Rendre interactive la découverte d'automates à partir de traces d'activités

Benoît Mathern, Alain Mille, Thierry Bellet

Rendre interactive la découverte d'automates à partir de traces d'activités

Benoît Mathern[1,2], Alain Mille[1], and Thierry Bellet[2]

[1] Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69621, France
benoit.mathern@liris.cnrs.fr et alain.mille@liris.cnrs.fr

[2] IFSTTAR, LESCOT, 25 avenue François Mitterrand, 69675 Bron cedex, France
benoit.mathern@ifsttar.fr et thierry.bellet@ifsttar.fr

Résumé : L'article s'intéresse à la découverte interactive de connaissances lors d'un processus d'analyse de comportements à partir de traces d'activités. Ce processus inclut une fouille de données produisant des motifs que les analystes doivent interpréter et synthétiser en connaissances nouvelles. La découverte de motifs est une tâche d'analyse tandis que la construction de connaissances est une tâche de synthèse. Une plateforme d'aide à l'analyse a été proposée avec ABSTRACT. Cet article propose d'aller plus loin en assistant la tâche de synthèse de connaissances. Un algorithme de synthèse d'automate est modifié de façon à impliquer l'analyste dans le processus de découverte, complétant les connaissances issues du corpus de traces utilisé. Nous nous sommes basés sur l'algorithme-α (Van Der Aalst et al.) pour illustrer ce principe. L'algorithme a été modifié pour permettre une convergence interactive vers un automate pertinent. Nous discutons cette nouvelle façon d'envisager un processus de découverte de connaissances.

Mots-clés : Découverte de connaissances, Réseau de Pétri, Découverte Interactive.

1. Introduction

La *découverte interactive de connaissances* est considérée ici comme une approche d'ingénierie des connaissances. Il s'agit d'intégrer données, informations et personnes dans une même perspective de construction de connaissances et un certain nombre d'outils traitent de cette intégration *sémantique* (Noy, 2004) : méthodes et outils de partage d'ontologies ; algorithmes, heuristiques et apprentissage automatique pour l'alignement d'ontologies ; et, quand l'expertise est peu explicite, techniques d'apprentissage automatique pour

fouiller les données, les textes et les logs (Omelayenko, 2001). Si la fouille peut être automatique, la découverte de connaissances relève de l'activité humaine. Les motifs *candidats* produits par une fouille sont interprétés comme connaissances par l'humain ; les sources fouillées (données, textes, événements, etc.) doivent être soigneusement *préparées*. Au-delà de ces tâches bien connues des analystes, notre objectif général est d'accompagner la création de nouvelles connaissances à partir des *motifs* produits par la fouille.

Cet article propose tout d'abord, à la section 2., une révision du cadre général de la découverte de connaissances lors de l'analyse de traces d'interactions issues d'une activité complexe pour la synthétiser sous une forme documentaire (interprétation humaine) et computationnelle (exploitable par un moteur). Nous appliquons ce cadre conceptuel à la découverte de connaissances lors de l'analyse de traces d'interactions issues de l'activité de conduite automobile en proposant une démarche interactive de découverte d'automate représentant les comportements observés dans les traces. Cette contribution, présentée à la section 3., vise à démontrer la faisabilité de l'approche en modifiant un algorithme classique de découverte d'automates pour introduire l'interactivité nécessaire à l'association des connaissances synthétiques de l'analyste et des connaissances potentielles représentées par les motifs découverts dans les traces. Enfin, la section 4. discute les résultats de la contribution et présente les perspectives dans le contexte de la découverte interactive de connaissances.

2. Découverte interactive de connaissances à partir de traces modélisées

2.1. Trace modélisée et Système à Base de Traces modélisées

Les *traces d'interactions* gagnent à être considérées comme des objets informatiques *à part entière*, c'est-à-dire possédant des propriétés précises, une représentation respectant une sémantique formalisée et des méthodes d'exploitation spécifiques. L'équipe SILEX du LIRIS a développé les notions de *traces modélisées ($\mathcal{M}$-Traces)* et de *Système à Base de Traces (SBT)* (Laflaquière *et al.*, 2006). Une *$\mathcal{M}$-Trace* est un couple constitué d'une séquence d'éléments observés temporellement situés (trace instanciée) et du modèle de cette séquence qui donne la sémantique des éléments observés et des relations qu'ils entretiennent. Un *Système à Base de Traces* est une structure informatique gérant de telles $\mathcal{M}$-Traces et fournissant des services orientés traces comme : collecter une $\mathcal{M}$-Trace, exécuter des requêtes de recherche de motifs

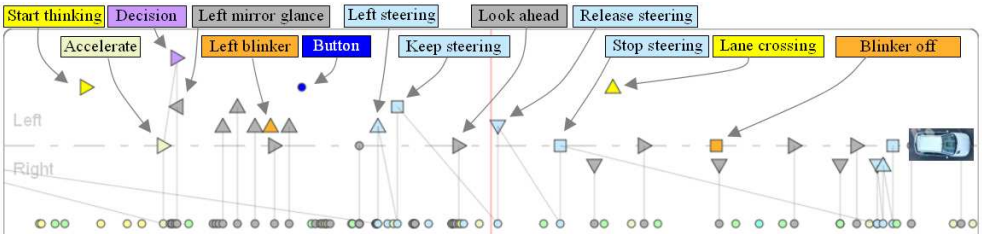

FIGURE 1: Interface ABSTRACT pour l'analyste : l'utilisation d'un SBT permet d'explorer plusieurs hypothèses, de naviguer entre traces transformées, de les comparer en ayant les modèles disponibles pour inspecter les éléments observés.

dans une $\mathcal{M}$-Trace, transformer une trace en une autre trace pour des besoins de reformulation, d'abstraction, de filtrage, de fusion, etc., gérer la navigation dans le graphe des transformations de $\mathcal{M}$-Traces... Dans le contexte de cette communication, ces services sont disponibles à travers le SBT implanté dans le cadre logiciel ABSTRACT. L'utilisation d'un SBT permet de *préparer* et *transformer* les séquences d'événements en tant que $\mathcal{M}$-Traces, dans le registre qui convient à l'observation, associé à une sémantique explicite et avec une procédure de préparation *tracée et navigable*, autorisant des retours en arrière et des reprises facilitées du processus de découverte de connaissances. Le cadre logiciel ABSTRACT (Georgeon *et al.*, 2006) a été développé de façon à assister le processus de découverte de connaissances mené par des analystes du comportement humain en situation de conduite automobile (Georgeon *et al.*, 2007). Un grand volume de données est collecté pendant des séances de conduite sur route. L'analyste utilise ABSTRACT (figure 1) pour progressivement transformer les traces issues de la collecte en traces représentant des connaissances découvertes. La requête permettant de trouver un motif est prise en compte comme la *signature* d'un nouveau concept qui peut être ajouté à l'ontologie utilisée pour décrire l'activité dans une trace.

2.2. Vers un processus de découverte interactive de connaissances

Fayyad *et al.* (1996) proposent une synthèse du processus général de découverte de connaissances à partir de données (figure 2). Nous proposons une approche qui reprend ce processus général dans le cadre de données issues de l'observation d'une activité (voir figure 3). L'utilisation d'un système à base de traces offre des possibilités d'interactivité dans le processus de préparation

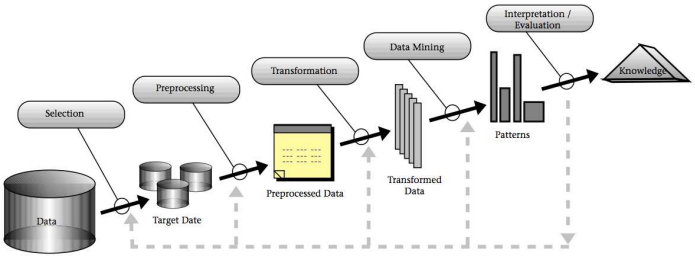

FIGURE 2: La découverte de connaissances est vue comme un processus itératif avec peu d'interactions. Figure de Fayyad *et al.* (1996).

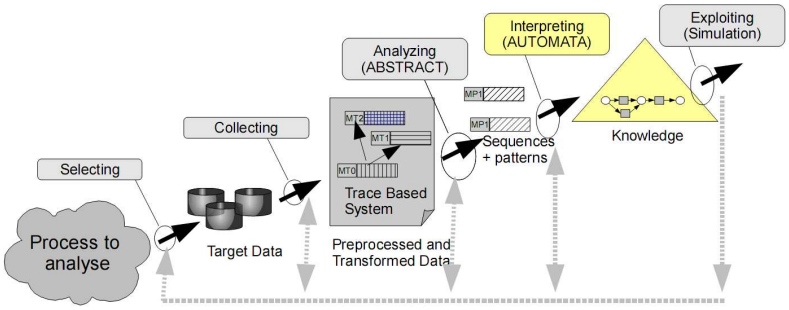

FIGURE 3: Dans notre approche l'analyste fouille interactivement les traces pour produire des motifs (étape ABSTRACT). La connaissance explicitée pilote les étapes de collecte et d'analyse. Une assistance (AUTOMATA) à l'étape de *synthèse* ouvre la voie à l'*exploitation des connaissances* qui conclut le processus. La dynamique du processus observé est représentée par un réseau de Pétri.

des données tout en introduisant progressivement les modèles de connaissances nécessaires à la description de l'observation. Ces connaissances sont explicites et sont directement exploitables pour guider les recherches de motifs et la disponibilité de traces transformées à des niveaux d'abstraction variés permet de considérer plusieurs points de vue simultanés, explicites et synchronisés. La visualisation interactive des résultats des hypothèses d'analyse, la navigation entre les différentes analyses et la synchronisation aux données sources fournissent des outils d'interaction exploitables par les analystes aux stades précoces de préparation de données, comme aux stades avancés de vérification d'hypothèses. De plus, et suivant en cela l'idée défendue par

Michalski (2003) puis plus récemment par Hammori *et al.* (2006), nous proposons de compléter ce processus de découverte de connaissances par une étape de synthèse interactive des motifs en une représentation opérationnelle des connaissances (figure 3), avec l'étape AUTOMATA (AUTOmata Modelling of the Activity, based on Trace Analysis). Cette approche *orientée connaissances* offre des opportunités de contrôles et de feedbacks à chaque étape du processus de découverte en fournissant à la fois une représentation formelle des connaissances et des interactions faciles avec les analystes. Nous visons une approche fournissant à l'analyste l'interactivité nécessaire à la compréhension, au contrôle et *in fine* à l'appropriation de l'ensemble du processus de découverte de connaissances.

3. Découverte interactive d'automates à partir de motifs séquentiels issus de traces de conduite automobile

Dans le cadre de la découverte interactive de connaissances, nous proposons d'établir une étape d'assistance à *l'interprétation* (description de l'automate) à l'aide d'un processus de fouille de traces se focalisant sur les occurrences de motifs satisfaisant une requête particulière. Ce processus est appliqué à l'analyse de données issues de la conduite automobile (Mathern *et al.*, 2010). Pour concevoir une telle assistance, nous modifions les algorithmes capables de synthétiser des automates à partir de traces de comportement. À titre d'illustration, nous avons choisi l'algorithme proposé par van der Aalst *et al.* (2002) dans le contexte de la gestion des workflows. Cet algorithme fait l'hypothèse d'une couverture complète du comportement observé par les traces, ce qui n'est naturellement pas le cas en situation de découverte initiale. Pour résoudre cette incomplétude, nous proposons une version *interactive* de l'algorithme, permettant à l'analyste d'exploiter ses propres connaissances.

3.1. État de l'art

3.1.1. Construire un automate à partir de traces

Un automate possède une facette documentaire et une facette computationnelle. La représentation descriptive est efficace par sa forme graphique normalisée accessible à l'humain formé à sa lecture et les moteurs de simulation permettent de compléter utilement les observations réelles qui ont mené à cette synthèse.

Les automates sont effectivement utilisés à la fois pour produire des comportements (modèles de comportement, simulation) et pour décrire, spécifier des comportements (par exemple les diagrammes d'états UML). Si l'automate décrit l'*intension* d'un comportement, les traces qu'il est capable de produire en fournissent l'*extension*. Cette dualité permet d'*apprendre* un automate à partir de l'observation des traces qu'il produit. La littérature propose plusieurs façons de découvrir des automates représentant un ensemble donné de traces. Puisque nous nous intéressons spécifiquement à l'observation de comportements, nous nous concentrons sur les approches de *process mining* (van der Aalst & Weijters, 2004; van der Aalst *et al.*, 2003). Typiquement, il s'agit d'analyser le workflow d'une entreprise à partir des traces extraites de progiciels tels que les ERP (Entreprise Resource Planning). Dans ce contexte, un modèle prescriptif du workflow existe (ce que les employés sont censés faire). Cependant, il n'y a aucune garantie que le personnel respecte ce modèle dans son activité. L'objectif est donc de découvrir le workflow réel, celui qui se produit en pratique, afin de le comparer au workflow *prescrit* et d'étudier comment résoudre les écarts observés en agissant sur la prescription et/ou sur la formation, les contrôles, etc.

3.1.2. Les défis de la fouille de workflows

Van der Aalst & Weijters (2004) ont effectué une revue de la littérature sur la fouille de workflows. Les algorithmes de fouille de workflow sont confrontés à un certain nombre de difficultés : les tâches « cachées » (une tâche qui n'apparaît pas dans les traces) ; les tâches « dupliquées » (une même tâche qui correspond à plusieurs transitions du modèle) ; les boucles ; les différents points de vue (à partir d'un même jeu de traces, étudier différents aspects de l'activité) ; le bruit des données ; l'incomplétude des données ; la visualisation des résultats…

Van der Aalst *et al.* (2003) présentent quatre algorithmes de fouille de processus : EMiT (van der Aalst & van Dongen, 2002), Little Thumb (Weijters & van Der Aalst, 2002), InWoLvE (Herbst, 2001) et Process Miner (Schimm, 2002). Les auteurs comparent les capacités de ces quatre algorithmes à redécouvrir des modèles connus. Dans le cadre de notre objectif applicatif, l'analyse du comportement de conduite automobile, nous avons sélectionné quatre propriétés : (1) la gestion du temps, puisque la conduite est une activité dynamique avec une forte composante temporelle, (2) la gestion du parallélisme, puisque plusieurs processus se déroulent en parallèle, (3) la gestion

TABLE 1: Comparaison de quatre approches de fouille de processus, adapté de van der Aalst *et al.* (2003).

Approche	Gestion du temps	Gestion du parallélisme	Gestion des boucles	Gestion du bruit
EMiT	Oui	Oui	++	Non
Little Thumb	Non	Oui	++	Oui
InWoLvE	Non	Oui	+	Oui
Process Miner	Non	Oui	+	Non

des boucles, car la conduite est une activité contrôlée, et (4) la gestion du bruit, lié d'une part à la collecte sur véhicule instrumenté ou sur simulateur de conduite et d'autre part à la difficulté de représenter les données collectées au bon niveau d'abstraction.

Le tableau 1 présente une comparaison de ces algorithmes au regard des critères que nous avons sélectionnés. Cette synthèse montre que l'approche EMiT est la seule à prendre en compte le temps (les autres algorithmes ne prennent en compte que la séquentialité), mais EMiT ne gère pas les données bruitées. Little Thumb et InWoLvE gèrent tous deux le bruit. Enfin, Process Miner ne prend en compte ni le temps ni le bruit. Tous ces algorithmes sont capables de gérer le parallélisme et les boucles.

La capacité de gérer le temps (EMiT) ou le bruit (Little Thumb ou InWoLvE) est discriminante pour notre contexte. Étant donné qu'EMiT et Little Thumb sont tous deux basés sur le même algorithme, l'algorithme-α, nous proposons de focaliser notre contribution sur cet algorithme commun. En rendant l'algorithme-α interactif, notre but est de pouvoir gérer la question de l'incomplétude des données : l'analyste pourra fournir à l'algorithme des connaissances supplémentaires à un niveau d'abstraction intermédiaire entre la trace et l'automate.

L'algorithme-α constitue notre première cible pour effectuer une preuve de concept du principe d'interactivité tel que nous le proposons. Cet algorithme ne permet pas par lui-même de gérer tous les types de boucles, mais constitue la base des approches EMiT et Little Thumb.

3.1.3. L'algorithme-α : définitions et principe de fonctionnement

L'algorithme-α (van der Aalst *et al.*, 2002) a pour objectif de découvrir un workflow à partir d'un ensemble de traces d'exécution de ce même workflow.

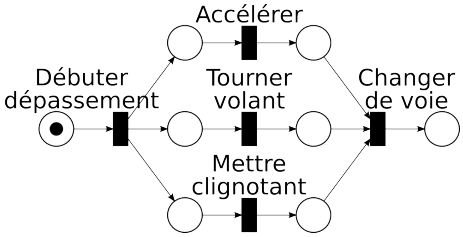

FIGURE 4: Un WF-net avec 3 transitions parallèles. Les cercles représentent les places ; le marquage des places caractérise l'état du réseau (ici le réseau est dans son état initial). Les rectangles sont les transitions ; Une transition représente une action du workflow.

3.1.3.1. Modèles (WF-nets) et traces (WF-logs)

L'algorithme-α s'appuie sur les *workflow-nets* (WF-nets) pour représenter le workflow. Un WF-net (figure 4) est un réseau de Petri répondant à certaines contraintes structurelles [1].

Un *workflow-log* (WF-log) est un ensemble de traces d'exécution d'un workflow. Une trace est une séquence d'événements produits par l'exécution du workflow. Chaque événement est caractérisé par un type et chaque type d'événement est associé à une transition du WF-net qui l'a produit. La trace d'un WF-net est la séquence de déclenchement des transitions du WF-net.

3.1.3.2. L'algorithme-α

L'algorithme-α (van der Aalst *et al.*, 2002) se déroule en 8 étapes construisant progressivement les ensembles nécessaires à la production du WF-net (dernière étape) :

– étapes 1 à 3. Construction de T_W, T_I et T_O, respectivement, l'ensemble des transitions, l'ensemble des transitions initiales et l'ensemble des transitions finales du WF-net. Les transitions correspondent aux types des événements observés dans les traces du WF-log.

– étape 4. Construction de X_W, caractérise un ensemble de places candidates pour le WF-net. Le calcul de cet ensemble constitue le point dur de cet algorithme. Chaque place candidate est caractérisée par ses transitions d'entrées et ses transitions de sorties.

1. Existence d'une place d'entrée et d'une place de sortie ; Si l'on connecte la place de sortie à la place d'entrée avec une transition, le réseau de Petri obtenu est fortement connecté.

– étape 5. Construction de Y_W, sous-ensemble de X_W, est calculé par l'intermédiaire d'une mesure d'extremum. Y_W caractérise l'ensemble des places candidates, épuré des places implicites, places qui n'ajoutent aucune information au WF-net.

– étapes 6 et 7. Construction de P_W, l'ensemble des places du WF-net reconstruit et de F_W, l'ensemble des arcs du WF-net (déduits directement de Y_W, T_I et T_O).

– étape 8. Construction de $\alpha(W)$ est le WF-net reconstruit, déduit directement de T_W, P_W et F_W.

L'étape 4 constitue le cœur de l'algorithme. Le calcul de X_W ne s'appuie cependant pas directement sur les traces du WF-log. En effet, la construction X_W s'appuie sur trois relations entre types d'événements : les relations $>_W$, $\rightarrow_W$ et $\#_W$. Notons a et b deux types d'événements présents dans les traces du WF-log W :

- $a >_W b$ signifie « a *peut être suivi directement de* b ». Autrement dit, il existe une trace de W où un événement de type a est directement suivi d'un événement de type b ;
- $a \rightarrow_W b$ signifie « a *peut être directement suivi de* b*, mais* b *n'est jamais directement suivi de* a » ; et
- $a \#_W b$ signifie « a *n'est jamais directement suivi de* b *et* b *n'est jamais directement suivi de* a ».

Notons ici que la notion de complétude d'un WF-log est intimement liée à la relation $>_W$. En effet, un WF-log est considéré comme complet au regard d'un modèle, si pour chaque couple de types d'événements, le WF-log suffit à caractériser correctement cette relation $>_W$.

Van der Aalst *et al.* (2002) prouvent qu'étant donné un WF-log complet d'un WF-net (respectant certaines contraintes structurelles), l'algorithme-α est capable de reconstruire ce WF-net, au renommage des places près.

3.2. Contribution : une variante interactive de l'algorithme-α

Nous proposons de rendre interactif l'algorithme-α. Ce que nous entendons par « interactif », c'est permettre à l'analyste, c'est-à-dire la personne qui manipule l'algorithme et interprète les résultats comme connaissances, d'introduire des connaissances intermédiaires complétant celles accessibles à partir des données (les traces du WF-log).

3.2.1. De l'algorithme-α à l'algorithme-α^i

Nous voulons rendre l'algorithme-α interactif. De cette manière, l'analyste peut utiliser ses propres connaissances sur l'activité pour aider l'algorithme-α à (re-)découvrir un WF-net.

Pourquoi est-il nécessaire à un expert d'introduire des connaissances dans l'étape de la fouille ? Parce que dans bien des cas, il n'est pas possible de savoir si la collection de traces que l'on fouille est complète ou non. Plus encore, lorsque l'on travaille sur des données relatives à une activité humaine telle que la conduite automobile, nous savons que la collection de traces est incomplète, bruitée et que les données ne décrivent pas forcément le niveau d'abstraction du modèle recherché : certains aspects du comportement ne sont pas directement observés dans les traces [2]. A contrario, un analyste dispose de connaissances sur l'activité qui sont complémentaires des données collectées.

Nous considérons trois possibilités d'interactions entre l'analyste et l'algorithme de fouille.

- Avant d'effectuer la fouille (via le SBT ABSTRACT) : en *préparant* les traces, en *ajoutant des traces-exemples*, ou encore en donnant des *exemples négatifs*.
- Après la fouille : en interprétant le modèle résultant, en *renommant les places* du WF-net produit, en *éditant* les places et les arcs, en *ajoutant* des transitions non observables.
- Pendant la fouille : sans changer l'algorithme, en *éditant* des résultats intermédiaires de l'algorithme de fouille, ou en changeant l'algorithme, de manière à implémenter un mécanisme de type chaînage mixte, demandant à l'analyste de compléter, si nécessaire, des informations potentiellement manquantes.

Nous nous sommes focalisés dans un premier temps sur l'édition des résultats intermédiaires de l'algorithme-α. Cela nous permet de conserver les bonnes propriétés de l'algorithme-α et en particulier d'intégrer nos modifications à moindre coût dans des versions améliorées telles que celles utilisées par EMiT ou Little Thumb. Nous nous intéressons à un résultat intermédiaire qui n'est pas explicité dans la description de l'algorithme-α : le calcul de la relation $>_W$. En effet, l'étape 4 de l'algorithme s'appuie sur les relations $\rightarrow_W$ et $\#_W$ pour caractériser l'ensemble de places candidates X_W. Ces deux relations sont elles-mêmes calculées à partir de la relation $>_W$. Nous proposons

2. Par exemple, un comportement peut être observable par un expert, à partir d'une vidéo de l'activité, mais pas être explicitement inscrit dans les traces exploitées informatiquement.

de présenter la relation $>_W$ à l'analyste, de manière à ce qu'il puisse l'éditer en ajoutant des connaissances qu'il possède sur la succession des types d'événements aux informations qui sont explicitement contenues dans les traces. L'analyste peut aussi valider ou invalider les informations contenues dans les traces.

Cette relation $>_W$ nous semble intéressante à présenter à l'analyste puisqu'elle peut être facilement exprimée en langage naturel et facilement comprise par l'analyste. En fait, $a >_W b$ signifie « il existe une trace où a est directement suivi de b. » Mais dans le cas de traces incomplètes, la question qui nous intéresse est légèrement différente, puisque nous savons que tous les comportements n'ont pas été observés dans notre ensemble de traces. La question qui nous intéresse devient donc : « est-il possible qu'une trace où a est directement suivi de b soit produite par le modèle recherché ? » La réponse à cette question conduit à la définition de la relation $>$, non pas simplement à partir du WF-log W comme c'est le cas de la relation $>_W$, mais aussi à partir des connaissances de l'analyste. Nous notons cette relation $>_\mathcal{A}$ ($\mathcal{A}$ pour « analyste »), de manière à la différencier de celle calculée uniquement à partir des traces. Cette nouvelle relation, contenant des connaissances provenant de l'analyste, permet de rendre l'algorithme-α utilisable dans le cas où le jeu de traces est incomplet.

Nous appelons l'algorithme modifié algorithme-α^i (i pour interactif). L'interaction que nous proposons intervient avant l'étape 4 de l'algorithme-α, pour la construction de X_W, les autres étapes restent inchangées :

- Les étapes de calcul 1 à 3 sont inchangées (calcul de T_W, T_I et T_O).
- Entre l'étape 3 et 4, nous introduisons 3 étapes supplémentaires, permettant à l'analyste de compléter ou corriger les résultats intermédiaires.
  (a) calculer la relation $>_W$ à partir des traces,
  (b) présenter la relation $>_W$ à l'analyste, en langage naturel pour la compléter de ses connaissances (relation $>_\mathcal{A}$), et
  (c) construire les relations $\rightarrow_\mathcal{A}$ et $\#_\mathcal{A}$ à partir de $>_\mathcal{A}$.
- Enfin, les étapes 4 à 8 sont inchangées, à l'exception du fait qu'elles contiennent maintenant des connaissances provenant de l'analyste ($\mathcal{A}$), et pas seulement de l'ensemble de traces (W). Pour expliciter formellement cette différence, nous modifions simplement la notation : calcul de $X_\mathcal{A}$, $Y_\mathcal{A}$, $P_\mathcal{A}$, $F_\mathcal{A}$ et $\alpha^i_\mathcal{A}(W)$ à la place de X_W, Y_W, P_W, F_W et $\alpha(W)$

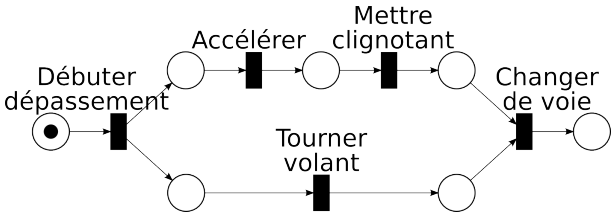

FIGURE 5: WF-net produit par l'algorithme-α sur un jeu de traces incomplet.

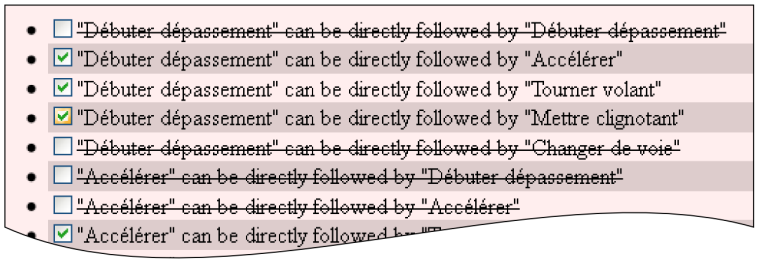

FIGURE 6: Prototype d'interface présentée à l'analyste pour interagir avec l'algorithme-α^i. L'analyste peut introduire ses connaissances au niveau des résultats intermédiaires (relation $>_{\mathcal{A}}$).

3.2.2. Exemple illustratif

Nous illustrons notre approche sur un exemple simple. À partir du WF-net présenté à la figure 4, nous produisons l'ensemble de traces suivant :

- Débuter dépassement, Accélérer, Tourner volant, Mettre clignotant, Changer de voie
- Débuter dépassement, Tourner volant, Accélérer, Mettre clignotant, Changer de voie
- Débuter dépassement, Accélérer, Mettre clignotant, Tourner volant, Changer de voie

Cet ensemble de traces est incomplet au regard du modèle donné.

Comme attendu, en utilisant l'algorithme-α sur un jeu de traces incomplet, le résultat (figure 5) ne correspond pas au modèle initial. Sur cet exemple, le résultat est un WF-net qui fait sens, sans pour autant correspondre au modèle que nous cherchons. Avec quelques connaissances de l'analyste (figure 6), l'algorithme est capable de redécouvrir le WF-net original.

4. Discussion et perspectives

Nous avons montré qu'il est possible de modifier un algorithme de fouille de processus, l'algorithme-α, pour le rendre interactif. L'utilisateur, un ana-

lyste qui cherche un modèle de workflow, peut interagir avec l'algorithme de manière à ajouter des connaissances qui n'étaient pas directement accessibles dans les traces. Cette interaction peut servir de base pour gérer le problème d'incomplétude des traces collectées.

L'interaction que nous proposons se base sur la relation $>_W$, qui peut être erronée à cause de l'incomplétude des données ou du bruit. Nous proposons à l'utilisateur d'enrichir la relation $>_W$, calculée à partir de l'ensemble des traces, pour produire la relation $>_\mathcal{A}$. Cela pose a priori un problème de complexité à deux niveaux. D'une part, l'algorithme-α s'appuie sur un calcul de cliques, avec une complexité exponentielle par rapport au nombre de types d'événements dans les traces. D'autre part, pour établir la relation $>_\mathcal{A}$ nous présentons à l'analyste une liste d'assertions de longueur n^2 où n est le nombre de types d'événements. Pour l'exemple de la figure 6, le WF-net étudié possède 5 transitions et l'analyste doit vérifier la relation $>_W$ pour chacun des 25 couples de types d'événements.

Nous rappelons que notre démarche s'intègre dans un processus global de découverte de connaissances (section 2.). Ceci nous permet de contrôler le nombre de type d'événements, et donc la complexité, au niveau de la phase de préparation. Dans notre contexte, la découverte de connaissances est guidée par un analyste qui sait quel type de connaissances il cherche. Lors de la préparation des données, à l'aide du système à base de traces ABSTRACT, l'analyste peut transformer (reformuler) les traces pour atteindre le niveau d'abstraction pertinent pour son analyse. Ce travail d'abstraction est accompagné d'un travail de filtrage pour ne conserver dans les traces que les informations utiles au regard de l'activité étudiée. À l'aide de ces connaissances *top-down*, il est donc possible de sélectionner les données dans les phases amont avant d'utiliser l'algorithme-α^i. Notons que si l'analyste se rend compte, au moment de la fouille, que les traces contiennent des informations non pertinentes, il peut interrompre la fouille et revenir à l'étape de préparation des données.

Pour la description d'activités riches, nécessitant de conserver un grand nombre d'informations dans les traces, nous proposons de diviser le problème en sous-problèmes liés aux différents niveaux de granularités de l'activité. Nous proposons de développer une méthodologie de découverte de connaissances sous forme de réseaux de Petri hiérarchiques, qui permettrait d'étudier une activité telle que la conduite automobile dans toute sa richesse, en limitant le problème de la complexité à un niveau hiérarchique à la fois.

Nous proposons deux manières de valider les connaissances découvertes. D'un point de vue épistémologique, comme les informations fournies par

l'analyste peuvent être stockées en tant que connaissances, l'approche que nous proposons est une approche de *construction* de connaissances. Le traçage de la construction des connaissances permet de vérifier et de remettre en question chaque étape de la construction des connaissances. De la même manière que pour l'établissement d'un fait par application de théorèmes, si chaque hypothèse ou donnée introduite explicitement dans le système de découverte de connaissance est vraie, alors la connaissance construite est valide par construction. Par ailleurs, il est possible de simuler le comportement du processus que le réseau de Petri représente. Le modèle découvert peut donc être validé par la vérification de conformité du comportement du modèle par rapport à celui du processus observé.

La première amélioration pour l'algorithme-α^i serait de décliner l'approche aux variantes de l'algorithme-α. Nous avons déjà produit une version interactive de l'algorithme-α^{++} (de Medeiros *et al.*, 2004), une extension de l'algorithme-α qui gère tous les types de boucles. Il serait intéressant d'étendre la notion d'interactivité à EMiT et Little Thumb, ou à d'autres classes d'algorithmes, tels que ceux présentés dans la revue de van Dongen *et al.* (2009).

Il serait aussi intéressant d'enrichir les possibilités d'interactions. Comme nous l'avons vu à la section 3.2.1., l'utilisateur peut interagir à trois niveaux : avant d'effectuer la fouille (sur les traces), après la fouille (sur le résultat) ou pendant la fouille (via l'interaction sur des résultats intermédiaires ou un mécanisme de type chaînage mixte). Nous avons ici proposé une interaction sur des résultats intermédiaires. Il serait intéressant dans le futur d'étudier plus spécifiquement la possibilité (1) de développer un chaînage mixte, et (2) d'interagir avec les traces. Le chaînage mixte permettrait de réduire l'effort de l'analyste, en lui présentant en priorité des informations susceptibles d'être fausses ou incomplètes. Par exemple, l'algorithme pourrait proposer à l'analyste des places « presque » découvertes, ou suggérer de vérifier des relations qui, si elles étaient avérées, permettraient de simplifier le modèle. Nous pensons que cela permettrait d'améliorer l'utilisabilité d'un tel système interactif.

Pour améliorer l'utilisabilité de l'algorithme dans le contexte de la découverte *interactive* de connaissances (figure 3), nous souhaitons intégrer l'algorithme de fouille interactif dans une plateforme qui permette d'effectuer facilement des allers-retours entre les traces (le logiciel ABSTRACT), les connaissances découvertes et l'algorithme de fouille. Cette plateforme devra enregistrer l'histoire de la construction des connaissances.

5. Conclusion

Ce papier propose une approche pour fouiller une activité observée et construire de manière interactive des connaissances, à la fois descriptives et opérationnelles. Nous montrons qu'il est possible de modifier un algorithme de fouille de workflow, l'algorithme-α, pour mettre en œuvre cette démarche de découverte interactive de connaissances. Les connaissances d'un analyste et les résultats intermédiaires de la fouille se combinent à travers l'interaction pour construire un automate. L'automate construit peut être utilisé en tant que représentation descriptive de connaissances, permettant à l'analyste de mieux comprendre l'activité, ou comme une représentation opérationnelle des connaissances, permettant l'exploitation directe dans un environnement informatique pour la simulation. Ce type d'interaction supporte l'idée générale de rendre le processus de découverte de connaissances interactif.

Remerciements. Ce travail a reçu le soutien de la Région Rhône-Alpes et du cluster de recherche « Transport, Territoires et Société ».

Références

DE MEDEIROS A., VAN DONGEN B., VAN DER AALST W. & WEIJTERS A. (2004). Process mining : Extending the α-algorithm to mine short loops. *Eindhoven University of Technology, Eindhoven*.

FAYYAD U., PIATETSKY-SHAPIRO G. & SMYTH P. (1996). From data mining to knowledge discovery : an overview. *AI-Magazine*, p. 37–54.

GEORGEON O., HENNING M. J., BELLET T. & MILLE A. (2007). Creating cognitive models from activity analysis : A knowledge engineering approach to car driver modeling. In *International Conference on Cognitive Modeling* : Taylor & Francis.

GEORGEON O., MILLE A. & BELLET T. (2006). Analyzing behavioral data for refining cognitive models of operator. *Philosophies and Methodologies for Knowledge Discovery*, p. 588–592.

HAMMORI M., HERBST J. & KLEINER N. (2006). Interactive workflow mining–requirements, concepts and implementation. *Data & Knowledge Engineering*, **56**(1), 41–63.

HERBST J. (2001). *Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen*. PhD thesis, Universität Ulm.

LAFLAQUIÈRE J., SETTOUTI L. S., PRIÉ Y. & MILLE A. (2006). A trace-based system framework for experience management and engineering. In

Second International Workshop on Experience Management and Engineering (EME 2006) in conjunction with KES2006 : Springer, Berlin.

MATHERN B., BELLET T. & MILLE A. (2010). An Iterative Approach to Develop a Cognitive Model of the Driver for Human Centred Design of ITS. In *Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems*.

MICHALSKI R. S. (2003). Knowledge mining : A proposed new direction. Invited talk at the Sanken Symposium on Data Mining and Semantic Web, Osaka University, Japan, March 10-11, 2003.

NOY N. F. (2004). Semantic integration : a survey of ontology-based approaches. *ACM Sigmod Record*, **33**(4), 65–70.

OMELAYENKO B. (2001). Learning of ontologies for the web : the analysis of existent approaches. In *In Proceedings of the International Workshop on Web Dynamics*.

SCHIMM G. (2002). Process Miner-A Tool for Mining Process Schemes from Event-Based Data. In *Proceedings of the European Conference on Logics in Artificial Intelligence* : Springer-Verlag.

VAN DER AALST W. & VAN DONGEN B. (2002). Discovering Workflow Performance Models from Timed Logs. In *Proceedings of the First International Conference on Engineering and Deployment of Cooperative Information Systems* : Springer-Verlag.

VAN DER AALST W., VAN DONGEN B., HERBST J., MARUSTER L., SCHIMM G. & WEIJTERS A. (2003). Workflow mining : A survey of issues and approaches. *Data & Knowledge Engineering*, **47**(2), 237–267.

VAN DER AALST W. & WEIJTERS A. (2004). Process mining : a research agenda. *Computers in Industry*, **53**(3), 231–244.

VAN DER AALST W., WEIJTERS A. & MARUSTER L. (2002). Workflow mining : Which processes can be rediscovered ? In *Eindhoven University of Technology*.

VAN DONGEN B., ALVES DE MEDEIROS A. & WEN L. (2009). Process mining : Overview and outlook of petri net discovery algorithms. In K. JENSEN & W. VAN DER AALST, Eds., *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 of *Lecture Notes in Computer Science*, p. 225–242. Springer Berlin / Heidelberg.

WEIJTERS A. & VAN DER AALST W. (2002). Workflow mining : discovering workflow models from event-based data. In *Proceedings of the ECAI Workshop on Knowledge Discovery and Spatial Data*.