréter les concepts qui lui sont proposés à partir de l'analyse des traces, ainsi que pour valider d'autres résultats obtenus par l'analyse. Ces principes ont été appliqués aux traces du jeu sérieux Tamagocours pour l'apprentissage des règles juridiques de diffusion de ressources numériques ainsi qu'aux traces de ClassCraft, une application ludique pour l'enseignement secondaire.

Mots-clés : Traces, système à base de traces, extraction de connaissances, analyse de concepts formels, modélisation de traces

1 Introduction

Les traces témoignent d'une activité passée et sont précieuses pour étudier et comprendre des phénomènes complexes. Parfois elles constituent le seul matériau disponible sur lequel on peut s'appuyer pour inférer des connaissances et construire un modèle du phénomène étudié. Plus précisément les *traces numériques d'interaction* sont issues de l'observation de l'activité d'un utilisateur lorsqu'il réalise une tâche assistée par un système informatique. Actuellement de plus en plus d'outils sont disponibles pour la gestion de traces permettant leur production, leur collecte et leur exploitation en aval pour des usages variés (analyse, visualisation, raisonnement, découverte de connaissances). Une des premières étapes en vue de l'exploitation des traces est la collecte : les traces doivent d'abord être recueillies à partir d'une source, l'environnement informatique où l'utilisateur réalise une activité, ou un fichier. Puis elles doivent être traitées afin d'être exploitables à des fins d'analyse ou de raisonnement par exemple. De ce fait, il est préférable de les associer à un modèle de sorte que les différents éléments qui les composent soient compréhensibles et que les résultats des diverses analyses réalisées en aval de la collecte soient interprétables. Généralement, les traces sont fournies après le déroulement de l'activité et se présentent sous différentes formes (fichiers texte, csv, tsv, *etc.*). Dans la plupart des cas elles ne sont pas associées à modèle explicite. L'utilisateur qui prend en charge leur intégration dans un environnement d'exploitation des traces, même s'il les connaît bien, ne pourra s'affranchir d'une étape de modélisation qui peut être difficile selon la complexité des traces dont il dispose.

Nous nous intéressons dans cet article à l'assistance à l'intégration de trace dans une base de traces qui explicite de façon automatique un modèle à partir des données qui lui sont fournies, en interaction avec l'utilisateur. Nous proposons une méthode d'analyse interactive à l'aide de l'analyse de concepts formels afin d'expliciter le modèle sous-jacent d'une trace à partir d'un fichier de données brut de type csv. Nous proposons une mise en oeuvre avec les traces de Tamagocours, un jeu sérieux pour l'apprentissage de règles juridiques de diffusion de documents

numériques à des fins pédagogiques.

Dans le premier paragraphe nous présentons les principes d'un système à base de traces modélisées, la difficulté de la collecte, et le cadre du jeu Tamagocours sur lequel nous nous sommes appuyés. Nous détaillons le processus de génération du modèle de trace sur cet exemple. Nous discutons des limites de cette approche puis nous concluons avec quelques perspectives.

2 Traces d'interaction

Un Système à Base de Traces modélisées (SBT) est un système dédié à la modélisation et la manipulation générique de traces numériques (Champin *et al.*, 2013). Les traces sont collectées en enregistrant les actions de l'utilisateur sous la forme d'un ensemble d'actions datées et situées et appelées *éléments observés* ou *obsels* pour *Observed Elements*. Dans un SBT, les traces sont associées à un modèle constituant un premier niveau d'interprétation de la trace. Le modèle de trace décrit les différents *types d'obsel* associés à un ensemble d'*attributs* et de *relations* entre types d'obsel. Un système dédié à la gestion de traces permet de collecter des traces, les stocker et les manipuler à l'aide d'opérations génériques appelées transformations qui sont de différents types : filtrage d'obsels, fusion de traces, etc. Nous utilisons la plateforme kTBS [1] qui réifie cette notion de SBT. Dans kTBS, les traces et leurs modèles sont décrits à l'aide du formalisme RDF et de ce fait nous utilisons par la suite le format TURTLE pour décrire le modèle de trace généré.

2.1 Collecte de traces

Lors de l'intégration de traces dans un kTBS, la première étape consiste à décrire son modèle à partir des informations disponibles sur les traces dont on dispose. L'alimentation de la base de traces ne pose pas de problème lorsque tous les types d'obsel sont caractérisés par les mêmes attributs, mais ce n'est généralement pas le cas. Les types d'obsel sont souvent différenciés ce qui rend plus difficile le processus d'intégration, d'autant plus que l'on ne dispose très souvent que de peu d'informations sur la structure des différents types d'obsel composant la trace. La construction du modèle nécessite alors d'étudier la trace pour répertorier d'abord quels sont les actions représentées dans la trace, et pour chacune d'elles, inventorier les attributs qui les décrivent. Or, les formats de fichiers sont souvent variés et conçus ad-hoc pour les besoins spécifiques d'une application particulière. Dans (Bouvier *et al.*, 2014) par exemple, les traces sont issues d'un jeu en ligne et se présentent sous la forme d'un fichier csv dont les premières lignes sont consacrées à une description très sommaire des différents types d'obsel associés aux attributs les caractérisant. Une colonne est susceptible de représenter différents attributs selon le type d'obsel. Dans l'approche de (Besnaci *et al.*, 2015), l'intégration de multiples sources hétérogènes est réalisée par l'utilisateur qui décrit d'abord son modèle dans le kTBS et le met en correspondance avec les éléments des traces à importer. Dans ces approches, la construction du modèle reste manuelle à la charge de l'utilisateur. Or, dès que le nombre d'attributs et de types d'obsel augmente, la conception du modèle est souvent fastidieuse. De plus les traces peuvent contenir des anomalies qui rendent difficile la conception du modèle et sont susceptibles d'avoir un impact non seulement sur le modèle mais aussi sur les analyses ultérieures des traces. Pour assister l'utilisateur dans le processus d'intégration d'une trace dans un kTBS, nous

1. kernel for Trace Based System

proposons d'exploiter l'analyse de concepts formels (ACF). L'ACF a été largement utilisée dans de nombreux domaines, en ingénierie des ontologies ou en génie logiciel et couvre un large spectre d'applications (Poelmans *et al.*, 2013). Dans le paragraphe suivant nous décrivons le cas d'étude sur lequel nous nous appuyons pour présenter notre approche.

2.2 Tamagocours

Tamagocours (Sanchez *et al.*, 2015) est un jeu destiné à l'apprentissage des règles juridiques auxquelles est soumis l'usage de ressources numériques dans le contexte éducatif. Il a été conçu à l'intention d'étudiants destinés à la carrière d'enseignant dans le cadre de la mise en place du C2I2e [2]. Les utilisateurs sont répartis en équipes de 2 à 4 joueurs et doivent alimenter un «Tamago» en ressources pédagogiques numériques (images, sons, vidéos, livres, articles, publications conçues à des fins pédagogiques, partitions, etc.). Les ressources sont disposées sur une étagère où les utilisateurs peuvent consulter leurs caractéristiques (nature, date de parution, taille de l'extrait, droits d'auteur, etc.), les récupérer et les associer à un mode d'utilisation (présentation orale, projection en classe, sujet d'examen, mise en ligne sur l'intranet, etc.). Les caractéristiques des ressources conjointement à leur mode d'utilisation déterminent si l'utilisation des ressources est autorisée ou non. Les utilisateurs peuvent ensuite placer des ressources dans un réfrigérateur commun à l'équipe où elles restent consultables de la même manière que sur l'étagère, puis les donner au Tamago pour le «nourrir». Le Tamago est associé à un score qui évolue au fur et à mesure des réussites (ressource autorisée) ou échecs (utilisation d'une ressource hors du cadre légal) des actions des utilisateurs du groupe. Une succession d'échecs peut mener au dépérissement du Tamago. Les actions d'un utilisateur sont visibles par l'équipe, et le jeu comporte 5 niveaux de difficulté croissante. Pour mener à bien leur activité, les utilisateurs ont à leur disposition une aide qui comporte l'ensemble des règles juridiques applicables, ainsi qu'une messagerie instantanée afin de dialoguer entre eux et se concerter.

2.3 Les traces

Les traces de 244 utilisateurs ont été collectées dans un fichier au format csv qui représente au total 25 944 actions de jeu. Un extrait de ce fichier est montré dans la table 1. Le fichier, tel qu'il a été reçu initialement, comportait 13 types d'action différents décrits à l'aide de 24 attributs. Les types d'action possibles sont décrits dans la table 2. La collecte initiale comportait l'attribut `logType` représentant le type d'action utilisateur qui a été ensuite combiné à des valeurs d'autres d'attributs pour obtenir l'attribut `actionType` (combinaison de `feedTamago` avec `isWon` pour obtenir `feedTamagoGood` et `feedTamagoBad`. D'autres attributs résultent de combinaisons tels que `resourceTypeMoU` (combinaison de `resourceType` et `mode_of_use`) et `grpus_id` (combinaison de `group_id` et `user_id`).

Pour procéder à l'importation de ces traces dans un système à base de traces, il a d'abord fallu s'intéresser à la construction du modèle, c'est à dire définir l'ensemble des types d'obsel associés chacun à un ensemble d'attributs, et organiser les types d'obsel en hiérarchie d'héritage.

2. Certificat Informatique et Internet niveau 2 enseignant

id	date	actionType	group_id	user_id	grpus_id	Codage	message	help	res_id	item_id	resourceType	mode_of_use	resource_title	creationDate	rightsAgreements	res_size	item_size	reason	game_id	level_id	isWon
5	30/03/2015 11:05:45	fill Cupboard	2													/			2	1	1
7	30/03/2015 11:06:45	help Link	2	3	2_3			Accès aide en ligne								/			2	1	1
8	30/03/2015 11:06:49	tuto	2	3	2_3											/			2	1	1
9	30/03/2015 11:07:00	showItem CUPBOARD	2	3	2_3				119	552	journal		Le Nouvel	1964	Droits d'auteur CFC	/	10 articles		2	1	1
12	30/03/2015 11:07:42	showItem CUPBOARD	2	3	2_3				43	339	book		Le Grand M	1913	Domaine Public	/	4		2	1	1
16	30/03/2015 11:07:52	showItem CUPBOARD	2	3	2_3				113	529	journal		Le Figaro		Domaine public	/	intégrale		2	1	1
17	30/03/2015 11:07:56	showItem CUPBOARD	2	3	2_3				42	326	book		La littéra	2013	Droits d'auteur CFC	/419	1		2	1	1
18	30/03/2015 11:07:59	showItem CUPBOARD	2	3	2_3				43	335	book		Le Grand M	1913	Domaine Public	/	intégrale		2	1	1
19	30/03/2015 11:08:02	showItem CUPBOARD	2	4	2_4				108	508	journal		The Washin	1983	Droits d'auteur	/	5 articles		2	1	1
20	30/03/2015 11:08:19	addTo Fridge	2	3	2_3				43	335	book	printedCopies	Le Grand M	1913	Domaine Public	/	intégrale		2	1	1
21	30/03/2015 11:08:20	showItem CUPBOARD	2	4	2_4				117	541	journal		Science et	1913	Droits d'auteur	/	1 article		2	1	1
22	30/03/2015 11:08:38	chat	2	3	2_3	OJ	Quelqu'un sait ce qu'il faut faire??									/			2	1	1
23	30/03/2015 11:08:39	showItem FRIDGE	2	4	2_4				43	335	book	printedCopies	Le Grand M	1913	Domaine Public	/	intégrale		2	1	1
24	30/03/2015 11:08:45	chat	2	6	2_6	OJ	aucune idée!									/			2	1	1
25	30/03/2015 11:08:45	tuto	2	3	2_3											/			2	1	1
26	30/03/2015 11:09:04	feedTamago Good	2	3	2_3				43	335	book	printedCopies	Le Grand M	1913	Domaine Public	/	intégrale		2	1	1
27	30/03/2015 11:09:25	chat	2	3	2_3	OJ	Je mets des trucs dans le frigo et je lui file à bouffer									/			2	1	1
28	30/03/2015 11:09:30	showItem CUPBOARD	2	4	2_4				108	506	journal		The Washin	1983	Droits d'auteur	/	1 article		2	1	1
...																					
29175	09/04/2015 15:38:42	showItem CUPBOARD	89	177	89_177				407	63	image		Le baiser	1950	Droits d'auteur	/	72 dpi		731	5	1
29177	09/04/2015 15:39:14	addTo Fridge	89	177	89_177				223	686	video	assessments	Le bon la	1966	Droits d'auteur	161 m/	6 min		731	5	1
29178	09/04/2015 15:39:16	feedTamago Good	89	177	89_177				223	686	video	assessments	Le bon la	1966	Droits d'auteur	161 m/	6 min		731	5	1

TABLE 1 – Un extrait du fichier des traces du jeu Tamagocours. Les attributs sont représentés en colonnes, et la colonne `actionType` représente le type d'action réalisé par un utilisateur. Chaque ligne est une action repérée par un identifiant (`id`) et par une estampille (`date`).

3 Génération interactive du modèle de trace

On peut remarquer que la table 1 comporte des intersections vides : pour certains types d'action, il n'y a aucune valeur pour certains attributs. Cela signifie que ces attributs ne s'appliquent pas à ces types d'action. Ces «valeurs manquantes» permettent d'identifier à quels types d'action les attributs correspondants doivent être rattachés. Il est ensuite possible d'en déduire quels types d'action partagent quels attributs. C'est précisément l'objet de l'analyse de concepts formels.

3.1 L'analyse de concepts formels

L'analyse de concepts formels (ACF, (Ganter & Wille, 1999)) vise à représenter sous forme de treillis une relation binaire entre un ensemble d'objets et un ensemble de propriétés, décrite dans un tableau à deux dimensions appelé contexte formel. Elle repose sur une théorie mathématiques (Barbut & Monjardet, 1970) issue de la théorie des treillis. Typiquement, un contexte formel décrit la relation binaire «a pour propriété» entre un ensemble d'objets et un ensemble de propriétés. Un concept formel est défini par une paire (E, I) où E est un ensemble d'objets et I un ensemble d'attributs où :

addToFridge	ajouter une ressource dans le frigo
chat	envoyer un message
feedTamagoBad feedTamagoGood	Nourrir le Tamago avec une bonne ou une mauvaise ressource
fillCupboard	Initialisation de l'étagère par le logiciel avec un ensemble de ressources
helpLink	affichage de l'aide
removeFromFridge	supprimer une ressource du frigo
showItemCUPBOARD showItemFRIDGE, showItemLEVEL showItemTAMAGO showItemSTOMACH	examiner une ressource placée sur l'étagère examiner une ressource dans le frigo, examiner une ressource dans le tableau de fin de niveau, examiner une ressource dans le Tamago. examiner les ressources dans l'estomac du Tamago.
tuto	Consultation du tutoriel

TABLE 2 – Les différents types d'action utilisateur dans le jeu Tamagocours.

— E est appelé l'extension et contient l'ensemble des objets partageant les attributs de I,
— I est appelé l'intension et contient l'ensemble des attributs partagés par les objets de E.

Les concepts formels peuvent être ordonnés en hiérarchie sous la forme d'un treillis de concepts appelé encore treillis de Galois, où les relations d'ordre partiel entre les ensembles d'objets d'une part et les ensembles d'attributs d'autre part forment une correspondance de Galois. L'ACF extrait les concepts à partir d'un contexte formel qui représente une relation binaire entre les individus et les propriétés, présenté dans la section suivante.

3.2 Contexte formel

Il a donc fallu dans un premier temps répertorier la liste des types d'action de Tamagocours et déterminer pour chacune d'elles la liste des attributs pour lesquels ils sont définis. Le tableau 3 représente cette relation binaire «a pour attribut», entre les types d'action et les attributs : c'est un contexte formel.

Définition 1 (Contexte formel)
Soient deux ensembles finis O et A, et une relation $R \subseteq O \times A$.
Deux fonctions sont associées à la relation R :

— *La fonction f permet de connaître les attributs partagés par un ensemble d'objets :*
$f : \mathcal{P}(O) \to \mathcal{P}(A), X \mapsto f(X) = \{y \in A | \forall x \in X, (x, y) \in R\}$
— *La fonction g permet de connaître les objets partageant un ensemble d'attributs*
$g : \mathcal{P}(A) \to \mathcal{P}(O), Y \mapsto f(Y) = \{x \in O | \forall y \in Y, (x, y) \in R\}$

Plus précisément, O est l'ensemble des types d'action Tamagocours, et A l'ensemble des attributs utilisés pour les décrire. Le contexte formel est défini de la façon suivante : pour chaque type d'action $o_i \in O$, pour chaque attribut $a_j \in A$, soit $V_{i,j}$ l'ensemble des valeurs prises par les objets de type o_i pour l'attribut a_j, $(o_i, a_j) \in R \iff V_{i,j} \neq \emptyset$. Il s'agit donc de répertorier les couples $(a, o) \in A \times O$ pour lesquels il existe au moins une ligne possédant une valeur.

	showItem STOMACH	addTo Fridge	help Link	tuto	fill Cupboard	showItem CUPBOARD	showItem FRIDGE	showItem LEVEL	showItem TAMAGO	remove FromFridge	feedTamago Good	chat	feedTamago Bad
actionType	X	X	X	X	X	X	X	X	X	X	X	X	X
codage						X						X	X
creationDate	X	X				X	X	X	X	X	X		X
date	X	X	X	X	X	X	X	X	X	X	X	X	X
game_id	X	X	X	X	X	X	X	X	X	X	X	X	X
group_id	X	X	X	X	X	X	X	X	X	X	X	X	X
grpus_id	X	X	X	X	X	X	X	X	X	X	X	X	X
help			X										
id	X	X	X	X	X	X	X	X	X	X	X	X	X
isWon	X	X	X	X	X	X	X	X	X	X	X	X	X
item_id	X	X				X	X	X	X	X	X		X
item_size	X	X				X	X	X	X	X	X		X
level_id	X	X	X	X	X	X	X	X	X	X	X	X	X
logType	X	X	X	X	X	X	X	X	X	X	X	X	X
message												X	
mode_of_use	X	X					X	X	X	X	X		X
reason	X	X					X	X	X	X			X
resource_id	X	X				X	X	X	X	X	X		X
resource_size	X	X	X	X	X	X	X	X	X	X	X	X	X
resource_title	X	X				X	X	X	X	X	X		X
resourceType	X	X				X	X	X	X	X	X		X
resourceTypeMoU	X	X				X	X	X	X	X	X		X
rightsAgreements	X	X				X	X	X	X	X	X		X
user_id	X	X	X	X		X	X	X	X	X	X	X	X

TABLE 3 – La relation entre les types d'action et les attributs. Les colonnes représentent les types d'action et les lignes les attributs utilisés pour les décrire.

3.3 Concepts formels

A partir du contexte formel, il est possible de construire un treillis de concepts. Il s'agit de déterminer les sous-ensembles maximaux d'objets (l'extension, ici les types d'obsel) qui partagent des sous-ensembles maximaux d'attributs (l'intension).

Définition 2 (Concept formel)
Un concept formel C *est un couple* (E, I) *tel que* $f(E) = I$ *ou de façon équivalente,* $E = g(I)$.
$E = \{o \in O | \forall a \in A, (o, a) \in R\}$ *est appelé l'extension du concept,*
$I = \{a \in A | \forall o \in o, (o, a) \in R\}$ *est appelé l'intension du concept.*

La figure 1 montre le treillis obtenu sur les traces à partir du contexte formel de la table 3. Les nœuds sont étiquetés par trois types d'informations : un numéro de nœud, l'ensemble des types d'action, c'est-à-dire l'extension, puis l'ensemble des attributs partagés c'est-à-dire l'intension. Le nœud le plus haut étiqueté c0 souvent appelé «Top» correspond à l'ensemble des attributs partagés par tous les types d'action. Ces attributs sont «hérités» par tous les nœuds placés «en dessous» dans la hiérarchie (ils ne sont pas répétés dans la figure pour une meilleure lisibilité). Le nœud le plus bas étiqueté c10 souvent appelé «Bottom» correspond à l'ensemble des types d'action partageant tous les attributs. Dans la figure 1, il n'y en a aucun. De la même façon que pour les attributs, les types d'action sont «hérités» mais cette fois de façon ascendante dans les nœuds situés «au dessus» dans la hiérarchie. Sur le treillis de la figure 1 des éléments sont mis en évidence car il montrent des anomalies au niveau des concepts c0-c1, c5-c7 et c2-c4, c2-c6-c9 : ces concepts ne peuvent être interprétés et semblent ne pas avoir de sens.

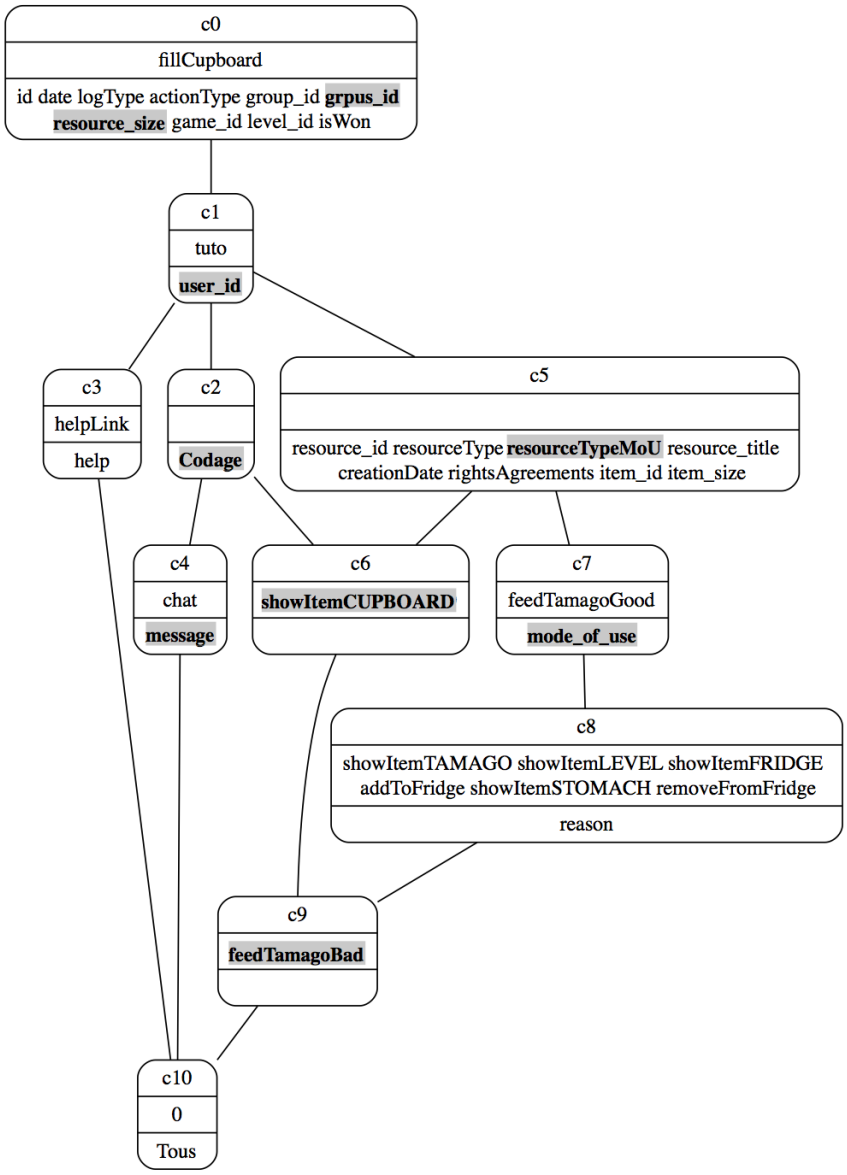

FIGURE 1 – Le treillis de concepts obtenu à partir du contexte formel de la table 3.

3.4 Rectification des données

Les modifications des données réalisées après la collecte ont généré des erreurs. C'est le cas en particulier pour les attributs `grpus_id`, `resourceTypeMoU` et `resource_size` :

— `grpus_id` est une concaténation de `group_id` et de `user_id`. L'action `fillCupboard` est générée lors du remplissage de l'étagère avec des ressources au début d'un jeu. Elle n'est pas associée à un utilisateur mais à un groupe et ne comporte donc pas de valeur pour `user_id`. La concaténation des deux attributs n'a pas pris la précaution de vérifier l'existence d'une valeur pour les deux attributs. Après vérification, 663 obsels possèdent une valeur pour `grpus_id` mais pas pour `user_id` :

ce sont toutes les actions de type `fillCupboard`. On retrouve la même anomalie avec l'attribut `resourceTypeMoU` qui résulte de la concaténation de `resourceType` et `mode_of_use`. Toutes les actions non associées à l'attribut `mode_of_use` ont une valeur pour `resourceTypeMoU`, sans `mode_of_use`. Après vérification, on constate qu'il s'agit de 9522 actions de type `showItemCUPBOARD`. Enfin de la même façon, `resource_size` ne se retrouve pas associé aux attributs liés à une ressource dans le concept pour la même raison.

— Les chats sont délicats à traiter car ils sont peu contextualisés : on ne peut pas connaître la ressource sur laquelle ils portent, ou leur objet. Pour cette raison un codage manuel a été réalisé afin de préciser leur objet. Compte tenu du nombre important d'actions dans la trace, des erreurs ont pu être introduites. D'après la figure 1, et après vérification, il s'avère que deux actions `feedTamagoGood` et `showItemCUPBOARD` ont été enrichies par erreur d'un champ codage qui a été introduit sur une mauvaise ligne, d'où la présence des concepts c6 et c7 qui n'ont pas lieu d'exister.

A l'issue de la correction des anomalies dans les données, on obtient le treillis de la figure 2.

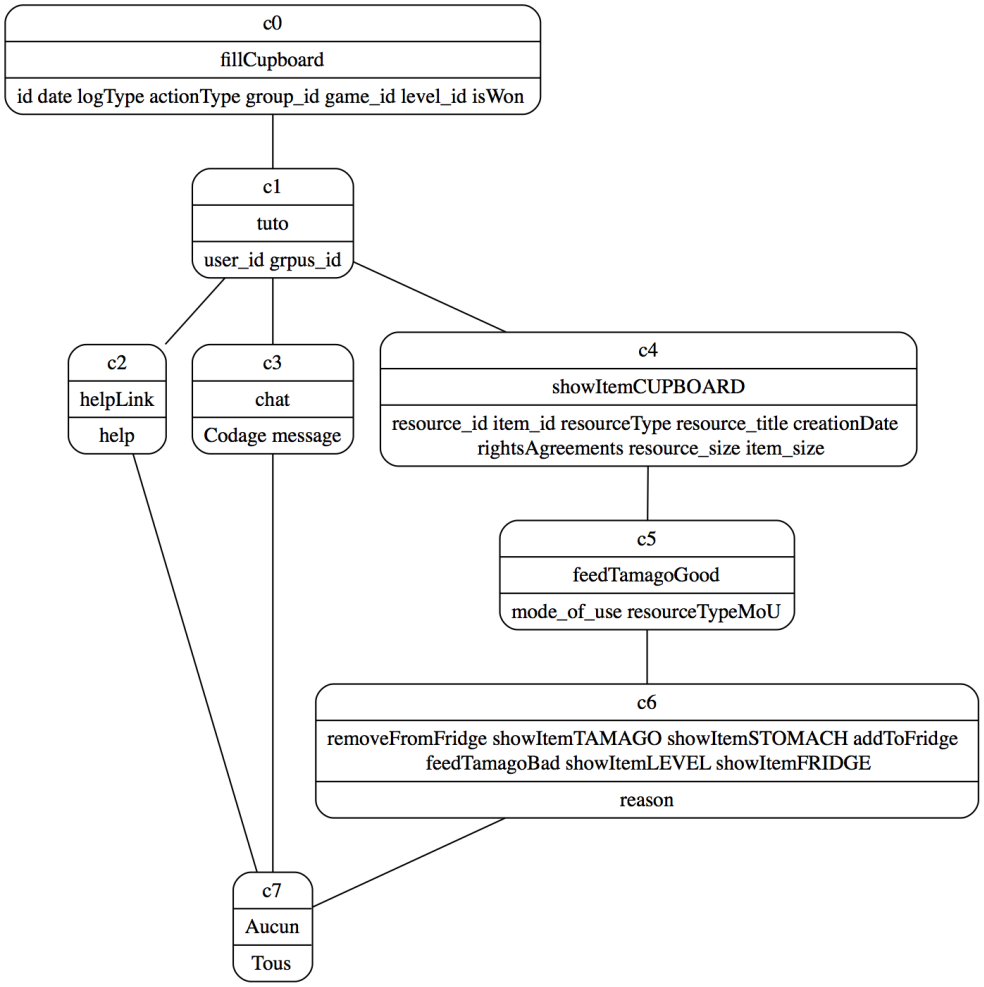

FIGURE 2 – Le treillis obtenu à l'issue de la correction des erreurs.

3.5 Interprétation du treillis

Après correction des erreurs il devient possible d'interpréter les différents concepts du treillis de la figure 2. On remarque qu'il est possible de supprimer le concept «Bottom» qui comporte tous les attributs mais aucun type d'action. On obtient la hiérarchie du modèle de trace de la figure 3.

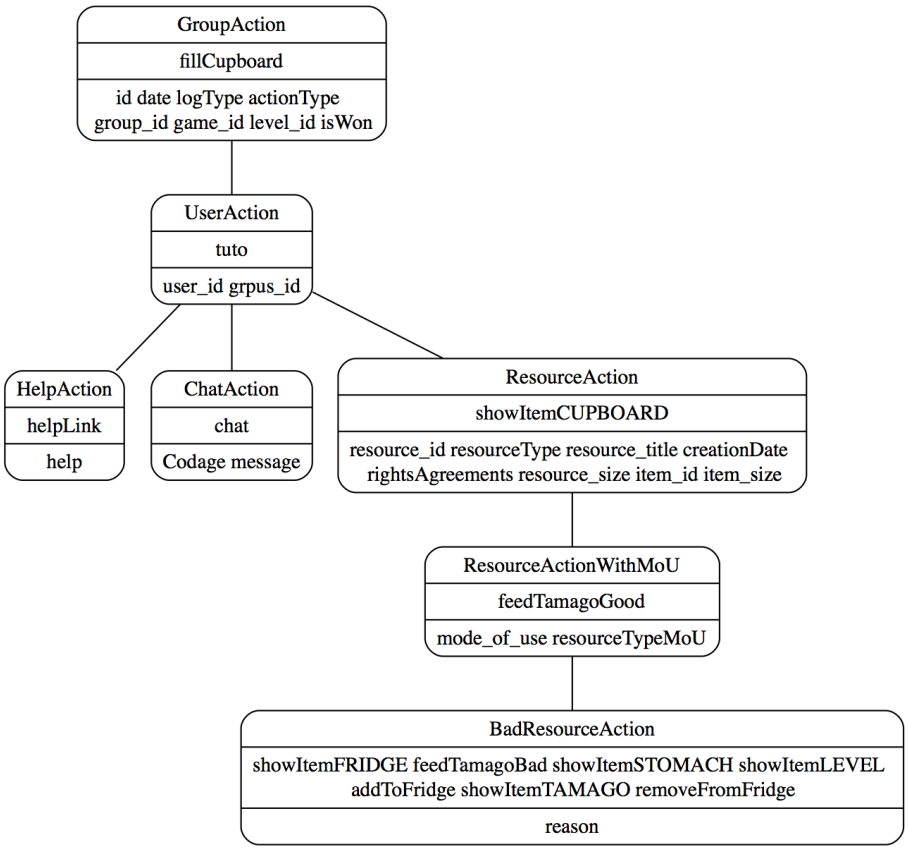

FIGURE 3 – Le modèle de trace obtenu à partir du treillis de concepts auquel l'élément «Bottom» a été supprimé et les concepts étiquetés.

Les concepts du treillis obtenu ont été interprétés de la façon suivante : Le type d'obsel `GroupAction` comporte les attributs qui sont partagés par toutes les actions, ce qui correspond au concept de groupe d'utilisateurs sur lequel le jeu est fondé. Le type d'obsel `UserAction` comporte les attributs correspondant à des actions d'utilisateurs, notamment l'identifiant de l'utilisateur. Le type d'obsel `ResourceAction` décrit les actions de manipulation des ressources sans mode d'utilisation. Le type d'obsel `ResourceActionWithMoU` décrit les actions de manipulation des ressources associées à un mode d'utilisation. Enfin le type d'obsel `BadResourceAction` représente les actions d'alimentation du Tamago avec des ressources et mode d'utilisation non autorisés et précise la raison de l'échec (`reason`).

Pour générer le modèle de trace, il suffit ensuite d'exploiter les informations présentes dans chaque concept du treillis de la figure 3 : les intensions et les extensions. Tout d'abord, les concepts formels sont utilisés pour générer des types d'obsel qui sont associés chacun à un

ensemble d'attributs (l'intension) partagé par les types d'action (l'extension). Les types d'action du jeu sont représentés par les types d'obsel qui héritent des définitions des types d'obsel issus des concepts formels ci-dessus. Par exemple le concept `UserAction` est représenté par un type d'obsel caractérisé par les attributs `user_id` et `grpus_id`, il hérite du type d'obsel `GroupAction` tous les attributs `id`, `date`, etc. Le type d'action `tuto` hérite du type d'obsel `UserAction`. Cette première étape de construction du modèle de trace permet d'obtenir un regroupement des attributs en types d'obsel et la liste des types d'obsel correspondant aux types d'action du jeu qui en héritent dans le modèle. Une deuxième étape précise le type et les caractéristiques des attributs et génère le modèle au format TURTLE pour le kTBS.

3.6 Génération du modèle de trace

Les données sont analysées de façon à déterminer les caractéristiques des valeurs prises par chaque attribut (liste des valeurs, intervalles de valeurs, etc.). En particulier pour chaque attribut, un type de données est détecté (nombre entier, nombre fractionnaire, booléen, chaîne de caractères, date et heure), et les attributs jouant le rôle d'identifiant sont détectés. Tous ces résultats sont proposés à l'utilisateur qui peut les valider ou les modifier. Cette analyse des types n'est pas détaillée davantage ici car elle sort du cadre de ce travail. Elle vise simplement à faciliter le travail de l'utilisateur. Ce processus est de plus perfectible, seuls les quelques types énumérés précédemment sont pris en compte. Ensuite, le modèle de trace est généré à l'aide de la syntaxe TURTLE. Un extrait du modèle de trace généré à partir du treillis de la figure 3 est présenté ci-dessous :

```
@prefix : <http://liris.cnrs.fr/silex/2009/ktbs#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://localhost:8001/TamagocoursBase/> :contains <> .
<> a :TraceModel ;
    :hasUnit :second .
<#GroupAction> a :ObselType .
    <#id> a :AttributeType ;
        :hasAttributeDomain <#GroupAction> ;
        :hasAttributeRange xsd:integer.
    <#date> a :AttributeType ;
        :hasAttributeDomain <#GroupAction> ;
        :hasAttributeRange xsd:dateTime.
    <#actionType> a :AttributeType ;
        :hasAttributeDomain <#GroupAction> ;
        :hasAttributeRange xsd:string.
    <#group_id> a :AttributeType ;
        :hasAttributeDomain <#GroupAction> ;
        :hasAttributeRange xsd:integer.
    [...]
<#fillCupboard> a :ObselType ;
    :hasSuperObselType <#GroupAction> .
<#UserAction> a :ObselType ;
    :hasSuperObselType <#GroupAction> .
    <#user_id> a :AttributeType ;
        :hasAttributeDomain <#UserAction> ;
        :hasAttributeRange xsd:integer.
    <#grpus_id> a :AttributeType ;
        :hasAttributeDomain <#UserAction> ;
        :hasAttributeRange xsd:string.
<#tuto> a :ObselType ;
    :hasSuperObselType <#UserAction> .
[...]
```

Une fois le modèle de trace généré, l'importation de la trace peut être réalisée. Cette étape est opérationnelle et brièvement décrite ci-après. Elle repose sur l'analyse des attributs, en particulier les attributs de type date. Il s'agit de déterminer l'unité de temps à utiliser pour calculer les estampilles associées à chaque obsel, conformément aux unités de temps utilisées par le kTBS : seconde, milli-seconde, ou numéro séquentiel. La plus petite date est utilisée comme référence et les estampilles de chaque obsel sont calculées comme un nombre d'unités de temps depuis depuis la date de référence.

4 Discussion

Actuellement, la génération du modèle implémentée dans CSV2KTBS, ainsi que celle de la trace est entièrement opérationnelle, mais plusieurs points peuvent être améliorés. Il reste à développer une interface graphique pour améliorer l'interactivité, à connecter CSV2KTBS à un kTBS pour procéder à l'importation en «temps réel» du modèle et de la trace, et à prendre en compte les relations entre les obsels. CSV2KTBS est capable de traiter très rapidement une trace de presque 25 944 obsels. La construction du modèle ne souffre pas de la taille importante de la trace puisque c'est le nombre d'attributs et de types d'action qui sont prises en compte pour la construction du treillis de concepts, et ce nombre est relativement limité. Antérieurement, le travail de modélisation avait été réalisé «à la main» en élaborant manuellement un contexte formel à partir des données. Le modèle obtenu était très similaire mais comportait de nombreuses imperfections liées aux erreurs dans les données. L'intérêt de l'ACF s'est imposé de façon assez évidente et son utilisation a montré que l'ACF peut constituer un outil puissant d'assistance à l'explicitation d'une sémantique «cachée» dans des données, difficile et fastidieuse à réaliser manuellement dès lors que le nombre de types d'action et d'attributs dépasse une ou deux dizaines. Par ailleurs, l'explicitation d'une hiérarchie d'héritage des attributs présente un intérêt important pour les utilisations en aval de la trace à des fins d'analyse. Certaines méthodes de fouille de données peuvent tirer parti d'une telle hiérarchie, par exemple (Marinica *et al.*, 2008) ou (Brisson & Collard, 2008). La principale limitation de l'approche présentée dans cet article est qu'elle ne peut s'appliquer que si chaque colonne ne représente qu'un seul attribut. Ainsi, lorsqu'un type d'attribut n'est pas défini pour un type d'action, alors les intersections correspondantes dans le tableau de données sont vides, et c'est cette caractéristique de la trace qui permet de construire le contexte formel de la relation binaire. La façon dont les données tabulaires sont organisées dans les fichiers de types `csv` est très variable et cette approche n'est pas toujours applicable. Cette limitation a été constatée dans les traces du jeu sérieux ClassCraft [3], une application ludique destinée à favoriser l'apprentissage dans l'enseignement secondaire. La trace comportait 13 488 lignes correspondant aux actions réalisées par les utilisateurs avec 11 types d'action et 22 attributs. Le treillis a été généré correctement, mais le travail d'interprétation a été rendu difficile par le manque de documentation suffisante sur la signification des attributs et des types d'action. De plus, certaines colonnes représentent plusieurs attributs différents selon le type d'action. Afin de mieux capturer la signification de chaque attribut «caché», une phase supplémentaire en interaction avec l'utilisateur est nécessaire afin de différencier et expliciter les attributs selon le type d'action.

3. `https://www.classcraft.com`

5 Conclusion

Dans cet article nous présentons une approche pour la génération automatique et interactive d'un modèle de trace à partir de données au format csv à l'aide de l'analyse de concepts formels, mise en œuvre dans le prototype CSV2KTBS. L'approche a été appliquée sur les traces du jeu Tamagocours, ce qui a permis de mettre en lumière les anomalies contenues dans la trace et de contribuer à améliorer la qualité des données. Le modèle est généré d'abord sous forme graphique pour être plus facilement compréhensible par l'utilisateur. Après l'interprétation des concepts, le modèle de trace est généré au format TURTLE, puis la trace elle-même. L'approche est opérationnelle, générique et s'est avérée applicable sur d'autres traces. Il reste à développer une interface graphique pour améliorer l'interactivité du prototype, et à prendre en compte la différenciation des attributs lorsqu'ils prennent des significations différentes en fonction des types d'action. Une autre perspective est la conception d'un processus interactif qui permette la détection et la correction d'erreurs dans la source de données analysée, et la génération de requêtes en conséquence afin d'aider l'utilisateur à repérer ces erreurs et les corriger.

Références

BARBUT M. & MONJARDET B. (1970). *Ordre et classification : Algèbre et combinatoire.* Hachette, Paris.

BESNACI M., GUIN N. & CHAMPIN P.-A. (2015). Acquisition de connaissances pour importer des traces existantes dans un système de gestion de bases de traces. In *IC2015*, Rennes, France : AFIA.

BOUVIER P., SEHABA K. & LAVOUÉ E. (2014). A trace-based approach to identifying users' engagement and qualifying their engaged-behaviours in interactive systems : application to a social game. *User Modeling and User-Adapted Interaction*, **24**(5), 413–451.

BRISSON L. & COLLARD M. (2008). How to semantically enhance a data mining process ?. In *ICEIS*, volume 19, p. 103–116 : Springer.

CHAMPIN P.-A., MILLE A. & PRIÉ Y. (2013). Vers des traces numériques comme objets informatiques de premier niveau : une approche par les traces modélisées. *Intellectica*, (59), 171–204.

GANTER B. & WILLE R. (1999). *Formal Concept Analysis.* Springer, mathematical foundations edition.

MARINICA C., GUILLET F. & BRIAND H. (2008). Post-processing of discovered association rules using ontologies. In *Workshops Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, p. 126–133.

POELMANS J., IGNATOV D. I., KUZNETSOV S. O. & DEDENE G. (2013). Formal concept analysis in knowledge processing : A survey on applications. *Expert systems with applications*, **40**(16), 6538–6560.

SANCHEZ E., EMIN-MARTINEZ V. & MANDRAN N. (2015). Jeu-game, jeu-play, vers une modélisation du jeu. Une étude empirique à partir des traces numériques d'interaction du jeu Tamagocours. *STICEF*, **22**.