Découverte interactive de connaissances à partir de traces d'activité : Synthèse d'automates pour l'analyse et la modélisation de l'activité de conduite automobile

Benoît Mathern

THÈSE DE L'UNIVERSITÉ DE LYON

Délivrée par

L'UNIVERSITÉ CLAUDE BERNARD LYON 1

ÉCOLE DOCTORALE INFORMATIQUE ET MATHÉMATIQUES

Découverte interactive de connaissances à partir de traces d'activité :
Synthèse d'automates pour l'analyse et la modélisation de l'activité de conduite automobile

THÈSE DE DOCTORAT EN INFORMATIQUE

soutenue publiquement par

Benoît MATHERN

le 12 mars 2012

Sous la direction de

Monsieur le Professeur Alain MILLE

Devant un jury composé de :

Guy André BOY :	Professeur des Universités, Florida Institute of Technology (rapporteur)
Fabien GANDON :	HDR, chargé de recherche, INRIA Sophia Antipolis (rapporteur)
Catherine GARBAY :	HDR, directeur de recherche, CNRS Université de Grenoble (rapporteur)
Pierre DERANSART :	HDR, directeur de recherche, INRIA Rocquencourt
Alain MILLE :	Professeur des Universités, Université Claude Bernard Lyon 1
Thierry BELLET :	Chargé de recherche, IFSTTAR Bron

Ce travail de thèse a bénéficié du soutien de la Région Rhône-Alpes à travers le cluster de recherche « Transport, Territoires et Société » et la bourse Explo'ra Doc.

Remerciements

Je tiens à exprimer ici ma gratitude envers celles et ceux sans qui ce travail de thèse n'aurait pas été ce qu'il est. J'ai eu la chance d'être soutenu par de nombreuses personnes, sur le plan professionnel comme sur le plan personnel.

Mes remerciements vont d'abord aux membres du jury, qui ont accepté d'évaluer mon travail. Je remercie sincèrement mes rapporteurs, Guy André Boy, Fabien Gandon et Catherine Garbay, pour leur travail de relecture attentive de ce manuscrit et l'ensemble des retours qu'ils m'ont fournis. Je remercie également Pierre Deransart d'avoir accepté de participer à mon jury de thèse.

Je remercie mes encadrants. Thierry Bellet, pour m'avoir fourni un environnement de recherche très riche et stimulant. Alain Mille, mon Directeur de thèse, pour son soutien infaillible. Alain, tu as toujours su te rendre disponible pour répondre à mes questions, alors que je sais que tu es quelqu'un de très occupé. Tu as toujours réussi à me guider et à faire en sorte que je garde le bon cap. Merci infiniment.

Je remercie l'ensemble de mes collègues et amis du LESCOT – Soizick De Bagneaux, Aurélie Banet, Evgueni Douissembekov, Catherine Gabaude, Maud Ranchet... et bien sûr l'ensemble de l'équipe technique ! –, du LIRIS – Olivier Champalle, Amélie Cordier, Olivier Georgeon, Brigitte Guyader... – et d'ailleurs – Arnaud Bonnard, Fanny Conte, Mohamed Houacine et Michael Regan. Vous avez tous participé, d'une manière ou d'une autre à me fournir un cadre de travail des plus agréables.

Je remercie ma famille et mes amis les plus proches. Merci à Charlene, mes parents, Anne-Gaël, Damien et Stéphanie. Vous avez toujours cru en moi. Vous m'avez toujours apporté votre soutien inconditionnel, en particulier dans les moments les plus difficiles. Charlene, c'est au quotidien que tu m'as apporté ton soutien. Maman, je ne sais comment te remercier. Ton aide m'a été inestimable, tu as toujours été là quand j'en avais besoin. Merci du fond du cœur.

Je remercie les relecteurs qui ont contribué à la qualité de ce mémoire grâce à leurs retours pertinents. Il s'agit de mes encadrants bien sûr, mais aussi Maman, Hélène Tattegrain, Damien Sornette, Arnaud Bonnard, Jean-Christophe Paris et Charlene Hallett.

Je remercie Steven Shladover et Christopher Nowakowski d'avoir accepté de m'accueillir à l'Université de Californie, Berkeley et Delphine Cody, pour son soutien sur place.

Enfin, je remercie sincèrement Damien Cram. Ce sont nos discussions, à l'époque où j'hésitais encore entre une carrière d'ingénieur et une carrière de chercheur qui m'ont aidé à prendre cette décision de me lancer dans la grande aventure qu'est une thèse. C'est un choix que je ne regrette pas. Merci.

Maintenant, c'est vous, lecteur, que je remercie. C'est votre intérêt qui fera vivre ce mémoire de thèse. J'espère que vous trouverez à sa lecture de quoi alimenter vos propres réflexions.

Découverte interactive de connaissances à partir de traces d'activité :

Synthèse d'automates pour l'analyse et la modélisation de l'activité de conduite automobile

---

Résumé

Comprendre la genèse d'une situation de conduite requiert d'analyser les choix faits par le conducteur au volant de son véhicule pendant l'activité de conduite, dans sa complexité naturelle et dans sa dynamique située. Le LESCOT a développé le modèle COSMODRIVE, fournissant un cadre conceptuel pour la simulation cognitive de l'activité de conduite automobile. Pour exploiter ce modèle en simulation, il est nécessaire de produire les connaissances liées à la situation de conduite sous forme d'un automate par exemple. La conception d'un tel automate nécessite d'une part de disposer de données issues de la conduite réelle, enregistrées sur un véhicule instrumenté et d'autre part d'une expertise humaine pour les interpréter.

Pour accompagner ce processus d'ingénierie des connaissances issues de l'analyse d'activité, ce travail de thèse propose une méthode de découverte interactive de connaissances à partir de traces d'activité. Les données de conduite automobile sont considérées comme des $\mathcal{M}$-Traces, associant une sémantique explicite aux données, exploitées en tant que connaissances dans un Système à Base de Traces (SBT). Le SBT permet de filtrer, transformer, reformuler et abstraire les séquences qui serviront à alimenter la synthèse de modèles automates de l'activité de conduite. Nous reprenons des techniques de fouille de workflow permettant de construire des automates (réseaux de Petri) à partir de logs. Ces techniques nécessitent des données complètes ou statistiquement représentatives. Or les données collectées à bord d'un véhicule en situation de conduite sont par nature des cas uniques, puisqu'aucune situation ne sera jamais reproductible à l'identique, certaines situations particulièrement intéressantes pouvant en outre être très rarement observées. La gageure est alors de procéder à une forme de généralisation sous la forme de modèle, à partir d'un nombre de cas limités, mais jugés pertinents, représentatifs, ou particulièrement révélateurs par des experts du domaine. Pour compléter la modélisation de telles situations, nous proposons donc de rendre interactifs les algorithmes de synthèse de réseau de Petri à partir de traces, afin de permettre à des experts-analystes de guider ces algorithmes et de favoriser ainsi la découverte de connaissances pertinentes pour leur domaine d'expertise. Nous montrerons comment rendre interactifs l'algorithme α et l'algorithme α^+ et comment généraliser cette approche à d'autres algorithmes.

Nous montrons comment l'utilisation d'un SBT et de la découverte interactive d'automates impacte le cycle général de découverte de connaissances. Une méthodologie est proposée pour construire des modèles automates de l'activité de conduite automobile.

Une étude de cas illustre la méthodologie en partant de données réelles de conduite et en allant jusqu'à la construction de modèles avec un prototype logiciel développé dans le cadre de cette thèse.

Mots-clés : Ingénierie de la dynamique des connaissances – découverte interactive de connaissances – modélisation de l'activité humaine – traces d'activité – réseaux de Petri – conduite automobile.

Interactive discovery of knowledge from activity traces:
A synthesis of automata in the analysis and modelling of the activity of car driving

Abstract

Driving is a dynamic and complex activity. Understanding the origin of a driving situation requires the analysis of the driver's choices made while he/she drives. In addition, a driving situation has to be studied in its natural complexity and evolution. LESCOT has developed a model called COSMODRIVE, which provides a conceptual framework for the cognitive simulation of the activity of car driving. In order to run the model for a simulation, it is necessary to gather knowledge related to the driving situation, for example in the form of an automaton. The conception of such an automaton requires : 1) the use of real data recorded in an instrumented car, and, 2) the use of human expertise to interpret these data. These data are considered in this thesis as activity traces.

The purpose of this thesis is to assist the Knowledge Engineering process of activity analysis. The present thesis proposes a method to interactively discover knowledge from activity traces. For this purpose, data from car driving are considered as M-traces – which associate an explicit semantic to these data. This semantic is then used as knowledge in a Trace Based System. In a Trace Based System, M-traces can be filtered, transformed, reformulated, and abstracted. The resulting traces are then used as inputs in the production of an automaton model of the activity of driving. In this thesis, Workflow Mining techniques have been used to build automata (Petri nets) from logs. These techniques require complete or statistically representative data sets. However, data collected from instrumented vehicles are intrinsically unique, as no two driving situations will ever be identical. In addition, situations of particular interest, such as critical situations, are rarely observed in instrumented vehicle studies. The challenge is then to produce a model which is a form of generalisation from a limited set of cases, which have been judged by domain experts as being relevant and representative of what actually happens.

In the current thesis, algorithms synthesising Petri nets from traces have been made interactive, in order to achieve the modelling of such driving situations. This then makes it possible for experts to guide the algorithms and therefore to support the discovery of knowledge relevant to the experts. The process involved in making the α-algorithm and the α^+-algorithm interactive is discussed in the thesis in a way that can be generalised to other algorithms.

In addition, the current thesis illustrates how the use of a Trace Based System and the interactive discovery of automata impacts the global cycle of Knowledge Discovery. A methodology is also proposed to build automaton models of the activity of car driving. Finally, a case study is presented to illustrate how the proposed methodology can be applied to real driving data in order to construct models with the software developed in this thesis.

Keywords: Engineering of the dynamic of knowledge – interactive discovery of knowledge – human activity modelling – activity traces – Petri nets – car driving.

Table des matières

Table des figures

Liste des tableaux

Introduction

Ce travail de recherche s'inscrit dans le cadre général de l'analyse de l'activité humaine et plus particulièrement l'analyse de la conduite automobile. Les analyses cherchent à répondre à des enjeux de sécurité routière et de confort du conducteur. Par exemple, en étudiant l'origine d'un accident de la route, des enquêteurs peuvent détecter des facteurs de risque. Ces facteurs peuvent être liés à l'environnement routier, par exemple, une intersection dangereuse avec une faible visibilité, ou à ce que fait le conducteur, par exemple, s'il était distrait et n'a pas repéré un véhicule dans son angle mort. Une fois les facteurs de risques identifiés, des solutions peuvent être mises en place. Par exemple, changer l'aménagement d'une intersection, ou proposer un système d'assistance à la conduite qui avertit le conducteur quand un véhicule est présent dans un angle mort.

Les analystes se focalisent sur les aspects cognitifs de la conduite automobile. De manière prosaïque, ils cherchent à savoir ce qui se passe dans la tête du conducteur. Mais est-ce possible ? Dans le cas du conducteur qui n'a pas détecté un véhicule dans son angle mort, un passager attentif pourrait s'en rendre compte, en interprétant le comportement du conducteur. Ainsi, pour comprendre le comportement du conducteur, il est nécessaire : 1) de l'observer et 2) d'analyser les comportements observés. Pour étudier les comportements du conducteur, les travaux de recherche en psychologie de la conduite mettent le conducteur en situation et observent son comportement. Ces expérimentations peuvent avoir lieu sur route ouverte ou, dans un environnement contrôlé, sur un simulateur de conduite. Pour pouvoir « observer » les comportements du conducteur, de nombreux capteurs instrumentent les véhicules expérimentaux et les simulateurs de conduite. L'objectif est de détecter les actions du conducteur qui ont un impact sur la conduite automobile (par ex., actions sur le volant, sur les pédales…) et tout indice qui permet de mieux comprendre son comportement (par ex., ses stratégies visuelles : a-t-il vérifié son rétroviseur ?). Les données enregistrées sont d'une grande richesse et d'une grande complexité. Encore faut-il pouvoir les analyser.

Autant le passager attentif est capable de comprendre ce que fait le conducteur, autant un calcul seul, aussi sophistiqué soit-il, ne peut rien faire de ces données brutes. Par exemple, analyser les actions sur les pédales du véhicule, c'est un problème aussi difficile que si l'on demandait à un passager de ne regarder que les pieds du conducteur et d'en déduire ce qui se passe. Les données enregistrées sont difficiles à analyser parce qu'elles constituent des indices épars de la conduite automobile. Pour comprendre l'activité, il faut retrouver l'unité entre ces différents indices.

C'est un processus qui semble difficilement automatisable, car il nécessite de nombreuses connaissances de la part des analystes, pour pouvoir inférer ce qui n'a pas été observé. Pour manipuler les données, l'ordinateur doit être capable d'exploiter les connaissances de l'humain. Et inversement, pour permettre à l'analyste de comprendre les données, il est important de définir leur signification (c.-à-d. comment les interpréter). L'ingénierie des connaissances vise à distribuer la gestion des connaissances

entre humains et systèmes informatiques dans l'objectif de pouvoir faciliter le calcul de données, leur interprétation et l'interaction entre l'humain et l'ordinateur.

C'est dans ce cadre de recherche que cette thèse se situe. L'analyse de l'activité de conduite est nécessairement transdisciplinaire.

Dans notre thèse, nous nous positionnons en faveur d'une ingénierie de la découverte de connaissances dynamique. Nous nous intéressons au processus de découverte de connaissances et proposons que chaque connaissance produite permette de nourrir à nouveau le processus de découverte dans un nouveau contexte. Les dispositifs d'assistance à la découverte de connaissances doivent donc intégrer de manière systématique (à chaque étape) cette nécessaire dynamicité. Nous soutenons que le processus de découverte de connaissances est distribué entre l'analyste, doué de ses connaissances expertes et construisant de nouvelles connaissances, et la plateforme informatique, support de la tâche de découverte. Les outils informatiques doivent permettre d'intégrer les connaissances expertes et les connaissances découvertes pour mener de nouveaux calculs dans un cycle enrichi. Plus précisément, dans cette thèse, nous nous intéressons à l'étape d'interprétation par l'analyste des motifs présents dans les données. Nous proposons d'assister cette interprétation en synthétisant les motifs découverts sous forme de modèles de connaissances exploitables par l'analyste pour la vérification même de ses hypothèses. Nous soutenons que le cycle de découverte peut être complété par une étape de synthèse interactive de connaissances, pour construire des modèles de l'activité.

Ces recherches s'inscrivent dans la collaboration entre deux laboratoires de recherche : le Laboratoire Ergonomie et Sciences Cognitives pour les Transports (LESCOT) et le Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS).

Le laboratoire LESCOT apporte les compétences nécessaires en ergonomie et en psychologie cognitive de la conduite automobile. C'est le laboratoire LESCOT qui recueille les données de conduite automobile et cherche à les analyser.

Le laboratoire LIRIS apporte les compétences en informatique et plus particulièrement en ingénierie des connaissances. L'équipe SILEX[1] du LIRIS propose des méthodes, des modèles et des outils permettant de manipuler et interpréter les données relatives à des activités humaines.

Un premier travail de thèse avait été effectué par Olivier Georgeon dans ce cadre de collaboration transdisciplinaire. La présente thèse s'inscrit dans le cadre du projet de recherche CRITICAL[2].

Le projet CRITICAL, consacré aux questions de sécurité routière, a pour objectif d'étudier les mécanismes cognitifs mis en œuvre par le conducteur automobile lors de la prise de conscience d'une situation critique. Pour cela il est nécessaire d'étudier la conduite automobile, telle qu'elle est mise en œuvre par le conducteur, et de disposer d'outils, à la fois théoriques et expérimentaux, pour étudier la cognition du conducteur.

L'objectif de ce projet est à la fois théorique et pratique pour le laboratoire LESCOT. Théorique, puisqu'il s'agit de comprendre, du point de vue des sciences cognitives et de l'ergonomie, le comportement du conducteur en particulier, et des erreurs humaines en général. Pratique, puisqu'il s'agit de développer des méthodes et des outils d'analyse de l'activité de conduite en situation critique et, à l'aide de ces outils, d'identifier les besoins du conducteur. Connaître les besoins du conducteur permettra, à terme, d'améliorer la formation du conducteur et de concevoir de futurs dispositifs

[1] Supporting Interaction and Learning by Experience

[2] Conscience et gestion des RIsques en situaTIon de Conduite AutomobiLe

(Mathern *et al.*, 2010a) assistant le conducteur à prévenir et à gérer les situations critiques.

La conduite automobile est une activité dynamique et complexe. Pour l'étudier, il faut à la fois observer le conducteur en train de conduire, si possible dans des situations les plus réalistes possibles, et analyser ces observations, tout en tenant compte des spécificités de cette activité. L'activité cognitive du conducteur, quant à elle, n'est pas accessible à l'observation directe. En outre, les situations accidentogènes sont difficiles à observer et à étudier. Il s'agit souvent de cas uniques et dont il faut comprendre en détail la genèse de la situation et la façon dont le conducteur essaye de récupérer cette situation critique.

Le modèle cognitif du conducteur COSMODRIVE[3] sert de cadre à la compréhension de l'activité de conduite automobile. Les outils d'analyse serviront non seulement à produire des analyses permettant de mieux comprendre l'activité de conduite en général, mais aussi à enrichir le modèle COSMODRIVE et sa version computationnelle COSMO-SiVIC.

Le **chapitre 1** donne un cadre épistémologique à cette thèse. Nous nous positionnerons face aux questions suivantes : qu'est-ce qu'une activité ? Comment l'observer ? Qu'est-ce qu'une trace ? Comment modéliser l'activité à partir de traces ? Nous verrons que ce n'est jamais l'activité elle-même que l'on analyse, mais les traces issues de son observation. Nous verrons que l'analyse de ces traces d'observation nécessite un travail d'interprétation, et donc l'intervention d'un analyste.

Le **chapitre 2** présente le contexte de la modélisation de l'activité de conduite automobile. Ce chapitre introduit des travaux d'ergonomie et de psychologie cognitive relatifs à l'activité humaine en général et à l'activité de conduite automobile en particulier. Une présentation de différents types de modèles de conduite permet de mieux comprendre les enjeux de la modélisation du conducteur. Nous présenterons le modèle de simulation cognitive COSMODRIVE, qui sert de cadre à ce travail de thèse. Enfin, nous définirons l'objectif applicatif de la thèse : assister l'analyste dans la création de connaissances de conduite exploitables par le modèle COSMODRIVE.

Le **chapitre 3** présente les travaux d'ingénierie des connaissances sur lesquels repose cette thèse. Nous présenterons le cadre méthodologique de cette thèse : le champ de la découverte de connaissances à partir de données. Puis, nous introduirons les théories sur lesquelles notre travail repose : la théorie des traces modélisées et la méta-théorie des traces. La théorie des traces modélisées formalise la notion de trace d'activité. La méta-théorie des traces formalise les relations entre les traces et les processus qui les ont générées.

Le **chapitre 4** fait un état de l'art des techniques de modélisation compatibles avec l'objectif de cette thèse. Différents formalismes de modélisation seront présentés et nous justifierons le choix que nous avons fait d'utiliser les réseaux de Petri. Puis nous présenterons un état de l'art des techniques de modélisation permettant de construire des réseaux de Petri à partir d'un ensemble de traces. Nous verrons en quoi ces algorithmes ne suffisent pas à répondre au problème posé dans cette thèse.

Le **chapitre 5** présente la première contribution (formelle) de cette thèse. Nous avons choisi une famille d'algorithmes de construction de modèles et nous montrons comment modifier ces algorithmes pour permettre d'utiliser les connaissances de l'analyste. Nous expliquons pourquoi et comment nous avons modifié un algorithme particulier. Nous donnons une définition formelle de l'algorithme modifié et nous montrons comment généraliser ces modifications à d'autres algorithmes de la même famille.

[3] Cognitive Simulation Model of the DRIVEr

Le **chapitre 6** présente la deuxième contribution (théorique) de cette thèse. Nous proposons de définir une nouvelle méthodologie de découverte de connaissances. La découverte de connaissances est un processus itératif et interactif. Nous insistons sur le rôle central de l'analyste et de ses connaissances. Nous montrons comment l'utilisation d'algorithmes de construction de modèles impacte ce cycle de découverte de connaissances. Et nous montrons comment instancier cette méthodologie à la question de la modélisation de l'activité de conduite automobile.

Le **chapitre 7** présente la troisième contribution (pragmatique) de cette thèse. Il s'agit de la mise en pratique de la méthodologie et des algorithmes développés. Des outils d'analyse préexistants et un prototype implémentant les algorithmes modifiés ont été utilisés pour mettre en œuvre la méthodologie de découverte de connaissances. Cette mise en œuvre offre une première évaluation du travail de thèse et montre l'intérêt des méthodes et des outils proposés.

Enfin, dans le **chapitre 8**, nous discuterons l'ensemble de ce travail. Nous prenons en compte les retours d'évaluation pour discuter les problèmes rencontrés, vérifier que les objectifs de la thèse ont été atteints et proposer de nouvelles perspectives de recherche.

CHAPITRE 1

Positionnement épistémologique de la thèse : activité, traces et connaissances

Les mots « trace » et « activité » font partie du vocabulaire courant. Une activité se rapporte à un processus, à un ensemble d'actions. Les traces relèvent de ce qu'il reste, de ce que l'on peut observer a posteriori de ce processus ou de ces actions. Dans ce chapitre, nous visons à clarifier le sens que nous donnons à ces deux termes, *traces* et *activité*. Nous verrons comment ces deux concepts sont liés l'un à l'autre. Nous donnerons les éléments permettant de situer le positionnement scientifique dans lequel nous nous inscrivons au regard de ces deux concepts.

Il ne s'agit pas ici de faire un travail de recherche en épistémologie. Ce n'est pas l'objet de cette thèse. L'objet est de donner un cadre général permettant d'éclairer l'ensemble de ce travail de thèse, en donnant des définitions intuitives des deux concepts que sont l'activité et la trace, ainsi que les relations entre les deux.

1.1 Activité

1.1.1 Qu'est-ce que l'activité ?

Le dictionnaire Le Petit Robert (2010) présente quatre acceptions du mot « activité » : la première relative à un processus physique ou chimique, la deuxième qualifiant une personne dynamique, la troisième relevant des actes d'êtres vivants et la dernière relative à l'exercice d'une profession. Dans le cadre de cette thèse, c'est la troisième acception qui est pertinente :

> 3. (XIX^{e}) Ensemble des actes coordonnés et des travaux de l'être humain ; fraction spéciale de cet ensemble. *Les produits de l'activité humaine. Activité économique* [...] PSYCHOLOGIE, BIOLOGIE Ensemble des phénomènes psychiques et physiologiques correspondant aux actes de l'être vivant, relevant de la volonté, des tendances, de l'habitude, de l'instinct, etc. ; série de phénomènes de cet ordre. *L'activité volontaire, réflexe, chez l'homme. Activité nerveuse supérieure [...] Activité mentale, motrice, sensitive.*
>
> (Petit Robert, 2010.)

Activité : processus unifiés visant à la réalisation d'un but global

L'activité relève de l'*action*. En effet, une activité est un ensemble « d'actions », de « phénomènes manifestant un processus, un fonctionnement ». Dans son acception large, l'activité est aussi bien produite par des organismes vivants, humains, animaux, que par des processus physiques ou chimiques : activité d'un volcan, activité du soleil. Dans cette thèse, c'est l'*activité humaine* qui est considérée.

L'activité humaine peut relever d'un niveau de granularité élevé au niveau d'un groupe d'humain, l'activité d'une entreprise, ou d'un niveau individuel, l'activité d'un chercheur. Dans cette thèse, puisque nous nous intéresserons à l'activité du conducteur automobile, c'est ce niveau individuel, le niveau de l'*être humain*, que nous allons considérer.

L'activité relève de processus unifiés – c'est l'« ensemble des actes *coordonnés* des travaux de l'être humain » (Petit Robert, 2010) – et vise à atteindre un but. L'activité du chercheur, par exemple, est l'ensemble coordonné de ses travaux – étude de la littérature, développement d'expérimentations, élaboration de théories, publication de son travail, évaluation du travail de ses pairs, etc. – pour atteindre un but : la création de connaissances. Ainsi, l'activité est caractérisée par un ensemble de processus *unifiés* (Leontiev, 1984) visant à la réalisation d'un *but* global, commun à l'ensemble de ces processus. Dans ce mémoire, il n'y a pas d'ambiguïté sur la nature humaine de cette activité. Ainsi, dans la suite nous utiliserons indifféremment les termes *activité de conduite automobile* et *activité du conducteur automobile*.

Contexte, part observable et inobservable de l'activité

Une activité, à travers son but, vise à changer l'état de quelque chose. Sans environnement où agir, il n'y a pas d'activité. L'activité est donc indissociable du contexte dans lequel elle se réalise : l'activité est *située*.

Ce que l'on peut observer de l'activité, ce sont les actions observables, c'est-à-dire une partie seulement des processus mis en œuvre pour réaliser l'activité dans le contexte dans laquelle elle prend place. Ces actions observables, ce sont les comportements. Cependant, l'activité ne se limite pas aux comportements, car elle contient aussi les processus qui engendrent le comportement. L'activité, comme un iceberg, n'est pas seulement constituée de sa partie observable, mais aussi de sa partie non observable, sous-jacente.

Par exemple, le choix que fait un conducteur au volant de sa voiture de doubler un autre véhicule fait partie de son activité de conduite automobile. Cependant, ce choix lui-même n'est pas observable. Ce qui est observable, ce sont les actions qu'il met en œuvre pour dépasser : regarder son rétroviseur, mettre le clignotant, changer de voie, etc.

Définition retenue dans le cadre de cette thèse : L'activité est un ensemble de processus plus ou moins observables, immergés dans une situation et visant à la réalisation d'un but global unifié.

1.1.2 Comment étudier l'activité ?

Avant de s'attaquer à la question de « comment étudier l'activité ? » et de présenter les difficultés de cette entreprise, développons rapidement *pourquoi* il est nécessaire d'étudier l'activité.

L'activité relève de nos actions et de nos possibilités d'interagir avec le monde. Étudier l'activité est donc essentiel pour comprendre comment l'homme interagit avec son environnement. Lorsque l'on rapporte ce travail à l'étude d'une activité particulière, il s'agit de la comprendre pour la rendre plus sûre (diminuer le risque d'accident sur un lieu de travail), plus efficace (améliorer les performances d'un sportif), plus facile (améliorer le confort d'utilisation d'un outil).

Caractéristiques de l'activité humaine

Comme nous l'avons souligné, l'activité n'est que partiellement observable. L'activité n'est pas limitée aux actions observables et mesurables, elle contient des processus sous-jacents aux actions : par exemple des raisonnements, des motivations, des intentions ou des prises de décision. Pour étudier ces aspects inobservables de l'activité, il est donc nécessaire de s'appuyer sur les connaissances que l'on peut avoir sur l'humain, en l'occurrence les connaissances issues de la psychologie.

Parmi les branches de la psychologie, la psychologie cognitive est particulièrement concernée, car elle s'intéresse au traitement de l'information par l'homme. Les théories de psychologie cognitive proposent des modèles cherchant à expliquer comment l'homme raisonne, comment il prend des décisions, comment il se représente le monde, comment fonctionne sa mémoire, etc.

Ces connaissances sur l'homme peuvent s'étudier en laboratoire, dans un contexte artificiel. Ce contexte artificiel permet un contrôle strict de l'activité, mais en contrepartie il *dénature* l'activité naturelle. Lorsqu'il s'agit d'étudier l'activité naturelle, il convient de considérer l'homme *in situ*, dans son contexte *écologique*[1]. Ceci est nécessaire pour appréhender l'activité dans sa complexité naturelle : la situation est spécifique et l'homme dispose d'un savoir, la situation lui est plus ou moins familière et il sait plus ou moins comment y répondre. De plus, l'activité est prise dans l'action, au sein de la situation. L'activité est par nature *dynamique*.

Dans le cadre d'activités complexes telles que la conduite automobile, chaque situation est singulière. Le conducteur ne rencontrera jamais deux fois exactement la même situation de conduite. L'étude de l'activité se fait à partir de l'observation de situations singulières.

Étudier l'activité est donc une tâche difficile : il faut observer des cas (parfois uniques) dans leur environnement et leur complexité naturelle et dans leur dynamique situationnelle.

1.1.3 Étudier l'activité humaine : le rôle de l'ergonomie

L'ergonomie est une des disciplines étudiant l'activité humaine. L'observation et l'analyse de l'activité sont même l'objet scientifique central de l'ergonomie.

Comment étudier l'activité humaine et expliquer le fonctionnement de processus difficilement observables tels que la prise de décision ? La psychologie cognitive apporte des éléments de réponse, mais ces éléments de réponse, lorsqu'ils sont étudiés en laboratoire, sont souvent désengagés de la situation concrète, c'est-à-dire de l'activité écologique. C'est le rôle de l'ergonomie (cognitive) que de faire le lien entre l'activité située et les théories de la cognition.

[1] Le mot « écologique » signifie ici que l'activité est étudiée en contexte naturel, au contraire d'une situation artificielle générée spécifiquement pour étudier l'activité.

Quelques éléments de vocabulaire

Pour pouvoir présenter les outils de l'ergonomie qui permettent d'étudier l'activité, nous devons nous arrêter un instant sur quelques éléments de vocabulaire.

L'humain qui réalise l'activité est appelé *opérateur*, en ergonomie, et *sujet*, en psychologie.

L'ergonomie distingue l'activité de la *tâche* (Leplat, 1990). Comme nous l'avons vu, l'activité est associée à un but. Dans le cadre d'activités humaines, ce but est une tâche à accomplir.

La tâche telle qu'on la donne à l'opérateur s'appelle la tâche *prescrite*. Par exemple, pour la conduite automobile le Code de la route définit un ensemble de règles qui relèvent des tâches prescrites.

La tâche que l'opérateur vise à effectuer s'appelle la tâche *effective*. La tâche effective traduit la façon dont l'opérateur s'approprie la tâche prescrite, elle constitue les objectifs qu'il se fixe à atteindre *dans une situation donnée*. Par exemple, la façon dont le conducteur veut dépasser, qui est éventuellement en contradiction avec les règles du Code de la route.

L'activité, quant à elle, est ce qui est réellement réalisé. L'activité est différente de la tâche *effective*, d'une part parce que l'opérateur ne réalise pas toujours l'action telle qu'il l'avait planifiée (par ex., la situation peut changer), d'autre part parce que l'opérateur n'a pas forcément conscience de la façon dont il met réellement en œuvre son activité.

Outils pour étudier l'activité

Pour étudier l'activité, l'ergonomie dispose d'outils et de méthodes (Gillet, 1987). Nous en présentons ici quelques exemples caractéristiques : l'observation, l'expérimentation, les entretiens et les enquêtes.

L'observation. L'observation de l'activité constitue le pilier de l'ergonomie. L'objectif de l'observation est de comprendre l'activité, dans sa complexité naturelle, c'est-à-dire telle qu'elle s'*exprime* naturellement en contexte écologique. L'observation consiste à observer l'opérateur en situation, dans son environnement naturel. « Ce qu'on observe, ce sont des *traces* (des actions, des verbalisations, des résultats d'activité) qui serviront à inférer l'inobservable. » (Gillet, 1987 p 57.) Ces traces peuvent être issues d'un cas unique, typiquement s'il s'agit d'un accident. Des techniques complémentaires à l'observation sont nécessaires pour « inférer l'inobservable », typiquement à l'aide d'entretiens semi-dirigés et en s'appuyant sur un cadre théorique. Notons que l'observation nécessite à l'ergonome de se familiariser avec l'activité qu'il étudie : il doit connaître la tâche à accomplir et le vocabulaire technique de l'activité (*langage opératif*, Falzon, 1991). L'ergonome peut aller jusqu'à exécuter lui-même l'activité à titre exploratoire.

L'expérimentation. Contrairement à l'observation qui vise plutôt à interpréter les faits dans leur complexité et à les confronter à un cadre théorique, l'expérimentation vise à tester une hypothèse en soumettant l'humain à la réalisation d'une tâche. L'expérimentation cherche à reproduire les conditions de l'activité, sur le terrain ou en laboratoire pour tester « l'effet de facteurs sur des variables dépendantes ». L'expérimentation offre un meilleur contrôle, mais s'éloigne nécessairement de la situation réelle étudiée.

Ceci est d'autant plus vrai lorsque l'expérimentation s'effectue en environnement simulé.

Les entretiens. Les entretiens avec le ou les opérateurs sont nécessaires pour étudier les aspects cognitifs de l'activité. Cependant, comme nous l'avons évoqué précédemment, la tâche telle que l'opérateur se la définit et l'activité telle qu'il l'exécute peuvent être différentes. L'opérateur n'a pas forcément conscience de cette différence. Ainsi, pour éviter que l'opérateur ne rationalise a posteriori un comportement qui ne l'est pas forcément au moment où il engage son activité ou dont il n'avait pas forcément conscience, il est nécessaire d'utiliser des techniques d'entretien semi-supervisés, comme l'entretien d'explicitation (Vermersch, 1994).

Par exemple, en ergonomie la méthode d'autoconfrontation consiste à filmer l'opérateur en activité, puis à visionner avec lui le film « à chaud ». Ce débriefing permet à l'opérateur d'expliciter ses actes (décisions, motivations, intentions, connaissances) et sa représentation de la situation. Cet entretien peut aussi être *critique* si l'opérateur déclare avoir fait telle chose alors que le film montre le contraire. L'enjeu peut alors être de permettre à l'opérateur de « prendre conscience » de son activité réelle.

Les enquêtes. Les enquêtes et les questionnaires sont des approches exploratoires qui permettent de guider l'ergonome vers les éléments de l'activité à étudier plus en détail, avec d'autres techniques d'investigation.

1.1.4 Activité – Conclusion

L'activité est un processus complexe qu'il est nécessaire d'étudier dans sa complexité et en condition naturelle en prenant en compte la dynamique de la situation. L'ergonomie propose des outils et des méthodes pour étudier l'activité humaine. Pour comprendre l'activité dans sa totalité, c'est-à-dire de sa genèse à sa traduction en actes observables, il est nécessaire de s'appuyer sur des théories, issues des sciences de la cognition, et de les combiner à des techniques d'observation et d'entretien. Les observations permettent la collecte des traces à partir desquelles l'ergonome pourra inférer les déterminants inobservables de l'activité.

1.2 Traces

Le mot « trace » possède quatre acceptions différentes selon le dictionnaire Le Petit Robert (2010). Les deux premières relèvent d'empreintes ou de marques résultant d'une action, la troisième correspond à la notion de très faible quantité et la dernière est spécifique aux mathématiques. Ce sont les deux premières définitions que nous retenons dans le cadre de cette thèse :

> 1. Empreinte ou suite d'empreintes, de marques que laisse le passage d'un être ou d'un objet. *« des traces de pas sur la neige conduisaient à un pavillon »* **CARGO.** *Disparaître sans laisser de traces. Perdre, retrouver la trace d'un fugitif.* ► **piste.** *Les traces d'une bête.* ► **brisées, foulée, pas, passée.** *Suivre un gibier à la trace* [...] ◇ Ce à quoi on reconnaît que quelque chose a existé, ce qui subsiste d'une chose passée. ► **reste, vestige.** *« la pioche minutieuse des archéologues découvre, couche par couche, la trace émouvante des civilisations »* **DANIEL-ROPS.**

> 2. Marque laissée par une action quelconque. *« il n'y avait nulle trace d'effraction »* **RENAN.** *Traces de freinage. Traces de coups.* [...]
>
> (Petit Robert, 2010.)

Pour illustrer ce qu'est une trace, nous allons nous appuyer sur l'exemple des empreintes laissées par le passage d'un animal dans la neige.

Cet exemple, bien que simple, permet d'illustrer les questionnements liés à l'utilisation de traces d'activité : que sont les traces ? Quelles sont leurs significations ? Que permettent-elles de déduire sur l'activité ? Dans quelle mesure ?

1.2.1 Qu'est-ce qu'une trace ?

Usages des traces

Comme le soulignent les définitions du dictionnaire, une trace traduit « le passage d'un être ou d'un objet » et est laissée par « une action quelconque » (Petit Robert, 2010). Une trace résulte donc d'une *action*. Et la trace est laissée, volontairement ou involontairement, par un être ou un objet.

Les traces de pas dans la neige sont le produit du passage de l'animal. Une trace est donc associée à une activité, ici le déplacement de l'animal.

La trace permet d'*inférer* l'existence d'une action passée (le déplacement de l'animal). Une trace permet typiquement d'apprendre des choses sur ce qui a laissé la trace. Une trace de pas dans la neige permet au pisteur d'inférer, non seulement qu'un animal se déplaçait, mais aussi de savoir quel type d'animal, quelle est sa taille approximative, dans quelle direction il se déplaçait, quelle était l'heure approximative de son passage, quelles étaient ses intentions, s'il cherchait de la nourriture, etc.

Interprétation des traces et choix d'observation

Une trace fait l'objet d'interprétations. Pour inférer toutes ces informations sur l'animal et son déplacement, le pisteur est amené à *interpréter* la trace. Le pisteur va chercher dans la trace les indices nécessaires à son interprétation. Mais cette interprétation ne pourrait se faire sans des connaissances supplémentaires dont dispose le pisteur : sur les animaux, par exemple leur mode de vie ; sur le contexte, par exemple les conditions météorologiques au cours des dernières heures ; ou sur les traces elles-mêmes, par exemple comment les empreintes dans la neige évoluent avec le temps.

Lorsque le pisteur regarde les traces d'un animal dans la neige, il va en déduire des informations sur l'animal. À partir exactement des mêmes traces, un physicien serait susceptible d'ignorer l'animal pour s'intéresser au fascinant phénomène de changement d'état de l'eau et de l'évolution de la forme des flocons de neige au fur et à mesure qu'ils fondent.

Pourquoi le physicien ignorerait-il les empreintes de l'animal ? Pourquoi le pisteur ignorerait-il les flocons de neige ? Parce que la trace est la trace de ce que l'on décide d'observer. Les empreintes ne sont que des *observables*. C'est l'observation, c'est-à-dire la collecte de la trace, qui donne le statut de trace aux éléments observés.

1.2.2 Observer n'est pas neutre : la collecte d'une trace

Une trace est issue d'une observation. Nous appelons cette observation qui *crée* la trace la « collecte de la trace ». Dans l'exemple du pisteur, la trace est laissée involontairement par l'animal et la trace est collectée a posteriori. Qu'en est-il de l'éthologue qui installe une caméra pour enregistrer le comportement des animaux ? On pourrait penser qu'il y a une différence de nature entre une trace fortuite laissée par l'activité d'un animal et une trace enregistrée intentionnellement par un dispositif technique. Cette question nous permet de distinguer trois dimensions de la trace : 1) la construction d'observables (ce qui peut-être observer), produits de l'activité, laissés intentionnellement ou non, 2) l'observation de ces observables, qui crée la trace et enfin 3) l'interprétation de la trace.

Observer modifie ce que l'on observe. Notons une première différence évidente : le dispositif technique d'enregistrement modifie l'environnement dans lequel interagit l'animal. En modifiant l'environnement, le dispositif peut influencer le comportement. Ainsi, l'éthologue prendra soin de faire en sorte que la caméra n'interfère pas avec les animaux qu'il étudie. Ceci est vrai lorsque l'on observe des activités animales, mais reste vrai lorsque l'on s'intéresse à l'humain. Par exemple, conduire un véhicule instrumenté avec des dispositifs dont on est conscient qu'ils enregistrent nos moindres comportements peut modifier notre façon de conduire.

Une observation est parcellaire et engagée. Nous l'avons déjà souligné, une trace est la trace de ce que l'on décide d'observer. La trace est donc une *construction*. Une trace n'est pas neutre. Elle est le résultat d'une observation *subjective*, qui répond au point de vue de l'observateur. La collecte n'est pas neutre. L'observateur décide de relever ou non un certain nombre de paramètres, que ce soit a posteriori, lorsqu'il collecte la trace (redessiner les contours des empreintes de pas, prendre des photographies, mesurer les dimensions des empreintes, la distance entre empreintes, etc.) ou a priori (installer une caméra, des capteurs de différente nature, etc.). Lorsque la trace est exploitée pour étudier scientifiquement des actions, l'observation fait l'objet de beaucoup de précautions pour permettre son objectivation et la construction de connaissances valides sur l'activité étudiée.

Les interprétations relèvent d'un point de vue. À partir de la même trace, c'est-à-dire des mêmes « données », les interprétations peuvent diverger. Tout d'abord, nous l'avons souligné, la même trace va produire des interprétations de nature différente en fonction de qui la regarde (le pisteur ou le physicien). Mais même lorsque deux pisteurs observent les mêmes empreintes, leur interprétation peut diverger. Ce n'est pas étonnant puisqu'il s'agit en réalité d'inférer ce qui n'a pas été observé à l'aide des simples indices présents dans une trace. Ces indices ne sont pas toujours suffisants, surtout lorsque l'on cherche à inférer des éléments, par nature non observables, comme l'intention d'un conducteur au volant de son véhicule. Notons que la situation est différente dans le cadre de l'interprétation scientifique des traces. Dans ce cadre l'interprétation d'une trace ne relève pas du point de vue d'une personne, mais de celui d'une communauté scientifique.[2]

[2] qui devrait interpréter les données de la même façon s'ils se réfèrent aux mêmes théories d'analyse.

Une trace est située. Une trace est liée à une situation d'observation. Tout comme l'activité s'inscrit dans un contexte, une trace est située dans un contexte. Par exemple, c'est parce qu'il a neigé que l'on peut observer les traces de pas de l'animal. Dans ce cas, le contexte météorologique est intimement lié à la trace. Dans d'autres cas, les éléments de contexte sont nécessaires à l'interprétation des traces. Ces éléments sont relatifs aux conditions de la collecte. Par exemple, dans le cadre de la conduite automobile, le brouillard détériore l'observation vidéo de la scène routière, un problème technique, ou la panne d'un capteur peut expliquer des mesures incongrues, etc. Il faut remettre la trace dans le contexte de sa collecte pour pouvoir l'interpréter correctement.

1.2.3 Traces – Conclusion

Une trace est produite par une activité, mais c'est l'observation (la collecte) qui lui donne son statut de trace. Une trace est une observation située et une lecture orientée de l'activité : elle résulte de choix d'observation. Une trace contient des éléments non seulement relatifs aux actions, mais aussi à leur contexte. Ces indices sur l'action et sur son contexte permettent d'inférer ce qui n'a pas été observé ou ce qui n'est pas observable.

1.3 Modéliser l'activité en analysant des traces

Activité et traces sont liées. Nous l'avons vu, pour étudier l'activité, la principale méthode de l'ergonomie est l'observation. L'observation consiste pour partie à collecter et analyser des traces de l'activité. Les traces, nous l'avons vu, sont des traces d'actions dans un contexte et contiennent des indices de l'activité. Étudier les traces permet donc d'appréhender l'activité. Étudier des traces, c'est interpréter les indices qui y sont présents pour leur donner du sens. Cette interprétation permet de construire des connaissances sur l'activité étudiée. La figure 1.1 résume l'ensemble du processus.

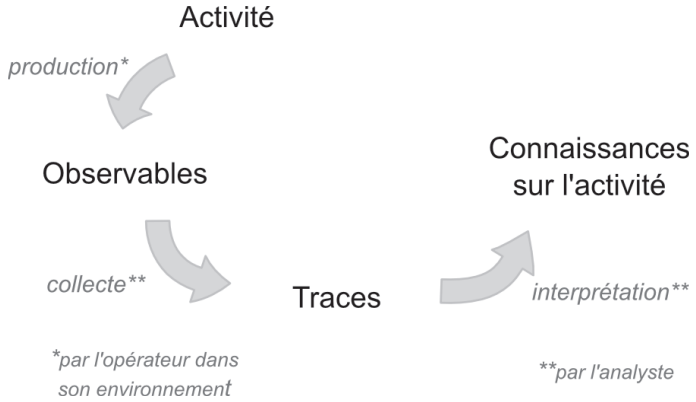


FIGURE 1.1 – Connaître l'activité à partir de traces. L'activité située produit des observables. Ces observables, collectés par l'analyste, forment des traces d'activité. En interprétant les traces, l'analyste construit des connaissances sur l'activité.

Cependant, nous n'avons pas encore précisé quel est le statut de la connaissance. Cette dernière section présente les questions relatives à l'ingénierie des connaissances :

qu'est-ce qu'une connaissance ? Au-delà des connaissances sur l'activité, peut-on construire des modèles de l'activité ? Quels sont les liens entre traces, connaissances et modèles ? À l'issue de cette section, nous détaillerons en quoi ce travail de thèse éclaire ces différentes questions.

1.3.1 Connaissances ? Ingénierie des connaissances

Lorsque nous disons que l'interprétation des traces d'une activité permet à celui qui observe la trace de connaître l'activité, nous introduisons le concept de connaissances.

La connaissance à laquelle nous nous intéressons est la connaissance relative à l'activité. Nous pouvons distinguer plusieurs types de connaissances, par exemple les connaissances qu'a l'opérateur sur sa propre activité et celles qu'a un observateur extérieur sur l'activité d'un opérateur.

Quelle est la connaissance sur l'activité et à quoi sert-elle ?

Dans le cadre de cette thèse, les connaissances auxquelles nous nous intéressons sont les connaissances qui permettent de produire l'activité étudiée. Une connaissance, comme le souligne Bachimont (2004a), est la « capacité à réaliser une action pour atteindre un but visé ». Ce sont donc en premier lieu les connaissances de l'opérateur qui nous intéressent, car c'est lui qui met en œuvre l'activité.

Les connaissances de l'opérateur sont en partie explicites et en partie implicites. Elles sont explicites quand l'opérateur sait « ce qu'il sait », et qu'il est capable de verbaliser ses connaissances. C'est le cas des connaissances scientifiques par exemple. Les connaissances sont implicites quand l'opérateur ne peut les verbaliser. Il s'agit par exemple des savoir-faire ou des connaissances relevant de l'« art de faire ». Ce sont les *connaissances procédurales* que décrit Anderson (1983).

Les connaissances de l'opérateur sur l'activité ne sont donc pas forcément accessibles à l'opérateur par introspection. Certaines connaissances ne lui sont accessibles qu'à travers l'exécution de l'activité. Autrement dit, l'opérateur possède des connaissances qui lui permettent de réaliser l'activité, mais il n'a pas conscience de posséder ces connaissances.

Pour étudier l'activité, un observateur extérieur, par exemple un opérateur expert ou un ergonome, peut construire des connaissances sur l'activité d'autrui (Georgeon, 2008). Dans ce cas, il est question d'analyser l'activité d'une personne, à partir d'observations de cette activité, pour mieux la comprendre et construire des connaissances permettant, par exemple, de modifier l'activité. Les connaissances construites sont relatives à l'activité, mais ne sont pas nécessairement les connaissances permettant à l'opérateur d'exécuter l'activité. Dans ce cas, les connaissances sur l'activité sont différentes des connaissances qui sont réellement mises en œuvre par l'opérateur pour réaliser l'activité.

C'est le rôle de l'ergonome cogniticien d'étudier les connaissances réellement mises en œuvre par l'opérateur pour réaliser son activité (Gillet, 1987). Bien qu'observateur extérieur, il cherche à reconstruire les connaissances de l'opérateur (aussi bien implicites qu'explicites), par le biais des observations et entretiens avec l'opérateur. Nous nous positionnons dans une approche complémentaire à celle développée par Georgeon (2008). En effet, il se focalisait sur l'analyse de la partie observable de l'activité, telle qu'elle est comprise par un observateur extérieur, alors que nous cherchons ici à modéliser la partie cachée de l'activité : les connaissances que l'opérateur utilise pour réaliser son activité.

Dans cette thèse, nous appellerons *analyste* la personne qui observe et analyse les traces d'activité. Dans ce document, l'analyste sera un *ergonome cogniticien*. L'activité étudiée sera l'*activité de conduite automobile* dans ses composantes comportementales et cognitives. L'objectif de l'analyste est de comprendre l'activité humaine dans sa complexité naturelle pour améliorer l'activité étudiée (la rendre plus efficiente, réduire les risques, améliorer le confort, etc.).

Les connaissances construites par l'analyste représentent les connaissances du conducteur automobile. L'objectif est donc de modéliser les connaissances du conducteur. Dans cette thèse, les connaissances sur l'activité de conduite automobile relèvent de deux niveaux : les connaissances qu'a le conducteur et qui lui permettent de conduire une voiture et les modèles que construit l'analyste de ces connaissances. Ces modèles sont aussi des connaissances dans la mesure où ils permettent, à un conducteur les apprenant, ou à un environnement informatique de simulation les utilisant, de conduire.

Notons ici que l'analyste est aussi un opérateur (Falzon, 1993). L'activité qu'il exécute est une activité d'analyse et de modélisation. L'analyste exploite des connaissances qui sont relatives à son activité (d'analyse). Il est important de distinguer les connaissances sur l'activité de conduite automobile, qui sont celles de l'opérateur-conducteur, des connaissances sur l'analyse de l'activité, qui sont celles de l'opérateur-analyste. Dans le cadre de ce travail de thèse, nous n'étudierons pas l'analyse de l'activité de l'analyste. Par contre, nous aurons l'occasion de souligner l'importance des connaissances de l'analyste pour mener à bien le travail de modélisation (de l'activité du conducteur).

Comment construire les connaissances sur l'activité ?

L'observation est une source de connaissances. Mais quel est le lien entre l'analyste, les traces qu'il interprète et les connaissances ? L'ingénierie des connaissances et l'épistémologie donnent un cadre à ce problème.

Une trace n'est pas en soi une connaissance. C'est l'interprétation de la trace qui constitue une connaissance. Ainsi, en reprenant l'expression de Bachimont (2004b), la trace serait une inscription de connaissances, du fait qu'elle est sujette à interprétation et que c'est cette interprétation de la trace qui est connaissance : « les connaissances ne s'appréhendent qu'à travers des inscriptions matérielles qui les expriment, et dont elles sont l'interprétation. »

Notons ici que la connaissance issue de l'interprétation de la trace n'est pas nécessairement une connaissance de conduite (pour l'opérateur), mais plutôt une connaissance pour l'analyste. À partir de ces connaissances d'analyse, l'analyste infère les connaissances de conduite qu'il estime être celles de l'opérateur.

Dans ce contexte, le travail d'analyse des traces pour leur donner du sens est un travail cognitif de la part de l'analyste. L'ingénierie des connaissances porte sur ces inscriptions de connaissances et vise à « instrumenter et outiller le travail de la connaissance ».

Ainsi, la construction de sens se fait toujours par l'humain. Cette interprétation s'appuie non seulement sur des éléments matériels, tels que les traces, mais aussi sur d'autres connaissances que possède l'analyste. Ces autres connaissances – sur le domaine étudié (la cognition humaine) ou sur l'activité étudiée (l'activité de conduite automobile) –, dont nous présenterons un aperçu au chapitre 2, permettent à l'analyste de construire l'interprétation de ce qu'il observe.

Conclusion

L'ergonome s'appuie sur des connaissances qu'il a déjà, sur l'activité étudiée et sur la cognition humaine. Ces connaissances lui fournissent un cadre pour interpréter les traces produites par l'activité d'un opérateur et pour ensuite construire de nouvelles connaissances sur cette activité.

Mais le travail de l'ergonome ne se limite pas à connaître l'activité. Pour mieux la comprendre, le chercheur en ergonomie cognitive cherche à construire un type particulier de connaissances : *des* modèles *de l'activité*.

1.3.2 Connaissances et modèles

Qu'est-ce qu'un modèle en ergonomie ? Leplat (2003) nous éclaire sur la question. Il cite deux définitions que nous reprenons ici. La première définition, de Régnier, est la suivante : « Un modèle d'un objet concret est un objet abstrait dont la description est tenue pour nous pour une description dudit objet concret. » La seconde définition formulée par de Montmollin présente un modèle comme « une représentation des comportements d'opérateurs dans une situation de travail [ou de vie[3]], et permettant d'agir sur cette situation ».

Leplat (2003) souligne la nécessité de distinguer le modèle et l'objet modélisé. Le modèle est nécessairement une simplification de l'objet concret. Un modèle vise à caractériser certains traits de l'objet modélisé, en l'occurrence l'activité. Et plus que ces traits, un modèle vise aussi « à définir leurs relations afin de mieux comprendre le fonctionnement de ce système et de donner les moyens de le modifier » (Leplat, 2003). Ainsi, l'ergonomie construit des modèles afin de mieux *comprendre* l'activité et disposer d'outils intellectuels pour modifier l'activité.

Objectif du modèle. Les modèles peuvent répondre à trois objectifs : décrire, expliquer ou prédire (Sperandio, 2003). Un modèle descriptif de l'activité permet de décrire ce qu'il s'est passé. Un modèle explicatif permet de comprendre les raisons sous-jacentes qui ont engendré cette activité. Un modèle prédictif permet de prédire le comportement futur de l'opérateur. Un modèle peut posséder une ou plusieurs de ces facettes. Nous l'avons vu, le caractère explicatif d'un modèle est particulièrement utile à l'ergonome. Dans l'idéal, un modèle est à la fois descriptif, explicatif et prédictif.

Une trace n'est que l'inscription d'une histoire produite par l'activité. Par nature, une trace contient donc des informations descriptives sur l'activité. C'est seulement par l'interprétation que l'analyste peut formuler des hypothèses permettant d'expliquer les comportements observés. Un modèle explicatif nécessite donc l'intervention de l'analyste qui va se servir de ses connaissances sur l'humain et sur l'activité pour produire ces explications.

Quelle est la différence entre une explication et un modèle explicatif ? La première relève d'une connaissance isolée sur l'activité, alors que la deuxième relève d'une représentation structurée et intégrée dans un cadre plus large : celui du modèle.

Une approche purement descriptive donne des informations sur l'activité dans son *extension*, c'est-à-dire sur des instances de l'activité. À ce titre, les traces constituent une description de l'activité. À l'inverse, la structuration des connaissances offerte par un modèle explicatif apporte des informations sur l'activité dans son *intension*, c'est-à-dire sur la nature même de l'activité. Neisser (1976), puis Stanton *et al.* (2009) et

[3] L'ergonomie a d'abord été focalisée sur l'étude des situations de travail avant d'être élargie aux situations de vie en général.

Bellet *et al.* (2009) présentent cette différence avec la métaphore du génotype et du phénotype de l'activité, empruntée au domaine de la génétique. Le *génotype* renvoie à une connaissance sur l'activité dans ses potentialités, alors que le *phénotype* correspond à une certaine expression de ce génotype lors de la réalisation effective, en situation, de l'activité. Les traces représentent ainsi l'activité telle qu'elle s'est exprimée en situation (phénotypes), alors que le modèle représente l'activité dans sa potentialité (génotype).

Quelle est la portée d'un modèle de l'activité ? Une activité implique l'humain à différents niveaux, par exemple, sociaux, cognitifs ou biomécaniques. Au niveau social, l'opérateur peut partager une certaine vision de son environnement ou de ses comportements, par exemple ce qu'est un comportement à risque, ce qui est acceptable ou ne l'est pas par la société ou par ses pairs, etc. Au niveau cognitif, l'opérateur raisonne, formule des intentions, prend des décisions. Le niveau cognitif peut inclure des éléments perceptifs (ce que l'opérateur perçoit de l'environnement), des représentations mentales (comment l'opérateur se représente le monde, la situation), des compétences sensori-motrices (des savoir-faire), il peut impliquer l'usage de la mémoire et du raisonnement, etc. Au niveau biomécanique, on peut étudier les postures de l'opérateur, les contraintes mécaniques liées à l'usage d'outils, le positionnement spatial de l'opérateur et de son environnement de travail.

L'activité est un processus unifié. L'opérateur n'est pas décomposé selon chacune de ces dimensions sociales, cognitives, biologiques lorsqu'il effectue une activité. Un modèle d'activité se doit donc de prendre en compte les composantes de l'opérateur qui sont pertinentes au regard des objectifs de l'activité et faire en sorte de proposer une vision unifiée de ces composantes.

Dans le cadre de la conduite automobile, par exemple, l'aspect social de l'activité permet d'étudier les comportements de certaines populations vis-à-vis de la prise de risque (Banet, 2010), alors que l'aspect biomécanique permet d'améliorer la disposition du poste de conduite ou de réduire la gravité d'accidents, en étudiant la position typique du conducteur dans des situations d'accidents (Hétier, 2008). Il est impossible de construire un modèle complet de l'humain décrivant toutes les dimensions de l'activité. Dans cette thèse, nous nous focalisons à la fois sur l'étude des aspects cognitifs de l'activité et sur celle de la performance de conduite qui en résulte.

Comment construire un modèle d'activité ? Un modèle d'activité se construit en structurant les connaissances issues de l'observation de l'activité, c'est-à-dire des traces d'activité. Cette structuration des connaissances sur l'activité s'effectue, lorsque cela est possible, en accord avec d'autres modèles (et théories) issus de disciplines telles que les sciences de la cognition.

La boucle complète de construction de modèles de l'activité est présentée à la figure 1.2. Cette figure souligne le rôle de l'analyste dans la construction des modèles de l'activité.

L'objectif de cette thèse est de permettre la construction de modèles de l'activité à partir de traces. Dans notre cas la collecte de la trace est déjà définie par l'expérimentateur. Il est donc nécessaire de s'attarder plus particulièrement sur la question de l'interprétation des traces et de la construction de modèles à partir de traces.

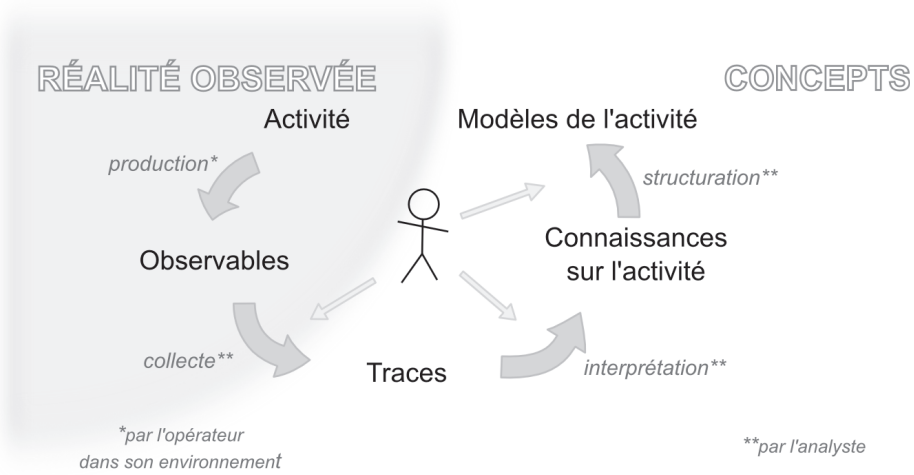


FIGURE 1.2 – Analyse et modélisation de l'activité à partir de traces. Au cours de son activité, l'opérateur produit des observables. L'analyste collecte ces observables pour en faire des traces d'activité. Les traces d'activité permettent, en les interprétant, de construire des connaissances sur l'activité, que l'analyste peut ensuite structurer en modèles de l'activité. L'analyste est donc au centre du processus qui permet de construire, à partir de ce que l'on observe de l'activité réelle, des connaissances et modèles conceptuels de cette activité.

1.3.3 Traces et modèles

Quels sont les liens entre traces et modèles de l'activité ? Nous l'avons vu, une trace est un ensemble d'informations relatives à l'activité telle qu'elle s'est *exprimée* dans un contexte donné. Ces informations sont parcellaires, car elles ne constituent que des mesures de l'activité, déconnectées de ses principes moteurs motivationnels. L'activité, elle, est unifiée. Plus précisément, l'activité ne se réduit pas à la somme des composantes que l'on peut mesurer, mais c'est un tout avec un objectif commun, immergé dans une situation unique. Un modèle de l'activité se doit d'être unifié. L'analyse de traces d'activité doit donc faire en sorte de retrouver l'unité de l'activité à travers la lecture morcelée qu'en donnent les traces.

Les traces seules ne sont pas suffisantes pour construire des modèles. Quelles sont les difficultés de cette entreprise ? Les difficultés sont de différentes natures. Premièrement, les traces ne contiennent pas explicitement les liens entre les différentes composantes de l'activité. La reconstruction d'un tout unifié se fait donc nécessairement par les connaissances qu'a l'analyste sur l'activité.

Contrairement à ce que l'on souhaiterait pour comprendre l'activité, les traces contiennent plus d'informations sur le produit de l'activité que sur les mécanismes qui ont été mis en œuvre pour générer l'activité. Il est donc nécessaire d'inférer ces processus non observés, voire non observables, de l'activité.

Ces processus non observables peuvent relever de différents niveaux de l'activité (actions mises en œuvre, mouvements de l'opérateur vs planification d'une série d'actions par l'opérateur). Elle repose sur différents niveaux de contrôle dont les

deux extrêmes sont l'activité contrôlée et l'activité automatisée. L'activité contrôlée (Schneider et Shiffrin, 1977) relève de processus de contrôle attentionnel relevant d'une conscience explicite (Bellet *et al.*, 2009). L'activité reposant sur des automatismes relève de compétences intégrées (*skill-based* d'après Rasmussen, 1986), ne nécessitant pas de ressources attentionnelles et dont l'opérateur a généralement une conscience implicite (Bellet *et al.*, 2009).

Relation épistémologique entre traces et modèles. Nous l'avons évoqué plus haut, les modèles explicatifs représentent l'activité en intension, alors que les traces représentent l'activité en extension. Il existe donc une relation épistémologique entre activité et traces. Cette relation entre traces et modèles est étudiée en mathématiques et il est possible de construire des modèles à partir d'un ensemble de traces (Diekert et Rozenberg, 1995). L'équivalence entre modèles synthétisant un ensemble de traces et les traces que ces modèles peuvent produire sont particulièrement étudiées dans le cadre de modèles automates tels que les automates à états finis ou réseaux de Petri.

En termes mathématiques, la question de construction de modèles à partir de traces se traduit ainsi : « considérant une activité dont on sait qu'elle est produite par un processus qui peut se modéliser avec le formalisme A, peut-on reconstruire un modèle de l'activité, à partir des traces d'activité de cette activité, en utilisant le formalisme A ? »

Cette question pose deux problèmes. Le premier problème est épistémologique : on ne peut pas savoir a priori si le processus peut se modéliser avec le formalisme A. Si comme dans les mathématiques, on recherche un modèle exact, il est impossible de savoir a priori si le formalisme A permet de modéliser le processus étudié. Si par contre on s'autorise un modèle simplificateur de l'activité, ce qui sera toujours le cas en ergonomie, il est possible de choisir le formalisme A en fonction de sa capacité à traduire les traits caractéristiques de l'activité étudiée.

Le deuxième problème est empirique. La construction mathématique de modèles à partir de traces pose des conditions sur l'exhaustivité de la trace et la présence de suffisamment d'informations pour reconstruire le modèle. Dans le cas de traces d'activité humaines, les traces ne sont pas exhaustives : elles sont parcellaires et certaines informations essentielles à la compréhension de l'activité manquent.

1.4 Problématique : méthodes, algorithmes et outils pour construire interactivement avec un analyste des modèles d'activité à partir de traces de l'activité observée

Ce chapitre a permis de poser les questions épistémologiques qui sous-tendent notre travail de thèse. Ce contexte épistémologique permet de situer notre travail de thèse vis-à-vis de questions qui dépassent une simple discipline. Au-delà des questions épistémologiques, l'articulation entre ergonomie, sciences de la cognition, ingénierie des connaissances et mathématiques a été évoquée.

Ce contexte épistémologique posé, il est à présent possible de présenter plus clairement les objectifs de cette thèse. Cette thèse vise à **concevoir des outils pour un analyste, lui permettant de construire des modèles d'une activité humaine à partir de traces de cette activité et de ses connaissances d'analyste**.

Analyste. L'*analyste*, ici, est un ergonome, car il étudie l'activité humaine. Dans cette thèse, l'activité étudiée est l'activité de conduite automobile, dans ses composantes comportementales et cognitives. L'analyste est donc un ergonome cogniticien, spécialiste de l'activité de conduite automobile.

Modèles. Le but de ces *modèles* est de mieux *comprendre* l'activité. Ces modèles doivent donc être des modèles en intension, plutôt que des modèles en extension. Ainsi, les modèles que l'on cherche à construire dans cette thèse doivent synthétiser l'activité. Il s'agit de construire des modèles explicatifs, éventuellement prédictifs, de l'activité.

Activité. L'*activité* est un processus unifié, visant un but. Une activité humaine prend place au sein d'une situation. L'activité implique des composantes non observables, telles que la cognition du sujet et des composantes observables, produits de l'activité au sein de la spécificité de la situation.

Traces. Les éléments observables de l'activité, tels qu'ils sont collectés par un observateur, constituent des traces de l'activité. Ces *traces* sont le matériau à partir duquel nous cherchons à reconstruire des modèles de l'activité. Cependant, les traces ne sont qu'inscriptions de connaissance. Pour produire des connaissances sur l'activité, il est nécessaire d'interpréter les traces. Pour produire des connaissances sur les aspects non observables de l'activité, il faut aussi effectuer des inférences à partir des traces.

Connaissances. Les traces seules ne peuvent donc pas permettre de connaître l'activité. Il est nécessaire d'exploiter les *connaissances* d'un analyste sur l'activité, sur l'humain, sur les processus de collecte des traces pour interpréter correctement l'activité, dans ses composantes observables et inobservables, à partir des traces.

Structure du travail de thèse

Ce travail de thèse est principalement un travail d'ingénierie des connaissances puisqu'il s'agira d'aider l'analyste à manipuler les inscriptions de connaissances que sont les traces pour construire des modèles descriptifs, explicatifs et prédictifs de l'activité.

Le cadre spécifique de l'activité de conduite automobile sera présenté au chapitre 2. Ce chapitre nous permettra de présenter les spécificités de l'activité étudiée et ainsi donner un cadre conceptuel (relevant des connaissances de l'analyste) de la modélisation de l'activité de conduite automobile.

Les concepts et méthodes d'ingénierie des connaissances existants et sur lesquels nous nous appuyons seront présentés au chapitre 3. Il s'agit notamment de la théorie de la trace modélisée, qui formalise les questions d'interprétation et de reformulation de traces.

Pour assister l'analyste dans son travail de modélisation, nous nous inspirons de travaux issus du monde de la fouille de données, recherchant des outils formels de construction de modèles à partir de traces. Ces outils sont présentés au chapitre 4 et au chapitre 5 nous proposons de modifier ces outils pour y intégrer les connaissances de l'analyste.

Le chapitre 6 revient sur l'approche globale d'analyse et de modélisation et propose une méthodologie permettant de mettre en œuvre notre approche. Le chapitre 7 présente la mise en œuvre concrète de cette méthodologie à travers une étude de cas.

CHAPITRE 2

Modéliser l'activité de conduite automobile

Nous avons vu dans le chapitre précédent que le travail de modélisation s'appuie sur une analyse combinant des traces d'activité et des connaissances qu'a l'analyste sur l'activité et sur la cognition humaine. Ce chapitre présente les connaissances dont dispose l'ergonome cogniticien pour construire des modèles de l'activité de conduite automobile.

Dans un premier temps, nous présentons des connaissances génériques sur la cognition et sur l'activité de conduite en particulier. Ensuite, nous introduirons la question de la modélisation de la conduite automobile à travers une approche historique. Puis, nous présenterons un modèle particulier, le modèle COSMODRIVE, et plus particulièrement la notion de *schémas de conduite*, qui sert de cadre à ce travail de thèse. Enfin, à partir des contraintes de la modélisation de l'activité de conduite automobile, nous préciserons les contours de ce travail de thèse.

2.1 L'activité de conduite automobile

La modélisation de l'activité de conduite automobile nécessite de connaître cette activité dans ses spécificités. Nous avons présenté au chapitre 1 ce qu'est une activité. Modéliser une activité humaine nécessite, nous l'avons vu, de s'intéresser à certains aspects, notamment cognitifs, de l'humain.

Cette section présente des caractéristiques de l'humain, principalement cognitives, qu'il faut considérer lorsque l'on étudie l'activité de conduite automobile. À ce titre, nous allons commencer par présenter des éléments de psychologie cognitive et introduire quelques modèles génériques qui serviront dans la suite de base sur laquelle nous nous appuierons pour présenter des modèles de l'activité de conduite automobile. Ensuite, nous présenterons les spécificités de l'activité de conduite automobile. Enfin, nous nous focaliserons sur les moyens d'observation dont nous disposons pour étudier l'activité de conduite automobile.

2.1.1 Cognition et modèles d'activité

La psychologie cognitive ainsi que l'ergonomie propose des modèles de la cognition. Nous allons présenter quelques modèles relatifs aux niveaux de contrôle (cognitif) de

l'activité et à la conscience de la situation. Il n'est pas question ici de donner une description détaillée de ces modèles cognitifs, mais plutôt d'introduire des concepts à la base de questions de recherche pour l'analyste.

Niveaux de contrôle de l'activité

Au cours d'une activité, plusieurs niveaux de contrôle sont mis en œuvre au niveau cognitif. Schneider et Shiffrin (1977) proposent un modèle cognitif différenciant les processus *contrôlés* et les processus *automatiques*. Les processus contrôlés sont séquentiels et requièrent l'attention de l'opérateur, au contraire des processus automatiques qui peuvent s'exécuter en parallèle sans engager d'effort attentionnel. Plus tard, Rasmussen (1986) propose un modèle distinguant trois niveaux de contrôle de l'activité. Les comportements mis en œuvre lors d'une activité sont contrôlés par :

- des habiletés sensori-motrices fortement intégrées (*Skill-based behaviours*),
- des règles de décision bien maîtrisées en vue de gérer des situations familières (*Rule-based behaviours*),
- et des connaissances plus abstraites et génériques, qui seront activées dans des situations nouvelles, pour lesquelles l'opérateur ne dispose pas d'expérience antérieure adéquate (*Knowledge-based behaviours*).

Pour une tâche donnée, par exemple dans le cadre de la conduite automobile, changer de vitesse (avec une boîte de vitesse manuelle) relèvera du niveau *knowledge-based* pour le conducteur débutant, alors qu'il s'agira d'une compétence automatique, *skill-based*, pour le conducteur expérimenté.

Rasmussen (1986) ne se contente pas de décrire ces trois niveaux, mais propose aussi un modèle de prise de décision explicitant l'articulation de ces trois niveaux au cours de l'exécution d'une activité. Ce modèle repose sur deux parties. La première partie correspond à l'analyse et à la compréhension de la situation, commençant par le niveau *skill-based* et remontant potentiellement jusqu'au niveau *knowledge-based*. La deuxième partie correspond à la planification de l'activité à mettre en œuvre, commençant potentiellement au niveau *knowledge-based* et redescendant jusqu'au niveau *skill-based* pour l'exécution concrète de l'activité. Le modèle de Rasmussen permet d'expliquer la résolution de problème, en allant des processus les plus intégrés, entièrement automatisés, jusqu'aux processus impliquant des raisonnements complexes. Le modèle explique comment les différents niveaux sont, ou ne sont pas, impliqués au fur et à mesure du traitement du problème.

Ces différents niveaux de contrôle de l'activité identifiés par Rasmussen (1986) peuvent être mis en parallèle de la conscience qu'a l'opérateur de sa propre activité. C'est par exemple la distinction que fait Piaget (1936) entre les actes *préréfléchis* – savoir-faire sensori-moteurs – et la *pensée réfléchissante* – schémas plus abstraits permettant au sujet de se représenter mentalement son activité (*abstraction réfléchissante*).

Conscience de la situation

Puisqu'une activité prend place dans le cadre d'une situation singulière, la conscience qu'a l'opérateur de sa propre activité est à mettre en parallèle avec la conscience qu'il a de la situation. Endsley (1995) propose une définition et un modèle de la conscience de la situation (*Situation awareness*). La conscience de la situation est ainsi constituée de trois niveaux : la perception des éléments de l'environnement de l'opérateur, la

compréhension de la signification de ces éléments et l'anticipation de l'évolution de l'environnement.

La conscience de la situation représente une sorte de modèle mental qu'a l'opérateur de la situation dans laquelle il évolue. Ce modèle mental permet de sélectionner les actions qu'il va mettre en œuvre avec des processus plus ou moins automatisés, plus ou moins conscients.

Bellet *et al.* (2009) proposent une évolution du modèle d'Endsley, présentée dans le cadre de l'activité de conduite automobile, faisant le lien entre la conscience de la situation et les niveaux de contrôle de l'activité (voir figure 2.1).

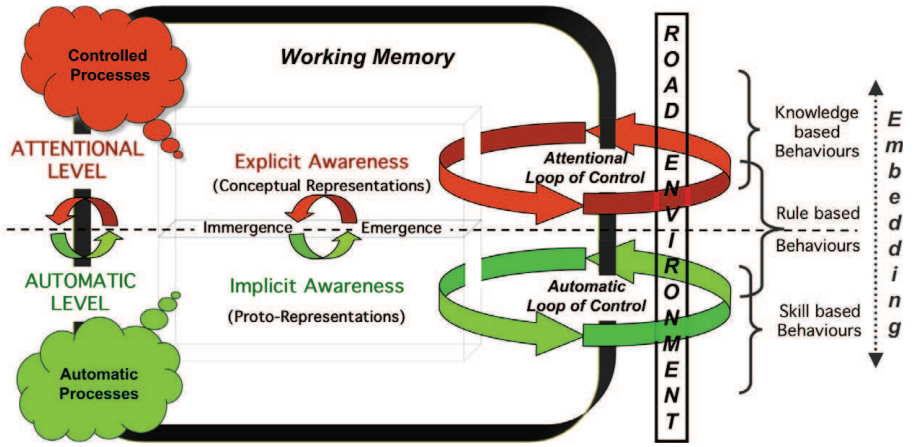


FIGURE 2.1 – Modèle de la conscience de la situation, d'après Bellet *et al.* (2009).

Bellet *et al.* (2009) distinguent ainsi les processus attentionnels, *explicites*, des processus automatiques, *implicites*. Les premiers sont explicites pour l'opérateur : l'opérateur a conscience de ce qu'il fait et peut le verbaliser, au contraire des processus automatiques, routiniers, dont l'opérateur n'a pas conscience explicitement pendant qu'il les réalise.

Un processus, même complexe, tel qu'effectuer un trajet routinier en voiture, peut relever d'un processus automatique. Dans ce cas, l'opérateur n'a pas forcément une conscience explicite de son activité, mais il en possède bien, néanmoins, une « conscience implicite » (Bellet *et al.*, 2009).

Bellet *et al.* (2009) s'intéressent à la relation entre processus explicites et processus implicites. Les deux processus coexistent lors d'une activité : conduire un véhicule est à la fois le fruit d'actions automatiques (changer de vitesse) et d'actions contrôlées (s'insérer sur une voie dans un trafic dense). Un processus explicite donné peut devenir implicite, c'est typiquement le cas lors de l'apprentissage. À l'inverse, un processus implicite, automatique, lorsqu'il conduit à une situation critique ou inattendue peut devenir brutalement explicite.

Motivation applicative de cette thèse

Le cadre d'application de cette thèse est de proposer un moyen d'étudier les différents niveaux de contrôle de l'activité (de conduite automobile), entre processus automatiques, implicites et processus contrôlés, explicites.

2.1.2 Spécificités de l'activité de conduite automobile

Alors que le contrôle du véhicule se résume grossièrement à deux composantes : la composante longitudinale avec la gestion de la vitesse et la composante latérale avec la gestion de la position sur la voie, l'activité de conduite cache une activité fortement dynamique, qui s'appuie sur des interactions complexes avec l'environnement routier et les autres véhicules.

Le conducteur doit collecter en permanence de nombreuses informations relatives à son environnement. Il doit en particulier être conscient de la présence des autres usagers de la route, avec lesquels il interagit, et de leurs intentions : tel véhicule va-t-il chercher à me dépasser ? Tel piéton va-t-il traverser la route ?

L'activité de conduite s'effectue sous une forte contrainte temporelle. En effet, un conducteur doit prendre des décisions dans un laps de temps très court. La contrainte temporelle est forte pour deux raisons : à cause de la dynamique du véhicule et parce que même si le conducteur ne fait rien, la situation continuera d'évoluer.

Pour comprendre les choix effectués par un conducteur, il faut non seulement comprendre la situation d'un point de vue objectif, mais aussi d'un point de vue subjectif. Il faut non seulement savoir quelle est la position des différents véhicules, mais aussi savoir comment le conducteur perçoit et se représente la situation, par exemple a-t-il pris en compte la présence de tel véhicule ? L'étude de la conscience de la situation prend ici tout son sens.

Les spécificités de l'activité de conduite impactent les moyens d'analyse de cette activité. La forte contrainte temporelle de la conduite automobile et sa complexité naturelle font que chaque situation de conduite est unique. Il est nécessaire de disposer de moyens permettant d'appréhender la conduite automobile dans sa complexité naturelle et dans la dynamique de la situation.

2.1.3 Observation de l'activité de conduite automobile

Pour comprendre l'activité de conduite automobile, il faut à la fois étudier l'activité réelle telle que l'on peut l'observer et l'activité telle qu'elle est vécue par le conducteur. Ces deux questions s'investiguent différemment.

Observer la conduite automobile

L'activité observable est étudiée à partir de traces d'observation. Les observations sont plus ou moins naturelles : elles peuvent être réalisées sur le terrain ou dans un environnement artificiel. Par exemple, l'ergonome peut observer les comportements de conducteurs au passage d'une infrastructure donnée, ou bien observer le comportement d'un conducteur à bord d'un véhicule (plus ou moins) instrumenté, sur route réelle ou dans un simulateur de conduite.

L'observation permet de mesurer le produit de l'activité : la performance de conduite. Les observations collectées correspondent à l'environnement de conduite (infrastructure, informations sur le trafic environnant, etc.), aux actions du conducteur sur les commandes du véhicule ou du résultat de ces actions (position du volant, enfoncement des pédales, vitesse du véhicule, etc.) ou aux actions perceptives effectués par le conducteur (regarder son rétroviseur, regarder devant, etc.).

On le voit bien, les observations sur le conducteur relèvent des actions motrices, qui elles-mêmes relèvent souvent de processus automatiques, implicites du point de

vue de l'activité (par ex., changer de vitesse). Faire le lien entre les différentes mesures, relevant d'actions implicites du conducteur, est difficile. Pour comprendre ce qui fait l'unité de l'activité, il faut accéder aux aspects cognitifs de l'activité ; et faire le lien entre cette activité cognitive et l'évolution de la situation de conduite dans son environnement.

Ainsi, les observations permettent d'étudier les actions mises en œuvre par les conducteurs. L'observation ne permet pas à elle seule d'étudier la situation telle qu'elle est vécue par le conducteur, ou d'inférer les processus cognitifs mis en jeu dans la conduite automobile. L'activité, telle qu'elle est vécue par le conducteur, s'investigue avec d'autres outils.

Étudier l'activité mentale du conducteur

La psychologie fournit des théories, des modèles et des outils pour étudier l'activité mentale du conducteur. Les modèles que nous avons présentés, par exemple sur la conscience de la situation, fournissent un cadre théorique permettant d'inférer des états mentaux ou des connaissances à partir d'observations et ainsi éclairer le fonctionnement de l'activité de conduite automobile. Un tel cadre théorique, combiné à des observations sur l'activité permet de faire des hypothèses sur la façon d'interpréter les traces d'observation, en faisant par exemple des hypothèses sur la façon dont le conducteur perçoit ou se représente la situation de conduite qu'il vit.

C'est typiquement dans ce contexte que la thèse se situe. L'analyste exploite des connaissances qu'il a sur la cognition pour émettre des hypothèses sur les relations entre états mentaux et traces, et ainsi évaluer la justesse des inférences produites (dans un cadre théorique donné).

Une partie des actions faites par un opérateur sont explicites, l'opérateur en a conscience au moment où il les exécute et pourrait verbaliser ce qu'il fait. Par exemple, il s'agit de choix conscients relevants de décisions, comme choisir d'éviter le périphérique parce que l'on sait qu'il y a des embouteillages. D'autres actions sont implicites, par exemple mettre sa ceinture de sécurité est une action machinale que l'opérateur n'a pas forcément conscience de faire.

Ainsi, comme nous l'avons présenté au chapitre 1, d'autres techniques peuvent être utilisées pour compléter l'observation. Il s'agit typiquement des entretiens. Les entretiens permettent d'étudier l'activité explicite du sujet. L'activité implicite est plus difficile à appréhender par entretien, parce qu'il s'agit de savoir-faire dont le sujet n'a pas conscience explicitement.

Pour étudier l'activité implicite, Vermersch (1994) propose une méthode d'entretien semi-guidé : l'*entretien d'explicitation*. Vermersch présente comment on peut, avec ce type d'entretien, interroger un sujet sur son activité implicite (ce que le sujet fait vraiment) sans l'inciter à construire une rationalisation (ce que le sujet croit avoir fait). Pour aider le sujet à expliciter son activité, il est possible de lui présenter des traces (typiquement des vidéos) de son activité. L'entretien d'explicitation devient alors un entretien d'autoconfrontation.

Les moyens d'observation exploités dans cette thèse

Dans le cadre de cette thèse, les observations sur lesquelles nous nous basons sont issues d'expérimentations effectuées par le laboratoire LESCOT. Deux types d'outils sont utilisés pour étudier l'activité de conduite automobile : des véhicules instrumentés et un simulateur de conduite.

Le véhicule instrumenté a l'avantage de permettre de confronter le conducteur à des situations de conduite réelles sur route ouverte, dans un environnement de conduite naturel. Le simulateur permet de mieux contrôler la situation de conduite et de confronter plusieurs sujets à la même situation.

Dans le cadre de cette thèse, nous nous focalisons sur des données de conduite issues d'un véhicule instrumenté, puisqu'il s'agit de comprendre la conduite dans sa complexité naturelle et dans la singularité de la situation.

Le véhicule instrumenté permet de collecter, à l'aide de capteurs, des informations sur les actions du conducteur (capteurs sur les pédales, le volant...) et sur la situation (télémètres, GPS et système de navigation). Ces informations numériques sont complétées de vidéos du conducteur et de l'environnement routier (vue avant, vue arrière).

Pour comprendre l'activité, ces observations numériques seules ne suffisent pas. Comme le soulignent Fastenmeier et Gstalter (2010), il est nécessaire de compléter ces mesures d'observations extérieures, faites par un expert en temps réel pendant l'expérimentation ou a posteriori, à l'aide d'enregistrements audio ou vidéo.

L'analyste s'appuie sur ces traces d'observation pour comprendre l'activité de conduite automobile et construire des modèles de cette activité. Il se sert de ses connaissances sur la cognition humaine et sur l'activité de conduite pour interpréter les traces.

L'objectif de cette thèse est de fournir des outils méthodologiques permettant à l'analyste d'exploiter ses connaissances pour interpréter les traces d'activité.

2.2 Des modèles du conducteur

Dans cette section, nous présentons les modèles du conducteur. Ce terme de *modèle de conducteur*, que nous n'avons pas encore défini, fait référence en réalité à des modèles de différentes natures, modèles à vocation descriptive, explicative, prédictive, voire même des modèles de simulation, et chaque modèle se focalise plus précisément sur un ou plusieurs aspects de la conduite automobile.

Pour mieux comprendre ces différentes dimensions de la modélisation du conducteur, nous commencerons par présenter, à titre illustratif, différents types de modèles remis dans leur contexte historique. Ces exemples nous permettront dans un second temps de présenter les questions relatives à la modélisation de la conduite dont nous dresserons un bref bilan.

2.2.1 Modèles du conducteur – approche historique

Dès la deuxième moitié du XX^{e} siècle, des modèles du conducteur commencent à faire leur apparition. Plutôt que de présenter une revue détaillée et exhaustive des différents modèles de conducteur disponibles dans la littérature, ce qui n'est pas l'objet de ce mémoire, nous préférons faire une revue rapide de différents *types* de modèles. Nous présentons ici quatre exemples de modèles. L'ordre dans lequel nous présentons ces exemples est cohérent avec l'évolution historique de la modélisation du conducteur. Les premières approches sont issues de la théorie du contrôle. Puis, dans les années 1970, des modèles taxinomiques, centrés sur la notion de tâche sont populaires. Face aux limites de cette approche, de nouveaux modèles, plus psychologiques, émergent. Ils sont focalisés sur la notion de risque et de prise de risque. Enfin, plus tard, dans les années 1990, une nouvelle génération de modèle apparaît, visant à combiner les approches antérieures aux théories de la cognition.

Modèles de régulation

Les modèles de régulation sont des modèles issus de la théorie du contrôle. Jürgensohn (2007) présente un bilan de cette approche de modélisation. Conçus par des ingénieurs, ces modèles visent à représenter le conducteur, non pas comme un sujet pensant, mais sous une forme mathématique. Concrètement, le conducteur est représenté sous la forme d'une équation différentielle, de la même manière qu'un contrôleur servomoteur. Le modèle mathématique est paramétré à l'aide de données issues de la conduite automobile. La figure 2.2 illustre ce type de modèle caractéristique de la théorie du contrôle.

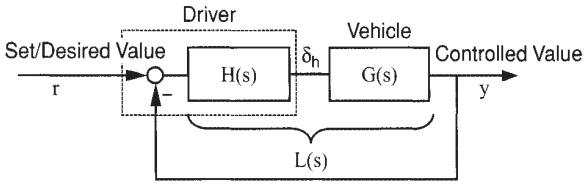


FIGURE 2.2 – Modèle de conducteur issu de la théorie du contrôle. Repris de Jürgensohn (2007).

Les modèles de régulation ne cherchent pas à modéliser la conduite automobile en général, mais plutôt à décrire ses aspects de régulation : régulation de la distance de suivi à un véhicule, régulation latérale du véhicule en fonction de la courbure de la route. Ce type de modèle, couplé à des modèles de véhicule – eux aussi représentés sous la forme d'équations différentielles –, permet de calculer la stabilité du couple conducteur-véhicule dans différentes situations de conduite.

Ces modèles sont loin d'avoir été abandonnés et sont encore raffinés aujourd'hui. En effet, ils sont tout à fait pertinents pour décrire le comportement du conducteur dans le contexte des situations de régulation (c.-à-d. suivi, maintien de la position latérale, etc.). Cependant, ces modèles ne décrivent qu'une partie de la conduite automobile, la partie la plus automatisée : ils ne permettent pas de modéliser les processus cognitifs (représentations et décisions sous-jacentes à l'activité) nécessaires pour pouvoir comprendre certaines ruptures dans l'activité que met en œuvre le conducteur (par exemple, prendre la décision de dépasser). Cette rupture justement traduit une différence de nature entre deux phases de conduite : le passage d'une situation de suivi à une situation de dépassement. Plus que des modèles de régulation longitudinale de suivi et latérale de maintien sur la voie, il faudrait disposer de modèles de régulation pour chaque phase d'activité (ou situation de conduite).

Modèles de la tâche

Au contraire des modèles de régulation, les modèles de la tâche visent à définir une taxinomie plus ou moins complète de la conduite automobile. Ce genre d'approche vise à décrire de manière plus ou moins exhaustive l'ensemble des tâches que doit réaliser un conducteur au volant.

La conduite est ainsi divisée hiérarchiquement en tâches et en sous-tâches. Chaque tâche représente un but que doit atteindre le conducteur en termes de performance. Les tâches sont organisées dans une taxonomie. Parmi les approches taxinomiques, celle de

McKnight et Adams (1970) est remarquable dans sa recherche d'exhaustivité puisque plus de 1700 tâches élémentaires ont été organisées en 45 tâches majeures.

Notons le travail d'Allen *et al.* (1971), qui propose une classification basée sur le niveau de performance. La performance du conducteur est distinguée selon trois niveaux, correspondant chacun à une fenêtre temporelle : la *microperformance* inférieure à la seconde s'intéresse aux actions du contrôle du véhicule, la *performance situationnelle* de l'ordre de la seconde à la dizaine de secondes relève du niveau de la manœuvre, la *macroperformance* représente la performance sur le long terme, de la dizaine de secondes à plusieurs heures et relève par exemple des tâches de navigation.

Cette deuxième taxinomie a l'avantage de proposer un modèle d'imbrication de tâches et de sous-tâches qui se rapproche de la conduite automobile telle qu'elle est effectivement mise en œuvre. Cependant, on atteint ici une des limites des approches taxinomiques : que les modèles soient exhaustifs ou non, ils ne permettent que de *décrire* la conduite automobile. Ces modèles taxinomiques se contentent de dire ce qu'il faut faire (tâche), mais n'expliquent pas comment le faire (activité). Aucun de ces modèles ne permet d'expliquer les comportements ou les mécanismes cognitifs sous-jacents dans leur mise en œuvre dynamique.

Modèles du risque

Afin de mieux appréhender les questions de sécurité routière et de mieux comprendre les mécanismes à l'origine d'accidents, de nouveaux modèles ont été proposés. Ces modèles se sont focalisés sur la notion de *risque* et de *prise de risque*. L'objectif était de mieux comprendre comment un conducteur estime les risques et prend des décisions. La notion de risque se ramène toujours au produit d'une probabilité et d'un coût (gravité). Plus que le risque « objectif » (extérieur), c'est la genèse du risque et l'évaluation subjective qu'en a le conducteur (risque perçu) que cherchent à représenter ces modèles. Une deuxième question centrale porte sur la prise de décision en fonction de cette estimation subjective du risque.

Certains auteurs postulent que le conducteur cherchera à éviter toute prise de risque (Näätänen et Summala, 1974), d'autres, tel Wilde (1982), postulent l'existence d'un principe d'« homéostasie », où le conducteur accepte un certain niveau de risque. Le modèle d'homéostasie du risque de Wilde (1982) postule que le conducteur ajuste en permanence sa conduite de manière à tendre vers un niveau de risque qu'il juge acceptable. Pour expliquer le mécanisme, ce modèle s'inspire des descriptions mathématiques du fonctionnement d'un thermostat.

Van der Molen et Bötticher (1988) proposent le modèle de hiérarchie du risque qui vise à fournir « un cadre structurel qui permette de décrire les processus de perception, de jugement et de décision à tous les niveaux de la tâche de conduite. » Ce modèle, présenté à la figure 2.3, reprend les trois niveaux de description de l'activité de Michon (1985) : le *niveau stratégique* relatif à l'itinéraire et à la navigation ; le *niveau tactique*, relatif à la manœuvre, au choix des actions à déployer dans le contexte de la situation ; et le *niveau opérationnel* relatif à la mise en œuvre concrète des actions. Un traitement particulier est réservé aux situations d'urgence, qui sont traitées exclusivement au niveau opérationnel, ce qui illustre la rupture effective dans le comportement de conduite pour répondre à une situation critique. Ce modèle décrit l'enchâssement des différents niveaux et illustre des mécanismes cognitifs de prise de décision en faisant appel à quelques expressions mathématiques.

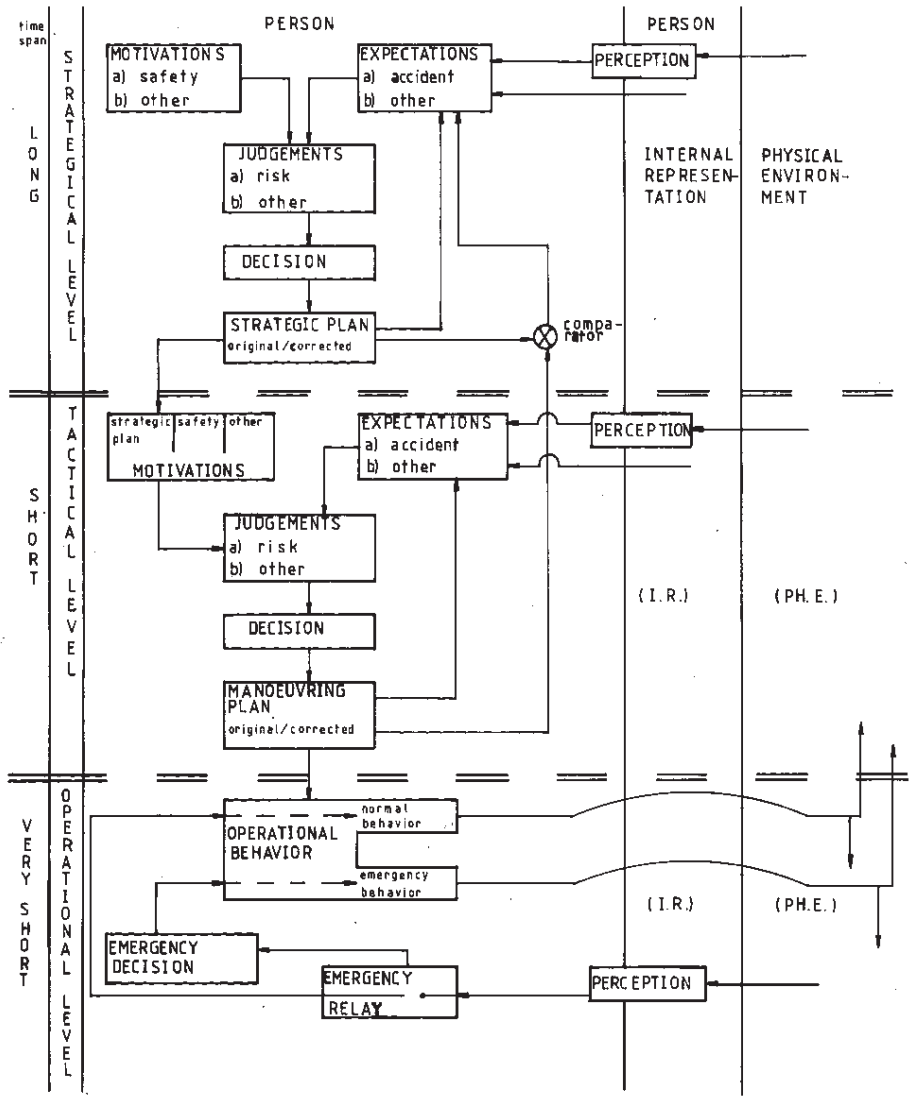


FIGURE 2.3 – Modèle hiérarchique du risque de van der Molen et Bötticher (1988). À tous les niveaux (stratégique, tactique et opérationnel), le conducteur perçoit son environnement, ce qui alimente ses *attentes*. En combinant ses attentes à ses *motivations* d'ordre stratégique, il construit un *jugement* de la situation qui lui permet de prendre des *décisions* et définir un *plan stratégique*. Ce plan stratégique détermine les motivations du niveau tactique, qui sont combinées à leur tour aux attentes du niveau tactique, amenant à un *jugement*, des *décisions* et un *plan de manœuvre*. Ce plan de manœuvre est comparé au plan stratégique. En cas de conflit, les attentes stratégiques sont revues. Sinon, le plan de manœuvre est mis en œuvre au niveau opérationnel. En cas de situation d'urgence, des *décisions d'urgence* court-circuitent l'ensemble du processus en agissant directement au niveau opérationnel.

Modèles cognitifs pour la simulation de conduite

Le modèle de hiérarchie du risque de van der Molen et Bötticher (1988), en intégrant des composantes cognitives, préfigure d'une nouvelle catégorie de modèles s'inspirant du fonctionnement cognitif humain pour simuler le comportement du conducteur. En effet, la psychologie cognitive propose des modèles expliquant comment l'humain traite l'information. Ces théories visent à expliquer à la fois comment l'humain perçoit le monde, comment il se le représente, comment il mémorise et traite les informations et comment il agit sur le monde.

Ces théories sont utilisées pour produire des modèles de conducteur capables d'aller au-delà de la simple imitation des performances du conducteur. Si l'on veut *comprendre* la genèse d'un comportement de conduite, il est nécessaire d'exploiter ces connaissances sur la cognition humaine.

Cette nouvelle génération de modèles s'appuie sur des architectures cognitives. Une architecture cognitive est une façon de concevoir la cognition humaine qui donne un cadre au modèle pour traiter les informations et interagir avec le monde.

Ces modèles possèdent des vertus explicatives, puisqu'ils permettent de décrire certains mécanismes de prise de décision. Cependant, de tels modèles ne permettent pas de prédire l'activité, car ils sont très macroscopiques à ce niveau et abordent plus le problème sous l'angle de l'intention (les raisons et les motivations d'une décision) que celui de l'action. La description de ces modèles n'est pas assez précise pour permettre d'être un support de simulation du conducteur et de son activité.

2.2.2 Modèles du conducteur – les enjeux

Sans chercher l'exhaustivité, nous avons présenté quelques types de modèles de conducteur pour en montrer la grande diversité. Les modèles sont construits selon plusieurs stratégies : directement à partir des données ou bien à partir du raisonnement et de la littérature. Les modèles répondent à différents objectifs : décrire, comprendre ou prédire le comportement de conduite automobile. Certains modèles se focalisent uniquement sur la performance observable, d'autres sur les façons de produire ces performances, c'est-à-dire sur les aspects cognitifs.

Modèles cognitifs et modèles de performance

Nous l'avons vu, certains modèles, tels que les modèles de régulation et modèles taxinomiques, se focalisent sur la performance du conducteur, c'est-à-dire ce que l'on peut observer, mesurer de l'activité du conducteur. Nous appelons ces modèles des *modèles de performance*. Ces modèles ont l'inconvénient de ne pas permettre de comprendre la genèse d'un comportement particulier. Sans cette genèse du comportement, difficile de modéliser le comportement imparfait de l'humain, typiquement les erreurs que peut commettre le conducteur. Les modèles de performance ne s'intéressent pas non plus aux états mentaux, à la façon dont le conducteur perçoit un risque, comprend une situation, ce qu'il ressent en terme de stress, peur, fatigue, etc.

Pour comprendre le comportement de conduite réel, dans son imperfection humaine, il est nécessaire d'étudier le fonctionnement cognitif humain. C'est en rentrant au cœur de l'inobservable, en reproduisant le fonctionnement des processus cognitifs de manière fidèle à ce qui se passe dans le système cognitif humain, qu'un modèle permettra de comprendre les raisons de telle ou telle performance observable. Nous appelons ces modèles des *modèles de simulation cognitive*.

Modèles descriptifs, explicatifs, de simulation

La plupart des modèles que nous avons pris en exemple sont des modèles conceptuels, décrits sous forme textuelle ou graphique. De tels modèles ne permettent pas la mise à l'épreuve du modèle pour le valider, contrairement aux modèles qui possèdent une traduction mathématique, comme les modèles de régulation.

Les modèles de tâche et du risque que nous avons présenté sont *documentaires*. Il s'agit d'une description de la conduite automobile. Cette description fournit un cadre pour étudier l'activité, mais ne permet pas toujours, par exemple pour les modèles de la tâche, de la comprendre.

Certains modèles ont une portée explicative. C'est le cas des modèles de risque. Mais comme nous l'avons noté, ces modèles ne sont pas assez précis pour permettre une évaluation du modèle. En fait, pour évaluer un modèle, au-delà de l'interprétation humaine que l'on peut en faire, il faut être capable de le mettre à l'épreuve dans une simulation.

Nous appelons *modèles descriptifs* les modèles qui permettent de dire ce qu'est la conduite. Nous appelons *modèles explicatifs* les modèles qui permettent de comprendre à l'aide d'une interprétation humaine comment le conducteur conduit. Enfin, les *modèles de simulation*[1] sont des modèles qui permettent de reproduire la conduite. Un modèle idéal, comme le souligne Sperandio (2003), réunit ces trois caractéristiques.

La simulation comme méthode d'investigation scientifique de la cognition et validation de modèle

Un modèle de simulation, tel que nous l'avons décrit dans le paragraphe précédent, est en fait un modèle « computationnel », c'est-à-dire un modèle dont le comportement peut être calculé. Le mot *simulation* traduit, en plus de l'aspect computationnel, la volonté de générer des comportements de conduite qui pourraient faire l'objet de comparaisons avec les comportements effectivement observés de conducteurs humains. Reproduire le comportement du conducteur permet de valider le modèle.

Pour tester un modèle de simulation, il est nécessaire de plonger le modèle dans un environnement de simulation, typiquement virtuel[2]. Dans son environnement de simulation, le modèle doit être capable de reproduire tout ou partie de la performance de conduite humaine.

Est-ce qu'il suffit à un modèle de reproduire le comportement de l'humain pour être valide ? Cela dépend de la portée du modèle (Sperandio, 2003). Si le modèle vise à reproduire la performance finale, tels les modèles de régulation, alors la simulation permet de valider le modèle. Si le modèle vise à reproduire le « fonctionnement » réel du conducteur, alors reproduire les performances ne suffit pas. Il faut aussi que l'architecture du modèle permette de rendre compte du fonctionnement du système cognitif humain, c'est-à-dire les processus cognitifs sous-tendant la performance observable.

Construire un modèle qui à la fois simule les activités cognitives humaines et à la fois imite les performances de conduite réelle est un problème difficile. En effet, la performance de conduite est de l'ordre de l'observable, alors que la cognition n'est pas directement observable. Ainsi, il est possible de construire des modèles de performance à partir de données collectées, mais un modèle cognitif, au contraire, doit être construit

[1] Nous ne parlons plus ici de modèles prédictifs. Les modèles de simulation sont des modèles prédictifs. Nous nous intéressons dans la suite spécifiquement aux modèles de simulation.

[2] Ou réel, en instrumentant le modèle de percepteurs et d'actuateurs

à partir des théories issues notamment de la psychologie cognitive. Employer la simulation cognitive comme méthode scientifique implique trois étapes. La première, c'est inférer le fonctionnement cognitif en combinant la théorie et les observations de la performance humaine. La deuxième étape, c'est « mettre en actes » cette théorie via la simulation (on fait l'hypothèse que le modèle est vrai). Enfin, l'évaluation du modèle se fait en comparant les performances du modèle à celle de l'humain.

Construire un modèle complet, qui soit à la fois un modèle cognitif et un modèle de performance, consiste en un difficile équilibre entre inférer un modèle à partir de données collectées et construire un modèle à partir de connaissances abstraites issues des sciences de la cognition.

Bilan relatif aux objectifs de la thèse

Pour *comprendre* la conduite automobile telle qu'elle est effectivement mise en œuvre, il est nécessaire de disposer de modèles explicatifs de la conduite. Pour valider ces modèles au-delà du simple consensus d'une communauté d'expert, il est nécessaire de construire des modèles de simulation. Simuler et comprendre l'activité de conduite automobile ne se fait pas sans étudier l'activité comportementale, ni sans la mettre en relation avec l'activité cognitive du conducteur.

Il est donc nécessaire de construire un modèle à la fois cognitif et capable de produire des performances de conduite. Ce modèle doit être à la fois descriptif, explicatif et capable de simuler la conduite automobile. Et pour construire ce modèle, il faut exploiter à la fois les connaissances sur l'humain et sa cognition, sur l'activité de conduite et sur les performances de conduite réelles telles que l'on peut les mesurer. C'est selon ces principes qu'a été développé le modèle COSMODRIVE.

2.3 Le modèle COSMODRIVE

Le modèle COSMODRIVE est un modèle de simulation cognitive (Bellet, 1998; Bellet et Tattegrain-Veste, 2003). Le modèle COSMODRIVE s'appuie sur une architecture cognitive et modulaire. Cette architecture est définie formellement : c'est à la fois un modèle cognitif se voulant explicatif de l'activité de conduite automobile, et un modèle spécifiant la construction d'une version informatique (computationnelle) du modèle.

Dans cette section, nous allons présenter le modèle COSMODRIVE, qui sert de cadre conceptuel dans lequel s'inscrit cette thèse pour comprendre l'activité de conduite automobile. Nous allons décrire rapidement les principes de l'architecture cognitive de COSMODRIVE et le rôle central de la représentation mentale. Puis, nous présenterons en détail la structure de représentation de connaissances utilisée par le modèle : les schémas de conduite.

2.3.1 Une architecture cognitive

Le modèle COSMODRIVE s'appuie sur une architecture fonctionnelle du système cognitif humain. La figure 2.4 présente de manière simplifiée cette architecture. Deux structures mnésiques sont distinguées : la mémoire de travail (MdT) et la mémoire à long terme (MLT). La mémoire à long terme stocke l'ensemble des connaissances du conducteur. La mémoire de travail contient seulement les connaissances mobilisées pour gérer la situation en cours et réaliser l'activité du moment.

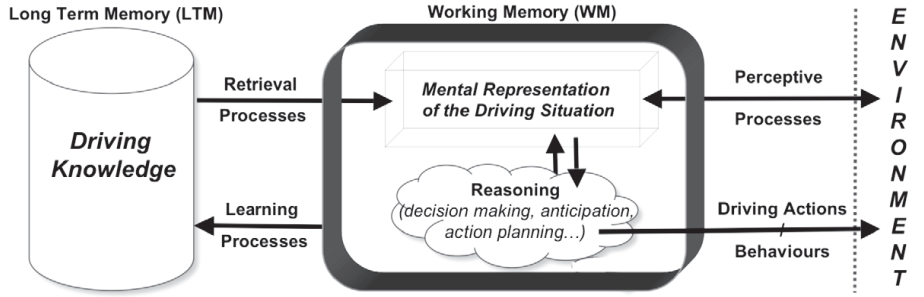


FIGURE 2.4 – Architecture élémentaire du système cognitif humain (Bellet *et al.*, 2009).

On distingue des processus : 1) liés à la mémoire (remémoration, apprentissage), à l'interface de la MdT et de la MLT, 2) liés à l'interaction avec l'environnement (perceptions, actions), à l'interface de l'environnement et de la MdT et 3) liés à la réalisation d'activités mentales en MdT (raisonnements au sens générique du terme). Le modèle COSMODRIVE détaille particulièrement les processus mis en œuvre dans la mémoire de travail : construction de représentations mentales et processus de manipulation de ces représentations mentales pour l'anticipation (simulation mentale), processus de décisions pilotant les actions (et stratégies perceptives) du conducteur.

2.3.2 Le rôle central de la représentation mentale

Pour traduire les différents niveaux d'activité, le modèle COSMODRIVE propose une approche modulaire, inspirée des modèles informatiques multiagents. Des modules spécialisés communiquent entre eux pour faire émerger la cognition. À l'instar du modèle hiérarchique du risque de van der Molen et Bötticher (1988), on retrouve les niveaux de contrôle stratégique, tactique, opérationnel et le niveau de gestion d'urgences. Chaque niveau est représenté par un module indépendant, ce qui permet des raisonnements parallèles. En plus de cela, Bellet (1998) introduit des modules de perception et d'exécution, qui explicitent l'interaction qu'a le conducteur avec son environnement. Enfin, un module lié au contrôle et à la gestion sert à contrôler les ressources cognitives attribuées aux différents modules.

Le module tactique est responsable de la gestion des manœuvres. C'est le module le plus intéressant à étudier, puisque c'est à ce niveau que la conduite automobile est la plus complexe à étudier et à modéliser.

Le module tactique s'articule autour de la représentation mentale de la situation de conduite. La représentation mentale contient une liste de buts que le conducteur vise à atteindre, une liste de faits sur la situation courante et une représentation spatio-temporelle de la situation. Cette représentation courante correspond ainsi à la conscience qu'a le conducteur de la situation telle qu'elle est vécue par le conducteur. Cette représentation est à la fois construite par l'intégration des informations perçues dans l'environnement via le module perception et par l'utilisation de connaissances extraites de la mémoire à long terme (processus d'activation).

À partir de la représentation courante de la situation de conduite, le module tactique dérive les évolutions futures possibles de la situation (simulation mentale). Ces anticipations, mises en regard des buts tactiques que le conducteur s'est fixés, servent

à la prise de décision, via une exécution mentale de l'activité avant même sa mise en œuvre.

Dans COSMODRIVE, les connaissances du conducteur sont de deux types : des connaissances catégorielles (catégories d'environnements routiers et lieux familiers) et des connaissances schématiques (schémas de conduite), combinant connaissances procédurales et déclaratives. Ce sont ces connaissances schématiques, instanciées à la particularité de la situation, qui permettent au conducteur de se représenter la situation de manière appropriée, de prendre des décisions, puis d'engager des actions. Le module tactique se charge d'identifier les situations de conduite (à l'aide des connaissances catégorielles) pour sélectionner le schéma (tactique) de conduite approprié à la situation, qu'il instancie à la situation courante dans sa spécificité (environnement, buts de conduite, etc.). Dans le modèle COSMODRIVE, les schémas de conduite sont au cœur : ils représentent la structure de représentation dynamique des connaissances de conduite.

2.3.3 Des schémas de conduite

Le concept de « schémas de conduite » vise à modéliser les connaissances humaines utilisées et mises en œuvre dans le cadre de la conduite automobile. Les schémas de conduite s'inspirent principalement des *schèmes* de Piaget (1936), en psychologie, et des *frames* de Minsky (1975), en intelligence artificielle. Les schémas de conduite sont des structures de connaissances contenant un savoir opérationnel, c'est à dire des connaissances pour l'action.

Un schéma de conduite est une structure de connaissances prototypiques (Rosch, 1975) représentant les compétences de conduite nécessaires à la réalisation de l'activité dans un type de situation de conduite, selon différents scénarios d'évolution possibles (en fonction des événements se produisant ou non, des décisions prises ou des actions engagées...). La figure 2.5 présente les schémas de conduite sous la forme d'un diagramme de classe UML[3] (Bellet *et al.*, 2009). Selon les auteurs, un schéma tactique est associé à un but tactique à atteindre dans une certaine infrastructure routière. Il se compose d'une trajectoire de conduite, elle-même définie comme une séquence de zones (de déplacement) et d'un ensemble d'actions (simples ou complexes). Ces actions peuvent être des sous-schémas, relevant du niveau opérationnel. L'exécution de ces actions est assortie de conditions à respecter et à vérifier, concernant l'occurrence d'événements dans certaines zones de l'infrastructure. Les zones relèvent de deux dimensions. Elles sont des zones d'évolution du véhicule ou des zones d'exploration perceptive d'une part. Elles sont absolues ou relatives d'autre part : absolues lorsqu'elles sont fixes par rapport à l'infrastructure, relatives lorsqu'elles dépendent de la vitesse et de la position du véhicule.

La figure 2.6 donne un exemple simplifié de schéma tactique et de ses ramifications en sous-schémas opérationnels (enchâssés dans le schéma tactique). Le schéma tactique « dépasser » est composé des sous-schémas opérationnels « changer de voie », « dépasser le véhicule », « se rabattre ». Chaque sous-schéma fait lui-même appel à des actions opératoires. Le sous-schéma « changer de voie », par exemple, sera composé de sous-schémas opératoires « mettre le clignotant », « accélérer », « tourner le volant », etc.

Comme pour les *frames* de Minsky, un schéma de conduite est une description de connaissance opératoire relative à une situation de conduite. Cela signifie que plus

[3] Unified Modeling Language

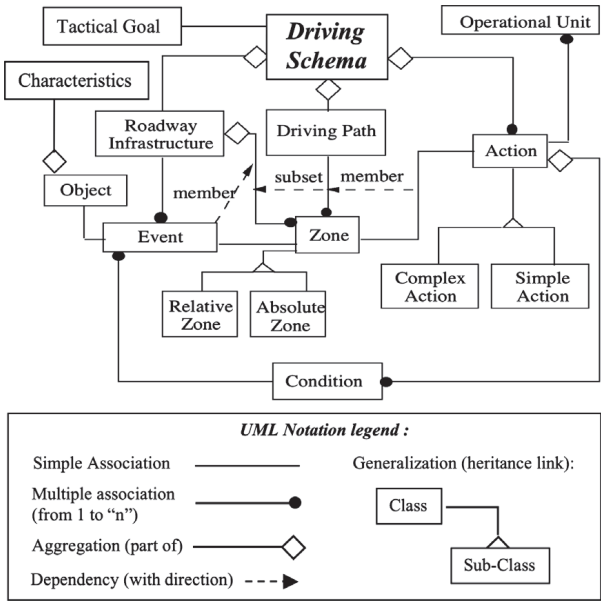


FIGURE 2.5 – Diagramme UML du schéma de conduite tactique, d'après Bellet *et al.* (2009).

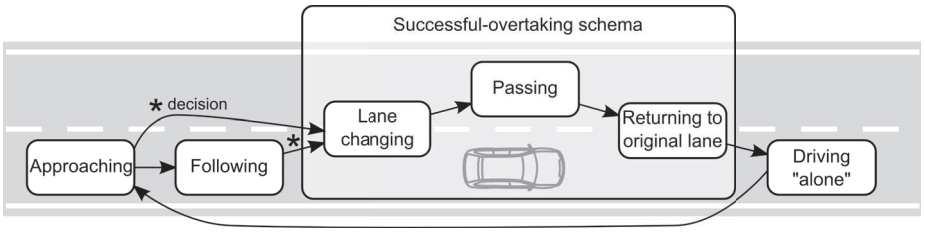


FIGURE 2.6 – Schéma tactique simplifié du dépassement sur autoroute et détail des sous-schémas opérationnels.

qu'un ensemble d'informations sur une situation de conduite prototypique, un schéma de conduite contient des éléments capables de piloter l'activité du conducteur dans la spécificité de la situation. Un schéma a pour but d'être déployé, c'est à dire exploité pour produire l'activité de conduite automobile. Dans COSMODRIVE, ce déploiement peut se faire soit à un niveau purement cognitif (pour anticiper des évolutions de la situation avant sa mise en œuvre), soit lors de la réalisation de l'activité (c'est-à-dire au moment de la mise en œuvre effective du schéma pour piloter le véhicule et progresser dans l'environnement). En ce but, un schéma est constitué de paramètres dont les valeurs par défaut correspondent à la situation prototypique. Lors du déploiement du schéma dans la dynamique d'une situation, le schéma pilote l'activité et les paramètres du schéma sont adaptés à la spécificité de la situation.

Suivant l'analogie empruntée au domaine de la génétique, formulée initialement

par Neisser (1976)[4], puis reprise par Stanton *et al.* (2009) et Bellet *et al.* (2009), le schéma de conduite relève davantage du génotype que du phénotype. Le schéma de conduite caractérise l'activité du conducteur dans sa potentialité, comme les gènes caractérisent un individu dans sa potentialité. Le schéma s'exprime à travers une situation de conduite pour produire un comportement de conduite, tout comme un gène s'exprime dans un certain environnement pour produire un phénotype particulier. Les schémas sont des connaissances génotypiques de l'activité, par opposition aux phénotypes que constituera l'activité telle qu'elle s'exprimera en situation lors de sa réalisation effective. Cette relation « génotype-phénotype de l'activité » est importante pour la thèse puisque par définition les traces nous éclairent sur l'activité phénotypique et les modèles de compétence de conduite que nous chercherons à appréhender à partir des traces se voudront génotypiques, à l'image des schémas de conduite.

Les schémas de conduite sont des structures de connaissances contenant un savoir opérationnel, c'est à dire des connaissances pour l'action. En fait, dans le modèle COSMODRIVE, les connaissances de conduite sont les schémas.

2.3.4 Un modèle de simulation

COSMODRIVE est un modèle de simulation. Le modèle COSMODRIVE s'appuie sur les schémas de conduite, qu'il utilise comme des connaissances de conduite. C'est le déploiement des schémas dans l'environnement de simulation qui produit l'activité de conduite automobile du modèle COSMODRIVE.

Pour produire une telle simulation, il est nécessaire de disposer d'une implémentation informatique du modèle COSMODRIVE, d'un environnement de simulation et de schémas de conduite pour alimenter le modèle.

Le programme COSMO-SiVIC[5], en cours de développement, vise à répondre à cet enjeu. COSMO-SiVIC (Bellet *et al.*, 2010, 2011b) est une implémentation, pour l'instant partielle, du modèle COSMODRIVE, dans l'environnement de simulation SiVIC[6].

Des schémas relatifs au niveau opératoire de l'activité ont déjà été développés : le schéma des zones enveloppes (voir figure 2.7), caractérisant un comportement très intégré (implicite) du conducteur sur sa régulation de vitesse en fonction de la présence d'obstacles.

2.4 Objectif applicatif de la thèse : modéliser les schémas de conduite

Dans cette section, nous allons faire un bilan des contraintes de la modélisation de schémas de conduite. Ces contraintes ont, dans la suite de ce manuscrit, des conséquences sur les choix qui sont faits dans cette thèse.

2.4.1 Les schémas de conduite comme connaissances

L'objectif de cette thèse est d'assister l'analyste dans sa démarche de modélisation de l'activité de conduite automobile et du conducteur. Nous avons vu que des modèles

[4]Cette analogie inspirée de la génétique a été formulée par Neisser (pp. 56–57, 1976) dans le cadre de schémas perceptifs.

[5]Modèle computationnel du conducteur automobile basé sur le modèle cognitif COSMODRIVE et implémenté dans l'environnement de simulation SiVIC

[6]Environnement de simulation informatique de capteurs.

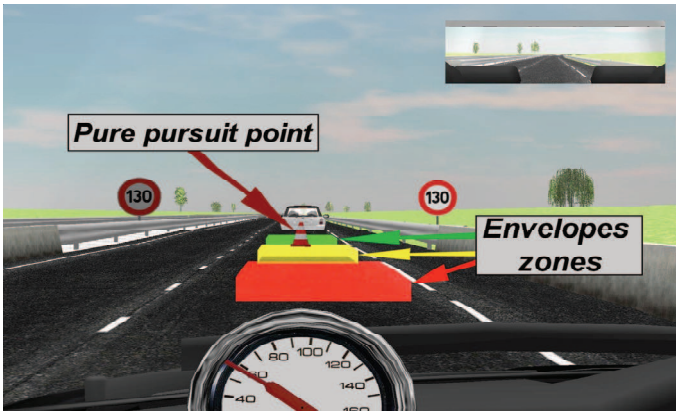


FIGURE 2.7 – Instanciation du schéma des zones enveloppes dans le modèle COSMO-SiVIC, d'après Bellet *et al.* (2011b).

du conducteur existent déjà. Dans le cadre de cette thèse, nous nous appuyons sur le modèle de simulation cognitive COSMODRIVE pour modéliser l'activité de conduite automobile.

Le modèle COSMODRIVE sépare les structures de connaissances (schémas de conduite) des processus mettant en œuvre ces connaissances (les agents cognitifs de COSMODRIVE). Les connaissances incluses dans les schémas de conduite traduisent des comportements possibles, susceptibles de s'exprimer ou non en situation (les schémas sont des génotypes). C'est de la confrontation de ces schémas avec une réalité situationnelle singulière qu'émergera le phénotype correspondant à l'activité observée.

Pour simuler l'activité de conduite automobile, le modèle COSMODRIVE doit être plongé dans un environnement de simulation et disposer d'une base de schémas de conduite. Sans environnement de simulation, le modèle ne peut produire d'activité. Sans les connaissances sur l'activité de conduite que sont les schémas, le modèle ne fera rien. Les schémas traduisent l'activité de conduite dans sa potentialité. Le modèle COSMODRIVE peut être vu comme un générateur d'activité permettant de produire l'activité en déployant progressivement les schémas au fur et à mesure de leur réalisation dans la situation de conduite.

Le modèle COSMO-SiVIC, en cours de développement, est une implémentation partielle de COSMODRIVE, connectée à un environnement routier virtuel. Cependant, il reste nécessaire d'alimenter ce modèle de connaissances sur l'activité, c'est-à-dire de schémas de conduite. Il est donc nécessaire de construire ces connaissances de conduite que sont les schémas à partir de l'activité réelle de conducteurs, afin d'accroître la validité du modèle.

L'objectif de cette thèse est **d'assister ce processus de construction des connaissances de conduite que sont les schémas de conduite** à partir de traces d'activité observées en conduite naturelle (Mathern *et al.*, 2010a). Les schémas de conduite sont des connaissances au sens défini au chapitre 1 pour le modèle COSMODRIVE. Les schémas de conduite sont des modèles des connaissances de conduite du conducteur. À ce titre, ils permettent à l'analyste de mieux comprendre l'activité de conduite automobile telle qu'elle est susceptible de s'exprimer en situation. La figure 2.8 présente les schémas de conduite comme connaissances. Les connaissances de conduite, dont

on fait l'hypothèse qu'elles existent chez le conducteur, sont utilisées par le conducteur pour produire l'activité de conduite telle que nous l'observons dans des traces. Les schémas de conduite constituent des modèles des connaissances de conduite, que l'on cherchera à construire avec des techniques d'analyse et de modélisation. Ces connaissances modélisées alimentent le modèle de simulation pour produire une activité de conduite et permettent à l'analyste de mieux comprendre l'activité de conduite.

L'un des enjeux applicatifs de la thèse est de permettre de générer des schémas à partir de traces (par ex., retrouver les schémas du conducteur et en produire une version computationnelle utilisable par un modèle comme COSMODRIVE). En partant de la trace d'activité réelle de conducteurs humains, on espère pouvoir découvrir des schémas proches des compétences de conduite effectivement utilisées par les conducteurs. À plus long terme (dépassant le cadre de cette thèse), l'objectif sera d'introduire les schémas découverts dans le modèle de simulation pour évaluer leur validité à travers leur mise en œuvre par l'activité de conduite du modèle. Si les performances générées et les traces d'activité observées du modèle et de l'humain sont proches, cela accréditera d'autant l'existence d'une équivalence entre les schémas découverts et les schémas de compétences utilisés par les conducteurs. Ce processus de découverte et validation est décrit à la figure 2.9.

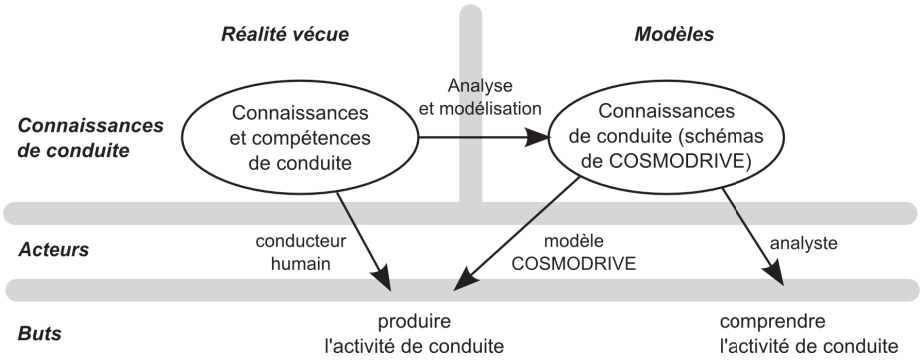


FIGURE 2.8 – Les connaissances de conduite du conducteur et leur équivalent dans le modèle COSMODRIVE.

Au-delà de l'objectif de modéliser ces connaissances que sont les schémas de conduite, l'analyste a aussi besoin d'explorer les relations qui existent entre différents schémas de conduite. Par exemple, quelles sont les relations entre un schéma tactique et ses sous-schémas ou des schémas opératoires ? Ou quelles sont les relations entre des schémas relevant de l'activité explicite ou de l'activité implicite ? Ces questions, bien que secondaires par rapport à l'objectif de modélisation de schémas de conduite, seront discutées dans le cadre de cette thèse.

Comment construire et valider les schémas de conduite ?

La figure 2.9 présente trois cycles de validation d'un modèle computationnel de l'activité. Ces cycles de validations font le parallèle avec la façon dont un modèle de l'activité est construit : par l'observation et la collecte de données d'une part, et par la simulation, quand le modèle est traduit sous forme informatique.

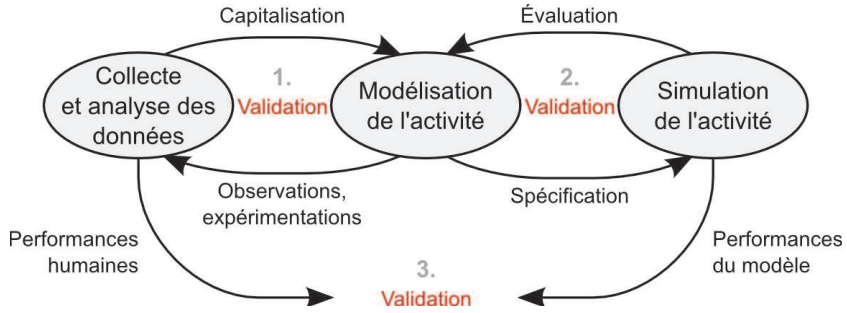


FIGURE 2.9 – Cycles de validation d'un modèle computationnel de l'activité. 1) Cycle où l'observation et l'expérimentation produisent des résultats que l'on capitalise dans un modèle. 2) Cycle spécifique où l'on vérifie que le modèle de simulation produit des résultats correspondants à ce qui est attendu du modèle. 3) Cycle de validation comparant performances humaines et performances du modèle.

Le premier cycle, vrai pour tout type de modélisation, consiste à collecter des données à travers des observations et des expérimentations. Ces données permettent de construire, raffiner et valider un modèle. C'est cette étape qui permet de construire les connaissances théoriques que sont, entres autres, les schémas de conduite.

Les deux autres cycles sont spécifiques aux modèles computationnels. Le deuxième cycle permet de spécifier la traduction informatique du modèle formel. Ce modèle informatique est plongé dans un environnement virtuel ou réel qui permet d'évaluer le modèle. Ce cycle permet d'évaluer la cohérence entre le modèle formel et sa traduction informatique exposée à l'environnement de simulation.

Pour pouvoir valider un modèle de simulation, il faut aller jusqu'à comparer les performances du modèle aux performances humaines. Cette dernière étape suppose d'avoir validé l'environnement de simulation et d'avoir validé la traduction du modèle formel sous sa forme informatique.

Dans notre cas, le modèle formel de conduite est constitué de deux composantes : les connaissances de conduite que sont les schémas et le « moteur d'interprétation » qu'est le modèle COSMODRIVE.

Le processus de validation complet d'une implémentation du modèle COSMODRIVE nécessite donc à la fois de valider le modèle COSMODRIVE, les schémas de conduite, et l'environnement de simulation.

2.4.2 Bilan

Au cours de ce chapitre, nous avons dressé le bilan des spécificités de l'activité de conduite automobile et de sa modélisation. Pour construire des modèles, en l'occurrence des schémas de conduite, il faut s'assurer que ces modèles soient capables de prendre en compte ces spécificités de l'activité de conduite automobile.

Contraintes liées à la nature de l'activité automobile

Nous avons vu au début de ce chapitre (p. 24) que l'activité de conduite est fortement *dynamique* et possède de fortes contraintes temporelles. Pour représenter ce caractère

dynamique, le modèle doit être capable de prendre en compte le temps, ou au moins l'ordre d'exécution de différentes actions et événements relatifs à la conduite.

La modélisation de la conduite automobile nous apprend que l'activité de conduite relève de différents niveaux de contrôles (automatisés, attentionnels) ou de différents niveaux d'activité (stratégique, tactique, opérationnel). La modélisation, même si elle ne porte que sur un niveau à la fois, doit permettre d'étudier les relations entre les différents niveaux de l'activité. Par exemple, la modélisation doit être capable d'explorer les relations hiérarchiques entre le niveau tactique et le niveau opérationnel de l'activité.

Les schémas de conduite visent à représenter l'activité dans sa potentialité (génotype de l'activité). L'activité de conduite s'exprime différemment en fonction du contexte. Les schémas capturent en quelque sorte les invariants de l'activité relatifs à une situation donnée. Ils capturent la structure de l'activité. La modélisation doit donc être capable, à partir de différentes situations observées, par nature singulières et uniques, de recréer un modèle *généralisant* capturant l'activité selon ses différentes potentialités.

Le schéma de conduite doit ainsi être capable de capturer des propriétés *structurelles* de l'activité telles que l'ordonnancement entre différents processus, potentiellement parallèles ou concurrents, c'est-à-dire des actions qui doivent être exécutées en parallèle (changer de voie et accélérer pour dépasser un véhicule) ou des actions qui nécessitent d'effectuer un choix (regarder la route ou le rétroviseur).

La capacité du modèle à généraliser l'activité (génotype) à partir de cas particuliers (phénotypes) est d'autant plus importante dès lors qu'il s'agit d'étudier des situations qui ne peuvent être observées qu'en petit nombre. C'est en particulier le cas des situations critiques ou atypiques. Ces situations sont pourtant des situations représentant un intérêt particulier pour l'analyste. La modélisation de ces situations particulières nécessite donc, en plus des capacités de généralisation du modèle, d'exploiter des connaissances de l'analyste pour assister à la construction d'un modèle pertinent.

Contraintes liées aux objectifs de modélisation

La modélisation sous forme de schémas de conduite vise à produire des modèles de l'activité qui sont descriptifs, explicatifs et capables de simulation.

Pour être descriptif et explicatif, le modèle doit être *interprétable* par un humain, en l'occurrence l'analyste. Le modèle a ainsi un rôle documentaire, expliquant ce qu'est l'activité de conduite. Les données de conduite collectées constituent des cas singuliers et ne sont pas en nombre suffisant pour construire des modèles statistiques. Dans ces conditions, le rôle explicatif implique d'avoir un modèle *déterministe*, qui dépend de l'environnement, de la situation et de paramètres liés au conducteur.

Enfin, pour la simulation, les modèles produits doivent être capables de représenter des connaissances sous une forme qui soit exploitable pour la simulation.

2.4.3 Conclusion

L'activité de conduite automobile est complexe et dynamique. Elle est complexe, car elle est multidimensionnelle et multi-niveaux : niveaux d'activité de Michon (1985), niveaux de contrôle de Rasmussen (1986) ou niveaux de conscience de Bellet *et al.* (2009).

L'étude de l'activité de conduite automobile se traduit par l'observation de performances de conduite au niveau des actions engagées par le conducteur. Pour comprendre l'activité, il est nécessaire d'inférer la partie cognitive de l'activité et de construire des

modèles qui synthétisent l'activité à partir d'actions singulières, comprises dans leur contexte.

Dans ce chapitre, nous avons présenté différents modèles du conducteur, en particulier le modèle COSMODRIVE qui sert de cadre conceptuel dans ce travail de thèse. Le modèle COSMODRIVE contient à la fois une base de connaissances qui permet de produire l'activité de conduite et des modules cognitifs visant à mettre en œuvre ces connaissances. Les modules cognitifs sont en cours de développement dans l'implémentation informatique COSMO-SiVIC. Dans le cadre de cette thèse, nous nous focalisons sur les schémas de conduite. Les schémas de conduite constituent les connaissances mises en œuvre par le modèle COSMODRIVE. Dans cette thèse, nous cherchons donc à assister l'analyste dans son travail de construction de schémas de conduite, c'est-à-dire dans la modélisation des connaissances du conducteur.

La construction des schémas de conduite est dans un premier temps un travail d'analyse exploratoire, puisqu'il s'agit de mieux comprendre l'activité de conduite en exploitant à la fois des observations singulières sur le terrain et des connaissances plus générales, sur l'humain ou le conducteur, dont nous avons présenté un aperçu dans ce chapitre.

Dans le chapitre suivant, nous présenterons comment, à partir de traces d'observations de la performance de conduite, il est possible de reconstruire des traces traduisant d'autres niveaux de l'activité, notamment cognitifs. Le chapitre 3 sera donc focalisé sur le phénotype de l'activité, c'est-à-dire des traces.

Le chapitre 4 s'intéresse à la question de reconstruire des modèles (descriptifs, explicatifs et de simulation) synthétisant l'activité à partir de traces. Suivant la métaphore de la génétique, le chapitre 4 porte donc sur la construction de génotypes à partir de phénotypes.

CHAPITRE 3

Ingénierie des connaissances tracées

Les chapitres précédents ont montré que le contexte de la thèse est l'analyse de l'activité de conduite automobile pour produire les connaissances nécessaires à la construction d'un modèle de simulation cognitive de la conduite automobile. Construire un tel modèle nécessite un travail multidisciplinaire en sciences humaines et en informatique. Le volet sciences humaines a été présenté dans les chapitres précédents : l'ergonomie et les sciences de la cognition permettent de fournir un cadre conceptuel pour construire un modèle fidèle à la cognition humaine. Le volet informatique est nécessaire pour construire un modèle computationnel, c'est-à-dire qui soit capable de simuler les activités mentales dans un environnement informatique virtuel. À partir de données de conduite réelles, il est question de construire un modèle qui soit conforme au cadre conceptuel défini par l'ergonomie et les sciences de la cognition.

Le problème informatique est donc le suivant : construire des modèles à partir de données, en tenant compte des connaissances qu'a déjà l'analyste sur les modèles et sur l'activité. Il s'agit donc de combiner une approche ascendante d'analyse de données et une approche descendante guidée par les connaissances d'un analyste.

Ce chapitre introduit les concepts informatiques sur lesquels nous nous appuyons dans le cadre de cette thèse. La première section présente le champ de la découverte de connaissances à partir de données. Ce champ fournit un cadre théorique et méthodologique en informatique sur lequel nous nous appuyons pour illustrer la découverte de connaissances. Nous insisterons sur le rôle de l'analyste dans cette approche méthodologique. La deuxième section introduit la théorie de la trace modélisée, un formalisme conçu pour favoriser l'interaction et la capitalisation de l'expérience à l'aide de traces d'activité. Enfin, la troisième partie présente la question formelle de découvrir le modèle d'un processus à partir de traces et fait le lien avec la théorie de la trace modélisée.

3.1 Découverte de connaissances

3.1.1 Naissance de la discipline

Le développement de l'informatique dans les années 1980 a permis de développer les techniques de calcul et d'envisager le développement de systèmes intelligents. Ces capacités computationnelles ont permis de développer des disciplines telles que la fouille

de données (*Data Mining*), pour trouver des motifs récurrents dans des données, ou des approches d'intelligence artificielle, basées sur les connaissances de l'humain, comme les systèmes experts. Ces capacités computationnelles ont permis le développement des techniques de l'intelligence artificielle (raisonneurs, classifieurs, etc.) se fondant sur des bases de connaissances explicites. Ces connaissances explicites sont acquises à partir de l'expertise (experts humains) ou de l'expérience (données, traces).

La question qui se pose est de découvrir des connaissances à partir de données. Dans tous les cas, les données ont une forme numérique et sont souvent stockées dans des bases de données. Les approches de fouille de données ont des bases théoriques solides, mais lorsqu'elles sont appliquées sur des données réelles, de nombreux problèmes pratiques se posent. Les bases de données réelles sont incomplètes, bruitées, contiennent des informations incohérentes ou non pertinentes, etc.

Ce constat a amené, au début des années 1990, à ouvrir un nouveau champ : la « découverte de connaissances dans les bases de données » (*Knowledge Discovery in Databases*). Frawley *et al.* (1992) en proposent la définition suivante :

> Knowledge discovery *is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data.*
> « La *découverte de connaissances* est l'extraction non triviale, à partir de données, d'informations implicites, précédemment inconnues et potentiellement utiles. »
>
> (Frawley *et al.*, 1992.)

L'objectif de cette discipline est d'étudier, quelle que soit la technique informatique d'intelligence artificielle utilisée, des méthodes et techniques permettant d'extraire des connaissances pertinentes à partir de données réelles. Ainsi, cette discipline englobe les questions d'accès aux bases de données, de préparation des données, d'utilisation de connaissances du domaine, de recherche de connaissances (quelle que soit la technique), d'interprétation et de réutilisation de la connaissance.

Contrastant avec les disciplines de fouille de données ou d'apprentissage automatique, l'analyste, expert du domaine d'analyse des données, est placé au cœur du processus de découverte de connaissances. L'analyste guide les algorithmes pour trouver les connaissances pertinentes, celles qui font sens pour lui et pour le domaine considéré. La présence de l'analyste est d'autant plus nécessaire que des connaissances du domaine ne sont pas nécessairement accessibles par le système.[1]

3.1.2 Le cycle de découverte de connaissances

À partir de ces bases fondatrices, Fayyad *et al.* (1996a) proposent de formaliser le processus de découverte de connaissances. Ils proposent une définition légèrement différente de celle de Frawley *et al.* (1992) qui insiste sur le fait que la découverte de connaissances est un *processus* et que les connaissances découvertes doivent faire sens et être valides :

[1] Notons que les acteurs de la découverte de connaissances reconnaissent (souvent) le rôle de l'humain pour des raisons pratiques (Brachman et Anand, 1996) plus qu'épistémologiques : l'humain pourrait ne plus avoir à intervenir le jour où les algorithmes seraient suffisamment performants sur des données réelles, incomplètes, bruitées, etc. Dans le cadre de cette thèse, nous considérons le rôle de l'humain comme indispensable d'un point de vue pratique *et* épistémologique, puisque les connaissances doivent faire sens *pour* l'analyste.

> Knowledge discovery in databases *is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*
> « La *découverte de connaissances dans des bases de données* est le processus non trivial d'identification, dans les données, de motifs valides, nouveaux potentiellement utiles et ultimement compréhensibles. »
>
> (Fayyad *et al.*, 1996a.)

La découverte de connaissances est un processus, dont Fayyad *et al.* (1996a,b) proposent une formalisation sous forme de cycle dont l'ultime étape est l'interprétation, par l'humain des motifs découverts. Le cycle de découverte de connaissances est composé de 9 étapes. La figure 3.1 représente une vue d'ensemble des étapes du cycle.

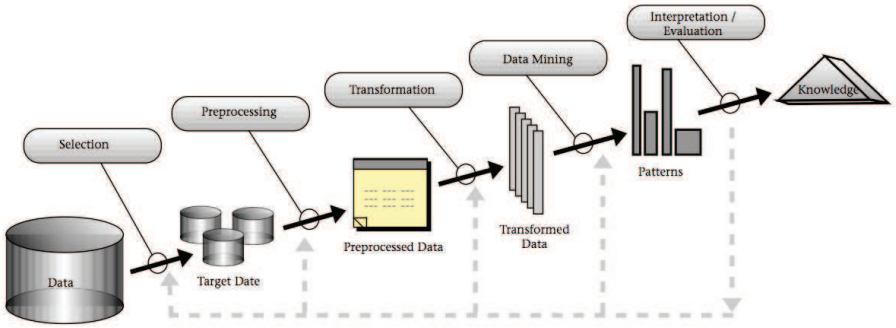


FIGURE 3.1 – Vue d'ensemble des étapes du cycle de découverte de connaissances à partir de données. Repris de Fayyad *et al.* (1996b)

Le processus de découverte de connaissances tel qu'il est présenté par Fayyad *et al.* (1996b) met en valeur le rôle de l'analyste dans le processus. En effet, le cycle de découverte de connaissances est fortement *itératif* et *interactif*. Itératif, car l'analyste peut revenir en arrière à n'importe quelle étape. Interactif, car c'est l'analyste qui pilote chaque étape.

Les étapes du cycle de découverte de connaissances sont les suivantes (Fayyad *et al.*, 1996a,b) :

1. Développer la compréhension du domaine d'application, les connaissances du domaine et déterminer les buts de la découverte de connaissances.

2. Créer un jeu de données cibles : sélectionner un jeu de données et les paramètres sur lesquels on va focaliser le processus de découverte.

3. Nettoyer et prétraiter les données : gérer le bruit, définir une stratégie pour gérer les défauts dans les données.

4. Réduire les données : réduire le nombre de dimensions présentes dans les données ou trouver des méthodes de transformation pour réduire le nombre de variables à considérer.

5. Choisir la méthode de fouille de données : en fonction de l'objectif définir la méthode de fouille à employer, par exemple une classification, une régression, etc.

6. Choisir un algorithme de fouille et son paramétrage : sélectionner une méthode et les paramètres de fouille, toujours en regard de l'objectif de la fouille.

7. Fouiller les données : appliquer l'algorithme de fouille et découverte de motifs.

8. Interpréter les motifs découverts : l'analyste visualise et interprète les motifs, parfois en regard des données.

9. Utiliser les connaissances découvertes : exploiter les connaissances découvertes pour la machine, en les intégrant dans un système capable d'utiliser ces connaissances pour agir, ou simplement pour l'humain, en documentant ces connaissances, et résoudre les éventuelles incohérences entre les connaissances découvertes et les connaissances a priori.

Notons que dans notre cas, la dernière étape, l'étape d'exploitation des connaissances découvertes, possède à la fois un but documentaire et un but applicatif. Documentaire, puisqu'il s'agit de comprendre l'activité de conduite automobile et donc enrichir les connaissances en ergonomie, en psychologie cognitive et dans le domaine du transport en général. Applicatif, puisqu'il s'agit d'intégrer ces connaissances dans un modèle de simulation cognitive de la conduite automobile : autrement dit, un modèle informatique capable d'exploiter ces connaissances pour simuler la conduite automobile.

Même si ce cycle est instancié aux techniques particulières de la fouille de données, qui représentent une étape du cycle, l'approche reste valide dans le contexte général de la découverte de connaissances, quelle que soit la technique d'intelligence artificielle exploitée. Les auteurs insistent bien sur l'importance de l'ensemble de la chaîne : la fouille de données seule, sans une préparation correcte des données, pouvant donner lieu à la découverte de motifs invalides ou n'ayant aucun sens.

Le cycle est présenté de manière linéaire, mais il s'agit en fait d'une tentative de linéarisation d'un processus fortement bouclé où l'analyste est amené à revenir à une étape antérieure à n'importe quel moment du processus. Cette formalisation du processus de découverte de connaissances suggère le rôle de l'analyste, mais sa nature linéaire cache la complexité réelle du processus et l'importance du rôle de l'analyste.

3.1.3 Le rôle de l'analyste

Lorsque l'on essaie d'appliquer les principes de la découverte de connaissances, on se rend compte que le rôle de l'analyste est déterminant. C'est le constat pragmatique qu'ont fait Brachman et Anand (1996) en analysant le travail d'analystes « cherchant » de la connaissance. Dans la réalité, le processus de découverte de connaissances est beaucoup plus coûteux et exploratoire que la structure linéaire de la figure 3.1 ne le laisse entendre. Brachman et Anand (1996) proposent d'étudier le cycle de découverte en se centrant sur l'analyste plutôt que sur les outils. En étudiant des cas réels d'analyse, ils redéfinissent la découverte de connaissances de la manière suivante :

> *Knowledge discovery is a* knowledge-intensive *task consisting of* complex interactions*,* protracted over time*, between a* human *and a (large) database, possibly supported by a* heterogeneous suite of tools*.*

> « La découverte de connaissances est une tâche *consommatrice de connaissances* constituée d'*interactions complexes*, *prolongées dans le temps*, entre un *humain* et une (grande) base de données, éventuellement assistée par une *suite d'outils hétérogènes*. »
>
> (Brachman et Anand, 1996.)

Cette définition montre le rôle essentiel de l'analyste et de ses connaissances dans le processus de découverte. Elle met en évidence l'importance des interactions entre l'humain et les outils informatiques mobilisés dans la chaîne de découverte. Ces outils, souvent développés séparément, ne sont pas homogènes. De nombreux efforts ont été faits pour intégrer des outils de préparation des données, des algorithmes de fouille et des outils de visualisation au sein d'une même suite logicielle (KNIME par exemple)[2]. Quoi qu'il en soit, même avec ces suites logicielles, le problème n'est pas complètement résolu. Il sera toujours nécessaire d'importer les données initiales dans un tel système. Et il existera toujours des situations où l'analyste utilisera des outils d'analyse ad hoc, spécifiques au domaine d'application et où il sera amené à manipuler des outils hétérogènes.

3.1.4 Interaction

L'étude de l'interaction entre l'homme et la machine (IHM) n'a été considérée en tant que telle que depuis la fin des années 1960 (Hollnagel, 2011). Puis, avec le développement de l'informatique, la question de l'interaction plus spécifiquement avec un ordinateur s'est posée (HCI pour *Human-Computer Interaction*).

Comme l'illustre la revue historique de Grudin (1990), la recherche en CHI s'intéresse principalement à la conception d'interfaces entre l'utilisateur et l'ordinateur. Bien que le mot « interaction » soit utilisé, c'est souvent l'interfaçage entre l'humain et la machine qui est étudié. En particulier, Hollnagel (2011) critique le point de vue qui consiste à voir l'humain et la machine comme deux entités « réactives » (*reactive*) : « ils sont supposés répondre aux signaux et informations reçues (et perçues), mais non agir indépendamment [l'un de l'autre]. » [3]

Le mot « interaction », nous dit le Petit Robert (2010), signifie « action réciproque ». Cette définition reste vraie lorsque l'on parle de systèmes informatiques. Il y a bien inter-action, car l'humain change le système informatique (données) et le système informatique change l'humain (informations, connaissances). L'humain apprend en interagissant avec la machine. Et il est souhaitable que la machine « apprenne » de l'interaction avec l'humain. En ce sens, nous partageons l'idée d'Hollnagel (2011) selon laquelle l'humain et la machine ne doivent pas être vus comme réactifs (répondre à un stimulus et produire une réponse), mais aussi comme proactifs (l'humain s'adapte à la machine et de son environnement et la machine pourrait apprendre de l'humain et s'adapter à lui et à son environnement). C'est dans cette logique que nous nous situons lorsque nous parlons d'interaction.

C'est cette interaction qui nous intéresse. Nous souhaitons que l'analyste, utilisateur des outils informatique, ajoute des connaissances dans le système informatique et

[2] KNIME : http://www.knime.org/ (Berthold *et al.*, 2008). Une liste conséquente de ces suites logicielles est accessible sur le site web KDnuggets : http://www.kdnuggets.com/software/suites.html.

[3] « they are assumed to respond to the signals and information they receive (and perceive), but not to act independently. » (Hollnagel, 2011)

que ce dernier en tire parti. Nous souhaitons que l'ordinateur effectue des calculs qui révèlent des connaissances nouvelles à l'analyste, ou qui au moins l'assistent dans sa compréhension des traces d'activité en particulier et de l'activité en général.

Pour répondre à cet objectif, nous nous appuyons sur des outils d'ingénierie des connaissances qui visent à exploiter cette interaction entre l'humain et la machine : les traces modélisées et les systèmes à base de traces.

3.2 Traces modélisées et systèmes à base de traces

3.2.1 Trace modélisée

Dans cette thèse, les données sur lesquelles nous travaillons sont des traces d'activité. Pour manipuler ces données particulières que sont les traces d'activité, nous proposons d'exploiter une ingénierie des connaissances particulière : l'ingénierie des connaissances tracées. Cette ingénierie des connaissances tracées fait partie des domaines d'intérêt de l'équipe SILEX du laboratoire LIRIS. Elle a cela de particulier que les connaissances manipulées réfèrent à des événements situés dans le temps (et potentiellement l'espace).

Pour manipuler les données que sont les traces d'activité, l'équipe SILEX a développé le concept de *trace modélisée* ($\mathcal{M}$-Trace) (Laflaquière *et al.*, 2008; Settouti, 2011).Une $\mathcal{M}$-Trace est une formalisation informatique du concept de trace d'activité décrit au chapitre 1. Ce concept informatique a donc été développé dans un esprit proche de celui de l'ergonomie. La $\mathcal{M}$-Trace permet en effet de capturer les différents événements observés lors d'une activité. Elle associe à chaque événement, appelé *obsel*, une sémantique. Cette sémantique, c'est-à-dire le vocabulaire de la trace, est enregistrée dans un modèle de trace (d'où le nom de *trace modélisée*).

Une $\mathcal{M}$-Trace contient donc plus d'informations que la simple valeur numérique d'une mesure. Le modèle contient des éléments d'interprétation des informations contenues dans les *obsel*s. Le modèle explicite aussi la manière de situer les *obsel*s dans le temps (représentation temporelle ou séquentielle, temps absolu ou relatif, unités de temps utilisées, etc.) Le modèle de la trace permet à l'humain d'expliciter l'interprétation qu'il fait d'une trace. Le modèle est utilisé par l'ordinateur pour exploiter la sémantique dans les calculs qu'il effectue. Les mêmes *obsel*s associés à des modèles différents constituent des $\mathcal{M}$-Traces différentes. D'un point de vue épistémologique, cela revient à interpréter différemment le même jeu de données. Une $\mathcal{M}$-Trace contient ainsi, non seulement des informations relatives à ce qui a été observé, mais aussi sur la manière d'interpréter ces éléments observés.

Exemple

Activité : Conduite automobile sur un véhicule instrumenté.

Données mesurées : Mesures d'enfoncement des pédales, position GPS du véhicule, vitesse du véhicule…

***Obsel*s :**

…

obsel 25 : date : 151,2 ; type : APPUI_FREIN,

obsel 26 : date : 151,3 ; type : DIMINUTION_VITESSE,

...

Modèle de trace :

- Les *obsel*s sont situés dans le temps avec le nombre de secondes écoulé depuis le début de l'expérimentation.
- Chaque *obsel* possède une « date » qui désigne l'instant où l'*obsel* a été observé. La valeur de cet attribut correspond au nombre de secondes écoulées depuis le début du trajet.
- Chaque *obsel* possède un type.
- Les types d'*obsel*s sont classés en trois catégories : ACTIONS_CONDUCTEUR, EVOLUTION_ETAT_VEHICULE, et EVOLUTION_ENVIRONNEMENT. La catégorie ACTIONS_CONDUCTEUR contient les *obsel*s relevant des actions du conducteur, etc.
- Un *obsel* de type APPUI_FREIN appartient à la catégorie ACTIONS_CONDUCTEUR et signifie « le conducteur appuie sur la pédale de frein » ;
- Un *obsel* de type DIMINUTION_VITESSE appartient à la catégorie EVOLUTION_ETAT_VEHICULE et signifie « la vitesse du véhicule diminue »...

Que peut-on faire avec des $\mathcal{M}$-Traces ? On peut les visualiser et les transformer. Transformer une $\mathcal{M}$-Trace, cela revient à construire une nouvelle $\mathcal{M}$-Trace dont on a modifié soit les *obsel*s, soit le modèle, soit les deux. Typiquement, une transformation va permettre de reformuler une trace, soit en filtrant des éléments qui ne sont pas pertinents, soit en agrégeant des *obsel*s qui constituent un motif particulier, etc. Le concept de $\mathcal{M}$-Trace permet de construire de manière rigoureuse et intelligible ces interprétations.

Le concept de $\mathcal{M}$-Trace s'appuie sur des principes constructivistes. La souplesse d'interprétation (via un modèle) et de reformulation (via des transformations) permet la construction de connaissances. La connaissance est à la fois connaissance pour l'humain (modèle qui permet d'expliciter la sémantique de la trace) et connaissance pour la machine (le modèle est exploité pour faire des calculs sur les données). Un expert peut donner du sens a priori aux données. Et à l'inverse, des données peuvent faire sens et être interprétées par un expert.

3.2.2 Système à Base de Traces et Système de Gestion de Bases de Traces

Une $\mathcal{M}$-Trace est un support de connaissances. Elle contient à la fois des données et un modèle pour interpréter ces données. On appelle Système à Base de Traces (SBT), un système à base de connaissances exploitant les $\mathcal{M}$-Traces (Laflaquière, 2009; Settouti, 2011; Settouti *et al.*, 2009). Un Système de Gestion de Bases de Traces (SGBT) est un logiciel permettant de gérer informatiquement les traces, leur collecte et leurs transformations. La distinction entre SBT et SGBT est semblable à celle que l'on peut faire

entre un logiciel exploitant des données et le système de gestion de bases de données (SGBD) utilisé pour gérer ces données et offrant des traitements canoniques. Le SGBT assure le stockage et effectue les modifications et transformations sur les $\mathcal{M}$-Traces, alors que le SBT constitue le niveau applicatif et à destination de l'utilisateur.

Un SGBT permet de gérer et transformer des traces modélisées. Le processus qui vise à créer une $\mathcal{M}$-Trace première dans le SGBT s'appelle la collecte de la trace. La collecte consiste en la création d'un modèle de trace et l'import des *obsel*s dans cette $\mathcal{M}$-Trace. Le SBT permet de visualiser les traces et d'offrir des outils permettant à l'utilisateur de faciliter son interaction avec les traces. Dans notre cas, le SBT a pour vocation de permettre la co-construction de connaissances entre l'analyste et la machine.

Dans le cadre de cette thèse, nous nous appuyons sur le logiciel ABSTRACT[4] (Georgeon, 2008; Georgeon *et al.*, 2011; Mathern, 2006). Le logiciel ABSTRACT est à la fois un SGBT et un SBT. Dans la suite, nous nous focalisons sur la partie applicative : le SBT ABSTRACT. ABSTRACT a été développé dans le contexte particulier de l'analyse de traces d'activité de conduite automobile. L'objectif était déjà d'exploiter le concept de traces d'activité pour produire des modèles descriptifs et explicatifs de l'activité de conduite automobile (Georgeon, 2008), par exemple, des modèles descriptifs portant sur le changement de voie sur autoroute (Henning *et al.*, 2008), illustré à la figure 3.2.

Lors de l'analyse de l'activité de conduite automobile, les *obsel*s que l'on collecte ne sont pas les états ou les sorties directes de ce que l'on modélise. En effet, on n'observe pas les états cognitifs du conducteur, on n'observe pas la décision du conducteur de dépasser et on n'observe pas directement le fait que le conducteur est en phase de préparation de dépassement. Ce que l'on observe, ce sont des données issues de capteurs, les actions du conducteur sur les actionneurs du véhicule. Il faut donc interpréter des données observées, les reformuler au fur et à mesure, jusqu'à ce que l'on obtienne une reformulation qui corresponde au modèle de l'activité de conduite. Notons que l'on ne connaît pas le modèle a priori et donc, on ne connaît pas a priori la « bonne » reformulation. Mais l'analyste se fixe un objectif d'analyse dans lequel il connaît au moins une partie du vocabulaire et la portée du modèle recherché. Ces objectifs permettent de guider progressivement la reformulation. L'itération du processus interactif permet de compléter au fur et à mesure les reformulations.

Le SBT ABSTRACT a permis de montrer qu'il est possible, à partir d'*obsel*s issus du niveau sensori-moteur (issus de capteurs sur des pédales d'un véhicule), de remonter à des niveaux plus cognitifs de l'activité (phase de dépassement). L'utilisation des $\mathcal{M}$-Traces et leurs transformations successives permettent de reformuler les traces et d'inférer des niveaux d'activité qui ne sont pas directement observables.

Le SBT ABSTRACT exploite des $\mathcal{M}$-Traces dont le modèle est représenté dans une ontologie. L'ontologie associée à une trace est un modèle d'interprétation des observations contenues dans les *obsel*s de la $\mathcal{M}$-Trace. Le SBT ABSTRACT propose de gérer les $\mathcal{M}$-Traces, de les visualiser, d'en éditer le modèle et d'effectuer des transformations sur les $\mathcal{M}$-Traces. L'histoire des transformations des $\mathcal{M}$-Traces est enregistrée, ce qui permet de savoir précisément comment une trace donnée a été construite. Nous insistons sur ce point, car en conservant l'histoire des transformations d'une $\mathcal{M}$-Trace, il est possible de savoir comment elle a été construite. C'est essentiel pour pouvoir argumenter les résultats obtenus à partir de $\mathcal{M}$-Traces. Ainsi, en combinant cette histoire

[4] Analysis of Behaviour and Situation for menTal Representation Assessment and Cognitive acTivity modeling

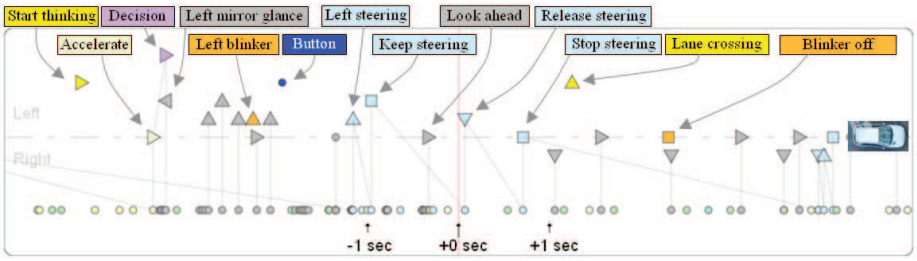


FIGURE 3.2 – Visualisation, dans le logiciel ABSTRACT, d'une $\mathcal{M}$-Trace de changement de voie sur autoroute (Georgeon *et al.*, 2011). L'axe horizontal représente le temps. L'axe vertical représente différents niveaux de lecture de l'activité. La capture d'écran est annotée pour expliciter le contenu.

des transformations avec le modèle de la trace, la construction de la trace est totalement argumentée et les choix de l'analyste sont enregistrés.

3.2.3 Découverte de connaissances à partir de $\mathcal{M}$-Traces

La construction de connaissances à partir de traces est conforme au principe de découverte de connaissances décrit par Fayyad *et al.* (1996b). La découverte de connaissances s'appuie sur une boucle itérative partant de la collecte des données, passant par des prétraitements, puis par l'analyse automatique avec des algorithmes de fouille de données et finissant par l'interprétation de la pertinence des motifs découverts par un expert humain. C'est habituellement à cette dernière étape, celle de l'interprétation, que la connaissance est construite.

Le fait de travailler sur des $\mathcal{M}$-Traces modifie la donne. En effet, cela nécessite en amont de donner un modèle, et donc une interprétation aux données. Mais dès lors que les données disposent d'un modèle, alors il ne s'agit plus de simples données, mais d'inscriptions de connaissances. L'interprétation par un humain est possible, car décrite dans le modèle. L'interprétation commence explicitement plus tôt dans le processus de découverte de connaissances.

Cela permet de pousser plus loin les possibilités de reformulation des données, car l'expert comprenant les données peut utiliser ses propres connaissances pour les ajouter dans le modèle ou pour transformer les traces[5]. Il existe donc déjà, sans même aller jusqu'à l'étape de fouille de données, une découverte ou plus exactement une construction de connaissances. Cette construction de connaissances est itérative et se base (explicitement) sur l'interaction entre un expert et la chaîne d'outils informatiques supportant l'analyse.

De la fouille de données à la fouille de traces

Que devient l'étape de fouille de données lorsque l'on travaille avec des $\mathcal{M}$-Traces ? Les $\mathcal{M}$-Traces, par nature, mettent l'accent sur l'aspect temporel de la trace. Bien entendu, un certain nombre d'algorithmes de fouille ne s'attaquent pas spécifiquement au problème de données temporelles ou séquentielles. Les algorithmes de fouille qui

[5] L'expert le faisait déjà avant, mais les $\mathcal{M}$-Traces permettent de capturer cette expertise de manière explicite.

exploitent l'aspect temporel ou séquentiel peuvent être exploités directement sur les $\mathcal{M}$-Traces. Cependant, les objectifs des algorithmes de fouille (trouver les motifs les plus fréquents, gérer au mieux le bruit des données, etc.) sont parfois différents des objectifs d'analyse d'un expert (comprendre un cas dans sa singularité, différencier un cas unique de cas fréquents, etc.). Là où certains cherchent des motifs fréquents avec un temps de calcul raisonnable, nous souhaitons pouvoir trouver à la fois des motifs fréquents et des motifs rares en un temps de calcul raisonnable, du point de vue de l'expert. Là encore, l'interaction entre l'expert et le module de calcul est primordiale : il est souhaitable de trouver des motifs rapidement, mais sans nécessairement calculer l'ensemble des solutions. L'expert peut interpréter les motifs découverts au fil de l'eau et juger (en temps réel) de la pertinence des paramètres de fouille (Cram *et al.*, 2011).

Les outils de fouille vont permettre à l'expert de découvrir des motifs dans les traces et d'exploiter ces motifs pour transformer les $\mathcal{M}$-Traces et enrichir le modèle associé à la trace. La fouille de données est vue ici comme un processus produisant des résultats dont l'analyste va se servir pour enrichir, reformuler, compléter ou simplifier la trace. Dans le cas du SBT ABSTRACT, la fouille de données est constituée de requêtes lancées par l'analyste permettant de découvrir des *obsel*s ou groupes d'*obsel*s possédant certaines propriétés.

Pour dépasser le simple niveau descriptif et explicatif des $\mathcal{M}$-Traces et comprendre l'activité non pas dans sa réalisation, mais dans sa potentialité, il est nécessaire de construire des connaissances sur les traces qui soient d'une nature différente des traces. Pour reprendre l'analogie présentée au chapitre 2, la trace représente le phénotype d'une activité (l'activité telle qu'elle s'est exprimée dans un contexte particulier), alors que ce que l'on recherche est un modèle (de simulation), de nature génotypique.

3.3 Dualité entre traces et modèles : observer et interpréter

Une trace traduit l'expression d'une activité dans un environnement. De la même manière qu'il existe une relation entre un génotype et un phénotype, il est possible d'étudier les relations entre les traces d'activité et l'activité. Pour construire des modèles de simulation tels que les schémas de conduite présentés au chapitre 2, il faut exploiter cette relation entre traces et modèles.

3.3.1 Une relation d'équivalence entre traces et modèles ?

Si l'on connaît le modèle et que l'on produit des traces à partir d'un modèle, il est possible d'étudier les relations entre traces et modèles d'un point de vue formel. Cette relation d'équivalence a par exemple été étudiée dans le cas des réseaux de Petri (Diekert et Rozenberg, 1995). Une trace correspond à une instanciation du modèle. Il y a une relation d'équivalence entre un modèle et l'ensemble de ses instanciations possibles. Cependant, dans de nombreux cas, il n'est pas possible de disposer de toutes les instanciations possibles d'un modèle, en particulier dès que l'on s'intéresse à des problèmes ayant lieu dans un environnement ouvert. C'est typiquement le cas de la conduite automobile : il est impossible de collecter toutes les traces de conduite possibles de tous les conducteurs dans toutes les situations.

3.3.2 Le problème de la construction d'un modèle à partir de traces

D'un point de vue pratique, dans le problème que nous étudions, nous ne connaissons pas le modèle. Nous supposons qu'il est possible de modéliser l'activité de conduite automobile avec un certain formalisme, par exemple les réseaux de Petri ou les chaînes de Markov, et nous cherchons à découvrir ce modèle à partir des traces collectées. Autrement dit, nous supposons l'existence d'un génotype qui s'exprimerait dans un certain formalisme, et on cherche à reconstruire ce génotype à partir des phénotypes observés et des connaissances de l'analyste.

Par rapport au cycle de découverte de connaissances, la construction d'un tel modèle ne relève pas de la fouille de données. Il s'agit plutôt de l'étape d'interprétation et de synthèse des connaissances, car le modèle génotypique constitue en soi une interprétation des données.

La question de la reconstruction d'un modèle à partir de l'interprétation de traces d'activité est un problème difficile, notamment parce que les traces sont issues d'une observation et génèrent des interprétations. Il est difficile de distinguer dans les traces ce qui relève d'artefacts liés à l'observation, de l'activité réellement mise en œuvre. Et quand bien même on observe bien l'activité produite par le modèle, rien ne nous garantit (lorsque l'on ne connaît pas le modèle) que la trace observée soit du même niveau que celle du modèle recherché. Par exemple, une trace de conduite correspond à des mesures d'enfoncement de pédales, alors que l'activité étudiée est cognitive. L'observation de l'activité est dans ce cas très éloignée de l'activité étudiée et le vocabulaire de la trace est différent du vocabulaire du modèle.

3.3.3 Une méta-théorie des traces

Pour étudier ces questions, Deransart (2010) propose une méta-théorie des traces qui formalise la question de la relation entre trace et modèle. Pour cela, il introduit la notion de trace virtuelle. L'activité étudiée produit des traces réelles, celles dont nous parlons dans cette thèse. L'ensemble des traces produites par le modèle recherché (dans le vocabulaire du modèle) n'est pas connu, il s'agit de la *trace virtuelle*. La trace virtuelle est un objet conceptuel qui n'a pas d'existence réelle. Elle est formellement équivalente au modèle recherché. L'objectif est donc, à partir de traces réelles, appelées *traces effectives*, de reconstruire la trace virtuelle. Ce travail de reconstruction, dans notre cas, s'appuie sur la transformation de $\mathcal{M}$-Traces. C'est l'expertise de l'analyste qui permet de reformuler les traces depuis le vocabulaire des traces collectées vers le vocabulaire du modèle recherché.

La méta-théorie des traces de Deransart (2010) explicite formellement les questions de la collecte de la trace et d'interprétation de la trace décrites au chapitre 1 (voir figure 1.2, p. 17). La collecte de la trace correspond à la production de la trace effective : elle est produite par une *sémantique observationnelle*. L'interprétation que l'on fait de la trace effective vise à associer à des éléments de la trace effective à ceux de la trace virtuelle : elle est produite par une *sémantique interprétative*. La sémantique interprétative consiste à faire le lien entre des événements observés et les processus réellement mis en œuvre (par ex., à partir de la mesure de l'enfoncement de la pédale de frein, on déduit que le conducteur a appuyé sur le frein). La sémantique interprétative peut consister à inférer des éléments de l'activité non directement observés à partir des traces, par exemple, le fait que l'action du conducteur sur la pédale de frein est liée à une décision vis-à-vis de la couleur du feu.

Lien entre traces et modèles

Appliqué au contexte d'étude de cette thèse, il n'y a pas *une* trace virtuelle, mais *des* traces virtuelles. Le processus réel est l'activité du conducteur. Cette activité que l'on cherche à modéliser est celle que l'on observe à l'aide des sémantiques observationnelles. Les interprétations que l'on fait des traces obtenues constituent des hypothèses sur l'activité de conduite. Chaque interprétation correspond à un modèle (potentiellement différent d'une interprétation à une autre). Notons que dans la théorie de la trace modélisée, la sémantique observationnelle correspond au modèle de la trace. Dans le SBT ABSTRACT, la sémantique observationnelle de la trace est son ontologie. La sémantique interprétative correspond aux requêtes de transformation de traces.

La figure 3.3 illustre les concepts de sémantique observationnelle et de sémantique interprétative. Dans le cas de l'activité de conduite, à partir d'une même trace de conduite, l'analyste peut s'intéresser à des modèles de différentes natures, par exemple un modèle comportemental vs un modèle cognitif.

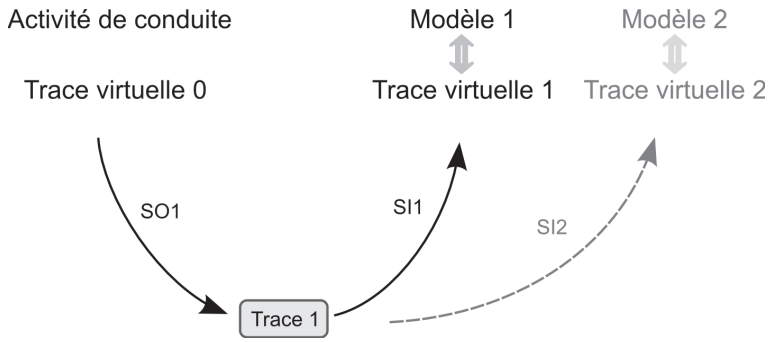


FIGURE 3.3 – L'observation de l'activité de conduite à travers une *sémantique observationnelle* (SO1) produit une trace (Trace 1). L'interprétation de cette trace avec une *sémantique interprétative* (SI1) permet de faire le lien avec une trace virtuelle correspondant à un modèle (Modèle 1) recherché. Une autre sémantique interprétative (SI2) pourrait permettre d'étudier un autre modèle (Modèle 2).

Transformations de traces

La méta-théorie des traces prend en compte le principe de transformation de traces (*dérivation* ou abstraction de traces). Comme nous en avons déjà discuté dans le cadre des SBTs, les traces peuvent être transformées, typiquement pour construire des abstractions à partir de ce qui a été observé (par ex., inférer les composantes non observées ou non observables de l'activité). Appliquer une transformation à une trace, c'est créer une nouvelle trace, dont la sémantique observationnelle et la (ou les) sémantique(s) interprétatives auront changé. La transformation de traces permet de rendre explicite la sémantique interprétative de la trace source dans la trace transformée. Au fur et à mesure des transformations, l'objectif est d'obtenir une sémantique interprétative de plus en plus explicite et une trace transformée de plus en plus proche de la trace virtuelle recherchée. La figure 3.4 illustre ces processus de transformation.

La méta-théorie des traces formalise ainsi le processus d'inférence que nous proposons de réaliser avec les traces modélisées.

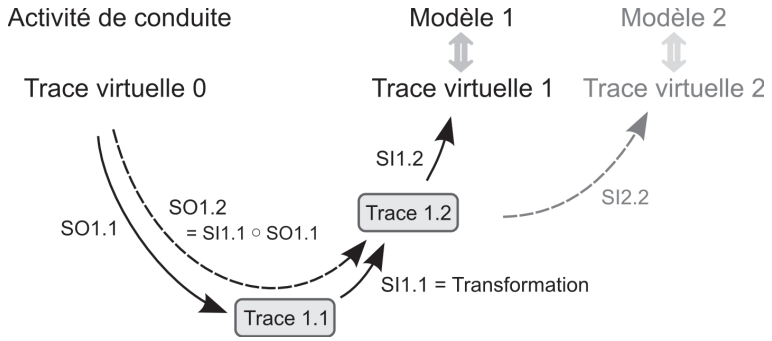


FIGURE 3.4 – L'observation de l'activité de conduite à travers une *sémantique observationnelle* (SO1.1) produit une trace (Trace 1.1). La transformation de cette trace, par exemple via un SBT, permet de produire une nouvelle trace (Trace 1.2) dont la sémantique interprétative (SI1.2) permet d'étudier un modèle (Modèle 1). Bien que la trace transformée (Trace 1.2) n'ait pas été directement observée, il existe une sémantique observationnelle équivalente (SO1.2) qui aurait permis son observation.

3.4 Conclusion

Dans ce chapitre, nous avons présenté le cadre informatique d'ingénierie des connaissances dans lequel ce travail de thèse s'inscrit. Tout d'abord, les connaissances (modèles) que nous souhaitons découvrir s'appuient sur une méthodologie de découverte de connaissances proposée par Fayyad *et al.* (1996b). Nous avons souligné le rôle de l'analyste et la nécessaire interaction de l'analyste avec un cycle de découverte de connaissances très itératif.

Puisque nous travaillons sur les données particulières que sont les traces d'activité, nous nous appuyons sur le formalisme des traces modélisées développées par l'équipe SILEX. Les $\mathcal{M}$-Traces permettent d'expliciter le travail d'interprétation des traces (via le modèle de la trace) et de l'abstraction des traces avec des transformations.

Les SBTs sont des systèmes permettant la gestion, la visualisation et la transformation de $\mathcal{M}$-Traces. Dans cette thèse, nous utiliserons le SBT ABSTRACT existant et développé dans le même contexte d'analyse de traces d'activité de conduite automobile (Georgeon, 2008) que celui de cette thèse.

La méta-théorie des traces permet de formaliser le processus d'abstraction et d'inférence que nous souhaitons faire avec les $\mathcal{M}$-Traces de l'activité de conduite. Le processus de découverte de connaissances et de construction d'un modèle génotypique de l'activité à partir de traces phénotypiques s'inscrit donc sur un cadre théorique formel sur lequel ce travail de thèse peut s'appuyer.

L'objectif du prochain chapitre (chapitre 4) est de faire une revue des formalismes capables de représenter la part génotypique de l'activité et des algorithmes permettant d'inférer ces modèles à partir de traces d'activité.

CHAPITRE 4

Reconstruire un modèle à partir de traces d'exécutions : le challenge du Workflow Mining

L'activité de conduite automobile est une activité complexe et dynamique. Pour modéliser cette activité, il convient de choisir un type de modèle adapté, capable de rendre compte de cette complexité. Il est nécessaire de disposer d'outils qui permettent, à partir des traces de conduite, de construire ces modèles.

Dans la première section de ce chapitre, nous présentons différents formalismes candidats pour représenter les *schémas de conduite* automobile. Parmi ces formalismes, nous justifierons le choix de travailler avec des modèles basés sur les réseaux de Petri. La deuxième section présente la question de la construction d'un tel modèle à partir des traces d'activité. Nous présentons en particulier le champ de recherche de la fouille de workflow qui propose et applique de telles méthodes avec des objectifs de modélisation similaires. La troisième section contient une revue de la littérature des algorithmes de fouille de workflow. Enfin, la quatrième section montre qu'il est nécessaire de modifier ces algorithmes pour atteindre notre objectif.

4.1 Quels modèles utiliser pour représenter les schémas de conduite ?

4.1.1 Critères de choix

Dans le cadre de cette thèse, l'objectif applicatif est de construire des modèles de l'activité de conduite automobile. Plus précisément, nous cherchons à représenter des modèles synthétiques de l'activité : les schémas de conduite.

Nous allons maintenant rappeler les caractéristiques que doit respecter le formalisme utilisé pour représenter un schéma de conduite et atteindre les objectifs applicatifs que nous avons définis au chapitre 2. Le formalisme choisi doit : 1) pouvoir représenter des schémas de conduite et 2) permettre à l'analyste d'exploiter les connaissances contenues dans ces schémas.

Représenter des schémas de conduite

Les schémas de conduite sont des modèles synthétiques de l'activité. Contrairement aux traces qui représentent l'activité telle qu'elle *s'est* instanciée, un schéma de conduite représente l'activité telle qu'elle *peut* s'instancier : l'activité dans sa potentialité.

Nous l'avons vu, l'activité de conduite est une activité dynamique, complexe et fortement contrainte. La conduite automobile implique différents niveaux hiérarchiques et les activités les plus automatisées peuvent se dérouler en parallèle (tourner le volant et appuyer sur l'accélérateur) et être impliquées dans des boucles de régulation.

Le modèle choisi doit être capable de représenter :

- La dynamique, donc le temps ;
- Des niveaux hiérarchiques ;
- Des processus contraints ;
- Des processus parallèles ;
- Des processus bouclés.

Notons aussi que lorsque l'on cherche à modéliser un schéma de conduite particulier, nous ne disposons que d'observations singulières. Il est donc nécessaire de construire des modèles qui généralisent ces observations singulières.

Exploiter les schémas de conduite

L'enjeu de la construction des schémas de conduite est double : premièrement, il s'agit pour l'analyste de mieux comprendre l'activité de conduite automobile et deuxièmement, il s'agit d'utiliser les connaissances produites dans le modèle de simulation cognitive COSMODRIVE, pour étudier la cognition du conducteur.

Les schémas de conduite doivent donc être interprétables par l'analyste, c'est-à-dire être *explicites*, et pouvoir être utilisés pour simuler l'activité. Enfin, pour être prédictive et explicative de l'activité, la simulation produite doit être déterministe.

4.1.2 Familles de modèles

Nous proposons une sélection de modèles candidats pour représenter l'activité de conduite automobile. Ces modèles sont tous capables de simulation. Nous les présentons en trois catégories : les modèles de type « boîte noire », comme les réseaux de neurones, les modèles stochastiques, tels les réseaux bayésiens ou les chaînes de Markov et enfin les modèles de type automate.

Modèles « boîtes noires »

Certains modèles permettent de modéliser des processus complexes sans représenter explicitement le processus modélisé. Ces modèles, que nous appelons modèles « boîtes noires », possèdent des entrées et produisent des sorties, mais le contenu du modèle n'est pas interprétable pour l'analyste. Il s'agit par exemple des réseaux de neurones formels (McCulloch et Pitts, 1943; Minsky et Papert, 1969).

Modèles stochastiques

Les modèles stochastiques sont des modèles construits à partir d'une analyste probabiliste de données statistiques.

Parmi les modèles stochastiques, les réseaux bayésiens (Charniak, 1991) sont classiquement utilisés en intelligence artificielle. Cependant, ils ne capturent pas l'aspect temporel, à l'inverse des modèles tels que les chaînes de Markov ou les réseaux bayésiens dynamiques (Russel et Peter, 2006).

Les réseaux bayésiens comme les chaînes de Markov peuvent être visualisés sous forme de graphes, ce qui facilite leur interprétation. Les connaissances capturées par le modèle (probabilités conditionnelles pour les réseaux bayésiens, probabilités de transition d'un état à un autre pour les chaînes de Markov) sont explicites et accessibles à l'analyste (Dapzol, 2006).

Les modèles stochastiques ne permettent pas de produire une simulation déterministe.

Modèles automates

Les modèles automates ont le double avantage de posséder une représentation graphique accessible à l'analyste et de disposer d'une traduction formelle permettant au modèle de piloter ou simuler le comportement modélisé. Nous nous intéressons dans cette section à trois types de modèles automates classiques : les automates à états finis, les statecharts (Harel, 1987) et les réseaux de Petri (Brams, 1983a,b; Peterson, 1977).

Les connaissances contenues dans un modèle automate sont explicites ce qui permet de se servir de ces représentations pour la conception ad hoc de modèles. Par exemple, les modèles de type statecharts (Harel, 1987) ont été développés dans le but de concevoir des systèmes complexes et sont utilisés dans le langage de modélisation UML.

Les modèles automates peuvent être utilisés pour produire une simulation déterministe.

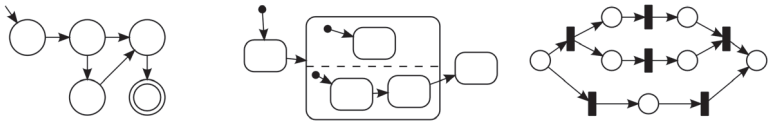
Choix d'une famille de modèles

Alors que toutes les familles de modèles permettent de piloter une simulation, les modèles de type « boîte noire » ne sont pas explicites et ne permettent pas à l'analyste de *comprendre* l'activité étudiée. Les modèles stochastiques sont capables de simulation, mais cette simulation n'est pas déterministe. Les modèles automates proposent à la fois une représentation explicite et une simulation déterministe des processus modélisés. Les modèles automates répondent donc à nos premiers critères de choix. Dans la suite, nous étudions plus précisément les différences entre modèles de type automate.

4.1.3 Choix d'un modèle automate

Automates à états finis

Les automates à états finis disposent des propriétés de base des automates. Ils offrent une représentation graphique sous forme de graphe orienté. Les noeuds représentent des états et les flèches représentent des transitions (événements) entre les états (voir figure 4.1). Les automates disposent d'une sémantique opérationnelle, permettant de traduire cette représentation graphique destinée à l'utilisateur humain, en un modèle formel.



(a) automate à états finis (b) statechart (c) réseau de Petri

FIGURE 4.1 – Exemples d'automates, illustrant les différentes représentations graphiques utilisées par les automates, les Statecharts et les réseaux de Petri.

Les automates à états finis permettent de rendre compte de la structure de l'activité. La représentation graphique permet de rendre compte de la dynamique des états et des événements. Le formalisme peut modéliser des boucles représentant un processus récurrent. Et la concurrence entre différents processus peut être représentée graphiquement. Un seul état est actif à la fois, ce qui induit certaines limitations, par exemple on ne peut pas modéliser le parallélisme.

Statecharts

Les statecharts ont été conçus pour faciliter la modélisation de processus informatiques complexes. Les statecharts permettent d'enrichir les propriétés des automates à états finis tout en conservant le même principe de représentation graphique, facilement interprétable par l'utilisateur humain (voir 4.1).

Les statecharts permettent entre autres de modéliser des processus hiérarchiques : des macroétats peuvent contenir des sous-états, décrivant plus finement l'évolution du processus. Les statecharts permettent aussi de modéliser des processus parallèles, mais avec certaines limitations.

Réseaux de Petri

Les réseaux de Petri disposent d'une représentation graphique différente des automates (voir 4.1). C'est un graphe bipartite orienté contenant deux types de nœuds : les places et les transitions. Des arcs relient les places aux transitions et les transitions aux places. Les places possèdent un marquage, représenté par la présence ou l'absence de jetons. Contrairement aux automates à états finis, il n'y a pas d'états. C'est le marquage du réseau de Petri qui correspondrait à la notion d'état dans un automate à états finis.

La représentation graphique des réseaux de Petri est donc un peu plus complexe que celle des automates à états finis, avec la notion de places et de marquage. Cependant, comme plusieurs places peuvent être marquées à la fois, cela permet de traduire une plus grande richesse de comportements : parallélisme et formes complexes de concurrence.

Il existe de nombreuses variantes des réseaux de Petri (Murata, 1989). Les réseaux de Petri temporels permettent de représenter la dynamique de manière plus fine, en associant des contraintes temporelles, aux places, aux transitions ou aux arcs. Dans les réseaux de Petri hiérarchiques, certains éléments du réseau de Petri peuvent être représentés eux-mêmes par un réseau de Petri. Enfin, les réseaux de Petri colorés, parfois appelés réseaux de Petri de haut niveau, permettent d'ajouter des contraintes (couleurs) de haut niveau sur le réseau (Jensen, 1997).

Notons que les modèles automates à états finis sont un sous-ensemble des modèles de type réseau de Petri (Murata, 1989). Il existe des méthodes pour convertir des statecharts en réseau de Petri, mais la conversion d'un réseau de Petri en un statechart n'est pas toujours possible (Eshuis, 2011).

Comparaison des différents modèles

Le tableau 4.1 présente les forces et faiblesses des trois formalismes automates que nous avons introduits. Les statecharts et les réseaux de Petri offrent le plus de possibilités. Le formalisme des statecharts a l'avantage d'être plus intuitif et donc plus accessible à l'analyste, la notion de hiérarchie permet aussi de permettre à l'analyste de naviguer entre les différents niveaux du processus modélisé. Les réseaux de Petri ont l'avantage de pouvoir décrire des formes plus complexes de concurrence. Les réseaux de Petri possèdent en outre de nombreuses extensions permettant de prendre en compte, entre autres, le temps et la hiérarchie. Statecharts et réseaux de Petri sont deux types de modèles candidats pour représenter l'activité de conduite automobile.

TABLE 4.1 – Comparaison de modèles automates au regard des propriétés recherchées pour comprendre et simuler l'activité de conduite automobile.

Formalisme	séquence	temps	boucles	concurrence	parallélisme	hiérarchie
Automates à états finis	+	-	+	+	-	-
Statecharts	+	-*	+	+	+	+
Réseaux de Petri	+	-*	+	+	+	-*

- : propriété absente du modèle. + : le modèle possède la propriété. * : possible par des extensions.

4.2 Reconstruire un modèle à partir de traces : la fouille de workflow

4.2.1 Dualité automates-traces

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.

4.2.2 La fouille de workflow

Contexte

Puisque nous nous intéressons spécifiquement à l'observation de comportements, nous nous concentrons sur les approches de fouille de processus (*process mining*) (van der Aalst *et al.*, 2003; van der Aalst et Weijters, 2004). L'objectif est de fouiller des séquences pour découvrir l'automate correspondant. La fouille de processus, aussi appelée fouille de workflow (*workflow mining*) ou découverte de processus (*process mining*)

vise typiquement, à analyser le workflow d'une entreprise à partir des traces extraites de progiciels tels que les ERP[1]. 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 ou sur la formation, les contrôles, etc.

La fouille de workflow propose d'étudier l'activité sous différents aspects : le flux de travail, l'aspect organisationnel et les propriétés des actions. Le flux de travail décrit l'activité et l'ordonnancement des tâches entre elles. L'aspect organisationnel s'intéresse aux ressources associées au flux de travail, par exemple qui est à l'origine d'une action. L'étude des propriétés des actions a pour objectif d'enrichir le modèle avec les informations implicitement contenues dans les traces à travers les attributs de chaque action. C'est l'étude du flux de travail qui est pertinent pour l'objectif de modélisation posé dans cette thèse.

Dans le cadre de l'étude du flux de travail, trois questions sont étudiées par la fouille de workflow : la question de la découverte, de la conformité et de l'extension. Pour la découverte, aucun modèle n'est connu a priori et il s'agit de découvrir un modèle à partir de traces. Pour la conformité, un modèle est connu a priori et il est question de mesurer les écarts entre le modèle et la réalité. Pour l'extension, un modèle est connu a priori et il est question d'enrichir ce modèle à partir des données réelles collectées. Dans notre cas, il n'y a pas de modèle connu a priori. C'est donc la question de la découverte qui nous intéresse.

Bien que le flux du travail d'une entreprise soit une activité très différente de l'activité de conduite automobile, le même genre de problèmes se pose, car il s'agit d'une activité humaine dynamique et complexe. Ainsi, la question de la synthèse d'un modèle à partir de traces est la même.

Modèles de workflow

De nombreux formalismes sont utilisés pour décrire des workflows. Van der Aalst (2004) compare plus de 20 formalismes et leur expressivité. Ces formalismes sont souvent conçus de manière ad hoc, mais il y a de bonnes raisons d'utiliser un formalisme basé sur les réseaux de Petri (van der Aalst, 1998b). En particulier en plus de leur nature graphique, les réseaux de Petri possèdent une sémantique formelle non ambiguë et de nombreuses techniques d'analyse existent pour les étudier. Van der Aalst (1998a) propose de représenter les workflows en s'appuyant sur une sous-classe des réseaux de Petri, les WF-nets.

La littérature de la fouille de workflow s'appuie largement sur ce formalisme. Van Dongen *et al.* (2009) présentent pas moins de 13 algorithmes de fouille de workflow permettant de reconstruire un réseau de Petri à partir de traces, dont 10 s'appuient sur le formalisme des WF-nets. Dans une étude plus ancienne, van der Aalst *et al.* (2003) font référence à quatre approches : deux s'appuient sur les WF-nets, une est basée sur un formalisme ad hoc qui peut se traduire sous forme de réseau de Petri et la dernière s'appuie sur une représentation ad hoc sous forme de blocs hiérarchiques enchâssés. Dans la suite, à la section 4.3 (p. 70), nous passons en revue ces algorithmes (à l'exception de l'approche par blocs trop limitée).

[1] Entreprise Resource Planning

À notre connaissance, il n'existe pas d'algorithmes de fouille générant des statecharts. Au-delà du domaine de la fouille de workflow, les approches visant à générer des statecharts s'appuient sur la conception par scénario (Harel et Kugler, 2001; Harel *et al.*, 2005). L'objectif est de construire un modèle à partir de spécifications : les concepteurs traduisent les spécifications en scénarios et les scénarios sont convertis en statecharts. La différence est significative, puisque le modèle, même s'il n'est pas encore explicite, est connu. Le langage de scénario possède une sémantique forte et capture ces connaissances, telles que les liens de causalité entre les différentes actions. Ces connaissances ne sont pas accessibles explicitement à partir de simples traces.

Dans la suite, nous nous focalisons sur les approches de fouille de workflow visant à construire des réseaux de Petri.

4.2.3 Réseaux de Petri et WF-nets

Nous présentons maintenant le formalisme des réseaux de Petri et des WF-nets. Cette présentation se veut être accessible à l'analyste voulant utiliser ce formalisme. Les définitions formelles sont présentées à l'annexe A. Le lecteur familier avec les réseaux de Petri peut directement poursuivre la lecture à la sous-section dédiée aux WF-nets, où les spécificités des WF-nets seront présentées.

Réseaux de Petri

Un réseau de Petri est un modèle possédant à la fois une représentation graphique et une sémantique formelle. Nous nous focalisons ici sur la représentation graphique en donnant une idée de la sémantique formelle associée.

La représentation graphique d'un réseau de Petri est un graphe orienté constitué de deux types de nœuds : les places et les transitions. La figure 4.2 montre un exemple de réseau de Petri. Les places sont représentées par des cercles. Les transitions sont représentées par des carrés ou des rectangles. Les arcs sont représentés par des flèches. Un arc relie une place à une transition ou une transition à une place[2].

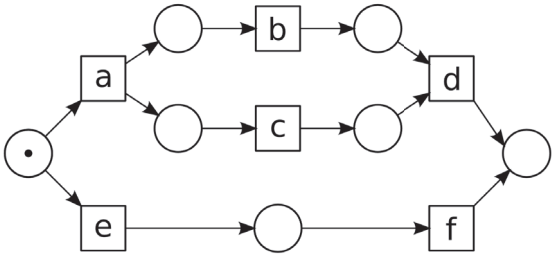


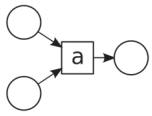
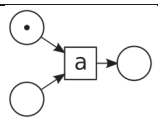
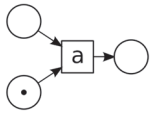
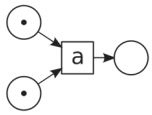
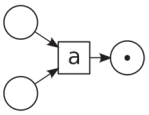
FIGURE 4.2 – Exemple de réseau de Petri. Les places sont représentées par des cercles. Une place est marquée si elle possède un jeton. Les transitions sont représentées par des rectangles. Les arcs relient les transitions aux places et les places aux transitions. Le réseau est dans son état initial.

L'état du réseau de Petri est caractérisé par son marquage : la présence ou non de jetons dans les différentes places du réseau. Lorsque le réseau de Petri est utilisé pour

[2]Un arc ne relie jamais une place à une autre place ou une transition à une autre transition

simuler le comportement modélisé, ce marquage évolue. En effet, les places marquées vont permettre de rendre certaines transitions franchissables (activables). Le franchissement d'une transition va consommer les jetons présents dans ses places d'entrée et produire des jetons dans ses places de sorties. Le tableau 4.2 illustre ce processus.

TABLE 4.2 – Une transition est franchissable quand toutes ses places d'entrée sont marquées. Le franchissement d'une transition consomme les jetons de ses places d'entrée et produit des jetons dans ses places de sortie.

Avant le franchissement	**Après le franchissement**
	Transition non franchissable
	Transition non franchissable
	Transition non franchissable
	

Les réseaux de Petri sont capables de représenter de nombreux types de structures : séquences, boucles, concurrence (choix entre plusieurs actions) et le parallélisme. Le tableau 4.3 présente des exemples basiques de ces motifs. Le choix exclusif entre plusieurs actions ou séries d'actions débute avec le motif XOR-SPLIT et finit avec le motif XOR-JOIN. Le déclenchement parallèle de deux ou plusieurs séries d'actions débute avec le motif AND-SPLIT et finit avec le motif AND-JOIN.

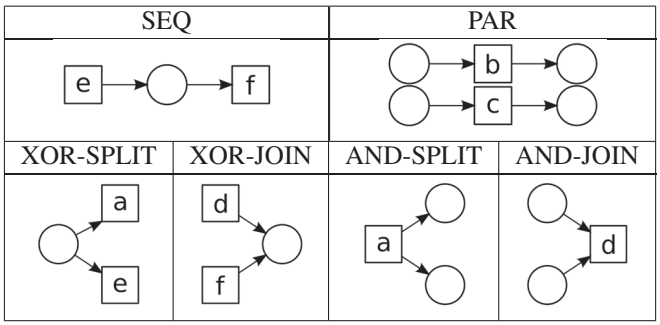
Les réseaux de Petri sont capables de modéliser des processus complexes allant bien au-delà de ces exemples simples et combinant différents motifs. Par exemple le choix entre plusieurs actions concurrentes peut être libre ou non libre (voir figure 4.3). Un choix est libre, au niveau d'une place donnée, s'il suffit que cette place soit marquée pour que plusieurs transitions deviennent franchissables. Le choix est non libre si le franchissement des transitions succédant à une place est conditionné par le marquage d'autres places. Le choix non libre combine les motifs XOR-SPLIT et AND-JOIN.

La figure 4.4 présente une synthèse de cette section. Sur la base du réseau de Petri présenté à la figure 4.2, les motifs sont mis en évidence et l'ensemble des traces que ce modèle peut produire est présenté.

WF-nets

Les WF-nets sont une spécialisation des réseaux de Petri (van der Aalst, 1998a). Ainsi, tout WF-net est un réseau de Petri. Pour qu'un réseau de Petri soit un WF-net, il doit

TABLE 4.3 – Motifs structurels caractéristiques des réseaux de Petri. Le motif SEQ est caractéristique des séquences. Le motif PAR représente des actions parallèles. Les motifs XOR-SPLIT et XOR-JOIN caractérisent le début et la fin d'un choix entre des actions concurrentes. Les motifs AND-SPLIT et AND-JOIN caractérisent le début et la fin de séries d'actions parallèles.



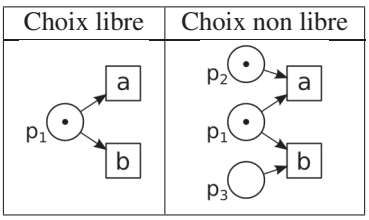


FIGURE 4.3 – Différences entre un choix libre et un choix non libre. Dans le cas du choix libre, le marquage de la place de choix (p_1) rend franchissables les deux transitions a et b, indépendamment du marquage du reste du réseau : le choix est libre. Dans le cas du choix non libre, le marquage de la place de choix ne rend pas nécessairement franchissables les deux transitions a et b : le choix est conditionné par le marquage des places p_2 et p_3.

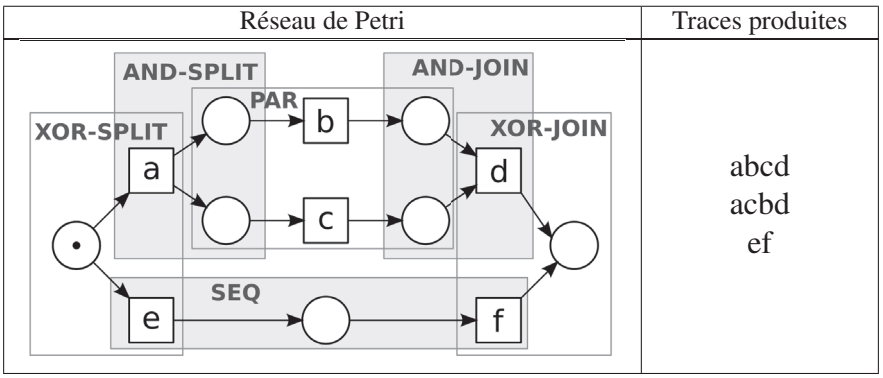


FIGURE 4.4 – Réseau de Petri avec ses motifs structurels mis en évidence. Le modèle peut produire exactement trois traces différentes.

posséder deux propriétés.

Première propriété : un WF-net possède une place initiale et une place finale, caractérisant le début et la fin du workflow. La place initiale ne possède pas d'arcs entrants et la place finale ne possède pas d'arcs sortants.

Deuxième propriété : si l'on ajoutait une transition en connectant la place de sortie à cette transition et cette transition à la place d'entrée (on fait boucler le WF-net sur lui-même), le réseau obtenu est fortement connecté, c'est-à-dire que si l'on prend deux nœuds (places ou transitions) du réseau au hasard, il existe un chemin reliant le premier au deuxième.

Le réseau de Petri présenté à la figure 4.2 possède ces deux propriétés, c'est donc un WF-net.

4.2.4 Défis de la fouille de workflow

Le problème de la fouille de workflow se traduit de la façon suivante : étant donné un ensemble de traces d'exécution d'un processus, trouver un modèle du processus étudié.

Complétude des traces et caractère généralisant du modèle

Comment traduire le comportement observé dans les traces ? Cette question simple amène d'autres questions. Au niveau de traces : représentent-elles l'ensemble exhaustif de tous les comportements possibles ? Au niveau du modèle découvert : doit-il généraliser les comportements possibles à partir des traces ou interdire les comportements non observés ? Ces deux questions sont évidemment liées.

La première question suppose de définir une notion de complétude des traces. Les algorithmes de fouille de workflow s'appuient sur différentes notions de complétudes : très stricte, où tous les comportements possibles doivent être observés et tout autre comportement est considéré comme faux, ou, plus souple, où les traces peuvent être bruitées.

La deuxième question pose la question du juste équilibre entre découvrir des modèles trop généraux où tout est possible et des modèles trop spécifiques, calquant les traces. Aucun des deux extrêmes n'apporte de connaissances sur le processus étudié.

Un exemple illustratif

Comment traduire en un réseau de Petri le comportement observé dans les traces ? Pour un ensemble de traces données, il est possible de construire de nombreux modèles différents. Pour illustrer les difficultés que l'on rencontre à construire un « bon » modèle à partir de traces, nous allons nous appuyer sur un exemple tiré de l'article de van Dongen *et al.* (2009).

Le tableau 4.4 reproduit un ensemble de traces d'un processus. Ce processus consiste à se déplacer pour assister à une conférence d'une journée. Différents événements sont enregistrés : se préparer (*Get Ready*), voyager en train ou en voiture (*Travel by Train* ou *Travel by Car*), début de la conférence (*Conference Starts*), faire une présentation (*Give a Talk*), poser une question (*Ask Question*), assister à la visite guidée (*Join Guided Tour*), assister au dîner (*Join Dinner*), rentrer à la maison (*Go Home*), et payer le parking (*Pay for Parking*).

À partir de ces traces dont la signification est relativement simple, on cherche à découvrir un modèle traduisant l'ensemble de ces comportements. Dans l'idéal, le modèle doit permettre de généraliser et autoriser des comportements non observés. Par

TABLE 4.4 – Exemple de traces d'un processus. Adapté de van Dongen *et al.* (2009)

Id	Traces
1	Start, Get Ready, Travel by Train, Conference Starts, Ask Question, Join Guided Tour, Join Dinner, Go Home, Travel by Train, End
2	Start, Get Ready, Travel by Car, Conference Starts, Give a Talk, Ask Question, Ask Question, Ask Question, Ask Question, Join Guided Tour, Join Dinner, Go Home, Pay for Parking, Travel by Car, End
3	Start, Get Ready, Travel by Train, Conference Starts, Give a Talk, Ask Question, Join Guided Tour, Join Dinner, Go Home, Travel by Train, End
4	Start, Get Ready, Travel by Car, Conference Starts, Give a Talk, Join Guided Tour, Join Dinner, Go Home, Pay for Parking, Travel by Car, End
5	Start, Get Ready, Travel by Train, Conference Starts, Give a Talk, Join Guided Tour, Join Dinner, Go Home, Travel by Train, End
6	Start, Get Ready, Travel by Car, Conference Starts, Join Guided Tour, Join Dinner, Go Home, Pay for Parking, Travel by Car, End

exemple, il est possible de voyager en train, ne pas faire de présentation et ne pas poser de questions, même si cette combinaison n'est pas observée. À l'inverse, on ne souhaite pas que le modèle soit trop généralisant. Par exemple, il est logique de conserver le même moyen de transport le matin et le soir[3].

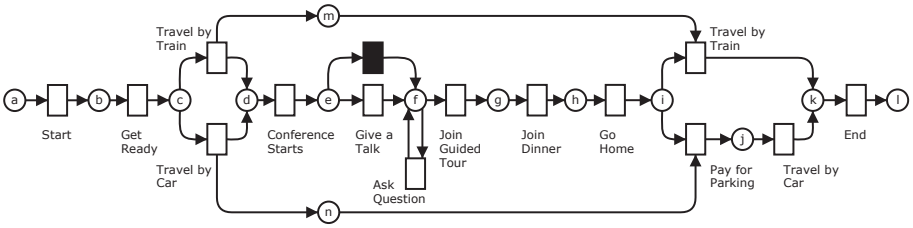


FIGURE 4.5 – Exemple de réseau de Petri pouvant représenter les traces présentées dans le tableau 4.4. Repris de van Dongen *et al.* (2009)

La figure 4.5 présente un modèle candidat que l'on aimerait pouvoir découvrir à partir des traces. Ce modèle est capable de représenter tous les comportements observés. Il généralise certains comportements : donner une présentation est facultatif, il est possible de poser un nombre quelconque de questions ou pas du tout. Il conserve certaines spécificités observées dans les traces : le mode de transport à l'aller est le même qu'au retour.

À partir de cet exemple, nous allons présenter les défis rencontrés par la fouille de workflow.

Les défis techniques à relever par les algorithmes

Les algorithmes construisant des modèles à partir de traces doivent relever un certain nombre de défis. Ces défis sont présentés par van der Aalst et Weijters (2004).

[3] On peut discuter cet argument. Il est possible de faire l'aller en train et de rentrer en covoiturage. Mais admettons, pour illustrer notre propos que cela n'est pas autorisé.

Van der Aalst et 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 transitions « cachées » (une action qui n'apparaît pas dans les traces) ; les transitions « dupliquées » (une même action 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...

Découvrir des transitions dupliquées. On appelle *transition dupliquée*[4] une transition qui apparaît plusieurs fois dans le modèle. Les transitions « Travel by Car » et « Travel by Train » sont dupliquées. Dans la plupart des cas, à chaque type d'événement, une seule transition est associée. Mais dans certains cas, il est nécessaire de représenter le même type d'événement par deux transitions. C'est le cas de l'événement « Travel by Car » par exemple, qui apparaît au début et à la fin de chaque trace et est représenté par deux transitions dans le modèle, l'une désignant le voyage aller, l'autre le voyage retour.

Les transitions dupliquées sont difficiles à découvrir, car il faut être capable de distinguer deux actions différentes portant le même nom, d'une même action répétée plusieurs fois.

Découvrir des transitions cachées. On appelle *transition cachée*[5] une transition qui apparaît dans le modèle, mais pas dans les traces. Ce genre de transition est parfois nécessaire pour modéliser certains comportements. Dans l'exemple, une transition cachée est nécessaire pour exprimer qu'il est possible d'aller à une conférence sans faire de présentation. Cette transition cachée est représentée par un rectangle noir à la figure 4.5.

Les transitions cachées sont difficiles à détecter, puisqu'il s'agit de traduire un comportement qui n'a pas été observé. Elles peuvent être utiles à la fois pour leur sémantique ou pour simplifier la structure du modèle.

Découvrir des boucles. Une boucle consiste en une ou plusieurs actions pouvant se répéter successivement. Elle peut être difficile à détecter dans le cas où d'autres actions ne relevant pas de la boucle sont exécutées en parallèle. En outre, il faut distinguer la répétition de la même action au sein d'une boucle des transitions dupliquées, représentant la même action dans différents contextes.

Découvrir des choix non libres. Un choix est non libre quand il est contraint par l'exécution d'autres actions. Par exemple pour le modèle présenté à la figure 4.5, à la fin de la journée, le participant peut soit voyager en train, soit voyager en voiture. Mais ce choix n'est pas libre : il dépend du mode de transport utilisé le matin. Ce choix non libre est matérialisé par la présence des places m et n qui contraignent le modèle en interdisant, par exemple, à une personne de voyager en train le matin et en voiture le soir.

[4] La traduction exacte est « tâche dupliquée » (*duplicate task*). Nous proposons une traduction différente pour éviter la confusion entre « tâche », au sens de la fouille de workflow : une action du processus et « tâche » au sens de l'ergonomie.

[5] Comme pour la note précédente, la traduction exacte est « tâche cachée » (*hidden task*).

Les choix non libres sont difficiles à détecter puisqu'ils nécessitent de détecter des relations de causalité entre des actions non locales (ici entre le début du modèle et la fin du modèle).

Exploiter le temps. Les traces présentées dans le tableau 4.4 sont des séquences d'actions. Ces traces ne contiennent pas d'information temporelle. Cependant, dans les traces réelles une estampille temporelle est souvent associée à chaque événement. On tire profit de cette information temporelle soit pour enrichir le modèle, soit pour améliorer le processus de construction du modèle.

Gérer le bruit. La plupart des algorithmes exploitent les traces en supposant qu'elles sont correctes. Cependant, les traces réelles sont souvent bruitées, par exemple, certaines actions peuvent ne pas avoir été enregistrées. Le défi consiste ici à distinguer les observations pertinentes de celles relevant du bruit. Ce qui est d'autant plus difficile lorsque l'on travaille sur des cas uniques ou avec un petit nombre de traces.

Les défis conceptuels

Certains défis à relever lors de la fouille de workflow dépassent le cadre technique et relèvent de problèmes conceptuels (van der Aalst et Weijters, 2004; van Dongen *et al.*, 2009). Si des données sont incomplètes comment peut-on espérer qu'un algorithme découvre les informations qui ne sont pas présentes ? Ces limites de la fouille de workflow ne peuvent être dépassées qu'en prenant en compte des connaissances extérieures, typiquement sur le processus à modéliser, soit lors de la préparation des données, soit au niveau de l'algorithme lui-même.

L'incomplétude des données. Les données ne sont pas toujours complètes, parce qu'elles sont bruitées, parce que toutes les situations n'ont pas été observées, parce que certaines informations sont manquantes, etc.

La sémantique des données. Les données ne sont pas qu'une série de chiffres, une sémantique peut leur être associée. C'est le sens que l'on donne à telle ou telle observation. Si l'on est capable d'associer une sémantique aux données, alors cette information peut être exploitée lors de la synthèse du modèle.

Les niveaux d'abstraction. Toutes les données collectées ne relèvent pas du même niveau d'abstraction. Par exemple, si l'on compare un événement « angle volant à 30 degrés » et un événement de type « tourner à gauche », même si les deux événements relèvent du même processus, ils relèvent de différents niveaux d'abstraction. En général, les traces n'explicitent pas les différents niveaux d'abstraction. Les traces sont « plates », comme si tous les événements appartenaient au même niveau d'abstraction.

Les différents points de vue. Plus que le simple ordonnancement des actions, un modèle peut contenir des informations sur les ressources ou sur des paramètres. Sur les ressources, pour les flux de travail d'une entreprise, il s'agit de savoir qui est à l'origine d'une action ou sur quoi porte l'action. Sur les paramètres, il s'agit de connaître l'impact des différentes actions sur un paramètre mesuré ou à l'inverse, l'impact d'un paramètre sur l'évolution du modèle. Dans le cas de la conduite automobile, il s'agit de savoir en quoi une situation va engendrer un certain niveau de stress ou bien de savoir

en quoi l'âge ou l'expérience du conducteur va impacter la conduite. Ces informations complémentaires sont souvent représentées dans les traces sous la forme d'attributs associés aux événements.

Défis – bilan. Les défis de la synthèse de modèles sont à la fois techniques et conceptuels. Dans la section suivante, une revue des algorithmes de synthèse de modèles compare les algorithmes entre eux et leur capacité à répondre aux défis techniques. Pour résoudre les problèmes conceptuels, il est nécessaire d'exploiter des connaissances expertes sur le processus à modéliser, en amont de la synthèse, lors de la préparation des traces, et au moment de la synthèse, au niveau de l'algorithme. Pour agir au niveau de l'algorithme, une solution consiste à intégrer ces connaissances expertes dans les résultats intermédiaires calculés par l'algorithme.

4.3 Revue des algorithmes

Van der Aalst *et al.* (2003) et van Dongen *et al.* (2009) proposent une revue de la littérature des algorithmes de fouille de workflow. Les auteurs comparent les algorithmes au regard de leur capacité à répondre aux différents défis de la fouille de workflow.

Dans leur revue de la littérature, van Dongen *et al.* (2009) classent les approches en cinq catégories en fonction du principe de fonctionnement : celles basées sur une abstraction à partir des données, celles basées sur des heuristiques, celles basées sur une exploration, celles basées sur les régions de langages et celles basées sur la construction d'un automate à états.

Nous proposons un autre classement de ces algorithmes, en quatre catégories, basées sur les structures formelles utilisées comme résultat intermédiaire : relations de causalité, matrices de causalité, automate ou régions de langages. Ce découpage nous permet de mettre en avant la possibilité d'interagir sur les résultats intermédiaires pour résoudre les défis conceptuels posés par la fouille de workflow.

Les algorithmes basés sur une abstraction s'appuient sur le calcul de relations d'ordre et de causalité pour en déduire la structure du réseau de Petri.

D'autres algorithmes s'appuient sur la notion de matrice de causalité (*causal matrix*). Les matrices de causalité sont utilisées comme une représentation des connaissances alternative aux réseaux de Petri dans les algorithmes GA[6] et DGA[7], basés sur une exploration. Une fois la matrice de causalité découverte, le WF-net correspondant en est déduit.

Une troisième approche consiste à construire dans un premier temps un automate ou un graphe intermédiaire, qui est traduit dans un second temps en réseau de Petri. C'est le cas des algorithmes basés sur la construction d'automates à états finis et de l'outil InWoLvE.

Enfin, une quatrième approche s'appuie sur les propriétés formelles des régions de langages. Un langage est l'ensemble des traces que peut générer un modèle. Une région est un ensemble de contraintes que le modèle doit respecter pour représenter correctement le langage. Une fois calculées, les régions servent à calculer des places candidates pour le réseau de Petri.

Le tableau 4.5 présente une synthèse des propriétés des algorithmes présentés dans la littérature (van der Aalst *et al.*, 2003; van Dongen *et al.*, 2009). Les algorithmes

[6]Genetic Algorithm Miner
[7]Duplicates Genetic Algorithm Miner

sont classés selon les quatre familles que nous venons de présenter. Notons que les algorithmes présentés sont tous capables de gérer la séquentialité, le parallélisme, la concurrence (choix) et dans une certaine mesure les boucles.

TABLE 4.5 – Comparaison des propriétés des différents algorithmes. Les algorithmes sont regroupés en quatre familles, chacune représentant un type de résultat intermédiaire. Inspiré de van Dongen *et al.* (2009) et van der Aalst *et al.* (2003).

Résultat intermédiaire	Algorithme	seq	par	chx	bcl	CNL	TD	TC	t	brt
Relations d'ordre et de causalité	algorithme α	+	+	+	+/-	-	-	-	-	-
	algorithme β	+	+	+	+	-	-	-	-	-
	algorithme α^{+}	+	+	+	+	-	-	-	-	-
	algorithme α^{++}	+	+	+	+	+	-	-	-	-
	algorithme α^{*}	+	+	+	+/-	-	+/-	-	-	-
	algorithme $\alpha^{\#}$	+	+	+	+	-	-	+	-	-
	EMiT	+	+	+	+/-	-	-	-	+	-
	Little Thumb	+	+	+	+/-	-	-	-	-	+
Matrices de causalité	HeuristicsMiner	+	+	+	+	+/-	-	+	-	+
	GA	+	+	+	+	+	-	+	-	+
	DGA	+	+	+	+	+	+	+	-	+
Automate ou graphe	InWoLvE	+	+	+	+/-	-	+	-	-	+
	Découverte d'automate	+	+	+	+	(+)	(+)	(+)	-	-
Régions de langages	*Basis representation*	+	+	+	+	+	-	-	-	-
	Seperation representation	+	+	+	+	+	-	-	-	-
	Integer Linear Programming	+	+	+	+	+	-	-	-	-

+ : propriété gérée par l'algorithme. - : propriété non gérée par l'algorithme. seq : séquentialité. par : parallélisme. chx : choix. bcl : boucles. CNL : choix non libre. TC : transitions cachées. TD : transitions dupliquées. t : temps. brt : bruit.

4.3.1 Algorithmes basés sur des relations d'ordre et de causalité

Les algorithmes basés sur des relations d'ordre et de causalité s'appuient sur l'algorithme α. L'algorithme α fonctionne en deux phases. Une première phase infère une relation d'ordre sur les transitions à partir des traces : telle action est directement suivie d'une autre action. Cette relation d'ordre permet d'inférer des relations de causalité, d'indépendance ou de parallélisme. Dans une deuxième phase, ces relations sont utilisées pour construire un WF-net candidat.

Les relations utilisées possèdent une interprétation accessible à l'analyste : telle action peut être directement suivie d'une autre action, ou bien deux actions ont lieu en parallèle, ou une action est la cause d'une autre action. Ce type de résultats intermédiaires est donc exploitable par un analyste pour y intégrer ses connaissances sur l'activité étudiée.

L'algorithme α (van der Aalst *et al.*, 2004) permet de reconstruire une classe spécifique de WF-nets : les WF-nets structurés et conformes sans boucles courtes. De nombreuses variantes de l'algorithme sont proposées pour permettre de construire des classes plus larges de WF-nets. L'algorithme β (Wen *et al.*, 2009) adapte l'algorithme α

dans le cas où le début et la fin des actions sont explicités dans les traces par deux événements différents. Cette approche permet de gérer les boucles courtes. L'algorithme α^+ (de Medeiros *et al.*, 2005) étend l'algorithme α en lui permettant de gérer les boucles courtes. L'algorithme α^{++} (Wen *et al.*, 2007a) étend l'algorithme α^+ pour détecter des choix non libres. L'algorithme α^* (Li *et al.*, 2007) permet de détecter et reconstruire des transitions dupliquées. L'algorithme $\alpha^\#$ (Wen *et al.*, 2007b) permet de découvrir des transitions cachées. L'outil EMiT (van der Aalst et van Dongen, 2002) prend en compte le temps et génère un réseau de Petri temporel[8]. Enfin, l'outil Little Thumb (Weijters *et al.*, 2006) exploite des heuristiques pour gérer le bruit des données.

4.3.2 Algorithmes basés sur les matrices de causalité

Les matrices de causalité (de Medeiros *et al.*, 2004b) servent à caractériser pour chaque transition, les transitions directement précédentes et les transitions directement suivantes. Les relations logiques *et* ($\wedge$) et *ou* ($\vee$) sont utilisées pour caractériser les transitions qui sont parallèles ou concurrentes. À titre d'exemple, la table 4.6 présente la matrice de causalité du réseau de Petri présenté à la figure 4.2, à la page 63.

TABLE 4.6 – Matrice de causalité du réseau de Petri présenté à la figure 4.2.

Transitions	**Entrées**	**Sorties**
a	-	b $\wedge$ c
b	a	d
c	a	d
d	b $\wedge$ c	-
e	-	f
f	e	-

La conversion d'un réseau de Petri quelconque en matrice de causalité est triviale, mais, la conversion d'une matrice de causalité en un réseau de Petri n'est pas toujours possible directement : dans certains cas, il faut ajouter des transitions cachées pour produire un réseau de Petri correspondant. De Medeiros *et al.* (2004b) proposent une méthode pour construire un réseau de Petri à partir d'une matrice de causalité en introduisant des transitions cachées. Ainsi, les algorithmes basés sur les matrices de causalité permettent tous de gérer les transitions cachées.

Les matrices de causalité peuvent constituer un résultat intermédiaire favorable à l'interaction, car elles contiennent une sémantique interprétable par l'analyste : « une action est précédée et suivie d'une combinaison logique d'actions d'entrée et de sortie. »

L'outil HeuristicsMiner (Weijters *et al.*, 2006) s'appuie sur des heuristiques pour gérer le bruit dans les données. Dans les algorithmes génétiques GA et DGA (de Medeiros, 2006; de Medeiros *et al.*, 2006; van der Aalst *et al.*, 2005), les matrices de causalité représentent des individus candidats. Le principe de sélection naturelle permet de proposer, au fil des générations, des modèles candidats plus pertinents. Ces algorithmes permettent de gérer le bruit et le choix non libre. L'algorithme DGA se distingue de l'algorithme GA par sa capacité à gérer les transitions dupliquées.

[8] Plus précisément un réseau de Petri P-temporel, la notion de temps étant associée aux places

4.3.3 Les algorithmes basés sur la construction d'un automate ou d'un graphe

La famille des automates à états finis est plus largement étudiée que les réseaux de Petri. Il n'est donc pas étonnant qu'un certain nombre d'algorithmes existent pour déduire un automate à états finis à partir d'un ensemble de traces. À partir de ce premier modèle, il est possible de déduire, un réseau de Petri équivalent (Kindler *et al.*, 2006). Le concept de régions est utilisé pour définir des ensembles d'états partageant des transitions d'entrée et de sortie, ces régions vont correspondre à des places du réseau de Petri (Desel et Reisig, 1996). D'autres approches analogues s'appuient sur la construction d'un graphe qui est converti dans un second temps en réseau de Petri.

Un automate ou un graphe est une structure de représentation des connaissances de haut niveau, proche du modèle. La sémantique est forte, puisque c'est une sémantique d'un niveau très proche à celle du modèle final.

Les algorithmes basés sur la construction d'automates sont nombreux. Nous ne faisons que les évoquer ici. Van Dongen *et al.* (2009) donne quelques références sur le sujet. En fonction des algorithmes utilisés, il est possible de gérer les différents défis de la fouille de workflow. L'outil InWoLvE (Herbst et Karagiannis, 2004) calcule à partir des traces un graphe stochastique d'activité, puis dans un second temps traduit ce graphe en réseau de Petri[9]. Cette approche permet de gérer le bruit et les transitions dupliquées.

4.3.4 Les algorithmes basés sur les régions de langages

Les algorithmes basés sur les régions s'appuient sur la dualité entre un réseau de Petri et les traces qu'il peut générer. L'ensemble de toutes les traces qu'un réseau de Petri peut générer est appelé *langage*. Les traces sont appelées des *mots*. L'objectif des approches basées sur les régions est de trouver un réseau de Petri, dont le déclenchement des transitions permet de générer tous les mots du langage et seulement ces mots.

La communauté de la fouille de workflow a adapté ces approches formelles aux problèmes appliqués de la fouille de workflow (Bergenthum *et al.*, 2007; van der Werf *et al.*, 2008).

Le principe est de définir un ensemble d'inéquations que doit respecter un réseau de Petri pour pouvoir produire le langage donné. Une solution à ce système d'inéquation est appelée une région et elle caractérise une place candidate et son marquage initial.

La notion de région est une notion abstraite qui semble a priori difficilement exploitable par un analyste.

Plusieurs méthodes existent pour déterminer ces régions, van Dongen *et al.* (2009) les classent en trois catégories : *Basis Representation*, *Separating Representation* et *Integer Linear Programming*. La méthode *Basis Representation* s'appuie sur la découverte d'une base de solutions à partir de laquelle toute autre solution peut être déduite par combinaison linéaire positive (Bergenthum *et al.*, 2007). La méthode *Separating Representation* construit itérativement des places solutions qui limitent le comportement du réseau de Petri, en interdisant les comportements qui ne sont pas présents dans le langage (Bergenthum *et al.*, 2007). La méthode *Integer Linear Programming* (van der Werf *et al.*, 2008) construit itérativement des places solutions qui matérialisent des relations de causalité, telles que celles utilisées dans l'algorithme α. Ce dernier algorithme constitue donc une approche hybride.

[9] Le formalisme utilisé n'est pas un réseau de Petri, il s'agit d'un formalisme ad hoc nommé ADONIS. Cependant, le formalisme ADONIS se traduit de manière directe en réseau de Petri.

4.4 Comment exploiter ces algorithmes dans le contexte applicatif de la conduite automobile ?

4.4.1 La fouille de workflow confrontée aux données réelles : les défis

Les algorithmes de fouille de workflow que nous avons présentés s'appuient sur des principes théoriques d'équivalence entre des traces et des modèles : un modèle connu produit des traces et l'algorithme doit être capable de retrouver un modèle équivalent (ou un modèle proche) à partir de cet ensemble de traces. Les défis théoriques à surmonter sont nombreux, mais certains algorithmes offrent des solutions pertinentes à certains de ces défis.

Cependant, lorsque l'on essaie d'appliquer ces algorithmes dans des conditions réelles d'analyse, de nouveaux défis apparaissent (Hammori *et al.*, 2006; van der Aalst et Günther, 2007; van Dongen *et al.*, 2009). Outre le fait que les traces peuvent être bruitées, une partie du processus peut n'avoir tout simplement pas été enregistrée. Les traces peuvent contenir des informations qui ne relèvent pas du processus étudié. Et surtout, les traces enregistrées n'appartiennent pas toutes au même niveau de description du processus.

Ainsi, les outils de fouille de workflow, doivent offrir des solutions pour exploiter les différents niveaux d'abstraction, la sémantique présente dans les données et les attributs éventuellement associés aux événements (van Dongen *et al.*, 2009). Van der Aalst et Günther (2007) affirme que les outils visant à des applications pratiques doivent être *exploratoires*[10]. Un analyste doit guider le processus de modélisation. Pour permettre de découvrir une structure dans des traces non structurées, van der Aalst et Günther (2007) pensent qu'il faut offrir des moyens de visualisation permettant d'agréger et d'abstraire les données, d'accentuer ce qui est important et plus généralement de personnaliser la visualisation. Hammori *et al.* (2006) insiste sur le fait que la fouille de workflow est un processus interactif et itératif. Il faut favoriser l'interaction pour permettre à l'analyste d'agir sur le processus de fouille. Et pour permettre à l'analyste de mettre en œuvre de nombreuses itérations, il faut lui faciliter la tâche avec des outils intégrés. Par exemple, toute manipulation qui nécessite d'ouvrir un fichier dans un logiciel externe pour visualiser le résultat introduit une lourdeur qui entrave le travail de l'analyste.

4.4.2 Introduire l'interaction

Nous proposons de rendre interactifs certains des algorithmes présentés. Pour favoriser l'interaction avec l'analyste. L'objectif de l'interaction est de compenser l'incomplétude des traces avec l'expertise de l'analyste. Nous proposons d'interagir au niveau des résultats intermédiaires de ces algorithmes.

Parmi les différentes classes d'algorithmes, nous proposons d'introduire l'interaction dans les algorithmes basés sur des relations d'ordre et de causalité pour les raisons suivantes :

- Les relations d'ordre et de causalité sont en effet facilement interprétables par l'analyste.

[10] « Useful tools for practical application mus be *explorative*. »

- Le réseau de Petri construit se veut simple et ne contient pas de transitions supplémentaires, contrairement aux algorithmes basés sur les matrices de causalité.
- De nombreux algorithmes les utilisent, chacun permettant de gérer un des défis théoriques de la fouille de workflow.
- De plus, tous ces algorithmes sont dérivés d'un algorithme de base : l'algorithme α. Nous pouvons donc commencer par rendre interactif l'algorithme α et avoir de bonnes chances de pouvoir adapter ces modifications à ses variantes.

4.5 Conclusion

Dans ce chapitre, nous avons d'abord introduit différents formalismes candidats pour modéliser l'activité de conduite automobile. Nous avons présenté la fouille de workflow, qui offre des techniques pour construire des modèles, à partir d'un ensemble de traces. Ces techniques produisent des modèles sous la forme de WF-nets, qui est une spécialisation des réseaux de Petri.

Les techniques de fouille de workflow doivent faire face à un certain nombre de défis théoriques que nous avons explicités. Nous avons effectué une revue de la littérature des algorithmes de fouille de workflow et comparé leur capacité à répondre à ces différents défis.

Enfin, ces algorithmes doivent répondre à des défis pratiques lorsqu'ils sont confrontés aux données réelles. Ces défis pratiques viennent par exemple de l'incomplétude des données, ou des différents niveaux d'abstraction présents dans les traces du processus étudié. Pour répondre à ces défis pratiques, il faut rendre le processus de synthèse de modèles interactif et exploratoire. Il faut faciliter les interactions et les possibilités de visualisation de l'analyste.

Dans le chapitre suivant, le chapitre 5, nous montrons comment nous proposons de rendre interactifs les algorithmes se basant sur le calcul de relations d'ordre et de causalité. Cette modification vise à compenser l'incomplétude des traces par l'expertise d'un analyste.

Dans le chapitre 6, nous montrons comment le processus de synthèse de modèle s'intègre, en réalité, dans le cycle de découverte de connaissances (voir chapitre 3). Puis, nous montrerons en quoi le cycle de découverte de connaissances apporte une réponse méthodologique aux défis pratiques de la fouille de workflow, par exemple en traitant les problèmes d'abstraction des données en amont de la synthèse de modèles.

CHAPITRE 5

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

Dans ce chapitre, nous nous intéressons au cœur de notre contribution en ingénierie des connaissances. Nous montrerons comment nous proposons de rendre interactive la découverte d'automates à partir de traces d'activité.

Comme nous l'avons vu au chapitre 3, il existe une dualité entre modèles automates et traces. Utilisé en simulation, un modèle automate peut générer des traces. Ces traces représentent l'activité de l'automate. À l'inverse, à partir d'un ensemble de traces, que l'on suppose issues d'un automate, il est possible de construire une estimation de l'automate, capable de reproduire cet ensemble de traces. Dans la partie 4, nous avons vu qu'une des problématiques de la fouille de workflow est de construire un modèle de type automate d'un processus à partir de ces traces d'exécution.

En fonction des connaissances que l'on a ou des hypothèses que l'on fait sur les traces et sur le processus à modéliser (traces représentant tous les comportements possibles et uniquement les comportements possibles, modèles ne possédant pas de transitions dupliquées, etc.), il est possible de redécouvrir un modèle du processus à partir de ses traces d'exécution.

Tous les algorithmes font des hypothèses, plus ou moins restrictives, sur le modèle de l'activité observée.

Les réseaux de Petri constituent une classe de modèles permettant de modéliser le parallélisme et la concurrence de plusieurs processus présents au sein de la même activité (par ex. en conduite automobile, la recherche d'informations visuelles en parallèle d'une action : regarder le rétroviseur et déclencher le clignotant). Les réseaux de Petri sont aussi capables de représenter des boucles (comportements régulés). Comme nous l'avons vu au chapitre 3 ces propriétés sont des propriétés recherchées pour synthétiser des connaissances sur l'activité de conduite automobile. C'est pourquoi nous focalisons notre travail sur les algorithmes qui permettent de découvrir des réseaux de Petri (ou des modèles qui possèdent ces mêmes propriétés et peuvent se traduire sous la forme de réseaux de Petri).

5.1 Adapter la fouille de workflow pour modéliser l'activité

5.1.1 Définition du problème

La fouille de workflow s'intéresse à construire un modèle à partir d'un ensemble de traces d'un processus. Appliqué aux traces d'activités, analysées et transformées dans un SBT pour représenter un processus plus ou moins abstrait, le problème devient le suivant : comment à partir d'un ensemble *homogène* de $\mathcal{M}$-Traces, possédant un même modèle et représentant un même processus, peut-on reconstruire un modèle (sous la forme d'un réseau de Petri) du processus analysé ?

Ce problème demande de :

1. Traduire le problème de la fouille de workflow dans le vocabulaire de la $\mathcal{M}$-Trace ;

2. Utiliser les connaissances complémentaires qu'a l'analyste sur l'activité pour éclairer la fouille. Favoriser l'interaction entre l'analyste et l'algorithme.

Le premier point fait l'objet de la section suivante 5.1.2. Le deuxième point sera développé tout au long de ce chapitre.

5.1.2 Traduction du vocabulaire de la fouille de workflow dans le vocabulaire de la $\mathcal{M}$-Trace

Dans le domaine de la fouille de workflow, les traces d'exécution du processus observé sont représentées par un WF-log (*workflow-log*). Un WF-log est un ensemble de séquences, appelé *cas* dans le vocabulaire de la fouille de workflow, chacune représentant une exécution complète du processus, de l'initialisation à la terminaison. Une séquence d'un WF-log est un ensemble ordonné d'événements. Chaque événement possède un type, représentant l'exécution d'une action du processus, et un ordre d'apparition. Dans certains cas, le temps est pris en considération : plutôt qu'un ordre d'exécution, une estampille temporelle est associée à chaque événement.

Une action[1] se traduit par un type d'événement dans les WF-log et par un type d'*obsel*s dans le vocabulaire de la trace modélisée.

Différences entre un ensemble de $\mathcal{M}$-Trace et un WF-log ? Un *cas* d'un WF-log est un objet similaire à la séquence d'*obsel*s d'une $\mathcal{M}$-Trace. Ainsi, le cas d'un WF-log diffère principalement d'une $\mathcal{M}$-Trace par l'absence de modèle associé à la séquence d'*obsel*s. Une autre différence porte sur la représentation du temps. En fonction du type d'algorithme utilisé, la représentation du temps dans un WF-log peut se réduire à la notion de séquentialité. C'est le cas de l'algorithme α et de ses variantes. Nous allons montrer qu'un WF-log peut se traduire par un ensemble de $\mathcal{M}$-Traces.

Théorème 1 *Un* WF-log *peut être représenté par un ensemble de $\mathcal{M}$-Traces partageant le même modèle de trace. Chaque* cas *d'un WF-log se traduit alors par une $\mathcal{M}$-Trace.*

[1] Une « action » est appelée « tâche » (*task*) ou « activité » (*activity*) dans le vocabulaire de la fouille de workflow

Pour convertir un WF-log en $\mathcal{M}$-Trace, il faut proposer un modèle de trace et une conversion de la temporalité. Nous proposons de manière naturelle d'associer les événements du WF-log aux *obsel*s de la $\mathcal{M}$-Trace. En ce qui concerne le modèle, comme chaque action est représentée par un type d'événement dans le WF-log, chaque action est représentée par un type d'*obsel*s dans le modèle de la $\mathcal{M}$-Trace.

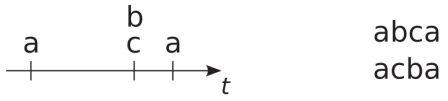
En ce qui concerne le temps, la transformation est immédiate si le WF-log se base sur une représentation temporelle. Si le WF-log s'appuie sur une représentation séquentielle, on associe de manière incrémentale une unité de temps à chaque *obsel* de la trace, en fonction de son ordre d'apparition.

La conversion d'un WF-log à un ensemble de $\mathcal{M}$-Traces se fait sans perte d'informations. Un ensemble de $\mathcal{M}$-Traces contient donc plus d'informations qu'un WF-log. Il est donc possible d'appliquer l'algorithme α ou tout autre algorithme d'analyse de WF-log sur un ensemble de $\mathcal{M}$-Traces.

Pour pouvoir exploiter, sur des $\mathcal{M}$-Traces, les algorithmes existants d'analyse de WF-log, il faut disposer d'un moyen de convertir un ensemble de $\mathcal{M}$-Traces en un WF-log. Les algorithmes présentés dans la partie 4 s'appuient sur des représentations des WF-log qui ne prennent en compte que l'ordre d'apparition (ou le temps) et le type des événements.

La conversion d'un ensemble de $\mathcal{M}$-Traces homogènes (partageant le même modèle) à un WF-log consiste : au niveau du modèle, à ne garder du modèle de trace que la liste des types d'*obsel*s présents dans les traces et, au niveau de chaque trace, de ne conserver pour chaque *obsel* que son type et son estampille temporelle.

Dans le cas d'un WF-log utilisant une représentation séquentielle du temps, il est nécessaire de transformer la représentation temporelle en représentation séquentielle. Cette conversion est a priori triviale : plutôt que d'associer à chaque événement une estampille temporelle, on lui associe un ordre d'apparition.



(a) Représentation temporelle (b) Représentation séquentielle

FIGURE 5.1 – Différence entre représentation temporelle et représentation séquentielle des séquences d'*obsel*s.

La conversion pose néanmoins problème dans le cas où des événements possèdent une estampille temporelle identique, puisque la relation d'ordre séquentiel des événements au sein d'une séquence d'un WF-log doit être totale. La figure 5.1 illustre le problème qui relève de la préparation des données. Une stratégie doit être adoptée au niveau de la conversion des $\mathcal{M}$-Traces en séquences : filtrer les $\mathcal{M}$-Traces, décider d'un ordre ou construire autant de séquences que d'ordres possibles.

5.2 Rendre interactive la fouille de workflow – le cas de l'algorithme α

Nous nous focalisons dans un premier temps sur l'algorithme α, algorithme dont il existe de nombreuses variantes. Dans cette section, nous verrons plus en détail le principe de fonctionnement de l'algorithme α. Puis, nous proposerons de rendre interactive une étape clé de l'algorithme, afin de permettre à l'expert de compléter (voire corriger) les connaissances déduites des traces. Nous verrons dans la section suivante comment généraliser cette interaction aux versions améliorées de l'algorithme α.

5.2.1 L'algorithme α : définitions et principe de fonctionnement

L'algorithme α (van der Aalst *et al.*, 2002, 2004) a pour objectif de découvrir un modèle de workflow à partir d'un ensemble de traces d'exécution de ce même workflow.

WF-nets, WF-logs et notations

L'algorithme α produit des *workflow-nets* (WF-nets) pour représenter le modèle du processus observé. Un WF-net est un réseau de Petri particulier. Nous donnons ici une définition formelle d'un réseau de Petri afin d'introduire les notations utilisées au long de cette thèse (définition 1).

Définition 1 (réseau de Petri) *Un réseau de Petri est un triplet (P, T, F) tel quel :*

1. *P est un ensemble fini de* places*,*
2. *T est un ensemble fini de* transitions*, disjoint de l'ensemble de places ($P \cap T = \emptyset$),*
3. *$F \subseteq (P \times T) \cup (T \times P)$ est un ensemble d'arcs orientés.*

Dans la suite, la lettre P représente un ensemble de places, la lettre T représente un ensemble de transitions et la lettre F représente un ensemble d'arcs. Un WF-net est un réseau de Petri répondant à certaines contraintes structurelles [2]. Un WF-net possède en particulier une place d'entrée, représentée par la lettre i (comme *input*) et une place de sortie, représentée par la lettre o (comme *output*). La définition formelle d'un WF-net est donnée à l'annexe A (p. 221). La figure 5.2 donne un exemple de WF-net.

Nous avons vu qu'un WF-log est un ensemble de séquences représentant l'exécution complète d'un processus. La définition formelle d'un WF-log est présentée à l'annexe A (p. 221). Nous reproduisons ici les notations utilisées. Un WF-log est représenté par la lettre W, une séquence d'un WF-log par la lettre σ.

La notation $\sigma = t_1 \dots t_n$ représente une séquence de n événements dont le premier événement est de type t_1, le deuxième événement de type t_2 et ainsi de suite. Les fonctions $first()$ et $last()$ renvoient respectivement le premier et le dernier événement de la séquence : $first(\sigma) = t_1$ et $last(\sigma) = t_n$.

Pour un type d'événement t et une séquence $\sigma = t_1 \dots t_n$, la notation $t \in \sigma$ signifie qu'un événement de type t est observé dans la séquence σ (c.-à-d. $\exists i \in [\![1; n]\!] : t = t_i$).

[2] 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.

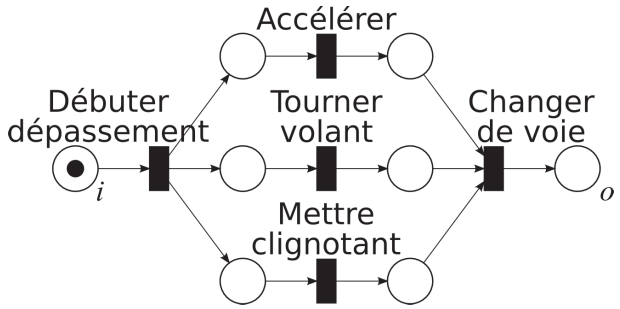


FIGURE 5.2 – Un WF-net avec 3 transitions parallèles. Les cercles représentent les places ; le marquage des places caractérise l'état du réseau (ici le réseau est dans son état initial). Les rectangles sont les transitions ; une transition représente une action du workflow.

Dans la fouille de workflow, les actions du WF-log sont représentées par des événements dans les séquences du WF-log et par des transitions du modèle WF-net. Le problème de synthèse d'un réseau de Petri consiste donc principalement en la construction des places et des arcs du réseau de Petri.

L'algorithme α

L'algorithme α (van der Aalst *et al.*, 2002, 2004) 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.

– étape 5. Construction de Y_W, sous-ensemble de X_W, 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 $\alpha(W)$, 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* » ;
- $a \#_W b$ 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 $>_W$.

Réécriture de l'algorithme α

Nous proposons de réécrire l'algorithme α pour prendre en compte toutes les étapes et clarifier les paramètres dont dépend chaque ensemble. Soit W un WF-log d'un processus observé $\mathcal{P}_{obs}$. Le modèle reconstruit $\mathcal{M}_{proc}$ du processus $\mathcal{P}_{obs}$ est calculé par l'algorithme $\alpha : \mathcal{M}_{proc} = \alpha(W)$.

1. $T(W) = \{t | \exists \sigma \in W,\ t \in \sigma\}$,
2. $T_I(W) = \{t \mid \exists \sigma \in W,\ t = \mathit{first}(\sigma)\}$,
3. $T_O(W) = \{t \mid \exists \sigma \in W,\ t = \mathit{last}(\sigma)\}$,

3′. a) $\forall a, b \in T(W), a >_W b \Leftrightarrow \exists \sigma = t_1 \dots t_n \in W, i \in [\![1; n-1]\!] : t_i = a$ et $t_{i+1} = b$,
   b) $\forall a, b \in T(W), a \rightarrow_W b \Leftrightarrow a >_W b$ et $b \not>_W a$,
   c) $\forall a, b \in T(W), a \#_W b \Leftrightarrow a \not>_W b$ et $b \not>_W a$,

4. $X(W) = \{(A, B) \mid A \subseteq T(W) \wedge B \subseteq T(W) \wedge \forall (a, b) \in A \times B : a \rightarrow_W b \wedge \forall a_1, a_2 \in A : a_1 \#_W a_2 \wedge \forall b_1, b_2 \in B : b_1 \#_W b_2\}$,
5. $Y(W) = \{(A, B) \in X(W) \mid \forall (A', B') \in X(W) :\ A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$,
6. $P(W) = \{p(A, B) \mid (A, B) \in Y(W)\} \cup \{i_w, o_w\}$[3],
7. $F(W) = \{(a, p(A, B)) \mid (A, B) \in Y(W) \wedge a \in A\} \cup \{(p(A, B), b) \mid (A, B) \in Y(W) \wedge b \in B\} \cup \{(i_w, t) \mid t \in T_I(W)\} \cup \{(t, o_w) \mid t \in T_O(W)\}$,
8. $\alpha(W) = (P(W), T(W), F(W))$,
9. $\mathcal{M}_{proc} = \alpha(W)$.

Van der Aalst *et al.* (2002, 2004) 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]Nous rappelons que i désigne la place d'entrée du WF-log, et o la place de sortie.

5.2.2 Comment interagir avec 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).

Introduire l'interaction

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[4]. 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 synthèse de réseau de Petri :

- Avant d'effectuer la synthèse, via le SBT ABSTRACT : en *préparant* les traces, en *ajoutant des traces-exemples*, ou encore en donnant des *exemples négatifs*.
- Pendant la synthèse : sans changer l'algorithme, en *éditant* des résultats intermédiaires de l'algorithme de synthèse, 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.
- Après la synthèse : 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.

Nous nous focalisons sur les interactions qui ont un impact direct sur l'algorithme de synthèse de réseau de Petri : les interactions pendant la synthèse. Les interactions qui impactent le principe de fonctionnement de l'algorithme sont celles liées à la visualisation et l'édition de résultats intermédiaires de l'algorithme. Nous présentons donc les différents résultats intermédiaires sur lesquels l'analyste pourrait interagir.

Les étapes de l'algorithme α

En décomposant l'algorithme α pour étudier toutes les étapes et leurs calculs intermédiaires, nous pouvons étudier un par un les éléments sur lesquels il est possible d'interagir. La figure 5.3 présente l'enchaînement des différentes étapes de l'algorithme α et permet donc d'expliciter l'interdépendance entre les étapes.

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

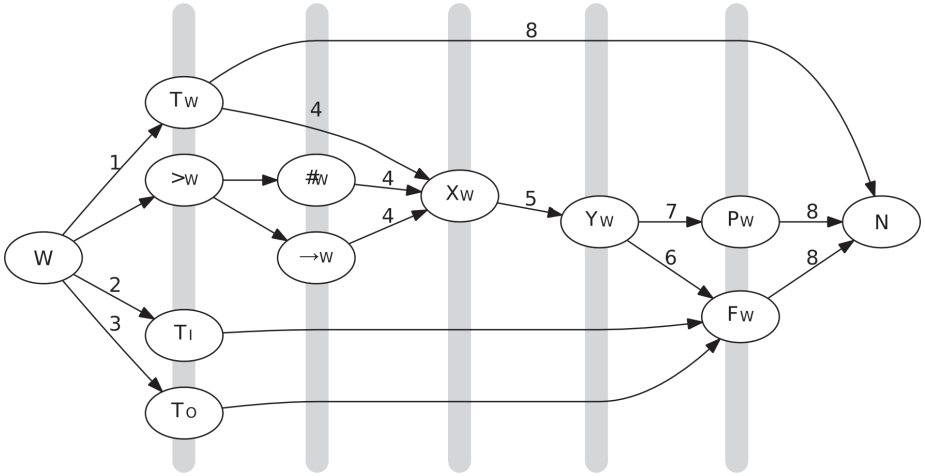


FIGURE 5.3 – Enchaînement des étapes de l'algorithme α. Ce schéma reprend les 8 étapes de l'algorithme α et montre les interdépendances entre ces étapes. Les lignes verticales matérialisent l'évolution à chaque étape de l'ensemble des résultats intermédiaires.

Ces étapes se regroupent en trois catégories : caractérisation des transitions (T_W, T_I et T_O), calcul de relations entre transitions ($>_W$, $\#_W$ et $\rightarrow_W$) et calcul de places candidates, représentées par les arcs qui les connectent aux transitions (X_W, Y_W, P_W et F_W). Pour chacun de ces résultats intermédiaires, nous allons étudier sa signification, en quoi il est possible de le modifier et en quoi cette modification impacte le reste de l'algorithme.

Caractérisation des transitions

L'ensemble des transitions. L'ensemble des transitions T_W est calculé à partir du WF-log. Une transition sera présente dans T_W si et seulement si un événement de ce type a été observé dans au moins une des traces du WF-log. Si une transition t n'est pas observée dans le WF-log, cela signifie qu'il sera impossible pour l'algorithme de calculer la relation $>_W$ (et les autres relations en découlant). Modifier l'ensemble T_W nécessite de corriger manuellement la relation T_W et peut avoir un impact aussi sur les ensembles de transitions initiales et finales.

Plutôt que de modifier l'ensemble T_W, il faut agir en amont, au niveau de la préparation des traces.

Les transitions initiales et finales. Les transitions initiales et finales, $T_I(W)$ et $T_O(W)$, sont aussi calculées à partir du WF-log. Le calcul est simple : une transition est une transition initiale, respectivement finale, si un événement du même type est le premier événement, respectivement dernier événement, d'une des traces du WF-log. Une transition initiale ou finale est nécessairement incluse dans l'ensemble T_W.

Pour respecter sa sémantique de transition initiale ou finale, une transition doit respecter certaines contraintes au niveau de la relation $>_W$. Une transition initiale t_i ne peut logiquement pas être précédée d'une autre transition : $\forall t \in T_W : t \not>_W t_i$. De

même une transition finale t_f ne peut pas être suivie d'autres transitions : $\forall t \in T_W$: $t_f \not>_W t$.[5]

Ajouter une transition de T_W dans l'ensemble de transitions initiales ou finales, ou supprimer une transition d'un de ces deux ensembles nécessite donc de vérifier la cohérence vis-à-vis de la relation $>_W$. Notons que la présence de bruit dans les traces peut engendrer ces mêmes incohérences. Notons que ces incohérences pourraient être facilement détectées et présentées à l'analyste.

En bref, modifier l'ensemble des transitions initiales T_I ou l'ensemble des transitions finales T_O peut se faire avec relativement peu d'impact sur les autres étapes de calcul de l'algorithme : il faut simplement ajuster la relation $>_W$ d'une manière équivalente à ce qu'il est nécessaire de faire pour des données bruitées.

Calcul de relations entre transitions

Le calcul des relations entre transitions (relations $>_W$, $\rightarrow_W$ et $\#_W$) n'est pas explicité dans les étapes de l'algorithme α tel qu'il est présenté dans la littérature. Pourtant, il est pertinent de présenter ces relations à l'analyste pour une éventuelle interaction.

La relation de succession $>_W$. La relation $>_W$ sert de base au calcul des relations $\rightarrow_W$ et $\#_W$ à partir desquelles les places candidates sont calculées. La relation $>_W$ est donc un résultat intermédiaire clé dans le fonctionnement de l'algorithme α.

La relation $a >_W b$ signifie, pour deux types d'événements a et b : « il existe une trace où un événement de type a est directement suivi d'un événement de type b ».

Dans le cas de traces incomplètes ou bruitées, il se peut que l'inférence de la succession ou non de deux types d'événements soit erronée : a est directement suivi de b dans une trace alors que cela ne s'est pas réellement produit ou à l'inverse on n'a jamais observé a directement suivi de b dans le jeu de données, alors que l'on sait que c'est possible.

En réalité, la sémantique recherchée de la relation $a >_W b$ est la suivante : « il est possible qu'une trace où a est directement suivi de b soit produite par le modèle recherché. » Cette formulation en langage naturel est facilement accessible par l'analyste. Dans le cas de traces complètes, non bruitées, quelle que soit l'interprétation, le résultat est le même. Dans le cas de traces incomplètes ou bruitées, il est nécessaire d'intervenir sur la relation $>_W$ pour répondre à cette deuxième interprétation que l'on ne peut pas, dans ce cas, inférer à partir des données seules.

Le fait d'ajouter une relation $a >_W b$ qui n'a pas été observée correspond à ajouter une information qui n'a pas été observée dans les traces. Le fait d'invalider une relation $>_W$ revient à dire que l'ensemble de traces qui a permis d'inférer $a >_W b$ est incomplet ou que les traces sont bruitées : la relation de succession observée n'a été observée qu'à cause de cette incomplétude des données.

La relation $>_W$ est le résultat intermédiaire clé sur lequel intervenir pour gérer les problèmes de bruit et d'incomplétude des données.

En bref, la sémantique de la relation $>_W$ est facilement accessible à l'analyste à travers sa formulation en langage naturel. Ce genre d'information sur la succession possible ou non de deux types d'événements est en outre une connaissance dont l'analyste peut disposer a priori. L'interaction au niveau de la relation $>_W$ est donc pertinente.

[5] La démonstration formelle est relativement triviale. Si une telle transition existait, alors le modèle découvert ne serait pas conforme.

Les relations de causalité $\rightarrow_W$ et d'indépendance $\#_W$. Les relations de causalité $\rightarrow_W$ et d'indépendance $\#_W$ se calculent directement à partir de la relation $>_W$. $a \rightarrow_W b$ 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 qu'il existe une troisième relation, la relation de parallélisme notée $||_W$, mais elle n'est pas exploitée par l'algorithme. Les relations $\rightarrow_W$, $\#_W$ et $||_W$ sont mutuellement exclusives. Il n'est donc pas possible de modifier l'une indépendamment de l'autre relation.

La relation $\rightarrow_W$ est appelée relation de causalité (*causal relation*) par van der Aalst *et al.* (2004), car elle détermine un ordre entre deux transitions et cette relation d'ordre est interprétée comme une relation de causalité.

Au niveau de la construction des places candidates, l'algorithme α s'appuie sur la relation de causalité pour déterminer les places candidates et le sens des arcs connectant places et transitions. L'absence de relation $a \rightarrow_W b$ entre deux transitions a et b garantit qu'il n'y a pas de place connectant a et b. Inversement, ajouter une relation de causalité $a \rightarrow_W b$ aura comme conséquence d'ajouter au moins une place connectant directement la transition a à la transition b. Modifier les relations de causalité a donc un impact direct sur le réseau de Petri produit par l'algorithme α.

L'algorithme α s'appuie sur la relation d'indépendance $\#_W$ pour regrouper les entrées et sorties de chaque place candidate. L'impact de la relation d'indépendance est donc moins direct que celui de la relation de causalité. Ajouter (respectivement supprimer) une relation d'indépendance peut n'avoir aucune conséquence sur le réseau final ou bien ajouter (resp. supprimer) des places candidates regroupant plus d'entrées ou de sorties.

Comme les relations de causalité et d'indépendance sont mutuellement exclusives, changer le type de relation pour un couple de transitions peut avoir un double impact. Modifier de manière conjointe les relations de causalité et d'indépendance revient, au niveau de l'algorithme à modifier la relation de succession seule.

Calcul des places candidates

Ensemble des places candidates X_W. Le cœur de l'algorithme α réside dans son calcul des places candidates. Ce calcul se fait au niveau du calcul de l'ensemble X_W. Cet ensemble caractérise des places candidates pour construire le réseau de Petri. Chaque élément de cet ensemble est un couple dont le premier élément est l'ensemble des transitions d'entrée de la place candidate et le second élément est l'ensemble des transitions de sortie de la place candidate. Il ne s'agit pas d'un ensemble de places à proprement parler, mais d'un ensemble caractérisant les places candidates.[6]

Ces places candidates sont calculées à partir de la relation de causalité et de la relation d'indépendance. Une relation de causalité entre deux transitions crée au moins une place candidate entre ces deux transitions. La relation d'indépendance, quant à elle, va servir à proposer des places candidates regroupant des transitions en entrée et en sortie.

Ajouter une place candidate revient à construire manuellement un ensemble de transitions d'entrée et de transitions de sortie que l'on ajoute à l'ensemble de places candidates. Cela ne signifie pas que cette place sera présente dans le réseau de Petri final, car l'étape suivante consiste à optimiser le réseau de Petri en supprimant les

[6]Par abus de langage, nous présentons l'ensemble X_W, et plus loin l'ensemble Y_W, comme des ensembles de places candidates.

places redondantes. Soit cette place ajoutée est implicite, c'est-à-dire elle n'ajoute pas d'information au réseau de Petri final, et elle n'apparaîtra pas dans le réseau final. Soit cette place ajoutée n'est pas implicite, ce qui signifie que l'ajout de cette place rend une ou plusieurs autres places implicites : la place ajoutée sera gardée dans le réseau de Petri final et les places devenues implicites n'apparaîtront pas dans le réseau final.

À l'inverse, supprimer une place candidate peut très bien n'avoir aucun effet sur le réseau de Petri final, si c'est une place implicite ; ou au contraire, cela peut modifier le réseau en faisant disparaître cette place et apparaître une ou plusieurs autres places initialement implicites.

En bref, modifier l'ensemble de places candidates peut n'avoir aucun effet s'il s'agit de places implicites ou modifier la structure du réseau de Petri final.

Ensembles Y_W, P_W et F_W. Comme l'ensemble X_W, l'ensemble Y_W ne représente pas à proprement parler des places candidates, mais il les caractérise.

Cet ensemble Y_W est une optimisation des places candidates : Y_W est obtenu en ne conservant de l'ensemble X_W que les places qui ne sont pas implicites. L'ensemble Y_W caractérise exactement l'ensemble des places qui seront présentes dans le réseau de Petri final.

P_W décrit, à proprement parler, l'ensemble des places du réseau de Petri et F_W les arcs du réseau de Petri. L'ensemble des places P_W est déduit directement de l'ensemble Y_W. L'ensemble des arcs F_W est déduit directement de l'ensemble Y_W pour les arcs à l'intérieur du réseau et des ensembles de transitions initiales et finales T_I et T_O pour les arcs reliant la place initiale aux transitions initiales et les transitions finales à la place finale.

En bref, les ensembles Y_W, P_W et F_W caractérisent le réseau de Petri final. Mais contrairement au réseau de Petri, elles ne disposent pas de représentation graphique permettant à l'analyste de les interpréter. Plutôt que de modifier ces ensembles, il vaut mieux modifier directement le réseau de Petri final.

Résultats intermédiaires : Bilan

Le tableau 5.1 présente l'ensemble des résultats intermédiaires et leurs caractéristiques en terme de sémantique et d'impact sur le reste de l'algorithme et sur le réseau de Petri final.

L'ensemble T_W ne se prête pas à l'interaction. Éditer les relations correspondant à une transition non observée est un problème difficile. Ce problème requiert de l'analyste qu'il puisse définir précisément l'action en question, alors qu'annoter les traces pour ajouter cette action est moins contraignant. Supprimer une transition de l'ensemble T_W nécessite d'éditer la relation $>_W$ alors qu'un simple filtrage des traces produit le même résultat. C'est au niveau de la préparation des traces qu'il faut gérer les problèmes de transitions absentes ou superflues.

Les ensembles de transitions d'entrée et de sortie T_I et T_O sont conformes à l'intuition et peuvent être modifiés, dans la mesure où certaines contraintes sont respectées dans la suite du déroulement de l'algorithme.

La relation de succession $>_W$ a une sémantique claire et simple que l'on peut présenter à l'analyste. L'algorithme α s'appuie sur cette relation pour tous les calculs ultérieurs. Cette relation est donc un candidat idéal pour l'interaction avec l'analyste.

Les relations de causalité $\rightarrow_W$ et d'indépendance $\#_W$ sont aussi interprétables par l'analyste, mais leur signification est plus ambiguë. La notion de causalité peut répondre à différentes définitions en fonction du contexte. Contrairement à la relation de

TABLE 5.1 – Synthèse des caractéristiques, pour l'interaction, des résultats intermédiaires de l'algorithme α.

Résultat intermédiaire	Compréhensible par l'analyste	Impacte le réseau de Petri final	Modifiable indépendamment des autres étapes
T_W	++	+	-
T_I et T_O	++	+	+/-
$>_W$	++	++	+/-
$\#_W$	+	+	-
$\rightarrow_W$	+	+	-
$(\#_W, \rightarrow_W)$	+	++	+/-
X_W	-	++	+
Y_W, P_W et F_W	-	++	++

+ : oui. ++ : oui, indéniablement. +/- : cela dépend. - : non.

succession, dont la sémantique relève de l'observation du processus, ces deux relations ont une sémantique de plus haut niveau, plus proche du but recherché : modéliser le processus. Enfin, ces deux relations sont mutuellement exclusives, modifier l'une peut donc impacter l'autre.

Les ensembles caractérisant les places relèvent du résultat final. Interagir sur ces résultats intermédiaires nous semble plus coûteux que de modifier directement le réseau de Petri final.

Sur l'ensemble des résultats intermédiaires, ceux qui nous semblent le plus à même de faire sens pour l'analyste sont les *ensembles de transitions* et les *relations*. Les relations font sens pour un analyste et sont la base sur laquelle s'appuie l'algorithme. Parmi les différentes relations, la relation de succession $>_W$ est celle dont la signification est la plus immédiate pour l'analyste.

Dans la suite, nous n'utilisons plus l'indice $_W$ pour alléger la notation. En outre, l'interaction introduit une autre source d'information. Tout n'est plus déduit du WF-log seulement : les connaissances peuvent aussi venir de l'analyste.

5.2.3 L'algorithme α_i – une version interactive de l'algorithme α

Réécrire l'algorithme α

Nous proposons de restructurer l'algorithme α pour le rendre interactif. Nous appelons cet algorithme modifié algorithme α_i (« i » pour interactif).[7] Pour cela, nous allons créer deux fonctions : *analyseLogs_NSL()* et *buildPlaces()*. La fonction *analyseLogs_NSL()*[8] regroupe les étapes d'analyse du WF-log, c'est-à-dire les étapes 1 à 3 de l'algorithme α. La fonction *buildPlaces()* regroupe les étapes de construction de places et des arcs, c'est-à-dire les étapes 4 à 7. La figure 5.4 représente de manière schéma-

[7] La façon dont nous présentons ici l'algorithme α_i est différente de nos précédentes publications (Mathern *et al.*, 2010b, 2011a). Les deux écritures sont équivalentes. Celle que nous présentons dans cette thèse a pour objectif de faciliter la généralisation aux autres algorithmes de la famille α.

[8] NSL pour « sans boucles courtes » (*No Short Loops*).

tique la version restructurée de l'algorithme α. Cette reformulation rend explicites les résultats intermédiaires sur lesquels il est possible d'interagir.

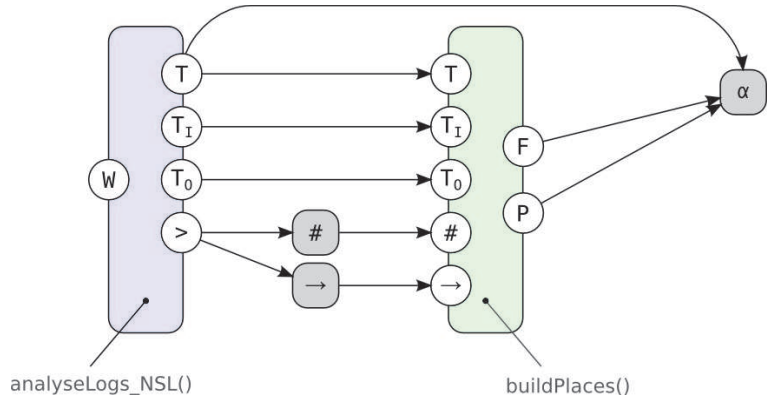


FIGURE 5.4 – Algorithme α_i. Proposition de regroupement des étapes de calcul de l'algorithme α en utilisant deux fonctions : *analyseLogs_NSL()* et *buildPlaces()*.
La fonction *analyseLogs_NSL()* effectue les calculs des étapes 1, 2, 3 de l'algorithme α et le calcul de la relation $>$. La fonction *buildPlaces()* effectue les calculs des étapes 4, 5, 6 et 7 de l'algorithme α.

Avec les fonctions *analyseLogs_NSL()* et *buildPlaces()*, la quasi-totalité de l'algorithme α est réécrite. Il ne manque plus qu'à expliciter le calcul des relations de causalité ($\rightarrow$) et d'indépendance ($\#$) à partir de la relation de succession ($>$). Pour cela, nous définissons deux fonctions, la fonction *buildDependencyRelation_NSL* et la fonction *buildUnrelatedRelation*.

Définition 2 (fonction *analyseLogs_NSL()*) *Soit W un WF-log. La fonction $analyseLogs\_NSL(W)$ se calcule de la façon suivante :*

1. $T = \{t | \ \exists \sigma \in W,\ t \in \sigma\}$,
2. $T_I = \{t \mid \exists \sigma \in W,\ t = first(\sigma)\}$,
3. $T_O = \{t \mid \exists \sigma \in W,\ t = last(\sigma)\}$,
4. $\forall a, b \in T, a > b \Leftrightarrow \exists \sigma = t_1 \dots t_n \in W, i \in [\![1; n-1]\!] : t_i = a$ *et* $t_{i+1} = b$,
5. $analyseLogs\_NSL(W) = (T, T_I, T_O, >)$.

La fonction *analyseLogs_NSL()* effectue les calculs des étapes 1, 2, 3 de l'algorithme α et le calcul de la relation $>$.

Définition 3 (fonction *buildDependencyRelation_NSL()*) *Soit T un ensemble de transitions et $>$ une relation de succession définie sur cet ensemble T. La fonction $buildDependencyRelation\_NSL(T, >)$ construit la relation $\rightarrow$, définie de la manière suivante :*

$$\forall a, b \in T : a \rightarrow b \Leftrightarrow a > b \wedge b \not> a$$

La fonction *buildDependencyRelation_NSL()* effectue le calcul de la relation $\rightarrow$.

Définition 4 (fonction *buildUnrelatedRelation()*) *Soit T un ensemble de transitions et $>$ une relation de succession définie sur cet ensemble T. Alors, la fonction $buildUnrelatedRelation(T, >)$ construit la relation $\#$, définie de la manière suivante :*

$$\forall a, b \in T : a \# b \Leftrightarrow a \not> b \wedge b \not> a$$

La fonction *buildUnrelatedRelation()* effectue le calcul de la relation $\#$.

Définition 5 (fonction *buildPlaces()*) *Soit T un ensemble de transitions, T_I et T_O deux sous-ensembles de T et $\rightarrow$ et $\#$ deux relations définies sur les transitions de T. T_I représente l'ensemble des transitions initiales, T_O l'ensemble des transitions finales, $\rightarrow$ la relation de causalité et $\#$ la relation d'indépendance.*
La fonction $buildPlaces(T, T_I, T_O, \rightarrow, \#)$ se calcule de la façon suivante :

1. $X = \{(A, B) \mid A \subseteq T \wedge B \subseteq T \wedge \forall (a, b) \in A \times B : a \rightarrow b \wedge \forall a_1, a_2 \in A : a_1 \# a_2 \wedge \forall b_1, b_2 \in B : b_1 \# b_2\}$,
2. $Y = \{(A, B) \in X \mid \forall (A', B') \in X : A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$,
3. $P = \{p(A, B) \mid (A, B) \in Y\} \cup \{i, o\}$,
4. $F = \{(a, p(A, B)) \mid (A, B) \in Y \wedge a \in A\} \cup \{(p(A, B), b) \mid (A, B) \in Y \wedge b \in B\} \cup \{(i, t) \mid t \in T_I\} \cup \{(t, o) \mid t \in T_O\}$,
5. $buildPlaces(T, T_I, T_O, \rightarrow, \#) = (P, F)$.

La fonction *buildPlaces()* effectue les calculs des étapes 4, 5, 6 et 7 de l'algorithme α.

Avec ces quatre fonctions, *analyseLogs_NSL()*, *buildDependencyRelation_NSL*, *buildUnrelatedRelation* et *buildPlaces()*, il est possible de totalement réécrire l'algorithme α. Toutes les étapes de calcul sont identiques à l'algorithme α tel qu'il est décrit dans la littérature (van der Aalst *et al.*, 2004). Nous appelons cet algorithme l'algorithme α_i, car il s'agit d'une réécriture de l'algorithme α permettant de mieux rendre compte des possibilités d'interaction sur des résultats intermédiaires.

Définition 6 (algorithme α_i) *Soit W un WF-log. L'algorithme α se déroule selon les étapes suivantes :*

1. $(T, T_I, T_O, >) := analyseLogs\_NSL(W)$,
2. $\rightarrow := buildDependencyRelation\_NSL(T, >)$,
3. $\# := buildUnrelatedRelation(T, >)$,
4. $(P, F) := buildPlaces(T, T_I, T_O, \rightarrow, \#)$,
5. $\alpha_i(W) = (P, T, F)$.

Sans interactions sur les résultats intermédiaires, l'algorithme α_i est identique à l'algorithme α, car toutes les étapes de calcul sont identiques. L'algorithme α_i dispose donc des mêmes propriétés que l'algorithme α. En particulier, l'algorithme α_i est capable de découvrir le modèle d'un workflow structuré, conforme et sans boucles courtes à partir d'un WF-log complet de ce workflow.

Interagir avec l'algorithme α_i

Nous proposons de présenter cette relation $>$ 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. Cela conduit à définir une nouvelle relation, notée $>_{\mathcal{A}}$ ($\mathcal{A}$ pour « analyste »), de manière à la différencier de celle calculée uniquement à partir des traces. Cette relation, issue de l'interaction, permet de rendre l'algorithme α_i utilisable dans le cas où le jeu de traces est incomplet. La figure 5.5 présente l'introduction de cette étape d'interaction dans l'algorithme α_i.

Intervention de l'analyste ($\mathcal{A}$) sur un résultat intermédiaire de l'algorithme α_i :

1. $(T, T_I, T_O, >) := analyseLogs\_NSL(W)$,
2. **Interaction :**

$$\text{Connaissances}_{\mathcal{A}} + > \quad \xrightarrow{\mathcal{A}} \quad >_{\mathcal{A}}$$

3. $\rightarrow := buildDependencyRelation\_NSL(T, >_{\mathcal{A}})$,
4. $\# := buildUnrelatedRelation(T, >_{\mathcal{A}})$,
5. $(P, F) := buildPlaces(T, T_I, T_O, \rightarrow, \#)$,
6. $\alpha_i(W) = (P, T, F)$.

FIGURE 5.5 – Intervention de l'analyste dans l'algorithme α_i. L'analyste combine ses connaissances (provenant de la littérature, de son expertise et de son interprétation des données) pour vérifier, compléter ou corriger les calculs effectués par l'algorithme.

Exemple illustratif

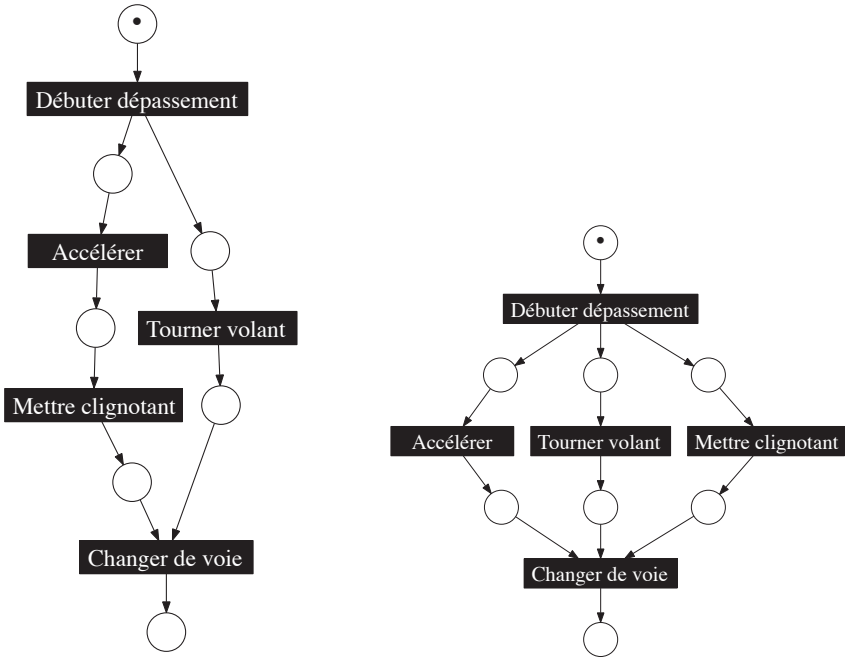
Nous illustrons notre approche sur un exemple simple. À partir du WF-net présenté à la figure 5.2, nous produisons l'ensemble de traces présenté à la table 5.2. Cet ensemble de traces est incomplet au regard du modèle donné.

TABLE 5.2 – Trois traces, produites à partir du WF-net présenté à la figure 5.2, formant un WF-log incomplet (au regard du modèle). Chaque trace est constituée d'une séquence d'événements. Les événements sont séparés par des virgules.

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.

Comme attendu, en utilisant l'algorithme α sur un jeu de traces incomplet, le résultat (figure 5.6a) 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 5.7), l'algorithme est capable de redécouvrir le WF-net original (figure 5.6b, équivalent au WF-net de la figure 5.2).



(a) Algorithme α (pas d'interaction)

(b) Algorithme α_i, avec interaction

FIGURE 5.6 – WF-nets produits à partir du jeu de traces incomplet présenté à la table 5.2 : algorithme α sans interaction (5.6a) et algorithme α_i avec interaction (5.6b).

L'algorithme α_i est une reformulation de l'algorithme α. Cette reformulation donne des points d'entrée à l'analyste pour interagir sur des résultats intermédiaires. Nous proposons d'interagir au niveau de la relation de succession $>$ qui s'exprime aisément en langage naturel.

Cependant, l'algorithme α_i hérite des mêmes limitations que l'algorithme α : même s'il est possible de contrôler le problème de la complétude du WF-log en entrée, il ne pourra découvrir que des modèles de type WF-log structuré, conforme et sans boucles courtes. Pour dépasser la limitation sur le type de modèles découverts par l'algorithme α_i, il faut adapter l'interaction à d'autres algorithmes de la famille α proposés dans la littérature pour étendre les capacités de l'algorithme α.

5.3 Généraliser l'interaction à d'autres algorithmes

Dans cette section nous allons montrer comment généraliser la formulation interactive de l'algorithme α à d'autres algorithmes de la famille α. Nous allons présenter le détail de ces modifications pour l'algorithme α^+ (de Medeiros *et al.*, 2005), qui permet

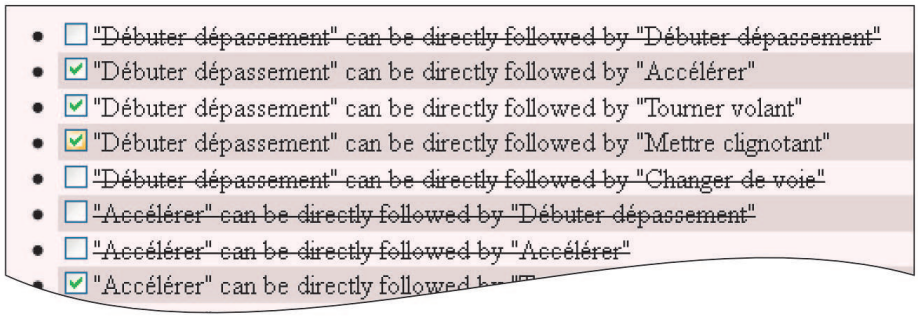


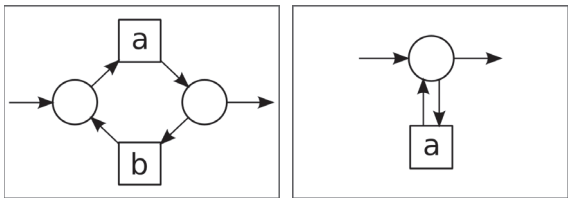
FIGURE 5.7 – Prototype d'interface présentée à l'analyste pour interagir avec l'algorithme α_i. L'analyste peut introduire ses connaissances au niveau des résultats intermédiaires (relation $>_{\mathcal{A}}$). Par exemple, l'analyste a indiqué au système que l'événement « Débuter dépassement » peut être directement suivi de l'événement « Mettre clignotant », alors que cette information n'a pas été observée dans les traces.

de gérer les boucles courtes, nécessaires à notre application. Puis, nous présenterons comment généraliser cette approche aux autres algorithmes de la famille α.

5.3.1 L'algorithme α_i^+ : interaction et boucles courtes

De Medeiros *et al.* (2004a, 2005) ont proposé l'algorithme α^+, une extension de l'algorithme α qui permet de gérer les boucles courtes. Contrairement à l'algorithme α, l'algorithme α^+ est capable de découvrir correctement tout type de WF-net structuré et conforme. Les boucles que l'algorithme α n'arrive pas à identifier sont de deux types (voir figure 5.8) : les boucles de longueur deux (deux transitions qui peuvent boucler l'une avec l'autre) et les boucles de longueur un (une transition qui peut se répéter).

Pour gérer ces deux types de boucles, de Medeiros *et al.* modifient l'algorithme α en deux étapes : l'algorithme α'[9], qui gère les boucles de longueur deux, puis l'algorithme α^+, qui s'appuie sur l'algorithme α' qui gère les boucles de longueur deux et de longueur une.



(a) Boucle de longueur 2 (b) Boucle de longueur 1

FIGURE 5.8 – Les boucles de longueur 2 (fig. 5.8a) et les boucles de longueur 1 (fig. 5.8b) ne sont pas gérées par l'algorithme α ; l'algorithme α^+ permet de les prendre en compte.

[9] De Medeiros *et al.* appellent cet algorithme l'« algorithme α ». Nous lui donnons le nom d'« algorithme α' » pour le différencier de l'algorithme α original qui ne gère pas les boucles de longueur 2.

Algorithme α'_i et boucles de longueur deux

Par rapport à l'algorithme α, la gestion des boucles de longueur 2 se fait par le calcul d'une relation supplémentaire : la relation $\triangle_W$. Cette relation identifie les types d'événements (actions de l'activité) impliqués dans des boucles de longueur 2. Pour identifier une boucle de longueur 2, il faut donc observer au moins une fois la répétition de deux types d'événements dans une trace.

Pour a et b deux types d'événements, la relation a $\triangle_W b$ signifie qu'il existe une trace σ dans le WF-log W telle que $\sigma = ...aba...$. Cela se traduit en langage naturel par « il existe au moins une trace où l'on observe un événement de type a suivi directement d'un événement de type b qui est à son tour suivi directement d'un événement de type a. »

Dans la suite de l'algorithme α', la relation $\triangle_W$ est combinée à la relation de succession $>_W$ pour inférer la relation de dépendance $\rightarrow'_W$[10]. Le reste de l'algorithme α est inchangé. Cette simple modification permet d'intégrer la gestion des boucles de longueur deux.

Définition 7 (Relation de causalité $\rightarrow'_W$) *Soit a et b deux types d'événements. La relation de causalité $\rightarrow'_W$ est définie par :*

$$a \rightarrow'_W b \text{ si et seulement si } \begin{cases} a >_W b \text{ et } b \not>_W a \\ \quad \text{ou} \quad a \triangle_W b \\ \quad \text{ou} \quad b \triangle_W a \end{cases}$$

Cette nouvelle définition de la relation de causalité permet de gérer les boucles courtes de longueur deux. Notons que de Medeiros *et al.* (2004a, 2005) utilisent une définition différente, où ils s'appuient sur la relation $a \diamond b$, qui signifie « $a \triangle b$ *et* $b \triangle a$ ». Nous montrons à l'annexe B l'équivalence des deux définitions[11]. La définition que nous donnons permet de découvrir les boucles de longueur deux avec une approche plus permissive de la complétude du WF-net. Il suffit d'observer une répétition « aba » pour déduire la présence d'une boucle entre a et b, plutôt que d'observer la répétition « aba » *et* « bab ». Comme nous ne savons pas si les WF-log sur lesquels nous travaillons sont complets, cette définition permet d'utiliser au mieux les indices laissés dans les traces.

Nous proposons de restructurer l'algorithme α' pour matérialiser les possibilités d'interaction. Seul le calcul de la relation de causalité est différent de l'algorithme α. Le reste de l'algorithme est identique. Nous nommons cet algorithme l'algorithme α'_i. La figure 5.9 présente l'algorithme α'_i.

Définition 8 (fonction *analyseLogs()*) *Soit W un WF-log. La fonction $analyseLogs(W)$ se calcule de la façon suivante :*

1. $T = \{t | \exists \sigma \in W,\ t \in \sigma\}$,
2. $T_I = \{t \mid \exists \sigma \in W,\ t = first(\sigma)\}$,
3. $T_O = \{t \mid \exists \sigma \in W,\ t = last(\sigma)\}$,
4. $\forall a, b \in T, a > b \Leftrightarrow \exists \sigma = t_1 \dots t_n \in W, i \in [\![1; n-1]\!] : t_i = a \text{ et } t_{i+1} = b$,

[10] De la même façon que nous appelons l'algorithme α' la version modifiée de l'algorithme α, nous renommons la relation de causalité en « $\rightarrow'_W$ » pour matérialiser la différence avec relation $\rightarrow_W$, qui ne permet pas de gérer les boucles courtes.

[11] Nous montrons que la relation $\triangle$ est équivalente à la relation $\diamond$ dans le cas des WF-log complets issus de WF-nets structurés et conformes sans boucles de longueur un.

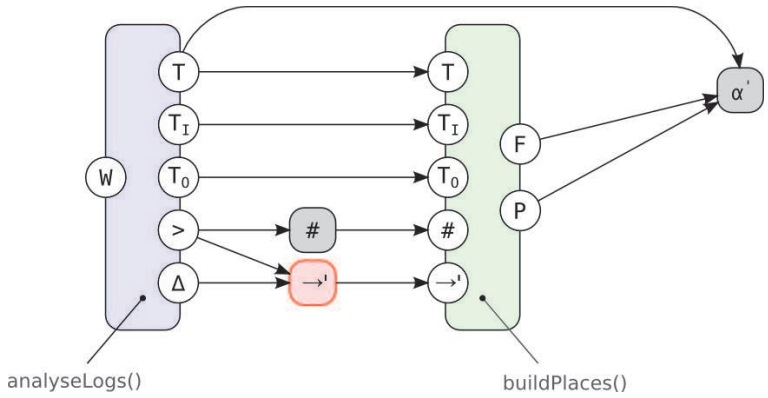


FIGURE 5.9 – Algorithme α'_i. Restructuration de l'algorithme α', permettant de gérer les boucles de longueur 2, en utilisant les fonctions *analyseLogs()* et *buildPlaces()*. La seule modification par rapport à l'algorithme α_i vient de l'utilisation de la relation $\triangle$ pour le calcul de la relation de causalité $\rightarrow$. Cette modification est représentée en rouge sur ce schéma.

5. $\forall a, b \in T, a \triangle b \Leftrightarrow \exists \sigma = t_1 \dots t_n \in W, i \in [\![1; n-2]\!] : t_i = t_{i+2} = a$ *et* $t_{i+1} = b$,
6. $analyseLogs(W) = (T, T_I, T_O, >, \triangle)$.

Comme pour passer de l'algorithme α à l'algorithme α', la seule modification consiste à modifier le calcul de la relation de causalité $\rightarrow$. Nous proposons donc de définir la fonction *buildDependencyRelation()*, qui remplace la fonction *buildDependencyRelation_NSL()*.

Définition 9 (fonction *buildDependencyRelation()*) *Soit T un ensemble de transitions et $>$ une relation de succession définie sur cet ensemble T et $\triangle$ une relation caractérisant les boucles de longueur 2. Alors, la fonction $buildDependencyRelation(T, >, \triangle)$ construit la relation $\rightarrow'$, définie de la manière suivante :*

$$\forall a, b \in T : a \rightarrow' b \Leftrightarrow (a > b \wedge b \not> a) \vee a \triangle b \vee b \triangle a$$

L'algorithme α'_i s'écrit donc de la manière suivante. La figure 5.9 met en évidence la différence entre l'algorithme α et l'algorithme α'.

Définition 10 (algorithme α'_i) *Soit W un WF-log. L'algorithme α'_i se déroule selon les étapes suivantes :*

1. $(T, T_I, T_O, >, \triangle) := analyseLogs(W)$,
2. $\rightarrow' := buildDependencyRelation(T, >, \triangle)$,
3. $\# := buildUnrelatedRelation(T, >)$,
4. $(P, F) := buildPlaces(T, T_I, T_O, \rightarrow', \#)$,
5. $\alpha'_i(W) = (P, T, F)$.

Nous proposons donc d'interagir avec l'analyste à deux niveaux maintenant : au niveau de la relation $>$ comme c'était déjà le cas dans l'algorithme α_i et au niveau de la relation $\triangle$. Par rapport à l'algorithme α, seul le calcul de la relation de causalité ($\rightarrow$, devenue $\rightarrow'$) est différent et le reste de l'algorithme n'est pas modifié.

Algorithme α_i^+ et boucles de longueur un

L'algorithme α^+ proposé par de Medeiros *et al.* (2004a) permet de gérer les boucles de longueur un. Une boucle de longueur un se caractérise par un événement qui peut se répéter plusieurs fois de suite. On observe une boucle de longueur un pour l'événement de type t s'il existe une trace σ dans le WF-log telle que $\sigma = ...tt...$[12].

L'algorithme α^+ se décompose en 3 grandes étapes :

1. Détection des boucles de longueur 1 et création d'un WF-log duquel on a supprimé les événements impliqués dans les boucles de longueur 1 (étapes 1–2 et 6–7).
2. Calcul du WF-net correspondant au réseau privé des boucles de longueur 1 (étape 8).
3. Connexion des événements impliqués dans les boucles de longueur 1 aux « bonnes » places du WF-net reconstruit (étapes 1, 3–5 et 9–12).

Nous proposons de restructurer l'algorithme α^+ de manière à rendre possible l'interaction sur les résultats intermédiaires. La figure 5.10 explicite l'ensemble des étapes de l'algorithme α^+ en détaillant l'appel de l'algorithme α'. De la même manière que

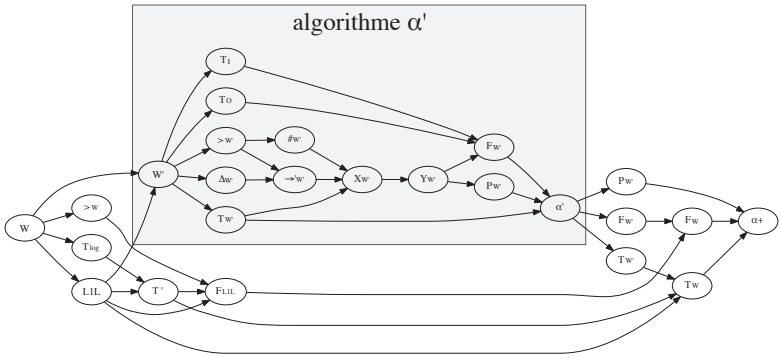


FIGURE 5.10 – Vue d'ensemble des étapes de l'algorithme α^+. Ce schéma explicite toutes les étapes de l'algorithme α^+ (dont l'appel à l'algorithme α'). Une version agrandie de la figure est reproduite en annexe, p. 226.

pour l'algorithme α, nous proposons d'utiliser les fonctions *analyseLogs()* et *buildPlaces()* pour restructurer l'algorithme α^+. Il faut en outre ajouter deux fonctions, *preprocess_L1L()* et *postprocess_L1L()*, pour calculer les résultats intermédiaires spécifiques aux boucles de longueur un. La figure 5.11 montre comment faire appel à ces fonctions sans modifier l'algorithme α^+.

[12]Notons que cela implique $t >_W t$

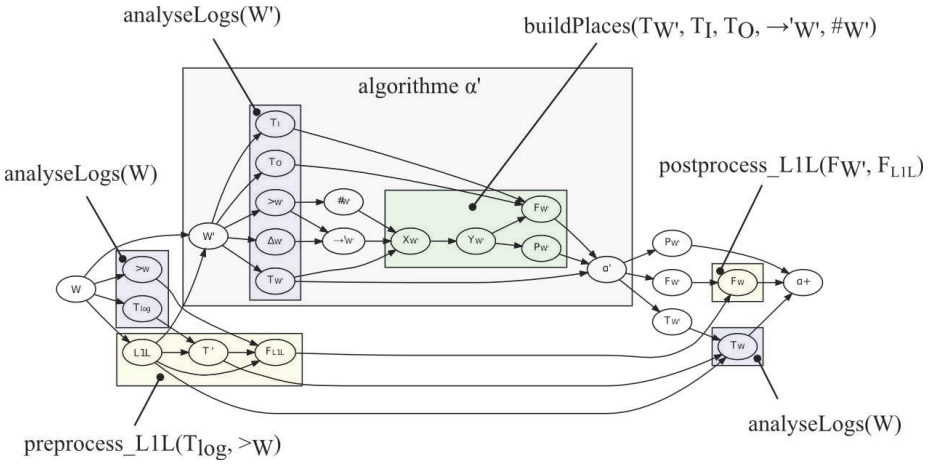


FIGURE 5.11 – Vue d'ensemble des étapes de l'algorithme α^+. Ce schéma reprend les étapes de l'algorithme α^+ en suggérant des regroupements possibles avec le même principe de reformulation que pour l'algorithme α. Une version agrandie de la figure est reproduite en annexe, p. 227.

En annexe B, nous montrons que plusieurs calculs sont redondants et que le calcul d'un nouveau WF-log sans boucles de longueur un n'est pas nécessaire. En particulier, la relation $>$ dont nous tirons parti pour l'interaction est calculée deux fois, une fois sur le WF-log (W), une fois sur le WF-log sans les boucles de longueur un (W'). Il est possible de simplifier l'algorithme α^+. Nous appelons cette version simplifiée, l'algorithme α_i^+. Les figures 5.12 et 5.13 présentent l'algorithme α_i^+. Nous démontrons à l'annexe B (pp. 224 à 231) l'équivalence de l'algorithme α^+ et de l'algorithme α_i^+.

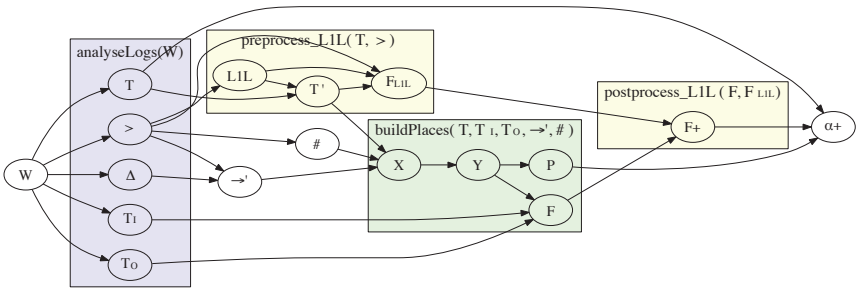


FIGURE 5.12 – Vue d'ensemble des étapes de l'algorithme α_i^+, reformulation de l'algorithme α^+.

L'algorithme α_i^+ s'appuie sur deux nouvelles fonctions. La fonction *preprocess_L1L()*, qui calcule des résultats intermédiaires relatifs aux boucles de longueur un, et la fonction *postprocess_L1L()*, qui connecte les transitions et les places impliquées dans les boucles de longueur un.

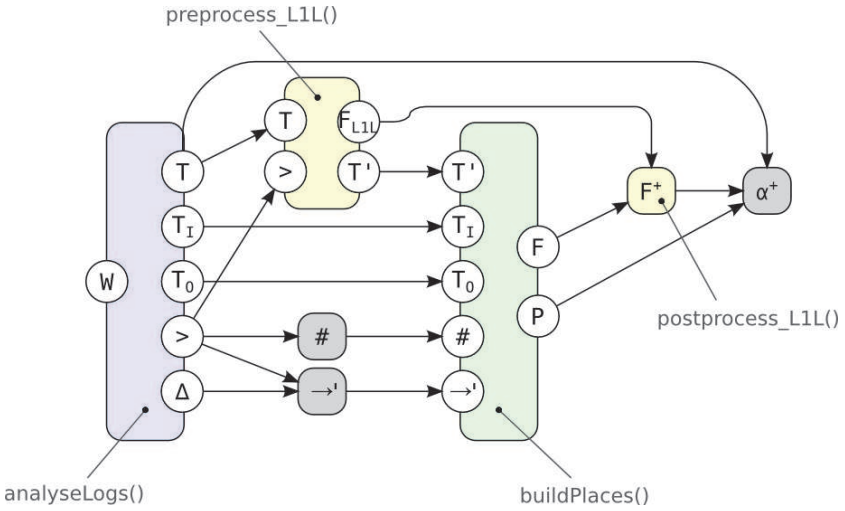


FIGURE 5.13 – Algorithme α_i^+. Réécriture, à l'aide des fonctions *analyseLogs()* et *buildPlaces()*, de l'algorithme α^+ qui permet de gérer tous les types de boucles d'un WF-net structuré et conforme.
Deux étapes intermédiaires sont ajoutées. La première étape, avant de construire les places : la fonction *preprocess_L1L()* extrait les informations sur les boucles de longueur 1 (F_{L1L}) et construit un sous-ensemble de transitions (T'), sans boucles de longueur 1, sur lequel la fonction *builPlaces()* est exécutée. La deuxième étape, après avoir construit les places : la fonction *postprocess_L1L()* connecte les transitions impliquées dans les boucles de longueur 1 au réseau découvert.

Définition 11 (fonction *preprocess_L1L()*) *Soit T un ensemble de transitions et $>$ la relations de succession définie sur les transitions de T.*
La fonction $preprocess\_L1L(T, >)$ se calcule de la façon suivante :

1. $L1L = \{t \in T | t > t\}$,
2. $T' = T \setminus L1L$,
3. $F_{L1L} = \emptyset$,
4. *Pour chaque $t \in L1L$:*
   a) $A = \{a \in T' | a >_W t\}$,
   b) $B = \{b \in T' | t >_W b\}$,
   c) $F_{L1L} := F_{L1L} \cup \{(t, p(A \setminus B, B \setminus A)), (p(A \setminus B, B \setminus A), t)\}$,
5. $preprocess\_L1L(T, >) = (T', F_{L1L})$.

Définition 12 (fonction *postprocess_L1L()*) *Soit F et F_{L1L} deux ensembles d'arcs d'un réseau de Petri.*
La fonction $postprocess\_L1L(F, F_{L1L})$ se calcule de la façon suivante :

1. $F^+ = F \cup F_{L1L}$,

2. $postprocess\_L1L(F, F_{L1L}) = F^+$.

La définition formelle de l'algorithme α_i^+ est la suivante :

Définition 13 (algorithme α_i^+) *Soit W un WF-log. L'algorithme α_i^+ se déroule selon les étapes suivantes :*

1. $(T, T_I, T_O, >, \triangle) := analyseLogs(W)$,
2. $(T', F_{L1L}) := preprocess\_L1L(T, >)$,
3. $\rightarrow' := buildDependencyRelation(T, >, \triangle)$,
4. $\# := buildUnrelatedRelation(T, >)$,
5. $(P, F) := buildPlaces(T', T_I, T_O, \rightarrow', \#)$,
6. $F^+ := postprocess\_L1L(F, F_{L1L})$,
7. $\alpha_i^+(W) = (P, T, F^+)$.

L'algorithme α_i^+, en simplifiant l'algorithme α^+, apporte des bénéfices en termes de simplicité de calcul (et donc potentiellement de performance), et aussi en termes d'interactions entre l'analyste et l'algorithme. En supprimant les calculs intermédiaires redondants, il n'est pas nécessaire d'interagir à plusieurs endroits pour modifier un résultat intermédiaire de manière pertinente. Par exemple, la relation de succession $>$ est calculée une seule fois dans tout l'algorithme. De même, le calcul des boucles de longueur un s'appuie sur la relation de succession. En interagissant au niveau de la relation $>$, l'analyste peut donc en même temps valider ou invalider la relation elle-même et valider ou invalider la présence de boucles de longueur un.

Algorithme $\alpha_i^{+\prime}$: généralisation de l'algorithme α_i^+

Enfin, une modification mineure de l'algorithme α^+ est proposée par de Medeiros *et al.* (2004a) pour généraliser à d'autres types de réseaux que les WF-nets structurés et conformes. Les auteurs nomment ce nouvel algorithme l'« algorithme α^{++} ». Pour éviter de le confondre avec l'algorithme α^{++} qui est couramment cité dans la littérature, nous appelons cette variante de l'algorithme α^+ l'« algorithme $\alpha^{+\prime}$ ».

La différence entre l'algorithme α^+ et l'algorithme $\alpha^{+\prime}$ se trouve dans la façon de connecter les transitions impliquées dans des boucles de longueur un aux places du réseau de Petri. Cette différence se répercute au niveau du calcul de F^+ et n'impacte pas les autres étapes de calcul l'algorithme α^+. En répercutant cette modification sur l'algorithme α_i^+, la fonction *postprocess_L1L()* devient *postprocess_L1L+()*.

Définition 14 (fonction *postprocess_L1L+()*) *Soit F et F_{L1L} deux ensembles d'arcs d'un réseau de Petri, P un ensemble de places candidates et $L1L$ l'ensemble des transitions impliquées dans des boucles de longueur un.*
La fonction $postprocess\_L1L+(F, F_{L1L}, P, L1L)$ se calcule de la façon suivante :

1. $F^+ = F \cup$
   $\{(t, p(A, B)) \in (L1L \times P) | \exists (t', p(A', B')) \in F_{L1L} : t = t' \wedge A \subseteq A' \wedge B \subseteq B'\} \cup \{ (p(A, B), t) \in (P \times L1L) | \exists (p(A', B'), t') \in F_{L1L} : t = t' \wedge A \subseteq A' \wedge B \subseteq B'\}$,

2. $postprocess\_L1L(F, F_{L1L}, P, L1L) = F^+$.

Notons que le calcul de *postprocess_L1L+()* nécessite d'avoir accès à l'ensemble $L1L$ qui est calculé au sein de la fonction *preprocess_L1L()*. Pour cela, nous appelons *preprocess_L1L+()* la fonction *preprocess_L1L()* dont on a modifié la signature.

En répercutant l'utilisation de ces nouvelles fonctions, on obtient l'algorithme ${\alpha_i}^{+\prime}$, présenté à la figure 5.14 dont la définition est la suivante :

Définition 15 (algorithme ${\alpha_i}^{+\prime}$) *Soit W un WF-log. L'algorithme ${\alpha_i}^{+\prime}$ se déroule selon les étapes suivantes :*

1. $(T, T_I, T_O, >, \triangle) := analyseLogs(W)$,
2. $(T', F_{L1L}, L1L) := preprocess\_L1L+(T, >)$,
3. $\rightarrow' := buildDependencyRelation(T, >, \triangle)$,
4. $\# := buildUnrelatedRelation(T, >)$,
5. $(P, F) := buildPlaces(T', T_I, T_O, \rightarrow', \#)$,
6. $F^+ := postprocess\_L1L+(F, F_{L1L}, P, L1L)$,
7. ${\alpha_i}^{+'}(W) = (P, T, F^+)$.

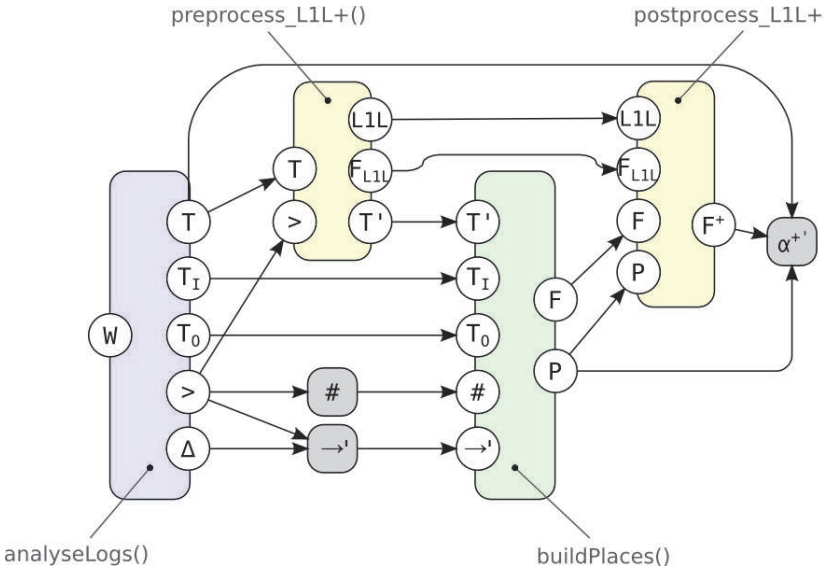


FIGURE 5.14 – Algorithme ${\alpha_i}^{+\prime}$. Réécriture, à l'aide des fonctions *analyseLogs()* et *buildPlaces()*, de l'algorithme ${\alpha}^{+\prime}$.
Par rapport à l'algorithme α_i^+, la fonction de post-traitement, ajoutant les boucles de longueur un au réseau final, est modifiée, ainsi que la signature de la fonction de prétraitement, donnant lieu à deux nouvelles fonctions : *preprocess_L1L+()* et *postprocess_L1L+()*.

Les algorithmes présentés jusqu'ici – les algorithmes α_i, α'_i, α_i^+, $\alpha_i^{+\prime}$ – ont fait l'objet d'une implémentation dans le cadre de cette thèse, dans le logiciel AUTOMATA[13].

5.3.2 Interaction avec les autres algorithmes de la famille α

Dans l'état de l'art proposé par van Dongen *et al.* (2009) sur les algorithmes permettant de construire un modèle de type réseau de Petri à partir de traces d'exécution d'un processus, les auteurs présentent et comparent six algorithmes basés sur l'algorithme α : l'algorithme α qui permet de retrouver des réseaux de Petri sans boucles courtes et à choix libre (van der Aalst *et al.*, 2002, 2004), l'algorithme α^+ qui gère les boucles courtes (de Medeiros *et al.*, 2004a, 2005), l'algorithme β qui gère les boucles courtes et les attributs (« début » et « fin ») associés aux événements (Wen *et al.*, 2009, 2004), l'algorithme α^* qui gère les transitions dupliquées (Li *et al.*, 2007), l'algorithme $\alpha^\#$ qui gère les transitions cachées (Wen *et al.*, 2007b), l'algorithme α^{++} qui étend l'algorithme à des modèles réseau de Petri à choix non libre (Wen *et al.*, 2007a).

Algorithme β

L'algorithme β, aussi appelé parfois « algorithme tsinghua-α », exploite les WF-log où chaque événement est associé avec un attribut « début » ou « fin », matérialisant le début ou la fin de l'exécution de l'action enregistrée (Wen *et al.*, 2009, 2004). L'algorithme β s'appuie sur le même principe que l'algorithme α : définir une relation d'ordre à partir de laquelle on déduit une relation de causalité. Puis, à partir de cette relation de causalité, construire un réseau de Petri. La principale différence réside dans la façon de calculer la relation de causalité.

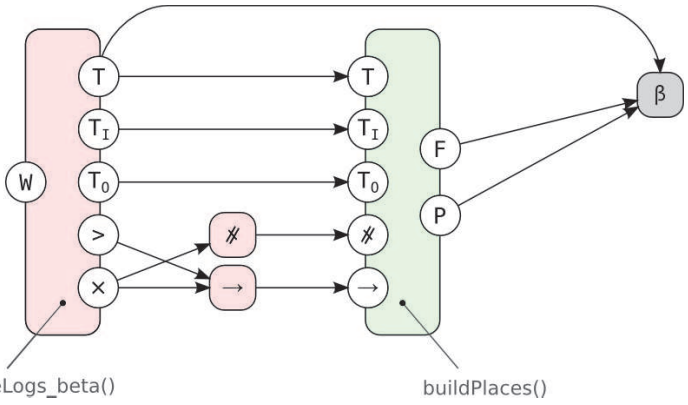


FIGURE 5.15 – L'algorithme β s'appuie sur le même principe que l'algorithme α. La différence principale, non visible sur ce schéma réside dans le format du WF-log W qui ne contient pas des événements atomiques, comme dans l'algorithme α, mais des événements associés à un attribut « début » ou « fin ». Cela nécessite de calculer différemment la relation de succession $>$ et introduit une autre relation, la relation d'intersection temporelle $\times$.
La fonction d'analyse du WF-log est donc différente. Les relations sont calculées différemment. Le reste de l'algorithme est identique.

[13] AUTOmata Modelling of the Activity, based on Trace Analysis

Ainsi, la figure 5.15 décrit le fonctionnement de l'algorithme β tel que l'on pourrait le reformuler selon notre approche. La relation de succession $>$ repose sur le même principe que pour l'algorithme α, en tirant parti, en plus, des dates de début et de fin associées aux événements. La relation d'intersection temporelle $\times$, aussi appelée relation de parallélisme $\|$, représente la possibilité pour deux types d'action de se dérouler en même temps.

La relation de causalité $\rightarrow$ est calculée à partir de la relation de succession $>$ et de la relation d'intersection temporelle $\times$. La relation de non-intersection, notée $\nparallel$, remplace la relation d'indépendance $\#$ dans la suite du déroulement de l'algorithme α.

Algorithme α^*

L'algorithme α^* vise à détecter les transitions dupliquées (Li *et al.*, 2007), c'est-à-dire les transitions associées au même type d'événement, mais représentées dans le modèle par des transitions distinctes. Pour ce faire, l'algorithme α^* analyse les traces du WF-log à la recherche de motifs permettant de distinguer des types d'événements réellement identiques, c.-à-d. correspondants à la même transition, des types d'événements identiques, mais devant être modélisés par des transitions différentes. Cette analyse permet de renommer les types d'événements détectés comme différents pour ensuite appliquer l'algorithme α sur le nouveau WF-log désambiguïsé. Une fois l'algorithme α appliqué, il suffit de décoder le résultat en renommant les transitions dupliquées pour leur attribuer un nom identique.

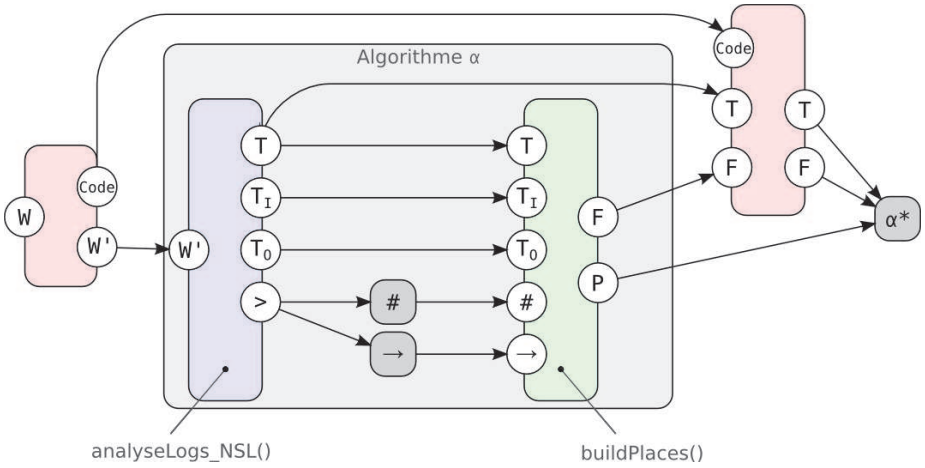


FIGURE 5.16 – L'algorithme α^* constitue un enrobage de l'algorithme α avec une étape de prétraitement et une étape de post-traitement. L'étape de prétraitement analyse le WF-log et encode les événements correspondant aux transitions dupliquées en leur donnant des noms différents. L'étape de post-traitement décode les transitions dupliquées en leur redonnant le même nom. Le paramètre « Code » définit la façon de décoder les transitions dupliquées dans l'étape de post-traitement.

L'algorithme α^* réutilise donc l'algorithme α en l'encadrant d'une étape de prétraitement et d'une étape de post-traitement pour découvrir les transitions dupliquées. La figure 5.16 illustre le fonctionnement de l'algorithme α^*.

Algorithme $\alpha^{\#}$

L'algorithme $\alpha^{\#}$ s'appuie sur l'algorithme α' et vise à découvrir les transitions cachées (Wen *et al.*, 2007b), c'est-à-dire des transitions n'apparaissant pas dans le WF-log, mais nécessaires à la construction du modèle pour expliquer les comportements observés. Comme l'algorithme $\alpha^{\#}$ s'appuie sur l'algorithme α', il gère les boucles courtes de longueur deux. L'algorithme $\alpha^{\#}$ s'appuie sur les transitions cachées pour reconstituer les boucles de longueur un.

Wen *et al.* (2007b) identifient quatre types de transitions cachées : les transitions cachées initiales ou finales (« SIDE »), les transitions cachées permettant de sauter une ou plusieurs transitions, pour sauter en avant (« SKIP ») ou revenir en arrière (« REDO »), et les transitions cachées permettant, pour deux branches concurrentes du réseau de Petri, de passer d'une branche à une autre (« SWITCH »).

L'algorithme $\alpha^{\#}$, comme l'algorithme α s'appuie sur les relations de succession et de causalité. Il s'appuie en outre sur une relation supplémentaire, appelée la « relation d'ordre avancée » (*advanced ordering relation*), notée $\rightsquigarrow_W$. Intuitivement, cette relation d'ordre avancée permet d'inférer une sorte de causalité qui nécessiterait une transition cachée pour être matérialisée dans le réseau de Petri.

La description de l'algorithme $\alpha^{\#}$ dans la publication de Wen *et al.* (2007b) est incomplète[14] et ne permet pas de faire une critique fondée.

Algorithme α^{++}

L'algorithme α^{++} de Wen *et al.* (2007a) s'appuie sur l'algorithme α^{+} et l'enrichit de post-traitements permettant de reconstituer un réseau de Petri à choix non libre. Le choix non libre est difficile à détecter puisqu'il représente une contrainte indirecte entre deux transitions. L'algorithme α et ses extensions s'appuient sur des relations locales entre les événements d'une trace. L'algorithme α^{++} introduit de nouvelles relations non locales, calculées sur les traces entières pour déterminer ces dépendances implicites.

Les auteurs identifient trois types de dépendances implicites notées $\mapsto_{W^1}$, $\mapsto_{W^2}$ et $\mapsto_{W^3}$. Le premier type de dépendance implicite, $\mapsto_{W^1}$, permet de résoudre des problèmes de construction où deux places devraient être fusionnées. Le deuxième type de dépendance implicite, $\mapsto_{W^2}$, détecte des situations où des arcs sont manquants. Enfin, le troisième type de dépendance, $\mapsto_{W^3}$, rajoute des places et les connecte aux transitions pertinentes pour marquer la dépendance implicite entre ces transitions.

L'utilisation de l'algorithme α^{+} sur un WF-log incomplet ou sur un WF-log complet d'un WF-net conforme non structuré peut aboutir à un WF-net non conforme. Les dépendances implicites des deux premiers types permettent de gérer ce genre de problème de construction et reconstruire un WF-net conforme. La dépendance implicite du troisième type, elle, n'a pas d'impact sur la conformité du WF-net généré. Elle ajoute une contrainte sémantique telle que « l'action B n'est possible que si l'action A a déjà été exécutée auparavant. »

L'algorithme α^{++} s'appuie sur l'algorithme α^{+} pour calculer un premier réseau de Petri candidat. Puis, l'algorithme α^{++} effectue un post-traitement sur ce réseau

[14] Trois fonctions sont utilisées sans être définies. La fonction *ConSideIT(W)* est censée détecter les transitions cachées de type SIDE et adapter en conséquence les transitions initiales et finales du réseau de Petri. Les fonctions *PreSet* et *PostSet* sont censées calculer respectivement les places d'entrée et les places de sortie d'une transition donnée, mais le calcul des places constitue justement le verrou que cherche à lever un algorithme comme l'algorithme α et l'appel à ces fonctions est fait avant même la construction d'un quelconque réseau de Petri.

de Petri. En combinant l'analyse du WF-log (et des relations non locales), avec une analyse du réseau de Petri, les relations de dépendances implicites sont calculées et le réseau est modifié en conséquence, d'abord en tenant compte des relations $\mapsto_{W^1}$ et $\mapsto_{W^2}$, puis de la relation $\mapsto_{W^3}$.

L'outil EMiT

L'outil EMiT (van der Aalst et van Dongen, 2002) permet de prendre en compte le temps. Cet outil s'appuie dans un premier temps sur l'algorithme α pour construire un premier réseau de Petri. Dans un second temps, l'outil construit un réseau de Petri temporel[15], à l'aide d'un post-traitement analysant les informations temporelles présentes dans les traces. Comme pour l'algorithme α, les traces sont supposées complètes : cette approche ne gère pas le bruit dans les données, ni au niveau de la construction du réseau de Petri, ni au niveau du post-traitement ajoutant la temporalité.

Comme l'algorithme α est utilisé tel quel dans l'outil EMiT, l'interaction que nous proposons pourrait y être intégrée avec l'algorithme α_i. Cependant, il faut gérer le problème du bruit et de l'incomplétude des traces au niveau de l'étape de post-traitement. Pour cela, il est nécessaire de modifier l'étape de post-traitement pour qu'elle prenne en compte les éventuelles incohérences entre les traces et le réseau de Petri construit.

Little Thumb

L'outil Little Thumb vise à résoudre les problèmes de bruit et d'incomplétude des données en utilisant une analyse fréquentielle des motifs d'une trace (Weijters et van der Aalst, 2003). Des heuristiques permettent d'inférer des relations de causalité et d'indépendance, nécessaires au fonctionnement de l'algorithme α à partir de mesures de fréquence de la succession ou non-succession de types d'événements dans les traces. Le reste de l'algorithme α est inchangé.

Il serait donc possible de profiter de la reformulation algorithme α_i dans l'outil Little Thumb pour proposer à l'analyste une autre méthode de calcul de succession et lui permettre d'interagir sur ce résultat intermédiaire.

5.4 Discussion

Dans ce chapitre, nous avons montré qu'il est possible de modifier un algorithme de synthèse de réseau de Petri, l'algorithme α, pour le rendre interactif. L'utilisateur, un analyste 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.

Nous nous sommes basés sur une famille d'algorithmes existants, la famille α et nous avons proposé une façon de reformuler ces algorithmes pour faciliter l'interaction avec un analyste. Cette interaction se base principalement sur la possibilité pour l'analyste de valider, modifier ou invalider des résultats intermédiaires qui se traduisent aisément en langage naturel.

[15] Plus précisément un réseau de Petri P-temporel, la notion de temps étant associée aux places

Développer l'interaction

Nous nous sommes focalisés sur un résultat intermédiaire particulier : la relation de succession $>$, car elle présente des propriétés intéressantes. Cette relation $>$ sert de base à toute la suite du calcul et elle se traduit très simplement en langage naturel : $a > b$ signifie qu'« il est possible qu'une trace où a est directement suivi de b soit produite par le processus à modéliser. » En outre, interagir sur la relation de succession $>$ permet de gérer les problèmes d'incomplétude des WF-log, dans la mesure où l'ensemble des transitions et les ensembles de transitions initiales et finales sont bien définis.

Il est pertinent d'interagir sur d'autres résultats intermédiaires, tels que les transitions d'entrée ou de sortie ou les relations de causalité et d'indépendance. La reformulation que nous proposons des algorithmes α, α' et α^+ rend d'ailleurs possible l'interaction sur ces résultats intermédiaires. Cependant, certains résultats intermédiaires sont dépendants les uns des autres. Ces résultats sont plus difficiles à appréhender indépendamment du reste de l'algorithme. Par exemple, si une transition est une transition d'entrée, alors aucune transition ne devrait pouvoir la précéder.

Pour aller plus loin qu'une simple interaction au niveau des résultats intermédiaires, il serait intéressant de tirer profit de ces contraintes pour développer une interaction de type chaînage mixte. Pour développer ce chaînage mixte, il faut étudier les cas où l'algorithme α (ou ses variantes) est mis en échec, typiquement avec un WF-log incomplet.

Un 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.

Quel algorithme choisir ?

Au sein de la famille d'algorithmes α, plusieurs évolutions de l'algorithme α permettent de gérer individuellement les problèmes des boucles courtes, des transitions dupliquées, des transitions cachées, des réseaux de Petri à choix non libre et des actions représentées par un début et une fin. Nous avons présenté en détail comment interagir avec l'algorithme α et son évolution permettant de gérer les boucles courtes, l'algorithme α^+. Si l'on disposait des versions interactives de tous ces algorithmes, il serait pertinent de se poser la question : quel algorithme utiliser pour quel problème ?

Si le modèle est susceptible de contenir des choix non libres, c'est-à-dire des choix dépendants d'actions passées, alors l'algorithme α^{++} est approprié. Si les actions sont représentées dans le WF-log par des événements de début et de fin, alors l'algorithme β sera plus adapté. Si des actions portant le même nom peuvent faire référence en réalité à des actions différentes, alors l'algorithme α^* est approprié. Dans ce cas, il est aussi envisageable de prétraiter le WF-log pour différencier en amont les différentes actions. Enfin, si l'on sait que certaines actions sont absentes des WF-log, on peut utiliser l'algorithme $\alpha^\#$ pour tenter d'inférer des transitions cachées.

On pourrait tenter de combiner les différentes approches de la famille de l'algorithme α pour pouvoir gérer au besoin les boucles courtes, les transitions cachées, les transitions dupliquées, le choix non libre, etc. L'algorithme α^{++}, qui consiste à un post-traitement du réseau de Petri découvert pour gérer les problèmes de choix non libres, pourrait par exemple être combiné à l'algorithme β pour gérer les actions non

atomiques. Ce travail d'intégration est difficile, puisque certains problèmes peuvent être abordés de différentes manières. Par exemple, les boucles courtes peuvent être traitées soit avec l'algorithme α^+, soit avec l'algorithme $\alpha^\#$, soit avec l'algorithme α^*. Ou encore, un choix non libre peut être exprimé avec des transitions dupliquées. Notons à ce titre que les algorithmes α^{++}, α^* et $\alpha^\#$ permettent d'analyser des WF-log d'un WF-net non structuré, alors que les autres algorithmes se limitent aux WF-logs d'un WF-net structuré.

Plutôt que de fusionner ces différentes approches en une seule, on peut imaginer que l'environnement d'analyse pourrait prendre en compte ces heuristiques pour proposer des choix à l'analyste.

Complexité

L'interaction que nous avons présentée se base sur la relation de succession $>$, qui peut être erronée à cause de l'incomplétude des données ou du bruit. Nous proposons à l'utilisateur d'enrichir la relation $>$, calculée à partir de l'ensemble des traces, pour produire une relation $>_\mathcal{A}$ complétée des connaissances de l'analyste. Cela pose a priori un problème de complexité à deux niveaux. D'une part, l'algorithme α s'appuie sur un calcul de cliques[16], 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 5.7 (p. 93), le WF-net étudié possède 5 transitions et l'analyste doit vérifier la relation $>$ pour chacun des 25 couples de types d'événements. Nous proposons de résoudre cette difficulté par notre approche méthodologique.

Le cycle de découverte de connaissances

Nous rappelons que notre démarche s'intègre dans un processus global de découverte de connaissances, que nous avons présenté au chapitre 3 et sur lequel nous reviendrons plus en détail au chapitre 6. Ceci nous permet de contrôler le nombre de types 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. Cet analyste sait quel type de connaissances il cherche. Le principe est donc de découper le problème extrêmement complexe de la conduite automobile en sous-problèmes plus simples que l'on peut facilement maîtriser. 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. Le chapitre 6 présente plus en détail cette méthodologie associée au cycle de découverte de connaissances.

Notons de plus que dans son ensemble, la méthodologie que nous proposons, combinée à l'utilisation de l'algorithme α_i, apporte une solution à certains défis de la fouille de workflow : le problème de la sémantique, des niveaux d'abstraction et de la gestion des attributs (van Dongen *et al.*, 2009). Les $\mathcal{M}$-Traces apportent la sémantique.

[16] En théorie des graphes, une clique est un ensemble de nœuds deux à deux adjacents. Les ensembles A et B de l'étape 4 de l'algorithme α (voir p. 82) sont des cliques sur la relation $\#$.

Les abstractions sont obtenues par reformulation dans le logiciel ABSTRACT Georgeon (2008). Enfin, à travers la transformation de traces, il est possible de prendre en compte les attributs associés aux événements.

Pour la description d'activités riches, nécessitant de conserver un grand nombre d'informations dans les traces, nous envisageons de diviser le problème en sous-problèmes liés aux différents niveaux de granularité de l'activité. Dans cet objectif, il serait pertinent de développer une méthodologie de découverte de connaissances sous forme de réseaux de Petri hiérarchiques, qui permettraient 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.

Validation

Nous proposons deux manières de valider les connaissances découvertes : une validation épistémologique et une validation par évaluation.

Les propriétés de reconstruction d'un WF-net à partir d'un WF-log ont été démontrées pour l'algorithme α et l'algorithme α^+ (de Medeiros *et al.*, 2004a; van der Aalst *et al.*, 2004). Nous avons montré que l'algorithme α_i, respectivement l'algorithme α_i^+, est formellement équivalant à l'algorithme α, respectivement à l'algorithme α^+. Ainsi, l'algorithme α_i et l'algorithme α_i^+ sont capables de reconstruire des WF-net, à partir d'un WF-log complet ou à partir d'un WF-log incomplet et des connaissances de l'analyste.

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 connaissances est vraie, alors la connaissance construite est valide par construction.

Il existe plusieurs façons d'évaluer les connaissances produites. Comme l'on dispose d'un modèle de type réseau de Petri, 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. Dans le domaine de la fouille de workflow, Rozinat et van der Aalst (2006) proposent différentes mesures pour qualifier l'adéquation du modèle à l'ensemble des traces : des mesures d'efficacité (*fitness*) et des mesures de pertinence (*appropriateness*). Les mesures d'efficacité déterminent à quel point le réseau de Petri construit est capable de produire tout ou partie des traces du WF-log. Les mesures de pertinence évaluent le juste équilibre entre la clarté du modèle au niveau de sa structure et sa capacité à décrire précisément le processus observé, sans être trop spécifique ni trop généralisant.

Cependant, comme le soulignent Wen *et al.* (2007a), les mesures d'efficacité et de pertinence comparent le *modèle découvert* aux *traces du processus réel* au lieu de comparer le *modèle découvert* au *modèle réel* du processus étudié. Van der Aalst *et al.* (2006) proposent de comparer différents modèles et d'évaluer leurs différences. Ces mesures sont exploitables dans le cas où un modèle est connu a priori ou dans le cas où il faudrait comparer plusieurs modèles candidats, ce qui n'est pas le cas de notre approche.

5.5 Conclusion

Dans ce chapitre, nous avons présenté la première contribution de cette thèse. Nous avons montré comment rendre interactive la découverte d'automates à partir de traces. Plus précisément, nous proposons à l'analyste d'interagir au niveau des résultats intermédiaires de l'algorithme α.

Nous avons ainsi proposé l'algorithme α_i, une version interactive de l'algorithme α. Nous avons montré comment utiliser ces mêmes principes d'interaction pour modifier des variantes de l'algorithme α, ce qui nous a amenés à proposer les algorithmes α'_i, α_i^+ et $\alpha_i^{+\prime}$. Ces variantes interactives permettent de compenser les problèmes d'incomplétude des traces avec les connaissances de l'analyste.

Dans le chapitre suivant, nous montrerons comment ces algorithmes s'inscrivent dans le cadre méthodologique global de la découverte de connaissances. Plus particulièrement, nous présenterons une méthodologie de découverte exploratoire de connaissances, pilotée par l'analyste, qui vise à prendre en compte les difficultés rencontrées lors de l'analyse de données complexes telles que les données de conduite automobile.

Enfin, dans le chapitre 7, nous présenterons une étude de cas où nous exploiterons les algorithmes interactifs sur des données réelles de conduite automobile.

CHAPITRE 6

Mettre en œuvre un cycle de construction interactive de connaissances

> *When you are solving a difficult problem re-ask the problem so that your solution helps you learn faster. Find a faster way to fail, recover, and try again. If the problem you are trying to solve involves creating a magnum opus, you are solving the wrong problem.* « Quand vous cherchez à résoudre un problème difficile, reformulez-le de telle manière à ce que votre solution vous aide à apprendre plus vite. Trouvez une manière d'échouer plus rapidement, d'apprendre de vos erreurs et d'essayer à nouveau. Si le problème que vous cherchez à résoudre implique de créer une chimère, vous vous attaquez au mauvais problème. »
>
> (Raskin, 2011.)

Ce chapitre détaille le cadre conceptuel dans lequel nous nous situons pour découvrir des connaissances. Nous souhaitons éclairer le lecteur sur un sujet qui nous semble nécessaire à la compréhension de toute notre démarche. La découverte de connaissances implique l'humain, car c'est lui qui donne à des données leur statut de connaissances et c'est lui qui discrimine ce qui fait sens ou non[1]. L'humain doit donc être intégré dans le cycle de découverte de connaissances. Nous souhaitons renforcer le couplage entre l'humain et la machine et ainsi profiter des capacités d'interprétation et de compréhension qu'a l'humain et des capacités de calculs que possède l'ordinateur.

Ce chapitre est composé de quatre sections. La première section explicite le fait que des connaissances sont en réalité créées à chaque niveau du cycle de découverte de connaissances. Il s'agit d'un autre cycle, complémentaire du cycle de découverte des connaissances, que nous appelons cycle d'appropriation. La deuxième section propose de revisiter le cycle de découverte de connaissances pour y intégrer l'humain et les connaissances créées tout au long du cycle de découverte de connaissances. C'est le cycle de construction interactive de connaissances. La troisième section propose une

[1] D'un point de vue philosophique, cela pourrait être la machine. Mais ce qui compte pour nous, humains, utilisateurs, ce n'est pas ce que la machine considère comme pertinent ou vrai, mais ce que nous *comprenons* comme vrai.

méthodologie basée sur ce cycle de construction interactive de connaissances et appliquée au domaine de la conduite automobile. Enfin, la quatrième section considère la question de la validation de notre travail d'ingénierie des connaissances.

6.1 Le cycle d'appropriation – un cycle complémentaire et orthogonal au cycle de découverte de connaissances

Le cycle de découverte de connaissances introduit par Fayyad *et al.* (1996b) est présenté comme une chaîne qui, à la dernière étape, permet de créer des connaissances via l'interprétation des motifs trouvés à l'issue d'une fouille de données. Ce processus d'interprétation peut donner lieu à des retours en arrière, que ce soit au niveau des réglages des paramètres de fouille (changement d'un seuil parce que l'algorithme trouve trop de motifs et qu'il n'est pas possible de discerner le bon grain de l'ivraie), ou au niveau de la sélection des données, du post-traitement des données ou de leur mise en forme.

Dans la pratique, le cycle est beaucoup plus « bouclé » que cela. Ce n'est pas seulement à la fin que l'analyste utilise ses connaissances pour revenir à une des étapes précédentes. En réalité, chaque étape permet à l'humain de produire des connaissances qui peuvent réalimenter les étapes précédentes du cycle de découverte. La capacité de l'humain à comprendre et à interpréter et les capacités de calcul de la machine sont toujours présentes dans ce processus de découverte de connaissances. C'est ce couplage entre l'humain et la machine que nous souhaitons expliciter dans cette section et plus généralement exploiter dans notre travail.

6.1.1 Raison conceptuelle d'avoir une interaction « à toutes les étapes »

Les itérations sont nécessaires, de la collecte à l'obtention d'un résultat

Exemple

Alors que dans un monde idéal les hypothèses sont bien définies, les données sont correctement collectées et toute la chaîne a été entièrement conçue et paramétrée a priori, avant d'y injecter un quelconque jeu de données, dans la réalité, de nombreux problèmes sont à traiter a posteriori. En effet, dans la pratique (nous parlons ici de notre expérience sur les données de conduite), nous disposons de données imparfaites, collectées avec des capteurs bruités ou tombant en panne. Les outils ne sont pas paramétrés a priori, mais après plusieurs tentatives. L'utilisation d'outils de post-traitement des données va mettre en avant la défaillance d'une des mesures et nécessiter un rebouclage. La mise en forme des données, qui paraissait logique a priori va s'avérer moins efficace ou moins pertinente que prévu en pratique. Bref, dans la pratique, une succession d'essais, de retours en arrière et d'événements inattendus va remettre en cause une mécanique semblant initialement bien huilée.

La raison pour laquelle il est essentiel dans notre cas de pouvoir interagir *à tous les niveaux*, c'est que nous nous situons dans une approche exploratoire. Les données fournies à l'analyste n'ont pas été exploitées antérieurement, ou seulement partiellement.

Par exemple, dans notre cas, personne n'a exploité de manière détaillée l'intégralité des données provenant des différents capteurs. Alors que des tests ont été effectués par l'équipe technique pour vérifier que les capteurs donnaient des résultats cohérents, les tests ne sont pas forcément menés en profondeur. Des problèmes peuvent apparaître lors de l'analyse plus poussée d'un de ces paramètres, par exemple des problèmes de synchronisation ou de calibration. Peut-on considérer que ce problème est en dehors du champ de l'analyste ? Nous pensons que non. Pourquoi ? Pourquoi ne pas dire que l'équipe d'instrumentation « n'avait qu'à bien faire son travail » ? Pour plusieurs raisons. Tout d'abord, quel que soit le niveau de test qui est fait au niveau de l'instrumentation, il restera toujours la possibilité qu'un bogue survienne. À partir du moment où le système d'acquisition atteint une certaine complexité, il est de l'ordre de l'impossible de créer des logiciels et des matériels d'acquisition qui soient fiables à 100 %. Ce n'est d'ailleurs pas le niveau d'exigence que l'on se fixe. Le mieux que l'on puisse faire, c'est assurer un certain niveau de fiabilité à l'aide de tests. Le deuxième élément, c'est que la question de fiabilité dépend du niveau d'analyse que l'on veut faire. Un capteur parfait n'existe pas et un capteur imparfait, imprécis, bruité, peut largement suffire à certaines analyses. Lorsque l'on collecte des données dans un objectif d'analyse exploratoire, les usages qui seront faits des données collectées ne sont pas toujours connus. On ne peut donc pas assurer que les données seront adaptées pour *toutes* les analyses possibles. Le mieux que l'on puisse faire, c'est de garantir un certain niveau de bruit, une certaine fiabilité du calibrage, etc. Autrement dit, on peut garantir la qualité de la mesure, critère technique qui n'est pas forcément à la portée de l'analyste, et ainsi documenter les limites du capteur. Cette documentation permet de trouver des parades aux défauts de collecte, à travers le post-traitement, l'intégration de plusieurs sources de données, de traitements manuels (enrichissement des données par codage de vidéos par exemple), ou enfin de simplement comprendre pourquoi telle analyse ne semble pas fonctionner ou n'est pas possible.

L'équipe technique n'est pas à blâmer dans la mesure où elle fournit suffisamment d'informations et documente son travail d'instrumentation. Notons dès maintenant que pour comprendre cette documentation qui, comme nous l'avons observé précédemment, est technique et donc pas forcément accessible à l'analyste, il est nécessaire de disposer d'équipes pluridisciplinaires. Nous reviendrons sur l'importance de ce point au fur et à mesure de ce chapitre.

Nous avons vu que des problèmes techniques peuvent apparaître lors de la collecte de données. Pour éviter ces problèmes ou chercher à comprendre leur origine, il est souvent nécessaire de réitérer la collecte, relancer des mesures en modifiant des paramètres. C'est ce que nous appelons une boucle d'appropriation. Un expert, guidé par l'analyste (ici probablement un ingénieur ou un technicien), va relancer plusieurs fois le processus de collecte jusqu'à comprendre l'origine d'un défaut de collecte et le résoudre.

Cette boucle d'interaction lors de la collecte se retrouve à chaque étape de la chaîne de découverte de connaissances : que ce soit au niveau du prétraitement des données, pour compenser les défauts d'une collecte que l'on ne peut réitérer ou pour améliorer les algorithmes de traitements, ou au niveau de la fouille de données, pour trouver les bons paramètres de fouille. Il paraît donc nécessaire de s'appuyer sur un environnement souple, permettant de facilement visualiser et tester telle ou telle hypothèse, ou simplement chercher des indices dans les données.

Comme il s'agit d'un processus exploratoire, il faut être capable de modifier les visualisations et de modifier les post-traitements au fur et à mesure des besoins.

Cela souligne le besoin d'avoir des équipes pluridisciplinaires. Certains post-

traitements peuvent être faits directement par un analyste-ergonome ayant quelques connaissances en traitement du signal, mais des traitements plus poussés tels que des questions de calibration de données, de bogues dans des scripts de traitement, etc., nécessitent d'interagir avec d'autres personnes, typiquement des informaticiens, développeurs ou spécialistes du traitement du signal.

Par expérience, nous avons noté que le travail isolé d'un ergonome d'un côté et d'un informaticien de l'autre ne communiquant l'un avec l'autre qu'à travers un cahier des charges ne fonctionne pas. Il est nécessaire à chacun de connaître un minimum le métier de l'autre pour pouvoir communiquer.

Par rapport au schéma de découverte de connaissances tel que présenté par Fayyad *et al.* (1996b), nous insistons sur le fait que ces itérations sont multiples et interviennent tout au long de la chaîne, de la collecte à la production de connaissances exploitables à l'issue du cycle.

De plus, nous souhaitons expliciter le fait que, dans la pratique, la connaissance n'est pas produite qu'en bout de chaîne, lors de l'interprétation des résultats d'un algorithme de fouille. En effet, l'interprétation se fait tout au long de la chaîne. Une itération est en soi un processus qui crée de la connaissance. Cette connaissance doit donc être gérée, capitalisée et partagée.

Chaque itération crée de la connaissance

Est-il vrai que l'interprétation intervient tout au long de la chaîne et qu'à chaque étape des connaissances sont créées ? Nous allons étayer cette proposition.

Exemple

Un ingénieur qui instrumente la collecte vérifiera que le processus qu'il a mis en place pour collecter les données est fonctionnel. Cela peut aller d'une vérification relativement minimale : vérifier qu'un fichier a bien été créé ou que les paramètres correspondants à différents capteurs sont bien présents, jusqu'à des vérifications plus poussées : recherche d'erreurs dans les données, test de la présence de problèmes d'enregistrements, de champs vides ou incohérents, vérification de la calibration, etc. Ces étapes de vérification sont généralement faites lors de la conception du système de collecte, mais pas remises en cause une fois le dispositif de collecte jugé robuste. Pour effectuer ces différentes vérifications, et pour chercher à comprendre l'origine d'un problème de collecte ou d'un bogue potentiel dans le dispositif de collecte, l'ingénieur visualise les données qu'il possède, c'est-à-dire les données collectées (ou des données intermédiaires). Il fait des tests en recommençant la collecte après avoir modifié le processus de collecte. Il interprète la source possible du problème, fait des hypothèses sur la source du problème et fait des tests confirmant ou infirmant cette hypothèse. Il *interagit* avec le système de collecte et *itère* le processus jusqu'à ce qu'il comprenne la source du problème et le résolve.

Ce processus itératif d'interaction, de visualisation et d'interprétation du résultat mobilise et produit des connaissances. Connaissances sur le processus de collecte, en comprenant mieux le fonctionnement du logiciel et en résolvant des bogues, ou en appréhendant les limites techniques ou technologiques des matériels et des logiciels de collecte. Connaissances sur les données, la signification physique de tel ou tel capteur, son intégrité, sa fiabilité, la structuration logique des données. Connaissances sur l'activité, qui peut parfois sortir du champ attendu ou prévu par le concepteur du dispositif

de collecte (par exemple, un sujet appuie à la fois sur la pédale d'accélération et de freinage).

Exemple

Un ingénieur qui traite le signal issu de la collecte effectue aussi différentes itérations. Il teste différents filtres, avec différents paramètres. Si le résultat d'un traitement s'avère aberrant, il revient en arrière et essaie de comprendre l'origine du problème. L'origine du problème peut venir du processus de traitement du signal (paramètres inadaptés, bogue dans le logiciel, etc.) ou bien des données (défaut dans la collecte), auquel cas, le rebouclage avec l'équipe en charge de la collecte est nécessaire, pour comprendre le problème et le corriger.

Enfin, l'analyste s'approprie lui aussi les données et les outils. L'appropriation des données correspond à l'acquisition de connaissances qui permettent de mieux comprendre les données et ainsi guider les différentes étapes du cycle de découverte de connaissances. L'appropriation des outils permet à l'analyste de mieux exploiter les fonctionnalités de l'outil et en particulier obtenir plus rapidement et plus facilement le résultat qu'il attend.

Exemple

L'ergonome visualise les données sous forme de courbes, regarde attentivement les vidéos correspondant aux cas intéressants pour y chercher des informations complémentaires permettant d'expliquer telle ou telle action du sujet. En regardant ces courbes et ces vidéos, l'ergonome apprend aussi bien à interpréter les données qu'à utiliser l'outil de visualisation. En visualisant les données, il peut découvrir les propriétés des données qui sont pertinentes pour son analyse, relevant directement de l'activité qu'il étudie, ou servant de base à des analyses plus poussées. L'utilisation des outils de visualisation et de traitement des données permet à l'analyste de se familiariser avec ces outils et leur mode de fonctionnement. Il apprend à être plus efficace avec les outils. Il peut découvrir des bogues dans les outils et les moyens de les contourner. L'analyste modifie donc les paramètres de visualisation et ce faisant s'approprie à la fois les outils et les données. Et enfin, c'est le but de sa recherche, il découvre, confirme ou infirme des connaissances sur l'activité de conduite automobile.

À chaque étape de la chaîne, la personne en charge de manipuler les données va chercher à visualiser ces données. Cette visualisation permet de s'approprier les données, d'interpréter la ou les activités que ces données capturent et enfin, de comprendre le fonctionnement du processus de traitement (au sens large : visualisation, manipulation de données aussi bien que de calculs). Ces connaissances sont directement utiles à la personne manipulant les données. Il s'agit des connaissances de l'analyse de données. Quand ces connaissances sont explicitées, elles peuvent être partagées et utilisées pour d'autres analyses.

Au fur et à mesure, la connaissance ainsi créée est plus abstraite : plus proche de l'activité et moins proche des données. En début de chaîne, les connaissances sont plus utiles à l'ingénieur et graduellement, elles deviennent plus pertinentes pour l'ergonome. Les connaissances sur le fonctionnement technique des processus de traitement

sont la spécialité de l'ingénieur. Les connaissances sur la sémantique du traitement et la signification d'une interprétation sont la spécialité de l'ergonome. D'où la nécessaire empathie professionnelle mutuelle pour effectuer l'analyse de données.

6.1.2 Le cycle d'appropriation

Nous proposons donc maintenant le cycle d'*appropriation* (c.f. figure 6.1). L'humain, qu'il soit ingénieur chargé du prétraitement ou analyste, travaille sur un jeu de données. Sur ce jeu de données, il effectue des traitements. Le processus de traitement de données peut correspondre à n'importe quelle étape du cycle de découverte de connaissances. Ces traitements[2] produisent des résultats que l'humain va visualiser. Les visualisations sont de différentes natures : vidéos, tableaux de données, représentations graphiques sous forme de courbes ou de diagrammes, etc. Les possibilités sont multiples et dépendent aussi bien des données, de l'analyse menée que des préférences des l'analyste. La visualisation permet à l'humain de découvrir des connaissances, au sens du cycle de découverte de connaissances, mais aussi d'un point de vue pratique : il s'approprie les outils qu'il utilise et les données sur lesquelles il travaille. Ces nouvelles connaissances servent à guider de nouvelles itérations de ce cycle.

Notons que le processus peut évoluer à chaque itération du cycle d'appropriation. Par exemple si le processus correspond au prétraitement des données, l'appropriation des données peut mener à modifier la sélection des données et réexécuter le même processus sur ce nouveau jeu de données, ou bien à valider le processus de traitement et passer à l'étape suivante du cycle de découverte de connaissances.

Ce cycle est très fortement itératif et interactif. C'est l'interaction avec les outils et les données ainsi que l'itération de ce cycle qui fait que l'humain s'approprie les données. Le cycle peut d'ailleurs être extrêmement court si le « processus » consiste à modifier un paramètre d'affichage ou extrêmement long, s'il s'agit de relancer l'ensemble des traitements et de la fouille sur un nouveau jeu de données. Ce qui nous intéresse dans ce cycle, c'est qu'il explicite l'importance de connaissances pratiques, auxquelles on fait référence comme un « art » d'analyser des données. Nous pensons qu'une interaction bien conçue entre l'humain et le processus de traitement favorise l'appropriation de l'outil et des données, ce qui permet une découverte de connaissances plus rapide, plus efficace et plus pertinente.

Ce schéma est complémentaire du schéma de Fayyad *et al.* (1996b). Il s'agit en fait d'un autre point de vue sur le processus de découverte de connaissances, moins axé sur les données et leur évolution logique prise a posteriori, mais axé sur les interactions lors d'un processus d'analyse médié par un environnement informatique. Un cycle du schéma d'appropriation peut correspondre à une ou plusieurs étapes du schéma de découverte de connaissances, par exemple, lorsqu'une chaîne de traitement est robuste, il est possible de traiter un nouveau jeu de données, du post-traitement à la fouille, avec les paramètres par défaut sans intervention humaine, ce qui constitue une itération du cycle d'appropriation et 3 étapes du cycle de découverte de connaissances. À l'inverse, pour arriver à trouver les bons paramètres d'un traitement, par exemple les paramètres de la fouille, l'analyste va exécuter plusieurs itérations de cette même étape du cycle de découverte de connaissances, ce qui constitue plusieurs itérations du cycle d'appropriation.

[2]Pour simplifier le schéma, nous ne différencions pas dans le « processus » les traitements qui sont purement calculatoires de ceux qui produisent des visualisations, puisqu'au final, l'humain va toujours chercher à visualiser le résultat.

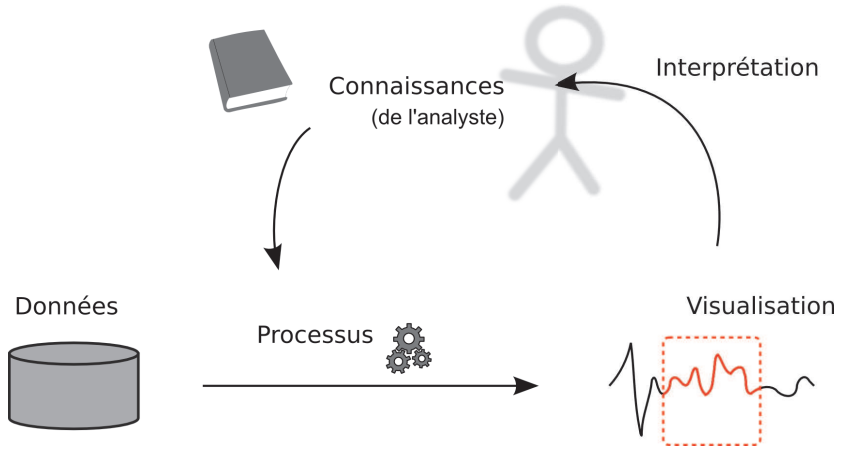


FIGURE 6.1 – Le cycle d'appropriation permet d'expliciter les différentes itérations et interactions qui ont lieu entre les outils d'analyse et de visualisation (le processus) et leur utilisateur (un analyste). Pour étudier un jeu de données, l'humain exploite un processus de traitement. La visualisation du résultat du processus lui permet d'interpréter les données. L'interprétation de ces données aboutit à de nouvelles connaissances, qui sont liées aux données, aux processus de traitement des données ou à l'activité dont ces données sont issues. Ces nouvelles connaissances vont permettre à l'humain de guider le processus de traitement et de visualisation, à l'itération suivante du processus.

6.1.3 Pourquoi expliciter le cycle d'appropriation ?

Présenter un cycle d'appropriation dans notre travail nous permet de souligner l'importance de l'interaction et de souligner le fait que l'analyste mobilise et produit des connaissances qui vont guider tout le processus de découverte de connaissances : il poursuit un objectif précis de production de connaissances. Ce cycle formalise le fait que ces itérations permettent de découvrir des connaissances sur l'activité observée et analysée, comme c'est le but, mais aussi sur les données elles-mêmes et sur les processus de traitement sur les données.

Ce cycle met donc en évidence plusieurs notions clés de la thèse : l'importance de l'interaction, le fait que la découverte est guidée par l'analyste et qu'il va s'approprier différents types de connaissances.

Interaction

C'est le point le plus important, celui que nous souhaitons souligner dans ce travail de thèse. Pour s'approprier l'activité, il est nécessaire à l'analyste de s'approprier les données et les outils. Cela passe potentiellement par de nombreuses itérations. L'analyste apprend à maîtriser les outils aussi bien que les données pour produire l'analyse qu'il souhaite. Il mobilise ses connaissances sur l'activité (connaissances théoriques, venant de la littérature, etc.) et produit des connaissances à partir de son expérience pour analyser les données.

La découverte est guidée

Une exploration de données, quelle qu'elle soit est associée à une motivation[3]. C'est cette motivation qui pousse l'analyste à s'intéresser à telles ou telles données, à faire tel ou tel traitement sur ces données et à visualiser de telle ou telle manière le résultat. Par exemple, même dans le cas où l'on cherche à comprendre l'activité selon toutes ses différentes dimensions et que dans ce but on mesure le maximum de paramètres sur l'activité, on ne pourra pas tout mesurer. En effet, il est toujours possible de trouver des paramètres supplémentaires à ajouter. Il faut donc faire des choix sur les paramètres à mesurer. Ces choix guident la découverte et les connaissances découvertes guident les choix, dans une boucle que l'on espère vertueuse et progressant vers des connaissances robustes sur l'activité. Les objectifs d'une analyse ont donc un rôle crucial à jouer dans le processus de découverte de connaissances. Ces objectifs ne sont pas toujours explicites et ils peuvent changer au cours de l'exploration, c'est-à-dire à chaque itération du cycle d'explicitation. Dans le cas d'une expérimentation, une partie des objectifs est explicite et vise soit à mieux comprendre une activité, soit à valider ou invalider des hypothèses de recherche. Les nouvelles connaissances produites sur l'activité, ou sur les défauts du processus de collecte (nouveau paramètre à collecter pour permettre des analyses plus poussées, défaut du système de collecte, etc.) amènent à la conception de nouvelles expérimentations.

Différentes connaissances sont impliquées

Nous distinguons les différents types de connaissances impliquées dans la découverte de connaissances. L'utilisation d'outils de découverte de connaissances ne se fait pas sans connaissances. Ces connaissances sont de différentes natures. Les connaissances peuvent être implicites ou explicites[4]. Les connaissances sont explicites lorsqu'elles sont verbalisables ou formulées dans une représentation formelle. Les connaissances sont implicites lorsqu'il s'agit de savoir-faire, ou lorsqu'elles sont codées sous une forme nécessitant une interprétation. Les connaissances sur l'activité peuvent provenir de la littérature, elles sont alors formelles et explicites. Elles peuvent provenir de l'analyste, elles sont alors implicites (savoir-faire) ou explicites (expertise). Les données permettent d'inférer des connaissances sur l'activité : la connaissance sur l'activité est alors implicitement contenue dans les données. Les connaissances sur les processus ou les outils d'analyse, elles, sont explicites lorsque le processus est documenté, implicites lorsqu'il faut utiliser la rétro-ingénierie.

Les « connaissances » que le cycle de découverte de connaissances cherche à découvrir sont celles portant sur l'*activité analysée*. Notre démarche propose de prendre du recul et d'inclure d'autres types de connaissances, liées au processus même de découverte : des connaissances sur les données, sur le processus de découverte et sur les outils de découverte. Ce que nous appelons « appropriation », c'est précisément l'émergence de ces connaissances. L'analyste s'approprie d'une part les données et d'autre part les outils de découverte de connaissances.

En synthèse

L'analyste poursuit un objectif qui motive son analyse. Cet objectif le pousse à utiliser des outils de découverte de connaissances et de visualisation pour comprendre l'ac-

[3] Ne serait-ce que la curiosité.

[4] Nous faisons ici une analogie avec la notion d'activité implicite et explicite dont nous avons parlé au chapitre 2.

tivité. En faisant cela, il s'approprie non seulement l'activité, mais aussi les données elles-mêmes et les outils de découverte de connaissances.

Nous pensons que l'interaction facilite l'appropriation. C'est pour favoriser cette appropriation, nécessaire à la production de connaissances, que nous favorisons dans cette thèse l'interaction entre l'analyste et le système informatique.

6.1.4 Travaux en relation

La visualisation interactive de données est reconnue comme un moyen de faciliter la découverte de connaissances pertinentes. Construire ce genre de visualisations est l'objet d'étude de la communauté de la *visualisation d'informations* (*information visualisation* ou *infovis*[5]). La fouille de données elle-même, dont l'objectif est de chercher à automatiser au maximum la découverte de motifs pertinents, ne peut éviter en bout de chaîne l'interprétation de la pertinence des résultats. Si l'interprétation est nécessairement humaine[6], les calculs sur de grandes masses de données sont effectués par l'environnement informatique, étendant les capacités globales de construction de connaissances. Toutefois, le repérage visuel de motifs « intéressants » est fait très efficacement par l'humain, guidé par une motivation échappant à l'environnement informatique tant que ces connaissances, élaborées par l'humain, n'y sont pas représentées par des requêtes ou des contraintes explicites. La fouille de données accorde de plus en plus d'importance à la présentation des données et des résultats à l'analyste, qui, au bout de la chaîne de découverte de connaissances, va interpréter les résultats et découvrir ou non des motifs d'intérêt.

Ces visualisations s'intéressent principalement à la synthèse de grandes masses de données, posant à la fois des questions de représentations graphiques adaptées à de grandes quantités de données et des questions de performances en terme de temps d'exécution. Cela pose aussi des questions en terme d'interaction. Plutôt que de visualiser en une seule fois l'ensemble des données, il est souvent préférable de proposer plusieurs vues ou de proposer un moyen de naviguer au sein d'une même vue.

Nous proposons donc non seulement d'instrumenter la phase de présentation des résultats (phase ultime du point de vue du cycle classique de découverte de connaissances), mais aussi de généraliser cette approche aux autres « moments » de la découverte et en particulier au moment de la production même des résultats : l'étape de fouille.

Comme nous l'avons vu, les interactions interviennent à chaque étape du processus. Nous proposons donc d'instrumenter chaque étape pour produire des connaissances explicites et mobilisables aussi bien pour le calcul que pour l'interprétation humaine.

De ce point de vue, notre positionnement est différent de celui de la communauté de la fouille de données et même de celui de la communauté de la visualisation interactive. Par contre, nous nous sentons beaucoup plus proches des questions de recherches posées par une discipline relativement récente : la visualisation analytique (*visual analytics*)[7]. Les bases de cette discipline, présentées par Thomas et Cook (2005) reposent sur

[5] http://www.infovis.org/

[6] Même dans les cas où l'on cherche à sortir l'humain de la boucle, en définissant des seuils de pertinence ou « d'intérêt », l'humain intervient avant, pour définir ces seuils et y intégrer ses connaissances dans le calcul du seuil, et après, en interprétant les motifs calculés pertinents par la machine.

[7] Cette discipline est née de la volonté de pouvoir anticiper, prévenir ou réagir face à des attaques terroristes sur le territoire américain. Même si l'application de notre travail est bien éloignée des questions de terrorisme, les questions de recherche que nous nous posons sont proches de celles de la visualisation analytique.

l'idée d'une analyse exploratoire, qui est guidée par un ou plusieurs analystes, de jeux de données complexes, multisources, dynamiques et potentiellement contradictoires. L'objectif de cette discipline est, en rassemblant de grandes quantités de données, de retrouver des informations pertinentes pour analyser et comprendre des comportements ou des événements complexes.

6.2 Le cycle de découverte de connaissances révisé au regard de la question de l'analyse de l'activité de conduite automobile

La section précédente nous a permis de mettre en évidence la présence et l'importance de l'interaction au sein de toutes les étapes de la découverte de connaissances. Cette dimension est complémentaire du cycle de découverte de connaissances présenté par Fayyad *et al.* (1996b), en affirmant le caractère exploratoire des analyses menées pendant la découverte de connaissances.

Les connaissances produites à l'aide de cette découverte exploratoire de connaissances peuvent permettre dans un second temps de développer des techniques d'analyse qui déploient toute une série de calculs inspirée du résultat de toutes les itérations précédentes. C'est l'avantage de représenter les connaissances dans le système informatique. Ainsi, un SBT[8] conserve en mémoire la façon de calculer une trace de haut niveau à partir d'une trace issue de capteurs. En conséquence, cette connaissance qui a nécessité de nombreuses itérations pour être construite peut être réutilisée en l'adaptant simplement à un nouveau contexte. Un SBT est par nature un support à la capitalisation de connaissances.

Le fait d'intégrer les connaissances au fur et à mesure dans le système informatique, dans un système à base de connaissances en général, ou dans un SBT en particulier, change la donne du schéma de découverte de connaissances (Fayyad *et al.*, 1996b). Nous proposons ici une version révisée du cycle de découverte de connaissances qui tient compte de l'approche, orientée connaissances et traces que nous adoptons pour la découverte de connaissances issues de la conduite automobile.

6.2.1 De l'expérimentation aux connaissances

Nous souhaitons adapter le cycle de découverte de connaissances au regard de trois points :

1. En amont, les données à partir desquelles on cherche à extraire des connaissances ne sont pas *données*, elles sont le résultat d'une expérimentation ou d'une collecte de données ;
2. L'utilisation d'un Système à Base de Traces (SBT) modifie le cycle et la logique de fouille de données et documente les connaissances produites ;
3. Nous transformons l'étape de fouille pour la rendre interactive : dans le contexte de notre travail, nous l'appliquons à la découverte d'un automate à partir d'un ensemble de traces.

Les sections suivantes argumentent les choix liés à ces trois questions. Le résultat est illustré par la figure 6.5, présenté à la page 124.

[8] Système à Base de Traces

Les données ne sont pas « données » : de l'expérimentation à l'analyse

Le point de départ du cycle de découverte de connaissances, ce sont les données. Les données ne sont jamais données, elles sont toujours collectées d'une manière ou d'une autre. Les données sont le fruit d'une observation, du choix d'observer ou de ne pas observer tels ou tels paramètres. Les données de conduite auxquelles nous nous intéressons par exemple sont le fruit d'une expérimentation. Les chercheurs ont décidé de mesurer certains paramètres de la conduite. Un véhicule ou un simulateur de conduite a été instrumenté pour enregistrer différents paramètres. À l'issue de l'expérimentation (ou plus généralement de la collecte de données), les données sont enfin présentes. Mais plutôt que de parler de données, il serait plus juste de parler « d'observés ».

Si Fayyad *et al.* introduit le cycle de découverte de connaissances en partant des données, c'est parce qu'il s'intéresse à la situation nominale de la fouille de données : la recherche de motifs pertinents au sein de bases de données déjà existantes.

Il nous semble pertinent de modifier le schéma du cycle de découverte de connaissances pour inclure la collecte des données, dans notre cas, une expérimentation. Ajouter ce niveau supplémentaire permet de matérialiser et d'expliciter plusieurs scénarios auxquels l'analyste est confronté :

Exemple

S1. La collecte des données n'a pas fonctionné correctement, il est nécessaire d'effectuer une nouvelle collecte de données. Cela peut être dû par exemple à un capteur défaillant, au scénario de l'expérimentation à améliorer, etc. C'est typiquement le rôle des préexpérimentations de tester la collecte pour corriger ce type de problèmes et permettre des ajustements.

S2. En analysant les données collectées, l'analyste découvre que d'autres données auraient pu être collectées pour approfondir son étude. Cela peut conduire à la création d'une nouvelle expérimentation, visant à collecter ces informations complémentaires.

S3. Après avoir analysé les données, l'analyste valide ou invalide certaines hypothèses. Les zones d'ombre peuvent éveiller sa curiosité. Il a acquis les connaissances nécessaires pour répondre au moins en partie à l'objectif de son expérimentation, mais il veut aller plus loin. Pour cela, l'analyste va préparer une nouvelle expérimentation visant à étudier les zones d'ombre découvertes lors de la première étude.

Le scénario S1 peut sembler trivial, mais dans la pratique, il n'est pas rare de découvrir les problèmes liés à la collecte seulement au moment de l'analyse des données. Comme une collecte de données est coûteuse, ce genre de problèmes est traité dans la mesure du possible par des post-traitements sur les données pour compenser les défauts liés à la collecte. Le scénario S1 correspond donc à une instanciation de la boucle de découverte de connaissances au niveau des données elles-mêmes.

Les scénarios S2 et S3 considèrent la production de connaissances sur les connaissances à découvrir (niveau méta), permettant de guider l'analyse et les calculs : la découverte de connaissances permet d'initier de nouvelles expérimentations qui vont à leur tour permettre de découvrir de nouvelles connaissances. Il s'agit du cycle de modélisation que nous avons présenté à la figure 2.9 du chapitre 2.

Première révision du cycle de découverte de connaissances

Le cycle de découverte de connaissances part des données et commence par sélectionner les sous-ensembles de données pertinents pour la découverte. Nous proposons de généraliser ce motif en incluant la construction des données par l'observation du processus. Cela nous permet d'expliciter la phase de collecte de données. La phase de collecte de données que nous proposons intègre l'étape de sélection des données du cycle de découverte de connaissances. En effet, nous considérons que dans la situation où l'analyse se fait sur des données existantes, la collecte se découpe en deux parties : la collecte initiale, qui a généré les données existantes et la phase de sélection des données pertinentes à l'analyse.

Nous proposons donc de modifier le cycle de Fayyad *et al.* (1996b) en un nouveau cycle, représenté à la figure 6.2, qui inclut la collecte de données.

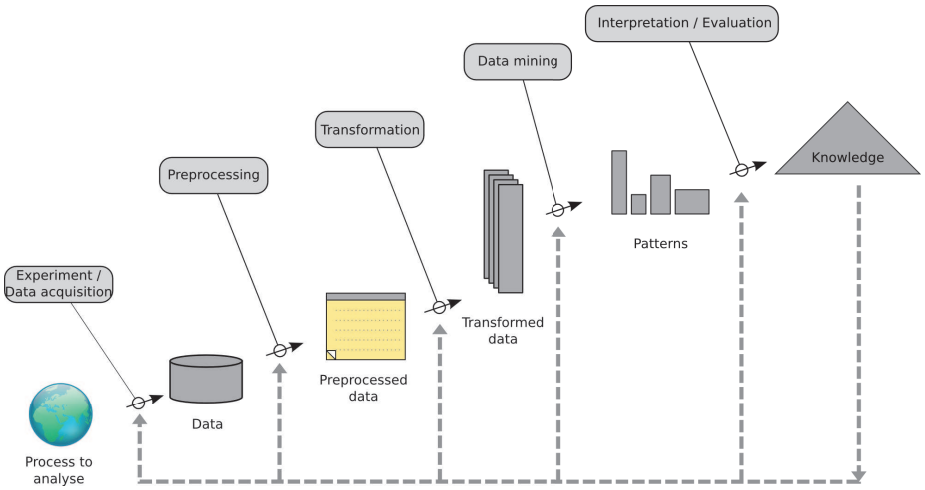


FIGURE 6.2 – Généralisation du cycle de découverte de connaissances, prenant en compte l'acquisition des données initiales. Les données sont issues d'une observation du monde à travers un dispositif de collecte.

6.2.2 Argumentation sur une approche « système à base de connaissances » pour soutenir le processus de découverte

Le cycle de découverte de connaissances met en évidence différentes étapes de préparation de la fouille : sélection des données, prétraitements, transformations, puis enfin, la fouille. Chacune de ces étapes vise à traiter et mettre en forme les données de manière à rendre la fouille productive et pertinente. Sauf à l'issue du processus, à aucun moment ces données ne sont traitées comme sources de connaissances. La sémantique de chaque donnée ne peut pas être définie et encore moins exploitée par les traitements intermédiaires dans le système. Pourtant, comme nous en avons discuté en présentant le cycle d'appropriation, ces données font sens pour l'analyste qui les traite. Les connaissances sur ces données et sur leurs traitements sont seulement présentes dans l'esprit de l'analyste, et de manière implicite, dans les scripts de calculs effectuant les traitements.

Nous souhaitons intégrer ces connaissances de manière explicite dans la boucle de découverte de connaissances. Il nous semble pertinent d'intégrer les connaissances au plus tôt dans le cycle de découverte de données, et non pas seulement à la fin du cycle, lorsque l'analyste interprète les résultats de la fouille.

C'est pourquoi nous proposons d'intégrer un système à base de connaissances, en l'occurrence un SBT, dans cette boucle de traitements. L'utilisation d'un système à base de connaissances permet d'améliorer les capacités d'interaction avec l'analyste, puisqu'à chaque donnée est associée une sémantique[9]. Les connaissances sur les données peuvent être enregistrées, de manière explicite, dans le système à base de connaissances (dans notre cas, dans l'ontologie représentant le modèle de la $\mathcal{M}$-Trace)

Un SBT est capable de sélectionner, traiter et mettre en forme les données. Toutes les étapes du cycle de découverte de connaissances en amont de l'étape de fouille peuvent donc être intégrées dans le SBT. Cependant, en pratique, le SBT dont nous disposons, ABSTRACT, n'est pas conçu pour traiter efficacement des données numériques « continues »[10]. ABSTRACT est conçu pour traiter des données symboliques et donc discrètes. Il n'est pas possible d'intégrer l'étape de prétraitement du cycle de découverte de connaissances au sein du SBT ABSTRACT.

Nous avons donc besoin, en amont du SBT d'effectuer des prétraitements sur les données. Puis, dans un second temps, il faut extraire les points d'intérêts, c'est-à-dire les données symboliques, à partir de l'ensemble des données collectées. Cette extraction de points d'intérêts s'effectue en parallèle de la définition d'un modèle de trace, c'est-à-dire d'une ontologie.

Une fois les données converties sous forme de $\mathcal{M}$-Traces, de nouveaux traitements et transformations peuvent être faits, tirant cette fois parti des connaissances mises dans le système. Le logiciel ABSTRACT facilite l'interaction avec l'expert, à la fois d'un point de vue pratique et théorique ; d'un point de vue pratique, à travers ses capacités de visualisation, sa souplesse d'utilisation et d'un point de vue théorique, par la rigueur de conceptualisation imposée par le développement du modèle de trace en parallèle de l'analyse. Plus qu'un outil de traitement, le SBT ABSTRACT est un outil d'analyse de données. ABSTRACT est en particulier capable de transformer les données, de les reformuler, pour représenter différents niveaux d'abstraction des processus étudiés.

Ainsi, nous proposons d'intégrer un SBT dans le cycle de découverte de connaissances (voir figure 6.3) pour permettre d'expliciter les connaissances au plus tôt dans la boucle de connaissances, et non pas à la toute fin, après l'étape de fouille. Le SBT ABSTRACT répond aux exigences d'interactivité qui font partie de notre démarche. L'utilisation d'un tel système nécessite en amont de prétraiter les données, d'en extraire des points d'intérêts et de construire des modèles afin de pouvoir les intégrer dans le SBT. Mais, à l'inverse, l'étape de « transformation » peut se faire au sein du SBT. En outre, le SBT permet d'analyser les données et donc permet d'aller plus loin dans les possibilités de fouille en gérant les problèmes de niveaux d'abstraction et de sémantique qui sont difficiles à traiter automatiquement au niveau de la fouille.

[9] Cette sémantique est souvent élémentaire au début du processus, mais peut être enrichie tout au long de l'analyse si nécessaire.

[10] Les données informatiques ne sont jamais continues à proprement parler, puisqu'elles sont échantillonnées.

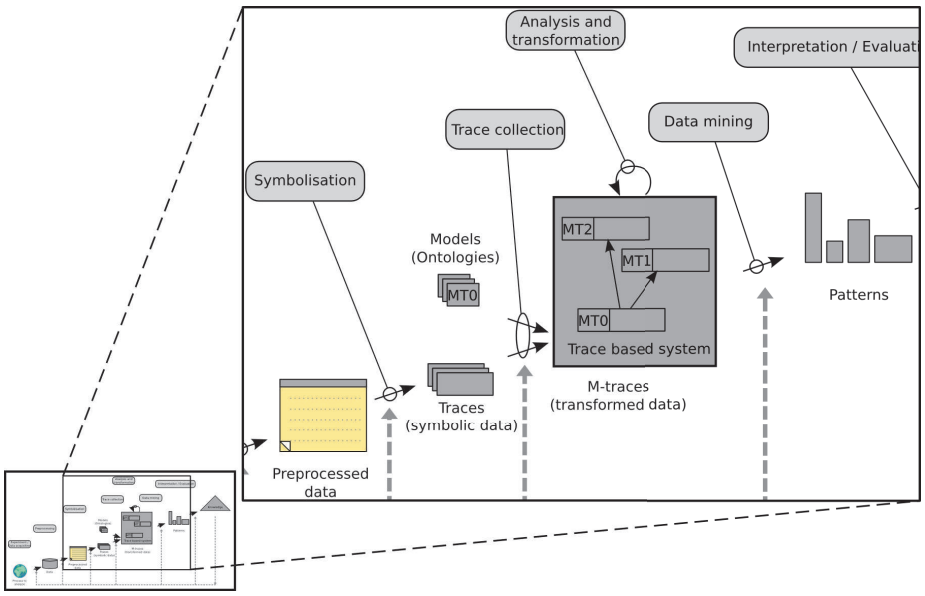


FIGURE 6.3 – Intégration d'un SBT dans le cycle de découverte de connaissances. Une étape de symbolisation construit des traces. Ces traces associées à des modèles ($\mathcal{M}$-Traces) sont importées dans le SBT. Le SBT permet d'effectuer des analyses, transformations et abstractions de $\mathcal{M}$-Traces, tout en profitant de la sémantique des modèles. Enfin, l'étape de fouille produit des connaissances candidates qu'il faut interpréter.

6.2.3 Rendre interactive l'étape d'interprétation des connaissances

Une fois les données introduites et traitées dans le système à base de connaissances, le cycle de découverte de connaissances pourrait s'arrêter, puisque l'on peut potentiellement déjà créer des connaissances. Cependant, il s'avère utile d'utiliser des techniques de fouilles classiques n'exploitant pas nécessairement les connaissances intégrées au SBT. Ces techniques de fouille produisent des connaissances candidates qui doivent ensuite être interprétées. Le cycle de découverte de connaissances ne s'arrête donc pas au système à base de connaissances.

Nous proposons de rendre interactive l'étape d'interprétation de données. Cela signifie deux choses : nous proposons des outils pour faciliter l'interprétation et ces outils sont interactifs et visent donc à assister l'analyste dans sa tâche de compréhension de l'activité étudiée.

Nous nous appuyons sur l'algorithme α_i qui permet de produire un modèle de type réseau de Petri. En effet, le modèle résultant est une représentation de connaissances, puisqu'une sémantique est associée à la structure des réseaux de Petri et que chaque transition est associée à une sémantique définie dans les modèles de traces. Nous avons choisi ce type de modèle dans l'objectif de faciliter l'appropriation du modèle. Les connaissances capturées par le modèle sont explicites et accessibles à l'analyste, contrairement à des modèles de type « boîte noire » (par ex., réseaux de neurones) où la connaissance capturée par le modèle est encapsulée et inaccessible à l'analyste.

En rendant l'étape d'interprétation interactive, nous voulons faciliter l'appropria-

tion du processus de construction du modèle (algorithme α_i). Contrairement à des approches de type algorithmes génétiques, la méthode de construction de connaissances est explicite et déterministe. En rendant interactive l'étape d'interprétation de connaissances, nous espérons faciliter l'appropriation de l'algorithme α_i par l'expert.

Autrement dit, cette étape d'interprétation interactive a pour objectif de diminuer la rupture de charge dans l'appropriation des connaissances produites qui interviendrait si on avait une étape incompréhensible par l'analyste.

Une interaction est « bien conçue » si elle permet de réduire cette rupture de charge. Nous proposons que cette interaction intègre des contraintes ou des requêtes formulées par l'analyste ou s'appuie sur des résultats intermédiaires de l'algorithme, en permettant à l'analyste de visualiser, interpréter, valider ou invalider ces résultats intermédiaires. Il est important que cette interaction puisse faciliter la compréhension de l'algorithme par l'analyste. Il faut donc que les requêtes, contraintes ou résultats intermédiaires soient présentés dans un formalisme qui lui soit familier.

La formulation de requêtes ou de contraintes ne modifie pas le cycle de découverte de connaissances, puisqu'il s'agit de paramètres de la fouille. Mais le fait d'interagir avec l'algorithme de fouille, en particulier en agissant sur des résultats intermédiaires, rajoute une sous-étape (ou de multiples sous-étapes) que nous explicitons à la figure 6.4.

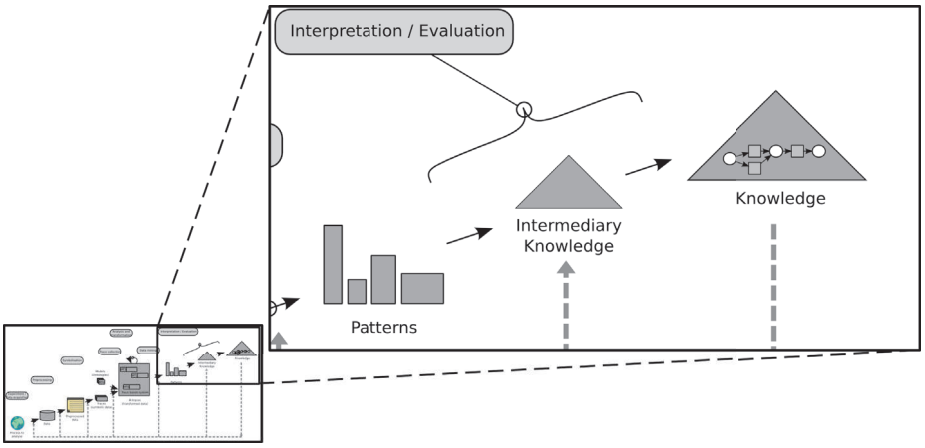


FIGURE 6.4 – Ajout d'une étape d'interprétation interactive des connaissances dans le cycle de découverte de connaissances.

6.2.4 Le cycle de découverte de connaissances révisé

Nous avons proposé de réviser le cycle de découverte de connaissances selon trois dimensions : 1) la prise en compte de la collecte initiale des données, 2) l'intégration d'un système à base de connaissances, un SBT, qui permet d'exploiter les connaissances produites plus tôt dans le cycle, et 3) l'ajout d'une étape d'interprétation interactive des connaissances candidates produites par la fouille.

La figure 6.5 présente le cycle révisé complet. La figure 6.6 présente une vision plus synthétique du cycle de découverte de connaissances révisé.

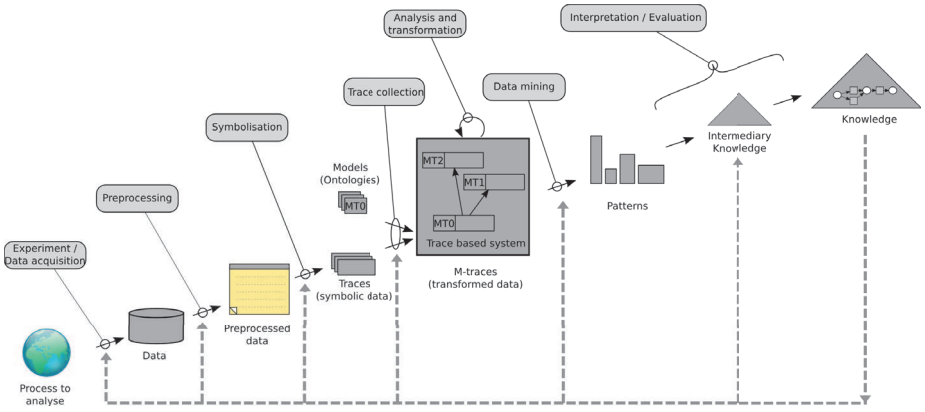


FIGURE 6.5 – Cycle de découverte de connaissances révisé pour tenir compte : de la collecte initiale des données, de l'utilisation d'un SBT pour exploiter au plus tôt les connaissances, et d'une étape pour l'interprétation interactive des connaissances.

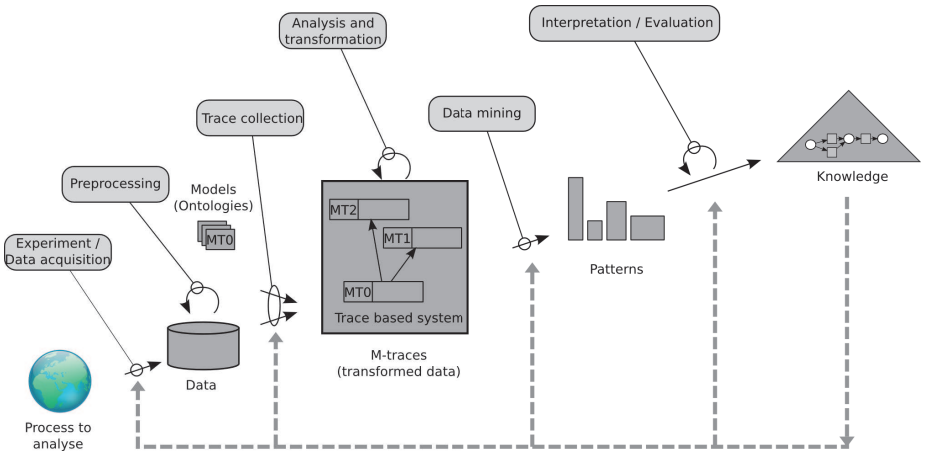


FIGURE 6.6 – Vue synthétique du cycle de découverte de connaissances révisé. Ce cycle révisé tient compte : de la collecte initiale des données, de l'utilisation d'un SBT pour exploiter au plus tôt les connaissances, et d'une étape pour d'interprétation interactive des connaissances.

6.2.5 Discussion

Le cycle de découverte de connaissances que nous proposons est une révision du cycle proposé par Fayyad *et al.* (1996b). Cette révision prend en compte le fait que nous utilisons un système à base de connaissances, un SBT. Le SBT permet d'intégrer, au plus tôt dans le cycle, des connaissances permettant de rendre l'analyse plus explicite. Nous avons aussi révisé la phase d'interprétation des connaissances, en la rendant interactive, avec un algorithme de découverte d'automates. Nous appliquons ce cycle révisé à notre contexte d'application : l'étude de la conduite automobile et en particulier des situations critiques.

Le cycle que nous proposons est donc une révision du cycle de découverte de connaissances. Il est conçu pour être adapté à d'autres contextes d'analyse de données. L'exemple de l'analyse de l'activité de conduite automobile, activité complexe et dynamique, est représentatif pour analyser de nombreux autres types de données, issues de l'activité humaine en premier lieu.

La découverte d'automates est liée au contexte de la recherche, mais il est tout à fait possible de remplacer cette étape de la chaîne de découverte de connaissances par d'autres algorithmes de fouille. Il est sans doute nécessaire de concevoir des variantes interactives des algorithmes de fouille déjà existants, au moins pour piloter la fouille par des contraintes (voir par ex., les travaux de Cram *et al.*, 2011, sur la fouille de chroniques).

6.3 Application : comment analyser les comportements de conduite automobile ? Une proposition de méthodologie

6.3.1 Introduction

Dans la section précédente, nous avons montré comment les différentes étapes d'analyse des comportements de conduite s'enchaînent d'un point de vue théorique. Mais certains problèmes pratiques restent entiers. Les données de conduite représentent des heures d'enregistrement, des giga-octets de données : comment analyser cette masse de données ? Comment interagir sur cette grande quantité de données dans un temps humainement acceptable et sans noyer l'analyste sous la quantité de données ? Et comment atteindre un niveau d'abstraction qui décrive l'activité de conduite automobile, et pas simplement l'état de capteurs, à partir de données qui ne sont que des observés issus de l'instrumentation d'un véhicule ?

Nous allons proposer dans cette section une méthodologie pour articuler cette approche conceptuelle à l'analyse de comportements de conduite. Nous verrons comment nous articulons l'approche descendante (*top-down*) et l'approche ascendante (*bottom-up*). En particulier, nous allons insister sur deux points, essentiels à cette thèse : 1) comment préparer, transformer, reformuler, abstraire les données pour analyser l'activité de conduite automobile à partir de simples données issues de capteurs, et ainsi préparer la fouille de données, et 2) comment exploiter un algorithme interactif de fouille de processus, tels que ceux présentés au chapitre 5, pour modéliser des comportements du conducteur automobile.

6.3.2 Comment préparer l'étape de synthèse de processus

Le problème de la quantité de données et du niveau d'abstraction

Nous l'avons vu au chapitre 2, les données collectées dans le cadre de l'analyse de l'activité de conduite automobile rassemblent des giga-octets de données provenant de capteurs instrumentant le véhicule. Ces capteurs permettent de capturer un certain nombre d'actions du conducteur (action sur les pédales, etc.), ou leurs effets sur le véhicule (vitesse, régime moteur, accélérations, etc.) et l'interaction avec l'environnement, l'infrastructure ou les autres usagers de la route (avec des télémètres, des vidéos, etc.). Les données numériques représentent à elles seules, sans compter la vidéo, des centaines de méga-octets pour une heure de conduite. Plus que le volume de données, la multiplicité des différentes sources de données, le nombre de paramètres collectés et la complexité des données (grande diversité de situations de conduite, de styles de conduite, etc.) induisent des difficultés d'analyse. Il est difficile de savoir a priori quelles informations constituent du bruit et quelles informations sont pertinentes par rapport à l'objectif d'analyse. D'autant plus que les paramètres collectés relèvent davantage de l'état du véhicule que des comportements de conduite.

Les données collectées ne représentent pas directement les comportements du conducteur. Pour étudier la cognition du conducteur, ou même les comportements de conduite, il faut donc effectuer de nombreuses abstractions et inférences.

Exemple

En fonction de l'expérimentation, nous pouvons connaître à l'avance l'itinéraire, mais comment savoir si le conducteur a déjà pris en compte le fait qu'il arrivait sur une intersection, sur un virage ? Comment savoir si le conducteur est distrait ? Comment savoir si le conducteur effectue les actions appropriées à la situation de conduite ? Toutes ces questions relèvent d'un niveau d'abstraction bien différent de celui des données collectées.

Sans connaissances supplémentaires sur l'activité de conduite automobile, sur la psychologie humaine, sur le fonctionnement d'une voiture, sur l'environnement routier, l'analyse serait impossible. Mais l'analyste dispose de ces connaissances. Pour pouvoir découvrir des connaissances relatives à l'activité de conduite à partir des données de conduite dont nous disposons, il est donc nécessaire d'exploiter ces connaissances. Nous proposons de combiner l'approche de découverte de connaissances à partir de données, approche ascendante (*bottom-up*), avec les connaissances provenant de l'analyste, approche descendante (*top-down*). Dans la pratique, les deux approches s'articulent : l'analyse du problème (approche descendante) prépare et guide la recherche de nouvelles connaissances dans les données (approche ascendante) ; et inversement les connaissances découvertes à partir des données (approche ascendante) permettent de mettre à jour les modèles et théories et ainsi alimenter une nouvelle analyse du problème (approche descendante).

La figure 6.7 met en perspective cette articulation entre les approches ascendante et descendante dans le cadre du cycle d'appropriation. La méthodologie que nous proposons dans cette section s'appuie elle aussi sur ce couplage entre approches ascendante et descendante.

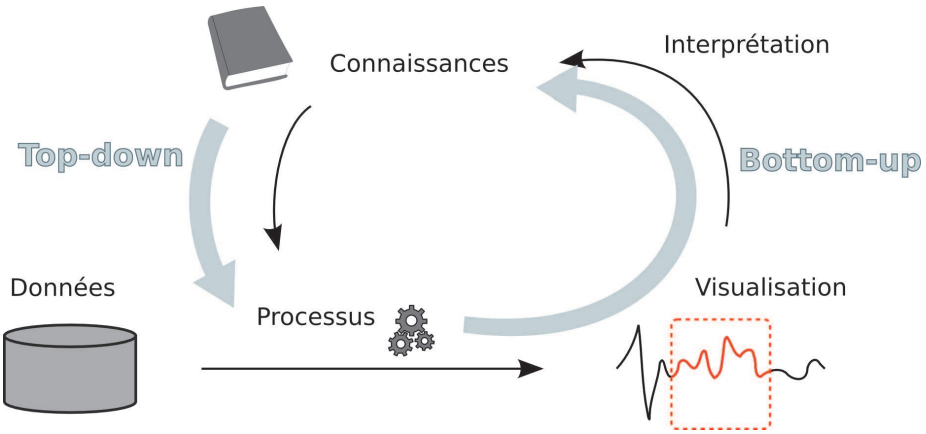


FIGURE 6.7 – Cycle d'appropriation des connaissances. À partir de connaissances explicites (littérature) et implicites (savoir-faire), l'analyste définit comment analyser les données. C'est l'approche descendante. Les données sont traitées et visualisées par l'analyste, qui va ensuite interpréter ces visualisations pour comprendre les données. C'est l'approche ascendante. Les deux approches, ascendante et descendante, s'alimentent l'une l'autre dans un cycle qui crée de la connaissance.

Le découpage de l'activité : une approche descendante

Nous allons ici montrer comment nous pouvons exploiter les connaissances disponibles sur l'activité de conduite automobile pour structurer la découverte de connaissances. Comme nous allons le voir, cette approche est un travail intellectuel, qui permet de définir, sur le papier, quels sont les différents éléments impliqués dans l'activité de conduite automobile. Méthodiquement, il s'agit de redescendre jusqu'aux données collectées et définir comment calculer ou inférer une partie de l'activité de conduite à partir de données issues d'une instrumentation.

Bien entendu, il n'est pas question d'étudier, d'un coup, toute l'activité de conduite automobile. Nous devons découper le problème en sous-problèmes intelligibles par l'analyste. C'est une chose naturelle de découper le problème en sous problèmes. L'activité de conduite automobile est une activité complexe et elle ne fait pas exception : les chercheurs étudiant cette activité vont soit isoler certains paramètres pour se focaliser sur un aspect précis de la conduite automobile (par exemple, étude d'un temps de réaction, en situation contrôlée, entre le moment où le feu tricolore devient orange et le moment où le conducteur commence à freiner), soit ils vont chercher à étudier l'activité dans sa complexité et définir des sous-activités ou des sous-tâches.

L'analyse de tâches Le découpage de l'activité de conduite en tâches et en sous-tâches est un champ de recherche en soi. C'est ce en quoi consiste « l'analyse de tâches » (*task analysis*) : l'analyse de tâches permet de dresser une taxinomie de l'activité de conduite, découpée en tâches et sous-tâches. L'analyse de tâches permet donc de partir de concepts de haut niveau et de descendre jusqu'à des tâches plus unitaires, qui sont celles que l'on peut observer, à l'aide ou non d'une instrumentation. Cependant, l'analyse de tâches n'est pas toujours adaptée à l'analyse que l'on cherche à effectuer d'une situation de conduite. En effet, elle s'intéresse à des situations de références et

se veut objective. Plus précisément, l'analyse de tâches va chercher à décrire quelles sont les tâches que le conducteur est censé effectuer pendant la conduite et représenter sous forme d'erreurs ou de remarques les autres activités dans lesquelles le conducteur s'investit. L'analyse de tâches peut donc se révéler pertinente pour décrire les situations de références, les situations normales, mais elle n'est pas adaptée à l'étude détaillée de situations atypiques ou critiques, ni à l'étude de la situation de conduite du point de vue, subjectif, du conducteur. Par exemple, le conducteur n'a pas vu qu'une voiture s'apprêtait à le dépasser et considère la situation comme normale quand il commence son dépassement : la situation sera normale du point de vue du conducteur, elle est critique du point de vue « objectif » de l'analyse de tâches.

L'analyse de tâches est donc un point de départ pour étudier les situations normales et l'analyste peut utiliser cette méthode pour proposer une première taxonomie adaptée aux objectifs d'analyse qu'il se pose.

Limite des taxonomies : tout n'est pas catégorisable Un problème se pose lorsque l'on cherche à définir une taxonomie des tâches : les situations de conduite sont complexes et il n'existe pas nécessairement de rupture nette entre une tâche et une autre tâche.

Exemple

Une action du conducteur, telle que *décélérer* peut se décrire de diverses manières. Le conducteur peut décélérer en lâchant la pédale d'accélération, en rétrogradant, en appuyant sur la pédale de frein... Maintenant si on met cette sous-tâche en lien avec un contexte, il est normal qu'un conducteur décélère en approche de virage, peu importe l'action qu'il effectue pour décélérer. Mais dans d'autres situations, la manière de décélérer aura un impact et ne sera pas neutre sur la tâche en cours. Un coup de frein en milieu de virage peut être jugé comme une tentative de récupération d'une situation critique : par exemple le conducteur avait mal anticipé la courbure du virage. Et cette même action de freinage dans un virage peut devenir normale en situation de suivi quand le véhicule précédent freine.

Cet exemple vise à illustrer la complexité du problème et le fait que de nombreuses dimensions entrent en jeu dans l'analyse de l'activité de conduite. Dans l'exemple précédent, les différentes dimensions évoquées sont : la dimension tactique vs opérationnelle de l'activité (décélérer vs relâcher l'accélérateur, rétrograder ou freiner), l'infrastructure (entrée de virage vs pendant le virage) et l'interaction avec l'environnement (seul vs en suivi).

Mais il n'est pas possible de trouver une taxonomie universelle capable de répondre à toutes les questions. Un certain nombre de critères ne sont pas fixes, mais dépendent de la situation, du conducteur (son expérience ou ses éventuels déficits), de l'infrastructure et même de la culture (limitations de vitesse différentes d'un pays à un autre, habitudes ou coutumes de conduite différentes, etc.)

Cette limite des taxonomies nécessite de faire des choix, plus ou moins arbitraires, permettant de fixer la différence entre concepts (par exemple la distance maximale au véhicule précédent à partir de laquelle on considère que le conducteur ne suit pas un véhicule).

Autres critères pour décomposer l'activité L'activité de conduite est complexe, l'analyste cherche à la décomposer pour structurer la construction de connaissances. Nous avons vu que l'approche taxonomique, notamment celle de l'analyse de tâches peut être un bon point de départ, mais possède ses propres limites.

Nous proposons une approche plus souple, compatible avec l'approche taxonomique. De la même façon que dans l'analyse de tâches, nous proposons de découper l'activité selon différents critères *objectifs* : infrastructure routière, différentes étapes d'une manœuvre, etc., mais aussi *subjectifs*, c'est-à-dire dépendants du point de vue du conducteur (par exemple, la manœuvre est-elle consciente/explicite ou inconsciente/implicite) ou de la situation de conduite, c'est-à-dire la corrélation de tous les paramètres qui font que chaque situation de conduite est unique : combinaison de l'infrastructure routière, des manœuvres engagées par le conducteur, du trafic environnant, des conditions météorologiques, et potentiellement aussi du passé (les situations de conduite rencontrées précédemment)...

Ces critères induisent différentes *dimensions d'analyse*, correspondant à des points de vue différents sur l'activité, que nous proposons d'organiser dans une ontologie sous forme de concepts. L'ontologie ainsi créée, dont nous donnerons un exemple au chapitre 7, permet de capturer l'ensemble de ces dimensions et de les organiser de manière relativement souple. Une ontologie pose les mêmes problèmes qu'une taxonomie en termes de difficulté de catégoriser les concepts, mais elle permet d'exprimer plus de sémantique. Le système informatique, à travers les SBT, peut ainsi exploiter cette sémantique pour faciliter l'analyse.

Le développement de l'ontologie permet de matérialiser un certain nombre de connaissances (ou d'hypothèses de travail) sur l'activité de conduite automobile. Ce travail conceptuel est extrêmement important pour analyser des données de conduite. Il permet à la fois à l'analyste de mieux savoir ce qu'il recherche en le formalisant explicitement et aussi de donner un cadre permettant de savoir comment inférer des concepts de haut niveau à partir de concepts de plus bas niveau. Le développement de l'ontologie est donc un travail top-down permettant de faire le lien conceptuel entre l'activité étudiée et les données. L'ontologie, issue d'une approche descendante, permet de guider l'approche ascendante d'analyse de données.

L'analyse de données : une approche ascendante

Le travail conceptuel seul n'est pas suffisant à l'analyse de l'activité. Encore faut-il pouvoir, en pratique, analyser et interpréter les données. L'interprétation des données suit une approche ascendante.

L'analyse des données nécessite un travail de transformation et de reformulation permettant, à partir des données collectées, de bas niveau, d'inférer des analyses de haut niveau, décrivant l'activité. L'activité de conduite automobile est complexe et les données collectées sont trop peu nombreuses pour pouvoir appliquer des méthodes de catégorisation automatique et découvrir la structure cachée de l'activité de conduite à partir d'outils de catégorisation (*clustering*). Le travail de conceptualisation décrit précédemment est donc nécessaire. L'ontologie permet de guider l'analyste dans sa façon de transformer les données de bas niveau en reformulations de haut niveau d'abstraction. Ce principe d'abstraction de l'activité à partir de données de conduite est décrit en détail dans la thèse de Georgeon (2008). Trois étapes sont nécessaires : a) effectuer le traitement du signal nécessaire pour traiter le bruit et calculer des indicateurs à partir des signaux collectés, b) extraire des données, continues ou discrètes, éléments nécessaires à la description symbolique de l'activité (dont le vocabulaire est décrit dans

l'ontologie), c) effectuer les transformations et reformulations nécessaires pour exprimer l'activité au niveau d'abstraction requis (en l'occurrence au sein du logiciel ABSTRACT).

En pratique, ces trois étapes, l'étape de traitement du signal, l'étape de discrétisation et l'étape de transformations sont itératives. Les signaux sont traités, discrétisés et transformés dans ABSTRACT. Si certaines reformulations ne sont pas possibles avec les données symboliques intégrées dans ABSTRACT, l'analyste effectue les traitements et les discrétisations nécessaires pour intégrer ces données manquantes dans ABSTRACT. Parfois, ce rebouclage n'est pas possible, car les données nécessaires n'ont pas été collectées. Dans ce cas, il est parfois possible d'analyser la vidéo, souvent par l'analyste lui-même. Ce codage des données à partir de la vidéo consiste à ajouter des marqueurs pertinents dans les données au fur et à mesure du visionnage de la vidéo. Enfin, s'il n'est pas possible d'enrichir les données avec les informations nécessaires aux reformulations des données, alors il est nécessaire de procéder à une autre expérimentation permettant de collecter ces données manquantes.

6.3.3 Comment exploiter l'étape d'interprétation et boucler avec les étapes précédentes

Une fois les données transformées et reformulées de manière à décrire l'activité de conduite telle que l'analyste la considère, l'étape d'interprétation peut commencer. L'outil d'interprétation que nous proposons est interactif et vise à découvrir, à partir d'un ensemble de traces décrivant une activité, un modèle de type réseau de Petri de cette activité. L'algorithme sur lequel nous nous basons, l'algorithme α_i, dérivé de l'algorithme α, permet de reconstruire *un* réseau de Petri représentant le processus, à partir d'un ensemble de traces d'exécution de ce processus.

L'algorithme d'interprétation n'est pas magique. Il ne sait pas trier les données pertinentes de celles qui ne le sont pas ; il ne sait pas découper les différents niveaux d'abstraction de l'activité par lui-même ; il ne sait pas regrouper des ensembles de tâches et de sous-tâches pour mieux structurer le modèle. L'interprétation arrive en bout de chaîne. Afin d'effectuer une interprétation maîtrisée, il est nécessaire de connaître quelques propriétés de l'algorithme que nous utilisons. Les deux propriétés principales qui influent sur la façon de préparer le traitement sont les suivantes : 1. Chaque trace correspond à une exécution du processus à modéliser. 2. Chaque type d'élément observé (ou *obsel*) sera représenté par une et une seule transition du réseau de Petri, chaque transition du réseau de Petri correspondant à un et un seul type d'*obsel*s.

Traces du processus à modéliser

Chaque trace en entrée du processus d'interprétation correspond à une exécution du processus à modéliser. Si l'on fournit à l'algorithme des traces complètes de conduite, alors le processus étudié est l'activité de conduite dans son ensemble. Il y a peu de chances que le résultat soit pertinent, car le processus est trop complexe et l'algorithme ne pourra pas distinguer la pertinence des différents éléments observés en fonction des différentes situations de conduite.

Nous proposons de découper le problème en sous-problèmes plus facilement maîtrisables : modéliser une sous-activité particulière, correspondant à une tâche de conduite ou à une situation de conduite. Comme chaque trace fournie à l'algorithme doit correspondre à une exécution du processus à modéliser, cela signifie qu'il faut segmenter chaque trace de conduite pour en extraire les traces d'exécution du processus

étudié. Par exemple, pour modéliser l'activité dépassement sur autoroute, il faut extraire chaque situation de dépassement sur autoroute, créant ainsi autant de traces que de dépassements. C'est sur cet ensemble de traces extraites que l'on propose d'exécuter la fouille (voir figure 6.8).

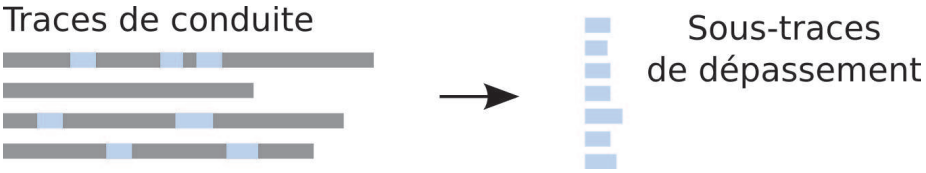


FIGURE 6.8 – L'algorithme d'interprétation que nous exploitons produit un modèle du processus observé à partir de l'ensemble des traces d'exécution de ce processus. Ainsi, pour modéliser un phénomène particulier, le dépassement sur autoroute, à partir de traces de conduite, il faut extraire des traces complètes de conduite, les situations correspondant aux dépassements.

Il faut donc que l'analyste définisse les situations d'intérêt, conceptuellement d'une part : « que modéliser ? » ; et en pratique d'autre part : « comment repérer ces situations dans les données ? » Avec le travail de conceptualisation fait en amont, les situations d'intérêts sont définies conceptuellement. La reformulation des données permet, par le calcul, de les définir en pratique. S'il n'est pas possible de faire ce travail par le calcul, alors l'analyste peut avoir recours au codage manuel, à partir de la vidéo, pour segmenter l'activité en différentes phases.

Un type d'*obsel*s de la trace = une transition du réseau de Petri

Avec l'algorithme α_i, chaque type d'*obsel*s est représenté par une et une seule transition, et inversement. Cela implique que : les traces d'exécution du processus observé doivent contenir les types d'*obsel*s caractéristiques du processus, et seulement les types d'*obsel*s caractéristiques du processus. Si d'autres types d'*obsel*s, non caractéristiques du processus sont présents, alors le modèle découvert sera bruité. S'il manque des types d'*obsel*s, alors le modèle découvert sera au mieux incomplet, au pire inutilisable.

Il est donc important pour l'analyste de connaître les mécanismes impliqués dans un processus pour pouvoir l'observer et donc le modéliser correctement. Cette connaissance relève du travail conceptuel qui est formalisé dans l'ontologie. Cependant, comme la découverte de connaissances est exploratoire, l'analyste ne dispose pas de toutes les connaissances a priori. Lorsque l'étape d'interprétation produit un modèle qui est insatisfaisant, l'analyste bouclera, soit en interagissant différemment à l'étape de fouille, soit plus en amont, en remontant à l'analyse des traces.

6.3.4 Bilan

La méthode que nous proposons pour analyser les données de conduite automobile se résume en trois points :

1. Analyse conceptuelle du problème (approche descendante) :
    - Découpage de l'activité en sous-activités, par rapport au contexte et en fonction des situations d'intérêt ;

- Définition d'une ontologie.

2. Analyse des données (approche ascendante) :
   - Traitement du signal et calculs numériques ;
   - Transformation des traces numériques en traces symboliques ;
   - Transformation, reformulation des traces pour représenter d'autres niveaux d'abstraction.
3. Construction de modèles d'activités :
   - Définir le processus que l'on cherche à modéliser ;
   - Définir les types d'*obsel*s pertinents pour modéliser ce processus ;
   - Extraire des traces de conduite les « sous-traces » correspondant au processus étudié.

Chacun de ces points correspond à une étape du processus de découverte de connaissances. Comme nous l'avons décrit au début de ce chapitre, ces étapes sont itératives, interactives et articulent l'approche descendante et l'approche ascendante (figure 6.7, p. 127). De nombreuses itérations de ces boucles sont effectuées afin de permettre à l'analyste de mieux s'approprier les données et le problème.

6.4 La question de la validation

La question de la validation d'un travail d'ingénierie des connaissances relève de l'épistémologie. À notre connaissance, il n'existe pas de travaux s'attaquant de manière spécifique à la validation de travaux de recherche en ingénierie des connaissances. Nous ne disposons donc pas de « règle d'or » pour valider notre travail. Cela ne nous dispense en rien de valider scientifiquement notre travail.

En préambule, nous allons présenter notre façon de considérer la question de la validation. Nous exposerons ensuite différentes méthodes pour valider notre travail, portant à la fois sur la validation des outils de découverte de connaissances et sur la validation des connaissances produites. Pour terminer, nous exposerons et discuterons la validité de notre contribution.

6.4.1 Valider un travail de recherche en Ingénierie des Connaissances

Une table ronde lors de la conférence « Ingénierie des Connaissances 2011 » portait spécifiquement sur « les questions épistémologiques de la recherche en Ingénierie des Connaissances ». Plus spécifiquement, il s'agissait de définir :

1. Qu'est-ce qu'une contribution scientifique en ingénierie des connaissances ?
2. Quelles sont les méthodes pour produire des connaissances en ingénierie des connaissances ?
3. Et enfin, comment valider une connaissance produite et quelles sont les différentes méthodes de validation ?

Toutes ces questions sont essentielles pour valider un travail de recherche en ingénierie des connaissances. À défaut de disposer de méthodes de validations partagées et bien établies, nous proposons de nous servir des discussions issues de cette table ronde pour définir les méthodes à utiliser pour valider notre contribution. Nous nous référons spécifiquement à la présentation de Gandon (2011), et aux notes de la table ronde, disponibles sur le Web[11].

Dans la suite, nous montrerons comment valider notre contribution. Pour cela, nous posons d'abord deux questions : quelle part de notre contribution faut-il valider ? Quelle est la contribution que nous apportons en ingénierie des connaissances ? Nous verrons qu'un travail en ingénierie des connaissances se valide à différents niveaux : au niveau des outils d'ingénierie des connaissances et au niveau des connaissances produites à l'aide de ces outils. Nous étudierons ces deux questions.

Que faut-il valider ?

Il est nécessaire de valider deux éléments : les outils et les connaissances produites. L'outil d'ingénierie des connaissances est l'objet de recherche du chercheur en ingénierie de connaissances, et l'outil de recherche de l'ergonome, dans le contexte d'application de la thèse. Les connaissances produites sont l'objet de recherche de l'ergonome et le résultat de la mise en œuvre des outils développés à la suite des recherches de l'ingénieur des connaissances.

Nous aborderons ces deux points différemment, puisque l'un relève de la recherche en ingénierie des connaissances et l'autre relève, nous le verrons, à la fois de l'ingénierie des connaissances et de la discipline dans laquelle on cherche à produire des connaissances.

6.4.2 Valider les outils d'ingénierie des connaissances

Valider les outils d'ingénierie des connaissances est une question qui relève de l'ingénierie des connaissances. Par conséquent, c'est une question transdisciplinaire, qui nécessite d'aller chercher des méthodes de validation issues des autres disciplines impliquées. Dans notre cas, les mathématiques, l'informatique, l'ergonomie, etc., et bien entendu, l'épistémologie. L'ensemble des méthodes de validation que nous proposons ici n'est pas exhaustif. Ce n'est pas l'objet de cette thèse. Nous avons choisi de présenter les méthodes qui nous semblent pertinentes dans le cadre de notre travail.

Validation formelle : démonstration mathématique

Cette validation consiste à démontrer les propriétés mathématiques d'un algorithme. C'est le cas par exemple de l'algorithme α, pour lequel il a été démontré qu'il est capable de retrouver le modèle d'un processus à partir de ses traces d'exécution (sous certaines conditions sur le modèle et les traces). Quand un théorème est démontré mathématiquement, il suffit de s'assurer que toutes les conditions requises pour l'utiliser sont réunies. Le résultat est alors valide formellement, il n'a pas besoin d'être redémontré.

[11] Présentation de Gandon : http://fabien-gandon.blogspot.com/2011/05/les-questions-epistemologiques-de-la.html (accédé en septembre 2011)
Notes de la table ronde : http://ic2011.liris.cnrs.fr/site/doku.php?id=tableronde (accédé en septembre 2011)

Validation épistémologique : validité par construction

La validation épistémologique consiste à vérifier que la méthode proposée vérifie point par point les attentes qui ont suscité le développement de la méthode. C'est l'équivalent de la validité de construit (*construct validity*) (Anastasi, 1976) utilisé pour l'évaluation de tests en psychologie. Nous pourrions appeler cette validation, la validation « par respect du cahier des charges ». En effet, il s'agit de montrer que l'outil ou la méthode proposée répond, par construction, aux questions posées.

Cette méthode de validation ne garantit pas que l'outil ou la méthode proposée est effectivement utile ou efficace. Mais cela démontre qu'elle répond au problème posé.

Évaluation expérimentale :

La validation expérimentale consiste à tester l'outil ou la méthode sur le terrain, avec ses utilisateurs. Cela nécessite d'effectuer une expérimentation. La difficulté d'une telle approche, c'est qu'elle nécessite d'avoir un nombre suffisant d'utilisateurs pour réellement évaluer l'outil ou la méthode. Or, notre usage est très spécifique et nécessite l'intervention de plusieurs domaines (en particulier : informatique pour traiter les données, psychologie pour définir les concepts). Il est inenvisageable de réunir un nombre suffisant d'analystes-ergonomes spécialistes de la conduite automobile pour évaluer la méthode. D'autant plus qu'il serait nécessaire de former des équipes à l'utilisation de la méthode.

On pourrait imaginer une expérimentation simplifiée où l'on traite d'un problème simple avec un public tout venant. Pour comprendre l'intérêt de la méthode, il faut comprendre les différents concepts sous-jacents et être formé à la création d'une ontologie/taxinomie, à la lecture et compréhension d'un réseau de Petri, etc. En outre l'objectif de la méthode est de s'intéresser aux cas atypiques, difficiles à étudier par définition. Comme ces cas sont difficiles à étudier, il faudrait que l'expérimentation pose un problème difficile à étudier, mais d'une manière suffisamment compréhensible pour que le public comprenne le problème. Une telle évaluation est donc difficile à réaliser.

Évaluation ergonomique :

L'ergonomie se distingue des sciences expérimentales dans sa façon d'étudier un problème. Plutôt que chercher à tester la validité d'une hypothèse et réduire au maximum le nombre de variables et de paramètres, comme c'est le cas dans les sciences expérimentales, l'ergonomie cherche à comprendre une situation dans sa complexité naturelle. L'ergonomie, en ce sens, permet des analyses plus exploratoires. Elle est aussi plus adaptée à l'étude de situations uniques ou difficilement reproductibles – c'est d'ailleurs pour cela que nous nous appuyons sur l'ergonomie pour étudier les situations de conduite critiques.

Une évaluation faite avec les méthodes de l'ergonomie permettrait donc de mieux comprendre quels sont les avantages et inconvénients de la méthode ou des outils que nous proposons. Une telle évaluation peut se faire dans un premier temps avec un nombre réduit d'utilisateurs. Il est préférable bien sûr de disposer d'un certain nombre d'utilisateurs pour pouvoir généraliser les qualités et défauts d'une méthode ou d'un outil. Mais une première évaluation avec un nombre réduit d'utilisateurs permet de disposer de premiers éléments d'évaluation.

L'ergonomie propose un certain nombre d'outils pour étudier une activité. L'un d'entre eux est l'entretien d'autoconfrontation. Ce type d'entretien permet de faire ver-

baliser l'utilisateur d'un outil sur ses actions. Pourquoi a-t-il fait telle ou telle action ? Comment pense-t-il avoir effectivement réalisé telle action ? Parfois, l'utilisateur croit avoir effectué une série d'actions alors qu'en réalité il en a fait une autre. L'objectif est d'expliciter ce que l'utilisateur croit avoir fait et le confronter avec ce qu'il a effectivement fait. Ce type d'entretien est plus pertinent qu'un simple retour de la part des utilisateurs, car il permet de mettre en évidence des problèmes dont l'utilisateur n'a pas lui-même conscience.

Il existe aussi d'autres méthodes plus lourdes à mettre en place, mais potentiellement très enrichissantes. Par exemple l'ethnométhodologie est une science qui vise à étudier une communauté de pratique de l'intérieur. Le praticien ethnométhodologue s'introduit et s'intègre dans une communauté et s'approprie l'état d'esprit de cette communauté pour comprendre les façons de penser, les relations et les interactions qui existent dans cette communauté. [12]

6.4.3 Valider les connaissances produites

Valider les connaissances produites peut relever à la fois de l'ingénierie des connaissances et de la discipline dans laquelle on cherche à produire des connaissances. La validation relève de la discipline cible (ici l'ergonomie), dans la mesure où, si les connaissances produites ne sont pas valides selon les critères de cette discipline, elles ne seront pas reconnues comme connaissances. Cette validation est donc indispensable.

Elle relève de l'ingénierie des connaissances, car si l'outil de production de connaissances a été démontré valide, alors les connaissances produites le sont, par définition. Ce deuxième cas n'est cependant que théorique. Dans la pratique, des erreurs dans l'utilisation des outils, dans les données utilisées (que cela vienne du protocole expérimental, de la collecte ou des traitements), peuvent amener à des connaissances erronées ou fausses. Ce que l'ingénierie des connaissances peut apporter pour valider scientifiquement les connaissances produites, c'est la possibilité d'argumenter la façon dont les connaissances ont été construites et de pouvoir remettre en question chaque étape de la construction de connaissances. Ce principe de remise en question est la base même de la science. S'il est possible de vérifier toute la chaîne de création de connaissances, alors, le fait d'expliquer comment une connaissance a été construite la valide.

Validation par l'expertise

Une première forme de validation des connaissances produites peut se faire par l'expertise, en présentant les résultats obtenus à d'autres experts de l'activité de conduite. Un expert est à même de juger de la pertinence d'une connaissance. Est-ce que cette connaissance est en accord avec l'état de l'art des connaissances du domaine ? Est-ce que cette connaissance est surprenante ou confirme les attentes ?

C'est ce genre de validation qui peut être effectuée dans l'étude de cas cliniques, qui se basent sur un petit nombre de sujets observés, voire même sur des cas uniques.

Validation en ergonomie

L'ergonomie est une science qui étudie l'activité humaine, en interaction avec des outils. Cependant, les connaissances ou les modèles produits à l'aide de cette discipline

[12] Je remercie en particulier Magali Ollagnier-Beldame pour les discussions passionnantes que nous avons eues à propos de l'évaluation d'outils d'ingénierie des connaissances.

relèvent potentiellement d'autres disciplines : ergologie, psychologie cognitive, etc. Les outils de validation des connaissances produites par des méthodes d'ergonomie relèvent donc de la discipline relative à la connaissance produite. Dans notre cas, l'activité cognitive du conducteur. La validation des connaissances produites relève donc de la psychologie cognitive (ou de l'expertise).

Validation en psychologie cognitive

La psychologie s'appuie sur l'observation de l'humain pour produire des connaissances. Certaines branches de la psychologie acceptent l'introspection du chercheur comme source d'alimentation de leurs théories. D'autres s'appuient sur l'observation « objective »[13] d'un sujet.

Les méthodes de validation des connaissances obtenues en psychologie relèvent de la psychologie expérimentale. Les expérimentations effectuées permettent soit de mesurer certains paramètres liés à la psychologie humaine, soit de tester et donc valider ou invalider des hypothèses sur la psychologie humaine. Comme les paramètres testés, liés à la psychologie humaine, ne sont pas directement observables, il est essentiel pour la psychologie expérimentale en particulier, et la psychologie en général de disposer de méthodes de validation.

Il existe plusieurs types de validation : la validation de construit, la validation de contenu, la validation de critère (Anastasi, 1976). La validité de construit s'intéresse à vérifier que l'expérimentation ou le test effectué mesure le bon paramètre théorique. Par exemple, si l'on s'intéresse à savoir si un conducteur a vu ou non un piéton sur la scène routière, on peut chercher à mesurer le concept « d'être conscient ou non de la présence du piéton dans la scène routière ». La validité de construction va s'assurer que les paramètres que l'on mesure (données oculométriques par exemple), permettent d'effectivement évaluer ce concept théorique non observable.

L'évaluation par la simulation

Certains types de modèles peuvent être implémentés informatiquement en vue de reproduire le comportement du sujet modélisé. Il s'agit des modèles de simulation que nous avons présentés au chapitre 2. Si l'on dispose d'une version informatique d'un tel modèle, il est donc possible de produire un comportement, le comportement du modèle, et de le comparer au comportement réel, le comportement observé.

C'est, en fait, ce qui est fait en physique. Des modèles sont établis et des expérimentations permettent de vérifier si les prédictions du modèle s'avèrent exactes.

Dans notre cas, il s'agirait d'effectuer une simulation du comportement du conducteur et comparer cette simulation aux comportements réels.

Dans le domaine de la fouille de workflow, plusieurs mesures servent à évaluer la pertinence d'un modèle, en comparant les traces produites par le modèle aux traces observées (Rozinat et van der Aalst, 2006). Ces mesures se basent sur plusieurs paramètres : 1) la capacité du modèle à rendre compte des comportements observés, de les généraliser ou au contraire à ne rendre compte que des comportements observés (notion de « fitness » : « underfitting » et « overfitting ») 2) la qualité structurelle du modèle.

[13] Nous écrivons « objective » entre guillemets, car nous pensons qu'aucune observation n'est objective, puisque le choix même des variables à observer est guidé par un analyste et relève donc d'une certaine subjectivité.

Assurer la réfutabilité : validation par la possibilité de remettre en question

Nous allons maintenant présenter une autre forme de validation qui s'inspire de la façon dont on peut différencier une connaissance scientifique d'une croyance. La différence principale entre les deux consiste en la possibilité de remise en question. C'est le principe de réfutabilité (ou falsifiabilité) de Popper (1979). Une croyance ne peut pas être remise en question. Un savoir scientifique doit pouvoir être remis en question. S'il n'est pas possible de remettre en question un savoir scientifique, alors, cela signifie qu'il n'est pas possible de l'examiner scientifiquement, c'est donc une croyance.

L'approche que nous adoptons pour produire des connaissances est une approche de *construction* de connaissances. Cela signifie que nous déduisons logiquement les connaissances de connaissances antérieures. Il s'agit de « connaissances argumentées, plutôt que démontrées », comme en parle Falzon (1998b). Logiquement donc, si les connaissances sur lesquelles on s'appuie pour produire les données sont valides d'une part, et que les processus de calcul que nous utilisons sont formellement démontrés d'autre part, alors le résultat, la connaissance construite, est vrai par construction. Il s'agit du pendant, pour la validation des connaissances construites, de la validation épistémologique, dont nous avons parlé pour la validation de la méthode.

Si la validation formelle, mathématique, du processus de construction de connaissances a été démontrée, alors il reste à valider les connaissances « initiales » introduites dans le processus de construction de connaissances.

Dans la mesure où ces connaissances « initiales » sont décrites et que le processus de construction de connaissances est démontré, alors, il est possible de remettre en question chaque étape de la construction des connaissances. La connaissance produite peut donc être contestée en remettant en question tout ou partie des connaissances « initiales ».

Pour permettre ce genre de validation, qui nous semble tout à fait pertinente pour l'approche que nous développons, nous suggérons de documenter chaque étape de la construction des connaissances produites. C'est justement possible à travers l'utilisation d'un SBT. En effet, chaque étape de la construction des connaissances est (ou peut être) tracée à partir du moment où les données sont mises dans le SBT.

Le SBT contient tous les éléments nécessaires pour remettre en question la connaissance produite : le modèle de trace, représenté sous forme d'ontologie contient les connaissances formelles et leur définition théorique ; l'arbre de transformation des traces contient les connaissances sur la façon dont une connaissance a été produite à partir d'une autre. Le tout est documenté et facilement présentable (ou exportable) à un expert qui voudrait contester la connaissance produite.[14]

[14] Notons qu'un programme informatique de traitement de données « construit » des connaissances et est potentiellement présentable à un expert. Mais il existe deux différences majeures, qui font qu'en pratique il est extrêmement difficile, voire impossible de vérifier la façon dont une connaissance a été construite.
La première différence, c'est l'absence de l'ontologie : un programme informatique ne contient pas de connaissances. Contrairement à l'ontologie qui regroupe, organise, donne une sémantique aux symboles manipulés par les programmes, un programme seul ne contient aucune sémantique et les raisons théoriques du calcul de telle ou telle variable ne sont pas accessibles. Le nom d'une variable produite par un programme informatique n'a pas de signification théorique, il s'agit juste d'un symbole utile pour le programme, mais n'est en soit pas porteur de sens.
La deuxième différence, c'est l'absence de l'arbre de transformation. Il n'est pas facile d'investiguer la façon dont une variable à été effectivement calculée à partir d'un programme informatique. Cela nécessite d'effectuer de la rétro-ingénierie, qui est un processus long et coûteux.

6.4.4 Notre contribution est-elle valide ?

Validation mathématique

L'étape AUTOMATA L'algorithme α est démontré (van der Aalst *et al.*, 2004). Nous avons démontré, au chapitre 5, que l'algorithme α_i est équivalent à l'algorithme α. L'algorithme α_i est donc démontré. Ceci nous assure que l'utilisation de cet algorithme sur les données complètes d'un modèle de type WF-net structuré et conforme, sans boucles courtes, est valide. Comme nous ne connaissons pas le modèle a priori, nous ne savons pas s'il peut s'exprimer sous forme d'un WF-net structuré et conforme sans boucles courtes. Nous faisons l'hypothèse qu'une telle modélisation est possible.

Comme nous avons de bonnes raisons de penser que les données sont incomplètes, nous proposons de rendre l'algorithme interactif pour y intégrer des connaissances de l'expert-analyste. Nous ne pouvons garantir que ces connaissances soient pertinentes. La validation de l'expertise ne peut se faire à ce niveau, mais dans d'autres boucles de validation.

Cette validation formelle n'est pas à négliger. Elle assure que notre travail est formellement correct sur une certaine plage de données. Elle nous garantit que si l'on observe un processus qui peut se modéliser d'une certaine façon (WF-net structuré et conforme sans boucles courtes), il sera formellement possible de retrouver le modèle du processus à partir de traces d'exécution du processus. Nous verrons aussi que cette validation formelle est nécessaire pour pouvoir garantir la validation épistémologique.

L'algorithme α^+ (et l'algorithme $\alpha^{+\prime}$), quant à lui, a été démontré dans un rapport de recherche (de Medeiros *et al.*, 2004a), qui n'a pas a priori été soumis à un comité de relecture scientifique.

Nous avons démontré que l'algorithme α_i^+, la version modifiée pour l'interaction de l'algorithme α^+, est équivalent à l'algorithme α^+. La démonstration formelle n'étant pas assurée, les modèles produits avec l'algorithme α_i^+ ou l'algorithme α^+ doivent nécessairement être validés avec les méthodes habituelles du domaine, dans notre cas, la psychologie cognitive.

Le cycle de découverte de connaissances dans son ensemble Toutes les étapes du cycle de découverte de connaissances, telles qu'elles sont mises en œuvre à l'heure actuelle, sont des étapes de calcul symbolique sur les données. Tous les calculs sont explicitables (il n'y a pas de boîtes noires). Il est seulement nécessaire de s'assurer que les calculs tels qu'ils sont implémentés reflètent bien les calculs que l'analyste veut effectuer.

La méta-théorie des traces, telle que proposée par Deransart (2010) s'attaque à formaliser la question de la découverte (ou redécouverte) du modèle d'un processus à partir des traces d'exécution de ce dernier. Cette méta-théorie nous donne un cadre formel pour valider les étapes de transformation de traces au sein du SBT.

Conclusion

Dans ce chapitre nous avons illustré le fonctionnement de la chaîne de découverte de connaissances dans son ensemble, d'un point de vue méthodologique. Ce sera l'objet du prochain chapitre de montrer comment cette méthodologie est concrètement exploitée, d'une part avec les outils existants ou développés dans le cadre de cette thèse et d'autre part avec la mise en pratique de la méthodologie sur un cas d'étude.

CHAPITRE 7

Mise en œuvre et évaluation sur des données de conduite réelles

L'algorithme ${\alpha_i}^{+\prime}$ de découverte interactive de connaissances sous forme de réseau de Petri a été présenté au chapitre 5. Cet algorithme s'intègre dans une démarche plus globale de découverte de connaissances présentée au chapitre 6. L'objectif de ce nouveau chapitre est de présenter une évaluation des propriétés de cette démarche de découverte interactive de connaissances, dans le cadre de l'objectif applicatif de la modélisation de l'activité de conduite automobile, présenté au chapitre 2.

Dans un premier temps, nous introduisons la démarche d'évaluation telle que nous la mettons en œuvre dans ce chapitre. Puis, nous présentons le prototype logiciel AUTOMATA, développé dans le cadre de cette thèse. AUTOMATA implémente l'algorithme ${\alpha_i}^{+\prime}$ que nous souhaitons évaluer dans un contexte réel d'utilisation. Le logiciel AUTOMATA permet de réaliser les dernières étapes du cycle de découverte de connaissances. Les étapes précédentes sont réalisées avec deux ateliers logiciels : BIND et ABSTRACT. Après avoir présenté ces différents outils, nous détaillons le travail préparatoire qu'il a été nécessaire d'effectuer pour analyser les données et exploiter AUTOMATA sur des données de conduite réelles. Puis, nous décrivons l'étude de cas exploitée dans le cadre de cette thèse.

7.1 La démarche d'évaluation adoptée

L'évaluation est une phase importante pour ce travail de thèse. Cependant, l'évaluation de méthodes et d'outils est un problème difficile, souvent oublié ou traité de manière inappropriée (Ellis et Dix, 2006). Ellis et Dix vont même jusqu'à affirmer que « la validation d'artefacts génératifs n'est pas robuste méthodologiquement. »[1] Les auteurs appellent « artefact génératif » un outil qui n'a pas de valeur intrinsèque mesurable, mais qui ne prend son sens que dans un contexte particulier : quand il est utilisé par un certain type d'utilisateurs, sur un certain type de données avec un objectif d'utilisation particulier. Notre approche s'appuie très largement sur des « artefacts génératifs ».

Nous souhaitons dans ce chapitre évaluer l'utilisation de l'algorithme ${\alpha_i}^{+\prime}$ dans le contexte applicatif de cette thèse : la construction de *schémas de conduite*, avec un analyste et à partir de données de conduite réelles sur véhicule instrumenté. Nous

[1] « Empirical evaluation of generative artefacts is methodogically unsound. » (Ellis et Dix, 2006.)

employons délibérément le terme « évaluation », et non pas « validation », car nous ne disposons pas des moyens nécessaires pour valider la démarche autrement que par une approche argumentative, ce que nous avons déjà fait. Nous avons déjà discuté des problèmes que pose la validation au chapitre 6. Nous allons maintenant identifier clairement les différents niveaux d'évaluation qui pourraient être envisagés, pour mieux définir l'objectif (réaliste) que nous nous fixons dans notre évaluation.

7.1.1 La gageure de l'évaluation de ce travail

Évaluer l'algorithme ${\alpha_i}^{+\prime}$

Pour exploiter l'algorithme ${\alpha_i}^{+\prime}$ et le principe d'interaction que nous défendons, il est nécessaire de disposer d'un logiciel : 1) implémentant l'algorithme et 2) permettant l'interaction avec un analyste, 3) et montrant que cette interaction est utile pour l'analyste. Évaluer ces propositions, pourtant triviales, a de nombreuses conséquences :

Il faut au moins un analyste.
(Pour une *validation*, il faudrait un nombre suffisant d'analystes pour pouvoir affirmer une évidence statistique.)

Il faut développer un logiciel.
C'est un travail d'ingénierie important, qui ne fait pourtant pas l'objet de cette thèse.[2]

L'évaluation de l'interaction est indissociable de l'évaluation de l'implémentation logicielle.
Ceci est extrêmement important. L'évaluation de l'approche ne peut se faire indépendamment de l'évaluation du logiciel (à moins de disposer de plusieurs logiciels implémentant les mêmes principes ?).

L'analyste, dans cette thèse, est un ergonome cogniticien spécialiste de la modélisation de l'activité de conduite automobile. Il est évident que nous ne disposons pas d'un grand panel d'analystes. Seule une évaluation est possible. Dans l'évaluation que nous faisons, nous avons travaillé avec un groupe de deux analystes.

Le logiciel développé dans cette thèse, que nous présentons à la section 7.2.3, n'est qu'un prototype. Son interface utilisateur est minimale et propose une façon de mettre en œuvre le principe d'interaction présenté au chapitre 5. L'évaluation que nous faisons porte donc sur l'évaluation de la démarche sur un prototype. Loin d'être inintéressant, cela permet de montrer la faisabilité de la démarche, et surtout de recueillir les retours précieux des utilisateurs. Ce retour d'utilisation permet, sur le plan scientifique, d'évaluer l'intérêt de la démarche, de discuter les choix de recherche, de proposer de nouvelles pistes de recherche et, sur le plan technique, de guider la conception d'un futur logiciel plus mature.

Évaluer le cycle de découverte de connaissances

Nous avons montré au chapitre 6, en replaçant l'algorithme ${\alpha_i}^{+\prime}$ dans le cycle de découverte de connaissances, que l'utilisation de cet algorithme intervient à la fin du cycle

[2] Et pour bien faire, il faudrait démontrer l'équivalence entre l'implémentation de l'algorithme et la définition formelle donnée au chapitre 5. Plus qu'anecdotique, il s'agit d'un véritable problème de rigueur scientifique.

de découverte de connaissances. Pour utiliser l'algorithme ${\alpha_i}^{+\prime}$ dans notre contexte applicatif, l'instanciation de toute la chaîne de découverte de connaissances est indispensable. Cela produit d'autres implications sur l'évaluation de notre approche. En effet, la pertinence des résultats produits par l'algorithme ${\alpha_i}^{+\prime}$ dépend directement de la pertinence du travail préparatoire dans le cycle de découverte de connaissances. Dans notre contexte applicatif, nous ne pouvons dissocier l'évaluation de l'algorithme ${\alpha_i}^{+\prime}$ de l'évaluation du cycle de découverte de connaissances. Et l'évaluation de la méthodologie de découverte de connaissances repose, à nouveau, sur des outils. L'évaluation de la méthodologie n'est donc pas indépendante de l'évaluation des outils qui permettent cette découverte de connaissances.

Bien que la chaîne d'outils que nous utilisons dans cette thèse pour alimenter la découverte de connaissances n'ait pas été validée scientifiquement, son potentiel a déjà été établi empiriquement. Le logiciel ABSTRACT, par exemple, a été utilisé avec succès (Georgeon, 2008; Henning *et al.*, 2008) dans cette démarche de découverte de connaissances.

Quel est l'objectif de l'évaluation ?

Quel est le résultat de l'évaluation ? Idéalement, c'est de montrer que l'algorithme ${\alpha_i}^{+\prime}$ permet d'atteindre son but : assister l'analyste dans son travail de modélisation des schémas de conduite. Cela revêt deux dimensions : celle de la qualité des schémas de conduite produits et celle de l'efficience de l'assistance que l'on offre à l'analyste.

La qualité des schémas de conduite produits s'évalue avec les techniques d'évaluation de l'ergonomie et des sciences cognitives. Cependant, la qualité du résultat produit n'est qu'un indicateur indirect de la qualité de la démarche. Le résultat produit est dépendant de l'algorithme, de la qualité de l'interaction avec l'analyste, mais aussi, ne l'oublions pas, de la qualité des données et des compétences et connaissances de l'analyste (ou plus exactement de l'équipe d'analyse).

Au vu des contraintes de temps et de la difficulté du problème, l'objectif de cette évaluation ne peut pas être de construire des schémas de conduite complets, exhaustifs et révélant des connaissances nouvelles dans le domaine étudié, mais, comme nous l'avons dit plus haut, de montrer que la démarche est pertinente.

7.1.2 Démarche d'évaluation

Alors, au final, que peut-on évaluer ? Comment évaluer le travail présenté dans cette thèse ?

L'évaluation que nous développons dans ce chapitre est celle préconisée par Ellis et Dix (2006) : une évaluation exploratoire et itérative. Cette évaluation, avec un petit groupe d'utilisateurs, permet de mettre le doigt sur une partie des forces et des limites de l'approche. Au-delà des retours spontanés des utilisateurs, nous souhaitons apporter des éléments de réponse aux questions suivantes :

Est-ce que l'approche permet à un analyste de construire des schémas de conduite ?

> Cette question vise à vérifier que la thèse apporte une valeur ajoutée à l'analyste dans son contexte de modélisation de l'activité de conduite automobile. L'analyste attend par exemple de pouvoir étudier les différents niveaux de l'activité de conduite. Est-ce que l'analyste aurait pu faire ces analyses sans ce que nous avons développé ?

Est-ce que les réseaux de Petri et l'algorithme $\alpha^{+\prime}$ sont des choix pertinents ?
Cette question vise à vérifier que les choix, faits pour des raisons théoriques, sont pertinents dans la pratique. Est-ce qu'un réseau de Petri permet de modéliser un schéma de conduite ? Oui, c'est pour cela que nous l'avons choisi. Est-ce que présenter un réseau de Petri à l'analyste l'aide à modéliser les schémas de conduite ? C'est ce que nous allons évaluer. D'autres questions sont relatives à l'algorithme choisi et à ses propres limitations (l'algorithme $\alpha^{+\prime}$ ne gère par les choix non libres, ni les transitions cachées ou dupliquées). Est-ce que ces limitations ont un impact sur l'analyse ?

Est-ce que l'interaction que nous proposons est pertinente pour l'analyste ?
Cette question vise à vérifier la pertinence des choix d'interactions entre l'analyste et la chaîne de cycle de découverte de connaissances. La question se décline à différents niveaux : le niveau de l'algorithme ${\alpha_i}^{+\prime}$ développé dans cette thèse, le niveau du logiciel AUTOMATA, mais aussi au niveau plus global du cycle de découverte de connaissances et des outils BIND et ABSTRACT.

À quel point les connaissances de l'analyste sont-elles nécessaires ?
Cette question vise à évaluer la pertinence de mettre l'analyste au cœur de la méthodologie. Nous avons insisté sur la nécessité pour l'analyste d'interagir à tous les niveaux du cycle de découverte de connaissances… Mais est-ce que l'analyse pourrait se faire sans interactions, sans connaissances de l'analyste, et à l'extrême limite, sans analyste ?

7.2 La plateforme logicielle de découverte de connaissances

Dans cette section, nous présentons les outils logiciels sur lesquels nous nous appuyons pour mettre en œuvre le cycle de découverte de connaissances présenté au chapitre 6 : les logiciels BIND, ABSTRACT et AUTOMATA. La figure 7.1 illustre le rôle de chaque logiciel dans le cycle.

Nous commençons par introduire les outils déjà existants, les logiciels BIND et ABSTRACT, qui permettent de préparer les données à travers les étapes du cycle de découverte de connaissances. Puis, nous présenterons le logiciel AUTOMATA, développé dans le cadre de cette thèse. Le logiciel AUTOMATA intervient à la fin du cycle de découverte de connaissances, lors de l'étape d'interprétation.

7.2.1 BIND

Description BIND est un framework logiciel permettant de développer des applications de traitement ou de visualisation de données[3]. Ce framework, développé en Matlab, est destiné à traiter les données de conduite, provenant de véhicules instrumentés ou de simulateurs de conduite. Les données manipulées par BIND sont structurées sous forme de *trips* (trajets). Un *trip* est une base de données contenant les mesures relatives à un trajet de conduite.

[3] Site web du projet : http://bind-project.inrets.fr/

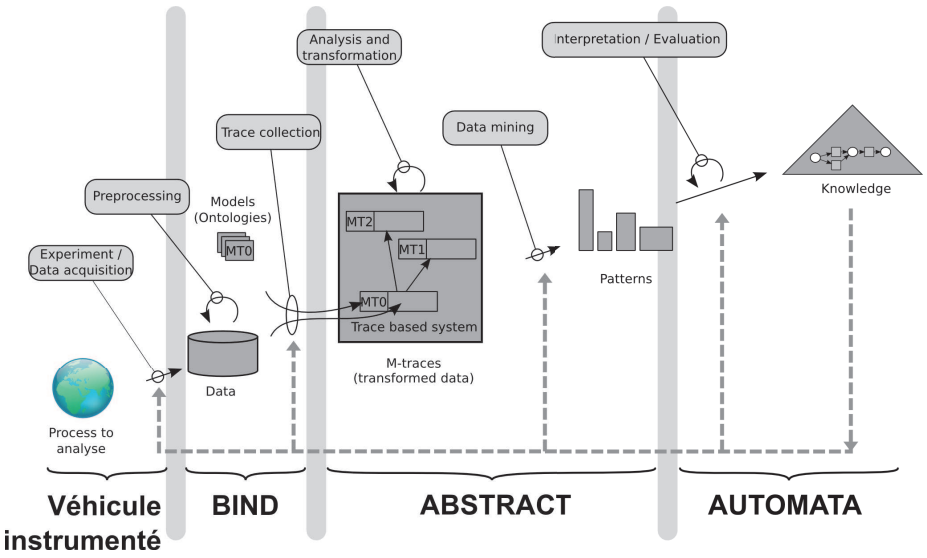


FIGURE 7.1 – Vue synthétique du cycle de découverte de connaissances révisé. Le logiciel BIND est utilisé pour les prétraitements, le logiciel ABSTRACT pour la gestion et la transformation des $\mathcal{M}$-Traces et le logiciel AUTOMATA pour l'aide à l'interprétation.

Contexte et origine de BIND Pour traiter le volume important d'informations recueillies lors des expérimentations, l'équipe technique du laboratoire LESCOT a développé un environnement logiciel préparé (*framework*) pour l'exploitation des données : BIND. Ce framework a été développé dans le but de structurer la collecte, la sauvegarde et l'archivage des données numériques et des vidéos en proposant des formats de référence. Il a d'ailleurs été conçu pour être en mesure de gérer les données collectées sur tous les moyens expérimentaux disponibles au LESCOT (véhicules instrumentés, mini-simulateur…) ainsi que des données recueillies par d'autres partenaires dans le cadre de projets de recherche.

L'usage de ce framework permet de capitaliser les différents développements spécifiques (*scripts*) aux exploitations de chaque expérimentation. Finalement, cet outil facilite l'analyse de l'activité de conduite en autorisant l'enrichissement des données par codage d'informations à partir de la vidéo puisque le framework prend également en charge la gestion d'un ensemble de fenêtres graphiques.

BIND a fait l'objet d'un dépôt à l'agence de protection des programmes en novembre 2010.

Notre usage En utilisant les fonctionnalités du framework, nos principaux objectifs seront de :

1. Importer des données brutes dans le format BIND ;
2. Créer des scripts pour traiter les données au format BIND ;
3. Créer des applications graphiques pour enrichir les données au format BIND et visualiser les vidéos et les données de manière synchrone ;

4. Créer des scripts pour discrétiser et exporter les données du format BIND vers un autre format, pouvant être lu par une autre application, par exemple le SBT ABSTRACT.

Lien avec la méthodologie Les deux premiers points permettent de réaliser l'étape de prétraitement des données de la méthodologie de découverte de connaissances que nous avons proposée au chapitre 6 (figure 7.1, p. 143).

Les applications de visualisation graphique des données (voir un exemple à la figure 7.2) sont utilisées tout au long de la chaîne de découverte de connaissances. Au

FIGURE 7.2 – Visualisation synchronisée de courbes de données, de vidéos et d'événements. Application développée avec le framework BIND.

début, elles permettent de visualiser les courbes de données brutes et vérifier la qualité de l'acquisition des données. Ensuite, elles permettent de visualiser les données traitées, et ainsi vérifier les résultats du traitement et continuer l'analyse. Enfin, elles permettent de visualiser les vidéos synchronisées avec les courbes de données, et potentiellement synchronisées avec les autres logiciels d'analyse de données, tels que le SBT ABSTRACT. La souplesse de visualisation offerte par BIND, la possibilité de synchroniser les données et la vidéo et de disposer d'accès rapides à certains points de la vidéo marqués dans les données favorise l'interaction et l'appropriation des données. Le framework logiciel BIND offre donc des solutions adaptées à la méthodologie présentée au chapitre 6. BIND permet l'itération du cycle de découverte de connaissances et l'interaction entre l'analyste et les applications de visualisation de données.

Enfin, BIND permet de construire des traces symboliques à partir des données numériques. Il permet donc d'exporter la matière brute exploitée par le logiciel ABSTRACT pour construire des $\mathcal{M}$-Traces.

Le framework BIND permet de répondre aux nécessaires interactions et itérations du cycle de découverte de connaissances (chapitre 6). L'utilisation de BIND (ou d'un

logiciel similaire) pour le prétraitement des données est indispensable pour notre objectif de modélisation.

Principes adoptés et qualité des développements. Le logiciel BIND est l'objet d'un projet, interne au laboratoire LESCOT. Son objectif est de proposer des outils d'analyse, de traitement et de visualisation qui soient d'une part robustes et fiables, et d'autre part permettent une grande souplesse d'utilisation. À ce titre, le logiciel BIND fait l'objet d'un développement de qualité « industrielle » : une documentation complète du framework est disponible et un outil de suivi de projet et de report de bogues a été mis en place[4]. Des tests unitaires permettent d'assurer la robustesse et la qualité du code au fur et à mesure des évolutions.

7.2.2 ABSTRACT

Description ABSTRACT[5] est un Système à Base de Traces (SBT). Il a été développé initialement dans l'objectif de produire des analyses exploratoires de données de conduite automobile (Georgeon, 2008; Georgeon *et al.*, 2011). Le logiciel ABSTRACT, développé principalement en PHP et Javascript, permet de gérer des $\mathcal{M}$-Traces, de les transformer pour créer des nouveaux *obsel*s correspondants à une représentation plus pertinente des données. Les $\mathcal{M}$-Traces sont stockées en RDF[6]. Le modèle de la $\mathcal{M}$-Trace est modifié avec un éditeur d'ontologie tiers, le logiciel Protégé[7]. ABSTRACT offre une grande souplesse de paramétrage de la visualisation des $\mathcal{M}$-Traces. Les transformations sont écrites sous la forme de requêtes SPARQL[8].

Notre usage

- Importer des $\mathcal{M}$-Traces ;
- Visualiser « interactivement » des $\mathcal{M}$-Traces : créer des visualisations pour faciliter la lecture et la compréhension des $\mathcal{M}$-Traces ;
- Effectuer des transformations de $\mathcal{M}$-Traces, afin de reformuler, enrichir et abstraire les $\mathcal{M}$-Traces ;
- Exporter une sélection d'*obsel*s vers le logiciel Google Earth, afin de disposer d'une visualisation spatiale des traces de conduite ;
- Exporter des séquences d'*obsel*s vers le logiciel AUTOMATA.

Lien avec la méthodologie ABSTRACT permet d'effectuer l'étape d'analyse, de transformation et de fouille des traces du cycle de découverte de connaissances (voir figure 7.1, p. 143). L'analyse des $\mathcal{M}$-Traces se fait d'abord par l'appropriation des $\mathcal{M}$-Traces par l'expert, en adaptant la visualisation. ABSTRACT permet de visualiser les $\mathcal{M}$-Traces, sous forme de bandeaux temporels, selon une échelle de temps locale (fenêtre d'une durée déterminée, 10 ou 20 secondes) ou globale (visualisation de la trace

[4] https://redmine.inrets.fr/projects/lescot-bind (il faut demander un compte utilisateur pour accéder au site)
[5] Site web du projet : http://liris.cnrs.fr/abstract/
[6] Resource Description Framework
[7] Protégé v.3.x : http://protege.stanford.edu/
[8] SPARQL Protocol and RDF Query Language

complète). L'axe horizontal désigne le temps et l'axe vertical permet de distinguer des observés de différents niveaux ou de différente nature. Chaque type d'*obsel*s possède sa propre visualisation et hérite des propriétés de ses types parents. Ainsi, il est possible de paramétrer avec une grande souplesse le type de visualisation (image ou forme), la couleur de la visualisation et la « hauteur » de visualisation (position sur l'axe vertical).

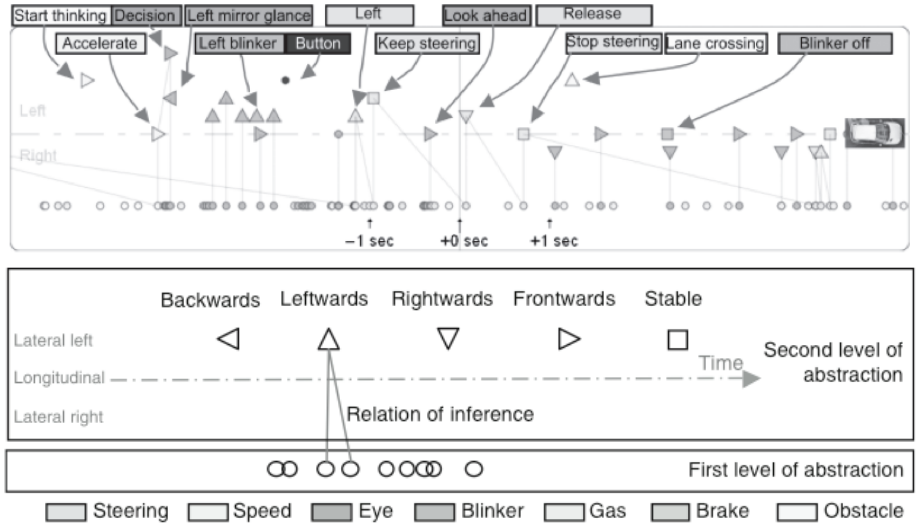


FIGURE 7.3 – La visualisation d'une $\mathcal{M}$-Trace de conduite dans ABSTRACT et une légende explicitant la logique de visualisation qui a été choisie par l'analyste (Georgeon *et al.*, 2011).

La figure 7.3 présente un exemple de choix de visualisation faits au cours d'une analyse de $\mathcal{M}$-Traces de conduite automobile. Le fait de permettre à l'analyste de modifier la visualisation qu'il a des traces lui permet de se les approprier plus facilement, en choisissant des motifs et couleurs faisant sens pour lui (orange pour le clignotant, rouge pour le frein, etc.). L'export de données vers le logiciel Google Earth offre des possibilités de visualisations supplémentaires permettant le rendu de la composante spatiale de l'activité de conduite.

La visualisation des $\mathcal{M}$-Traces permet à l'analyste de définir des transformations visant à reformuler les traces et ainsi en faciliter l'interprétation. Par exemple, il est possible de créer un nouvel *obsel* quand un certain motif est découvert dans les traces (lorsque le conducteur appuie sur le frein fortement et rapidement, on crée un nouvel *obsel* de type FreinBrusqueFort). L'ontologie doit naturellement être mise à jour pour tenir compte des nouveaux types d'*obsel*s construits au fur et à mesure des transformations.

Notons qu'en utilisant le SBT ABSTRACT, les connaissances produites sont argumentées. En effet, les sémantiques observationnelles et interprétatives présentées au chapitre 3 sont explicitées. L'ontologie contient la sémantique des traces (sémantique observationnelle). L'arbre de transformation et les transformations elles-mêmes contiennent la connaissance sur la façon dont les $\mathcal{M}$-Traces abstraites ont été construites (sémantique interprétative).

Enfin, l'export de séquences vers AUTOMATA permet de poursuivre le cycle de

découverte de connaissances avec l'interprétation des séquences étudiées.

Principes adoptés et qualité des développements Initialement développé comme un prototype de SBT permettant d'analyser des traces de conduite automobile (Mathern, 2006), le logiciel ABSTRACT, dans sa version actuelle, est le fruit d'un travail d'ingénierie important. Il a fait l'objet d'un redéveloppement progressif, visant à la fois à maintenir le logiciel et à améliorer sa maintenabilité future, par exemple en séparant plus clairement la partie gestion des données (SGBT) de la partie applicative (SBT) du logiciel. L'architecture du logiciel a fait ses preuves dans la pratique. Le développement d'ABSTRACT s'appuie sur des outils de gestion de versions[9] et de suivi de projet logiciel[10].

Les retours d'expérience (de développeur) sur le logiciel ABSTRACT nous ont encouragés à proposer des améliorations au niveau de l'architecture logicielle améliorant ainsi d'autant plus la flexibilité d'utilisation pour l'utilisateur, et de maintenance pour le développeur. C'est l'origine du projet logiciel SAMOTRACES[11]. Le projet SAMOTRACES vise à la production d'un logiciel de qualité « industrielle ». Il s'appuie sur le SGBT développé par l'équipe SILEX, le KTBS[12].

7.2.3 AUTOMATA

Nous avons développé le logiciel AUTOMATA pour servir les besoins de cette thèse. Le logiciel AUTOMATA est un *prototype* permettant de tester les principes d'interaction présentés dans le cadre de cette thèse au chapitre 5. Il ne s'agit en aucun cas d'une version finalisée destinée à être mise entre les mains de l'expert. Le prototype a pour objectif de tester le concept d'interaction. Pour réaliser un logiciel qui soit directement exploitable par l'analyste, ce qui ne fait pas l'objet de cette thèse, un important travail d'ingénierie serait nécessaire.

Notre usage

- Implémenter différentes versions de l'algorithme α ;
- Implémenter des versions interactives de ces algorithmes, en particulier l'algorithme ${\alpha_i}^{+\prime}$;
- Tester les interactions possibles avec l'analyste ;
- Faire en sorte de pouvoir exploiter les traces issues du logiciel ABSTRACT, pour pouvoir généraliser les tests à des données réelles ;
- Effectuer des études de cas sur des données réelles de conduite, avec l'aide d'un analyste, pour évaluer les principes proposés par cette thèse.

[9] ABSTRACT peut être téléchargé à l'adresse suivante : https://svn.liris.cnrs.fr/abstract/branches/v0.2/

[10] https://redmine.inrets.fr/projects/abstract

[11] Scanning, skiming, supervising, apprehending, analysing, manipulating, managing and organizing observed traces (SAMOTRACES). Site du projet SAMOTRACES : http://sourceforge.net/projects/samotraces/

[12] a kernel for Trace-Based Systems (KTBS). Documentation du KTBS : http://liris.cnrs.fr/sbt-dev/ktbs/. Site du projet : https://github.com/ktbs

Lien avec la méthodologie

L'utilisation du logiciel AUTOMATA intervient à la fin du cycle de découverte de connaissances présenté au chapitre 6 (figure 7.1, p. 143). Il permet, à partir de séquences d'*obsel*s d'une même activité observée, de reconstruire un modèle. AUTOMATA assiste l'analyste dans l'étape d'interprétation des séquences extraites des $\mathcal{M}$-Traces avec le logiciel ABSTRACT.

Le logiciel AUTOMATA met en œuvre les principes d'interaction et d'itération présentés au chapitre 6. L'analyste peut visualiser et modifier des résultats intermédiaires. Le modèle reconstruit peut facilement être visualisé.

Par construction, AUTOMATA est capable de construire des réseaux de Petri représentant l'activité de conduite. Nous avons argumenté théoriquement (au chapitre 4) le choix des réseaux de Petri pour représenter les connaissances de conduite. L'utilisation concrète d'AUTOMATA nous permettra d'évaluer la pertinence de choix dans la pratique. Nous pourrons aussi évaluer la pertinence du choix de l'algorithme de construction de réseau de Petri (algorithme $\alpha^{+\prime}$) et des choix d'interaction que nous avons proposés au chapitre 5.

Généralités techniques

Le logiciel AUTOMATA embarque une implémentation des algorithmes α, α', α^+, $\alpha^{+\prime}$ et des versions interactives α'_i, α^+_i et $\alpha^{+\prime}_i$. Le logiciel est développé en PHP[13]. L'interface utilisateur est accessible via un navigateur web[14]. Les traces sont représentées dans un format de fichier ad hoc (basé sur un format CSV[15]). Le logiciel AUTOMATA peut exporter le résultat, c'est-à-dire le réseau de Petri reconstruit, dans plusieurs formats : un format XML, utilisé par le logiciel d'édition de réseau de Petri Pipe[16], le format GRAPHML, exploité par le logiciel yEd Graph Editor[17] et enfin le format DOT, exploité par le logiciel Graphviz[18]. AUTOMATA intègre le logiciel Graphviz pour proposer une visualisation du réseau de Petri directement depuis AUTOMATA, simplifiant ainsi le flux de travail. Cette visualisation prend en charge le positionnement automatique des places, des transitions et des arcs de manière à proposer un résultat tout de suite lisible.

L'implémentation des algorithmes de la famille α et de leur variante interactive au sein d'AUTOMATA a fait l'objet d'un travail attentif d'optimisation pour offrir un temps de calcul qui soit suffisamment court pour permettre à l'utilisateur d'interagir avec le système[19]. Sur les jeux de données testés, la construction des résultats intermédiaires à partir des traces est instantanée pour l'utilisateur. Le temps nécessaire à la construction du réseau de Petri par l'algorithme et de la construction de sa visualisation reste acceptable pour l'utilisateur (de l'ordre de la seconde à la minute).

[13] Nous avons fait le choix de ce langage pour deux raisons : 1) c'est le langage avec lequel nous étions le plus familier et donc le plus opérationnel ; 2) les technologies du Web permettent un prototypage rapide et peu coûteux d'interfaces utilisateurs.

[14] C'est le navigateur Firefox qui a été utilisé. Aucun test n'a été effectué avec d'autres navigateurs.

[15] Comma-Separated Value

[16] Platform Independent Petri net Editor 2 : `http://pipe2.sourceforge.net/`.

[17] yEd Graph Editor : `http://www.yworks.com/en/products_yed_about.html`.

[18] Graphviz – Graph Visualization Software : `http://www.graphviz.org/`.

[19] Ces algorithmes s'appuient sur un calcul de cliques dont la complexité algorithmique est exponentielle (sur le nombre de transitions, c'est-à-dire sur le nombre de types d'*obsel*s présents dans les traces). C'est principalement sur ce calcul de cliques que nos efforts d'optimisation se sont portés.

Présentation de l'interface

Le logiciel AUTOMATA est accessible à partir d'un navigateur web. Le « menu » principal d'AUTOMATA permet à l'utilisateur de choisir un jeu de traces préenregistrées, de choisir l'algorithme qu'il souhaite utiliser et enfin de visualiser ou d'exporter le résultat (voir figure 7.4).

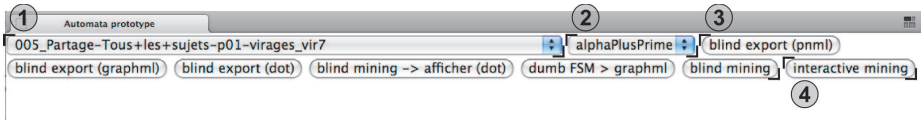


FIGURE 7.4 – Menu principal du logiciel AUTOMATA. 1) Sélectionner un jeu de traces, 2) sélectionner un algorithme, 3) appliquer l'algorithme sans interaction et exporter ou visualiser dans un format de son choix, 4) utiliser la version interactive de l'algorithme.

Si l'utilisateur souhaite utiliser la version interactive des algorithmes, alors il aura accès à une interface (figures 7.5, 7.6 et 7.7) lui permettant de visualiser les résultats intermédiaires (relations $>$ et $\triangle$) qu'il pourra éditer (figure 7.6) et filtrer (figure 7.7).

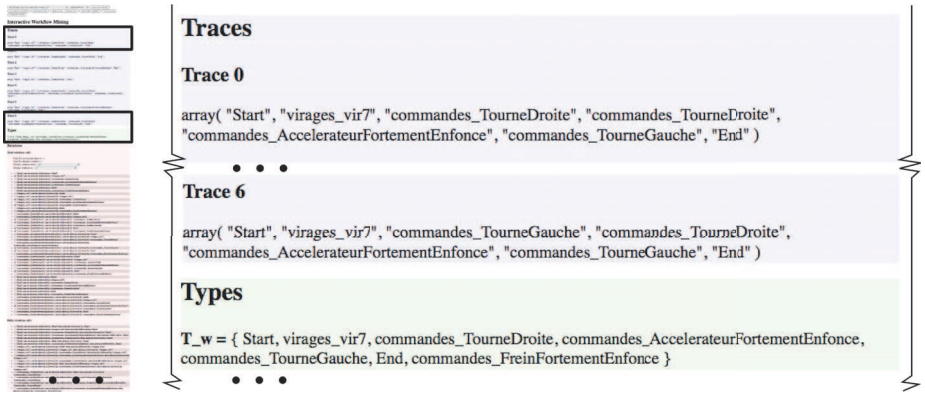


FIGURE 7.5 – Interface d'AUTOMATA : présentation de la liste des séquences (dont les doublons ont été retirés), et de la liste des types d'*obsel*s présents dans ces séquences. À gauche, la vue globale de l'interface montre où se situe cette capture d'écran.

D'autres informations sont présentées à l'analyste, lui permettant d'avoir une vue synthétique du contenu des traces. Par exemple, AUTOMATA affiche l'ensemble des séquences (c'est à dire des traces simplifiées, ne présentant que l'ordre d'apparition des différents événements) dont les doublons ont été retirés (figures 7.5).

Après édition (ou non) des résultats intermédiaires, l'analyste soumet ses modifications à l'algorithme choisi. Le réseau de Petri reconstruit peut être soit exporté pour être lu dans un logiciel externe, soit visualisé directement depuis AUTOMATA (voir par exemple la figure 7.8, p. 152). L'export vers un logiciel externe offre des possibilités supplémentaires de visualisation et d'analyse de réseau de Petri ou de graphes en général, voire de simulation. Cependant, cet import dans un logiciel tiers ajoute des manipulations supplémentaires, qui sont d'autant plus fastidieuses que le processus de découverte de connaissances est fortement itératif. La visualisation intégrée au

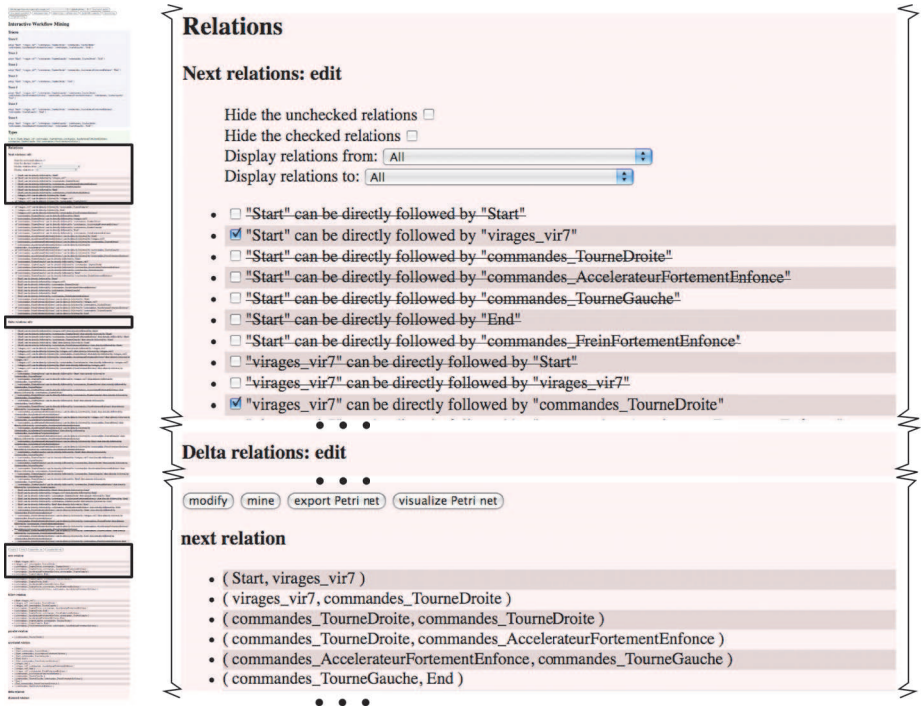


FIGURE 7.6 – Interface d'AUTOMATA : présentation de la liste des relations. Quelques filtres permettent d'accéder directement aux types d'*obsel*s pertinents. Les relations $>$ (« next ») et $\triangle$ (« delta ») peuvent être éditées par l'analyste. Puis, il peut enregistrer les modifications, construire le réseau de Petri et l'exporter ou le visualiser. Les relations calculées par l'algorithme ($>$, $\rightarrow$, $\#$, $||$, $\triangle$ et $\diamond$) sont affichées à la suite. À gauche, la vue globale de l'interface montre où se situe cette capture d'écran.

logiciel AUTOMATA résout ce problème en offrant une interaction facilitée, autorisant de nombreuses interactions et itérations entre l'utilisateur-analyste et l'algorithme de découverte de connaissances.

7.3 Travail préparatoire pour mettre en place l'analyse des données

7.3.1 Prérequis : extraire des traces depuis le logiciel ABSTRACT pour les traiter dans AUTOMATA

Un premier travail de mise en place des outils sur des données réelles a été fait avec des données de conduite automobile déjà analysées avec le logiciel ABSTRACT dans la thèse de Georgeon (2008). Ces analyses se focalisaient sur les situations de dépassement sur autoroute. Dans cette expérimentation, chacun des 22 sujets conduisait un véhicule instrumenté sur un parcours identique, sur route ouverte, pour un trajet d'une heure environ.

FIGURE 7.7 – Interface d'AUTOMATA : présentation du filtrage des relations. Les filtres permettent à l'analyste de rapidement accéder aux relations qu'il souhaite éditer. Dans le cas de cette figure, l'analyste édite les relations $>$ et $\triangle$ dont le premier argument est le type d'*obsel*s « Start ». Il est aussi possible d'afficher seulement les relations détectées dans les traces, ou seulement celles qui n'ont pas été détectées, etc. À gauche, la vue globale de l'interface montre où se situe cette capture d'écran (les zones grisées représentent la partie cachée par le filtre).

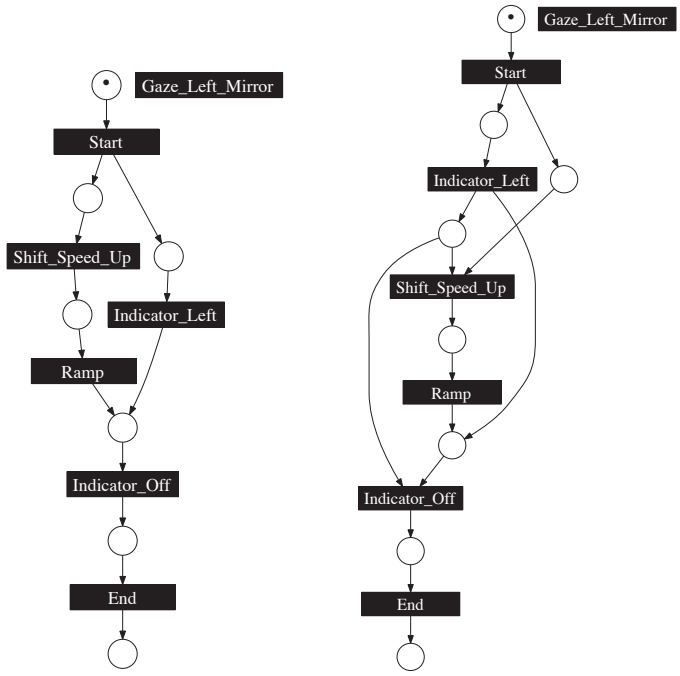
Nous avons implémenté une fonction d'export de données afin de rendre possible l'export de traces depuis le logiciel ABSTRACT, en vue de leur import de ces mêmes traces dans le logiciel AUTOMATA. Cette première fonctionnalité d'export permettait de sélectionner un temps de début et un temps de fin de la séquence à exporter. Tous les *obsel*s de la $\mathcal{M}$-Trace compris entre ces deux temps et répondants à un certain niveau d'abstraction (déterminé par le modèle de la trace) étaient alors exportés sous forme de séquence ordonnée d'*obsel*s (dans un format CSV où l'on ne retient que le type de l'*obsel*) à destination d'AUTOMATA, produisant ainsi une séquence d'événements.

À titre d'exemple, les réseaux de Petri de la figure 7.8 ont été construits de cette façon à partir de quatre situations de dépassement extraites de la même trace et recombinées en un seul fichier pour AUTOMATA. Nous reviendrons en discussion sur le fait que, même sans interaction de la part de l'analyste, l'algorithme $\alpha^{+\prime}$ et l'algorithme ${\alpha_i}^{+\prime}$ ne produisent pas les mêmes résultats[20].

Cette première implémentation a permis de mettre en connexion les deux outils que sont ABSTRACT et AUTOMATA. Cependant, il fallait choisir manuellement les temps de début et de fin de chaque séquence à exporter puis les combiner au sein d'un même fichier pour permettre à AUTOMATA de travailler sur cet ensemble de séquences.

Cette première implémentation de l'export depuis ABSTRACT vers AUTOMATA a permis de poser trois questions :

[20]Ce résultat peut paraître étonnant étant donné que nous avons démontré l'équivalence entre les deux algorithmes, mais cette équivalence ne tient que pour des traces *complètes* au sens défini par de Medeiros *et al.* (2004a).



(a) Avec l'algorithme $\alpha^{+\prime}$ (b) Avec l'algorithme ${\alpha_i}^{+\prime}$

FIGURE 7.8 – Réseaux de Petri produits à partir de données de conduite extraites du logiciel ABSTRACT, avec l'algorithme $\alpha^{+\prime}$ et l'algorithme ${\alpha_i}^{+\prime}$ (sans que l'analyste n'ajoute de connaissances).

Comment exporter les séquences de manière automatique sur plusieurs traces ? Nous proposons une méthode dans la section suivante.

Comment exporter des *obsel*s non ponctuels ? Pour un *obsel* non ponctuel (possédants un temps de début et un temps de fin), faut-il l'exporter sous la forme de deux événements (l'un correspondant au début, l'autre à la fin), ou sous la forme d'un seul (mais dans ce cas comment l'ordonner par rapport aux autres événements ?). Le choix, retenu de manière arbitraire pour sa simplicité de mise en œuvre, a été d'exporter un tel *obsel* sous la forme d'un événement dont le temps est le temps de début de l'*obsel*.

Comment gérer les débuts et les fins de séquences ? Exporte-t-on les séquences d'événements telles quelles ou ajoute-t-on, sous la forme de posttraitement une transition de début et une transition de fin (arbitrairement nommées « Start » et « End ») pour chaque séquence observée ? Nous avons retenu cette dernière option, que nous discuterons dans le chapitre suivant.

Malheureusement, le travail d'analyse sur ce jeu de données a été contraint par plusieurs éléments. D'une part, les données collectées n'avaient pas été converties dans le format du logiciel BIND et l'ancien format de données n'est plus supporté par les

outils du laboratoire. Le retour en arrière pour enrichir les traces de données complémentaires, ou simplement pour visualiser les vidéos et les données numériques était donc rendu difficile. D'autre part, l'analyse des données qui avait déjà été produite avec ABSTRACT n'avait pas été faite de manière équivalente sur tous les sujets et tous les dépassements. Dans la suite, nous montrons comment nous avons mis en œuvre la connexion entre BIND et ABSTRACT sur un autre jeu de données.

Réponses aux questions d'évaluation

Est-ce que l'approche permet à un analyste de construire des schémas de conduite ?

Le premier résultat obtenu à titre de test est prometteur. À défaut d'être un schéma de conduite, le réseau de Petri produit illustre différentes séquences d'actions possibles.

Est-ce que les réseaux de Petri et l'algorithme $\alpha^{+\prime}$ sont des choix pertinents ?

On peut constater que les boucles de longueur un ne sont pas nécessairement gérées de la manière attendue. Est-ce un problème de l'algorithme ou de la préparation des données ?

Est-ce que l'interaction que nous proposons est pertinente pour l'analyste ?

Pas d'interaction dans cet exemple.

À quel point les connaissances de l'analyste sont-elles nécessaires ?

Les connaissances de l'analyste sont dans les $\mathcal{M}$-Traces exploitées. Les essais ont été faits sans analyste, pour tester les outils.

7.3.2 L'ensemble de la chaîne : des données brutes jusqu'au réseau de Petri

Il a donc été nécessaire de remettre en œuvre l'ensemble du cycle de découverte de connaissances à partir du logiciel BIND, en passant par ABSTRACT et en allant jusqu'à AUTOMATA. À partir des outils existants, BIND et ABSTRACT, nous avons construit notre propre environnement d'analyse. L'objectif était double : capitaliser le travail de développement informatique pour l'analyse (par ex., développer des scripts qui soient facilement réutilisables et modifiables) et pouvoir itérer les traitements facilement, aussi souvent que possible, conformément à la méthodologie du cycle de découverte de connaissances.

Ce travail a été fait dans le cadre d'un stage doctoral[21] à l'Université de Californie, Berkeley. L'ensemble du travail est présenté dans un rapport de recherche (Mathern *et al.*, 2011b). Nous avons travaillé sur des données de conduites collectées par le laboratoire PATH[22] dans le cadre de l'évaluation d'un nouveau système d'assistance à la conduite appelé CACC[23] (Nowakowski *et al.*, 2010; Shladover *et al.*, 2010a,b). Dans

[21] Le stage a été soutenu par la Région Rhône-Alpes par l'intermédiaire de la bourse Explo'ra Doc
[22] Partners for Advanced Transportation TecHnology
[23] Cooperative Adaptative Cruise Control

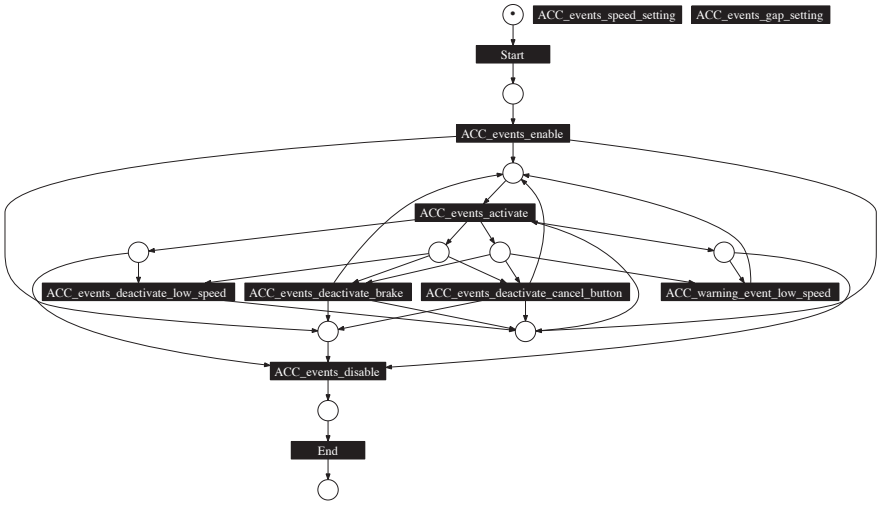
cette expérimentation, 16 conducteurs ont conduit le véhicule instrumenté. Chaque conducteur a, pendant deux semaines, effectué ses trajets quotidiens avec le véhicule fourni par PATH à la place de son véhicule personnel. Les données collectées sont donc des données sur route ouverte dont le parcours n'est pas connu à l'avance. Chaque sujet effectuait une trentaine de trajets, d'une durée de l'ordre de la minute à celle de l'heure. Pour un même sujet, les trajets empruntés pour se rendre de chez eux à leur travail et pour en revenir étaient souvent identiques, ce qui offre des possibilités de comparaison (intra-sujet pour un trajet donné).

Les données produites par le laboratoire PATH ont été importées dans le logiciel BIND. Pour des raisons de temps de traitement (liés à la grande quantité de données), seules les données d'un sujet ont été importées dans le logiciel BIND. Ainsi, la totalité des 36 trajets effectués par le sujet 3 a été convertie. Cette conversion a pris environ une semaine de temps de calcul (de l'ordre de 150 heures). Des fonctions d'export de données depuis le logiciel BIND vers le logiciel ABSTRACT ont été mises en place, permettant la mise en œuvre concrète du cycle de découverte de connaissances avec les outils logiciels BIND et ABSTRACT.

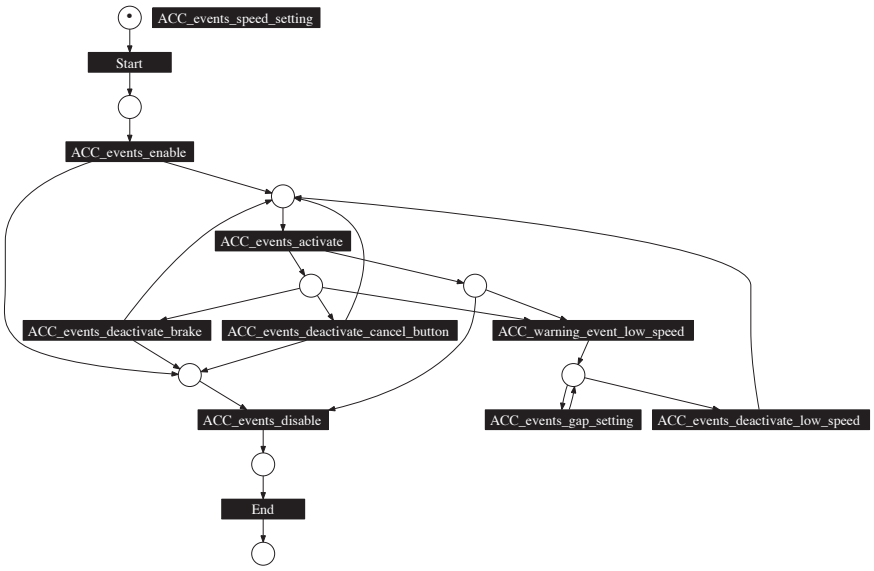
La spécificité des données (multi-trajets et multi-sujets) nécessitait la mise en place d'outils permettant de gérer des groupes de traces (partageant un même critère d'analyse). Nous avons donc du développer des solutions permettant de travailler sur plusieurs $\mathcal{M}$-Traces en parallèle (Mathern *et al.*, 2011b), ce qui n'avait pas encore été développé dans le cadre du logiciel ABSTRACT. Concrètement, nous avons développé cinq fonctions pour le travail sur plusieurs $\mathcal{M}$-Traces en parallèle :

- L'import par lot de traces liées à une expérimentation. C'est important, car quand on itère avec BIND, il faut pouvoir facilement réimporter les nouvelles $\mathcal{M}$-Traces premières.
- La visualisation d'un ensemble de $\mathcal{M}$-Traces en parallèle pour permettre leur comparaison (voir figure 7.10 p. 160).
- L'exécution d'une transformation (ou un ensemble de transformations) sur un ensemble de $\mathcal{M}$-Traces. Cela permet de garantir une homogénéité au niveau des $\mathcal{M}$-Traces de différents trajets, en leur appliquant les mêmes transformations et disposant d'un modèle de trace homogène.
- L'export de tous les *obsel*s d'un type donné (sous-types inclus) en KML pour la visualisation dans le logiciel Google Earth.
- L'export des séquences d'*obsel*s (traces) vers AUTOMATA.

Pour exporter des séquences vers AUTOMATA depuis plusieurs traces, l'idée présentée au chapitre 6 (figure 6.8, p. 131) a été mise en place. Plutôt que d'exporter des séquences en définissant manuellement les temps de début et de fin de chaque séquence, nous avons exploité des *obsel*s déjà présents dans les traces délimitant des situations de conduite. Par exemple, dans le cadre des données du laboratoire PATH, les séquences pertinentes étaient celles où le système CACC était allumé. En s'appuyant sur l'*obsel* « ACC_situation_enabled » dont le début marque l'allumage du système d'assistance et la fin marque son extinction, il est possible de détecter systématiquement toutes les séquences qui nous intéressent. Nous avons donc mis en place une fonction d'export permettant, dans ABSTRACT, de choisir un ensemble de traces et un type d'*obsel*s et d'exporter toutes les séquences correspondantes.



(a) Sans interaction avec l'analyste



(b) Avec interaction avec l'analyste

FIGURE 7.9 – Réseaux de Petri produits avec l'algorithme ${\alpha_i}^{+\prime}$ traduisant l'utilisation d'un système d'assistance à la conduite.

La figure 7.9 présente les réseaux de Petri des interactions entre le conducteur et le système d'assistance, générés par l'algorithme ${\alpha_i}^{+\prime}$, sans et avec interaction avec l'analyste. La visualisation du premier résultat a permis de mettre en évidence un problème relatif au prétraitement des données : on s'attend à ce que l'alarme avertissant de la désactivation imminente du système en cas de vitesse trop lente (ACC_warning_event_low_speed) apparaisse toujours avant la désactivation effective du système (ACC_events_deactivate_low_speed). Or, le réseau de Petri généré ne traduit pas cette assertion. Après vérification au niveau des relations de succession $>$ présentées à l'analyste et au niveau des traces, il se trouve que l'*obsel* ACC_warning_event_low_speed apparaît tantôt avant, tantôt après l'*obsel* ACC_events_deactivate_low_speed. Cela pourrait traduire un problème de conception de l'assistance, mais en l'occurrence, ce n'est pas le cas. Il s'agit d'un problème de posttraitement des données, les données collectées sur le véhicule instrumenté n'étant pas complètes, le calcul des événements relatifs à la désactivation du système et aux alarmes a fait l'objet d'une reconstruction a posteriori qui a produit quelques erreurs dans les données. Idéalement, cette erreur devrait être corrigée au niveau de l'instrumentation du véhicule, mais il est évidemment impossible de le faire sur les données existantes. Toutefois, en prenant en compte cette connaissance, l'analyste peut modifier les résultats intermédiaires au niveau des relations $>$ et $\triangle$ pour obtenir un résultat plus pertinent.[24]

Le résultat avec interaction est plus pertinent et permet déjà à l'analyste de dégager des informations intéressantes sur l'activité du conducteur (en l'occurrence son interaction avec le système d'assistance). Par exemple, on retrouve des résultats logiques : il faut d'abord allumer le système avant de l'éteindre, il faut d'abord allumer le système avant de l'activer et d'abord activer le système avant de le désactiver... On peut relever un point intéressant pour l'analyste ici. Il y a trois manières de désactiver le système dont une est automatique (lié à des conditions de vitesse du véhicule). Or, le système n'a jamais été éteint après la désactivation automatique (ACC_events_deactivate_low_speed), mais toujours après une des deux méthodes de désactivation manuelle : ACC_events_deactivate_brake et ACC_events_deactivate_cancel_button.

Ces données méritent de plus amples analyses. Par exemple, les transitions ACC_events_speed_setting et ACC_events_gap_setting ne sont pas correctement placées dans le réseau de Petri. Ces transitions constituent des boucles de longueur 1 (voir chapitre 5). Une solution consisterait à traiter ces transitions comme des transitions dupliquées. Pour compléter l'analyse, il faudrait donc différencier plusieurs types de ACC_events_speed_setting et ACC_events_gap_setting en fonction de leurs contextes d'utilisation. Ce premier exemple d'analyse avec AUTOMATA, sur les données du laboratoire PATH a été fait au retour du stage, et sans l'accès aux données sources et aux vidéos[25], il est difficile de poursuivre cette analyse plus en avant.

[24] En l'occurrence, l'analyste invalide cinq relations observées dans les traces : ACC_events_activate $>$ ACC_events_deactivate_low_speed, ACC_events_deactivate_low_speed $>$ ACC_warning_event_low_speed et ACC_events_deactivate_low_speed $\triangle$ ACC_events_activate (car l'alarme intervient entre l'activation du système et sa désactivation), ACC_warning_event_low_speed $>$ ACC_events_activate et ACC_warning_event_low_speed $\triangle$ ACC_events_activate (car la désactivation du système intervient avant sa réactivation et après l'alarme).

[25] Les données et vidéos ne sont accessibles uniquement sur place, dans les locaux de l'Université de Californie, Berkeley, pour des raisons évidentes d'éthique et de propriété intellectuelle.

Réponses aux questions d'évaluation

Est-ce que l'approche permet à un analyste de construire des schémas de conduite ?

Ces premiers résultats sont prometteurs. Plutôt que des schémas de conduite, il s'agit plutôt de schémas d'interaction avec un système d'assistance à la conduite. Une analyse plus poussée serait nécessaire, mais l'on est déjà capable avec ces quelques données de retrouver un schéma d'interaction qui traduit en partie le fonctionnement technique du système (on ne peut activer l'assistance qu'après l'avoir allumée) et les comportements particuliers de l'humain (le conducteur n'a jamais éteint le système après qu'il ait été désactivé automatiquement suite à une vitesse lente).

Est-ce que les réseaux de Petri et l'algorithme $\alpha^{+\prime}$ sont des choix pertinents ?

Le choix des réseaux de Petri est ici totalement pertinent. Par contre la limitation de l'algorithme ${\alpha_i}^{+\prime}$ vis-à-vis des boucles de longueur 1 se fait sentir à nouveau : l'algorithme n'arrive pas à placer correctement la transition « ACC_events_speed_setting » correctement dans le réseau.

Est-ce que l'interaction que nous proposons est pertinente pour l'analyste ?

Cette évaluation a été la première occasion de tester le cycle de découverte de connaissances dans son ensemble. Les outils BIND et ABSTRACT se sont révélés être des bases solides avec lesquelles l'analyste peut travailler, mais nécessitent un apprentissage. Notre connaissance du logiciel ABSTRACT (en tant que concepteur et développeur) a facilité l'ajout ou l'amélioration de fonctionnalités pour le travail en parallèle sur plusieurs traces et l'export vers AUTOMATA.

À quel point les connaissances de l'analyste sont-elles nécessaires ?

L'analyste est nécessaire. C'est avec les connaissances de l'analyste (et grâce à la forte sémantique des données) qu'il a été possible de détecter le problème où l'alarme de vitesse lente apparaissait parfois après la désactivation du système pour des raisons techniques liées à la mauvaise collecte des données.

7.4 Évaluation de la démarche sur les données de conduite du projet Partage

7.4.1 Présentation du cadre d'analyse Partage et de l'expérimentation

Les exemples présentés précédemment ont permis de mettre en œuvre techniquement l'ensemble de la chaîne de découverte des connaissances. Dans cette section, nous nous focalisons sur une étude liée au projet ANR Partage[26]. Le projet Partage se concentre sur la « fonction de prévention des sorties involontaires de voie, en virage ou en ligne droite, grâce à des observateurs de défaillance de trajectoire pour la prévention et de détection de situations à risque et de défaillances humaines ». Pour cela, une expérimentation sur route ouverte avec le véhicule instrumenté du laboratoire LESCOT a été effectuée pour observer et analyser des situations de sortie de voie dans leur contexte naturel (Bellet *et al.*, 2012). L'analyse des données collectées par cette expérimentation est en cours. Dans ce contexte d'analyse, l'ensemble du cycle de découverte de connaissances a été mis en place (BIND, ABSTRACT et AUTOMATA) ce qui nous permet d'effectuer des évaluations sur des données que nous maîtrisons totalement (nous avons accès aux données à tous les niveaux du cycle).

Dans cette expérimentation 8 sujets (en réalité 9, mais sur un des sujets, un problème d'instrumentation rend les données inexploitables) ont effectué un trajet de trois heures environ. Le trajet contenait de nombreux virages et les situations de sortie de voie ont été annotées dans les données par l'expérimentateur. Une autoconfrontation avec le sujet après l'expérimentation a permis de définir la perception du conducteur face à ces situations de sortie de voie (était-ce volontaire, conscient, critique, légitime, etc. ?). Une analyse plus particulière de 43 virages a été effectuée, pour lesquels un enrichissement des données associées a été effectué par codage vidéo (pour indiquer si le conducteur était en situation de suivi, s'il y avait du trafic dans le sens opposé, etc.).

Les études de cas que nous présentons dans la suite se sont focalisées sur ces 43 virages. En profitant de ces données riches aussi bien quantitativement (43 virages $\times$ 8 sujets) que qualitativement (enrichissement des données par le codage et l'autoconfrontation), nous proposons, avec l'aide de l'analyste, de construire des schémas de conduite de franchissement de virage.

Nous illustrons ici le travail de découverte de connaissances qui a été fait avec l'analyste. Un premier travail préparatoire a été effectué par l'analyste seul[27], puis les dernières étapes, impliquant ABSTRACT et AUTOMATA, ont été exécutées en notre collaboration :

- Faire passer les expérimentations (conduite, observations, questionnaires et entretiens) ;
- Recueillir des informations pertinentes relatives aux virages (à droite ou à gauche, léger ou fort, en montée ou en descente, etc.) et à l'activité observée (pour chaque sujet, les situations d'intérêt et les résultats d'autoconfrontation) ;
- Importer les données de conduite dans le logiciel BIND ;

[26] Site Web du projet Partage : http://www.projet-partage.fr/

[27] Après une phase de formation à l'utilisation des différents outils, l'analyste a exploité lui-même les outils dont nous parlons. D'autres personnes ont soutenu l'analyste, des ingénieurs pour des questions techniques, des étudiants pour enrichir les données à partir de la vidéo.

- Effectuer des prétraitements dans le logiciel BIND (traitement du signal, déterminer le franchissement de virages prédéterminés à partir des données GPS[28]) ;
- Visualiser les courbes de données et la vidéo (par ex. pour vérifier que les virages ont été identifiés correctement) ;
- Utiliser la vidéo pour enrichir la base de données (ajout d'informations sur le trafic, sur les caractéristiques des sorties de voie observées, etc.)
- Symboliser les données pour leur traitement ultérieur avec ABSTRACT (extraire des extremums ou des franchissements de seuils, par ex. des seuils d'enfoncement de la pédale de frein) ;
- Exporter les données symboliques avec leurs propriétés en CSV ;
- Définir une ontologie associée aux données symboliques avec le logiciel Protégé, c'est-à-dire construire le modèle de la $\mathcal{M}$-Trace première ;
- Importer les $\mathcal{M}$-Traces premières (données symboliques CSV + modèles) dans le logiciel ABSTRACT ;
- Visualiser les $\mathcal{M}$-Traces et adapter la visualisation aux besoins d'analyse.
- Transformer les $\mathcal{M}$-Traces dans ABSTRACT pour interpréter l'activité mesurée ;
- Exporter des séquences depuis ABSTRACT dans un fichier CSV ;
- Importer les séquences CSV dans AUTOMATA et visualiser les résultats intermédiaires (synthétiques) sur les données ;
- Synthétiser de manière interactive ou non (avec l'algorithme ${\alpha_i}^{+\prime}$) un réseau de Petri à partir de séquences, et visualisation du résultat ;
- Analyser le réseau de Petri produit et retour à une des phases antérieures jusqu'à la finalisation de l'analyse.

Toutes ces étapes font partie du cycle de découverte de connaissances. Elles sont itératives (l'analyste peut revenir en arrière pour modifier un prétraitement) et interactives (grâce aux interfaces de visualisation des logiciels utilisés).

Notre processus de découverte s'arrête là pour l'instant. Mais l'objectif à plus long terme sera de constituer une base de schémas de conduite et d'exploiter ces schémas dans le modèle COSMO-SiVIC pour simuler l'activité de conduite. Les performances de conduite générées en combinant les schémas utilisés par le modèle de simulation cognitive COSMODRIVE pourraient être comparées à des performances de conduite humaine. Permettant ainsi d'exploiter le modèle de simulation cognitive COSMODRIVE comme méthode d'investigation scientifique.

[28] Global Positioning System

Nature des $\mathcal{M}$-Traces avec lesquelles on travaille dans AUTOMATA

Après plusieurs cycles, d'import de $\mathcal{M}$-Traces dans ABSTRACT et de visualisation de ces $\mathcal{M}$-Trace, l'analyste a déterminé un certain nombre d'*obsel*s qu'il souhaite observer. Ainsi, les $\mathcal{M}$-Traces auxquelles nous nous intéressons contiennent des *obsel*s relatifs à l'angle du volant (supérieur à un certain seuil), aux pédales (enfoncement au-delà d'un seuil, par ex. où le freinage est ressenti physiquement comme fort), aux virages franchis (identifiés par un numéro) et à une tendance de vitesse (plutôt croissante, décroissante ou stable) et un marqueur indiquant qu'une sortie de voie a été effectuée. La figure 7.10 présente un exemple de visualisation parallèle des traces des 8 sujets synchronisées sur le même virage (en l'occurrence sur des $\mathcal{M}$-Traces transformées). La figure 7.14 présente plus en détail une de ces traces.

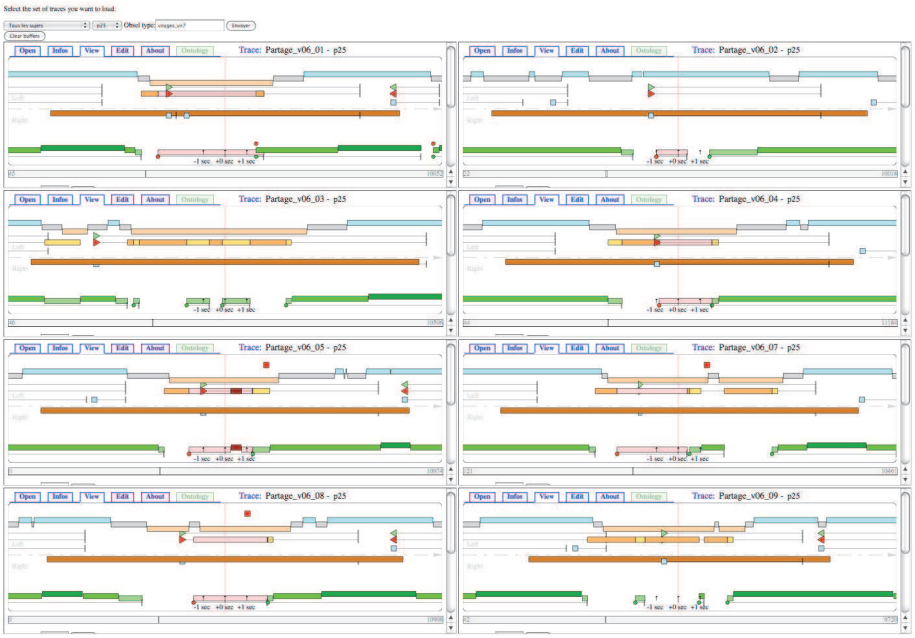


FIGURE 7.10 – Visualisation parallèle, centrée sur le virage numéro 7, des traces des 8 sujets de l'expérimentation Partage.

7.4.2 Une première analyse « naïve »

La première idée a été de sélectionner un virage déjà étudié par l'analyste (le virage 25) et d'exporter tous les types d'*obsel*s présents relatifs à ce virage. Dans cette analyse, nous aurons recours à des tableaux pour présenter les résultats de manière synthétique. La figure 7.11 explique comment lire ces tableaux. Les tableaux se succèdent montrant l'évolution progressive de l'analyse. Lorsque cela nous semble nécessaire, nous donnons des explications supplémentaires dans le texte.

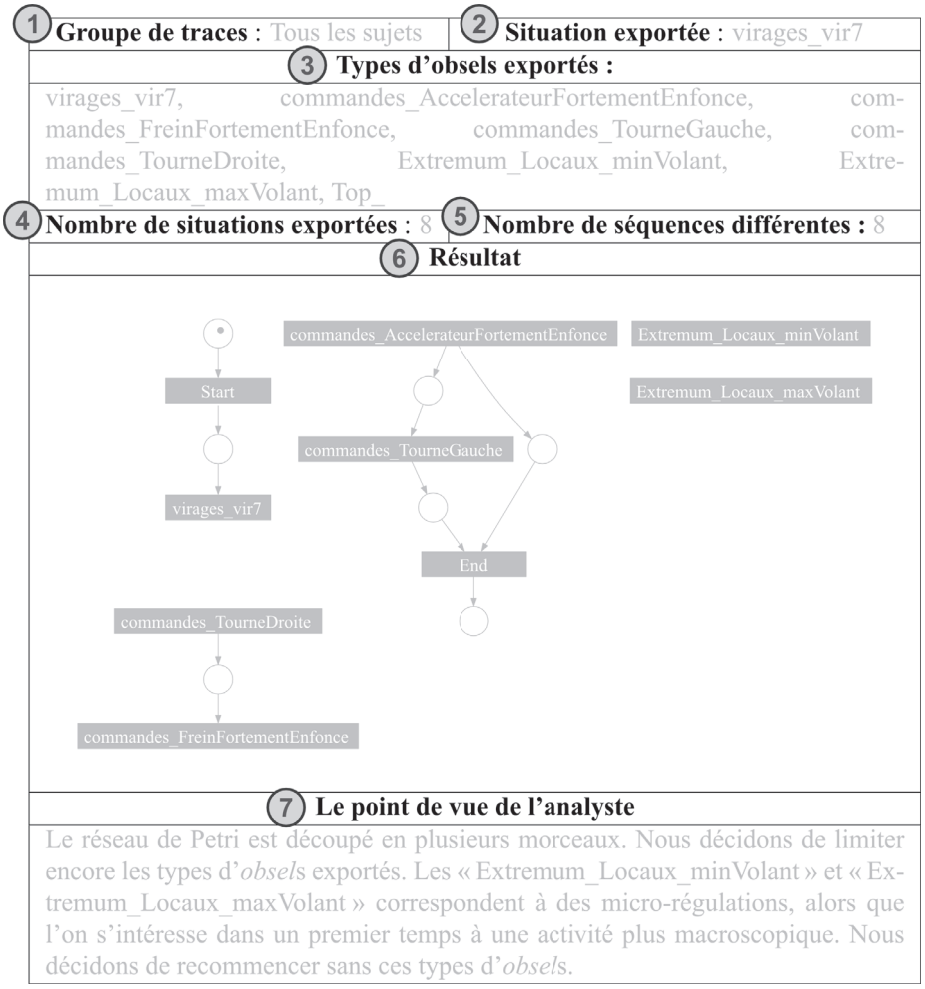
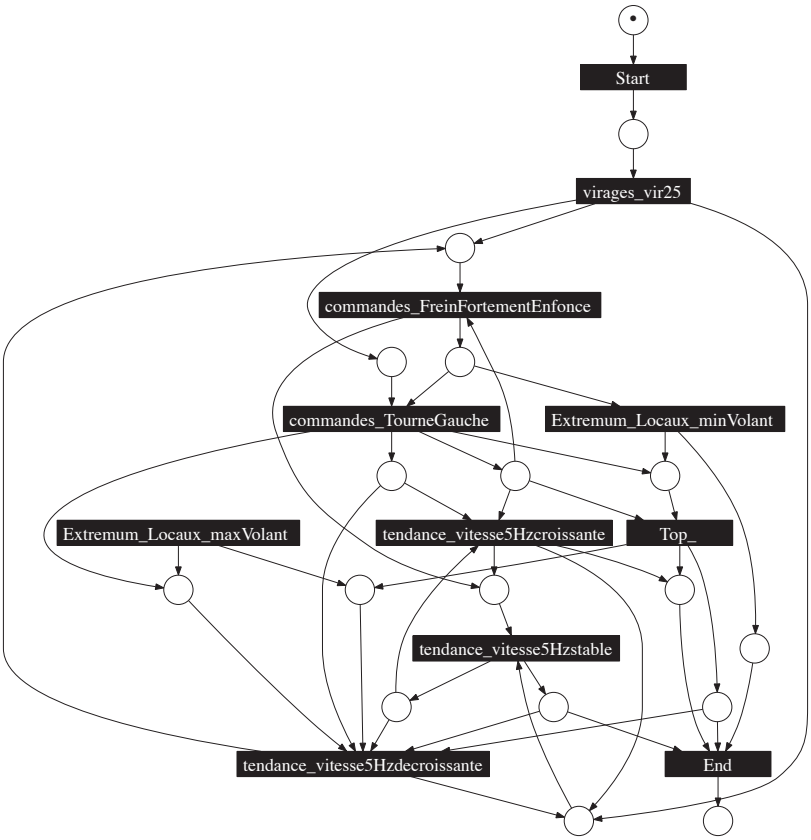


FIGURE 7.11 – Nous allons présenter les différentes analyses sous la forme de tables de résultat dont nous présentons chaque partie sur cet exemple.
1) Groupe de traces à partir duquel on exécute l'export.
2) Type d'*obsel*s définissant les situations à exporter (voir figure 6.8, p. 131).
3) Les types d'*obsel*s qui sont exportés. Pour chaque *obsel* du type désignant la situation à exporter, une séquence est créée avec tous les *obsel*s cooccurrents qui sont listés dans les « types d'*obsel*s exportés » (voir figure 7.13, p. 171).
4) Nombre de fois où un *obsel* du type de la situation à exporter (« virages_vir7 ») a été rencontré dans le groupe de traces (« Tous les sujets »).
5) Nombre de séquences différentes exportées.
6) Le réseau de Petri produit par l'algorithme ${\alpha_i}^{+\prime}$ (sauf mention contraire), avec ou sans interaction.
7) Le point de vue de l'analyste sur le réseau de Petri construit ou sur les résultats intermédiaires.

TABLE 7.1 – Table de résultat 1 : virages, 1re tentative

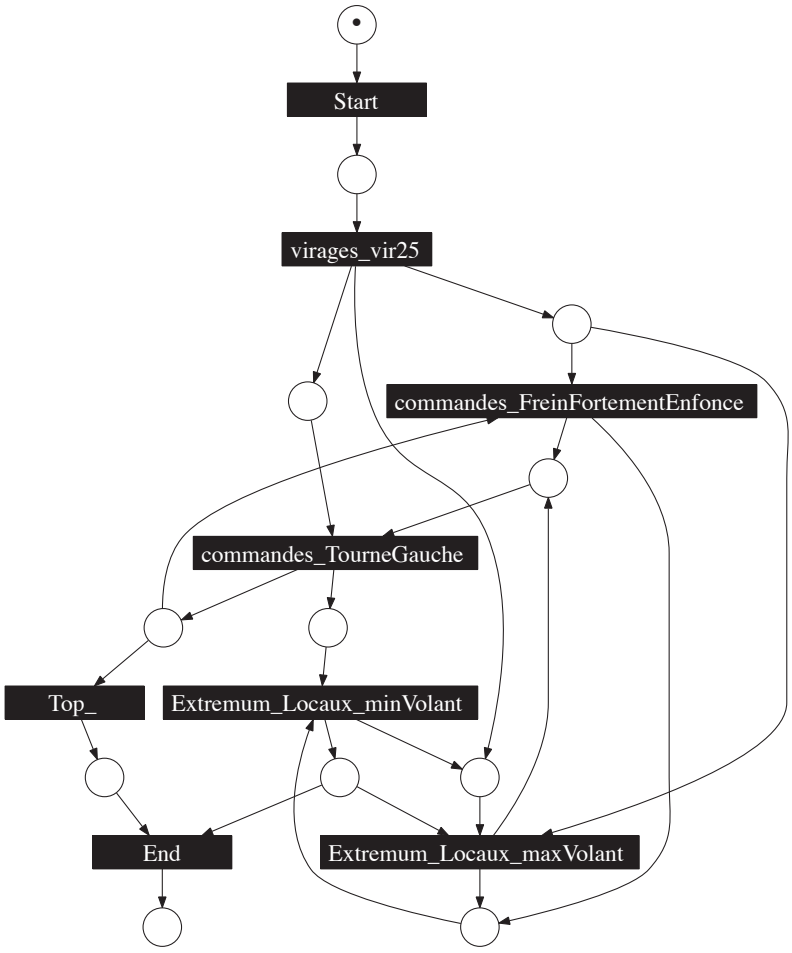
Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir25
Types d'obsels exportés :	
virages_vir25, commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Extremum_Locaux_minVolant, Extremum_Locaux_maxVolant, tendance_vitesse5Hzcroissante, tendance_vitesse5Hzstable, tendance_vitesse5Hzdecroissante, Top_	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 8
Résultat	



Le point de vue de l'analyste
Certains types d'*obsel*s ne sont pas pertinents dans un premier temps. Nous décidons de nous focaliser uniquement sur les *obsel*s relatifs au volant (supprimer ceux correspondants à la tendance de la vitesse).

TABLE 7.2 – Table de résultat 2 : virages, 2e tentative

Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir25
Types d'obsels exportés :	
virages_vir25, commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Extremum_Locaux_minVolant, Extremum_Locaux_maxVolant, Top_	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 8
Résultat	



Le point de vue de l'analyste
Le réseau de Petri est plus accessible. Mais il faut retourner au niveau des traces pour pouvoir faire le lien avec l'activité et en comprendre la signification. L'analyste est surpris de ne pas voir apparaître d'*obsel* de type « commandes_AccelerateurFortementEnfonce ». Il se rend compte alors que nous ne travaillons pas sur le virage qu'il croyait. Le virage auquel il pensait initialement était le virage numéro 7.

TABLE 7.3 – Table de résultat 3 : virages, 3e tentative

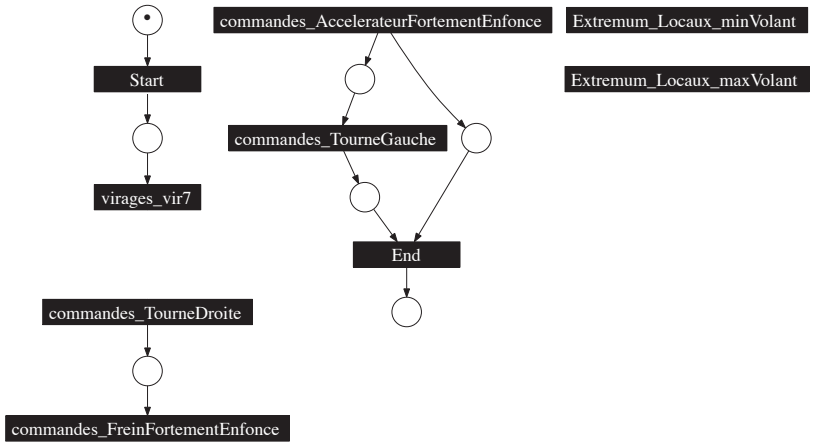
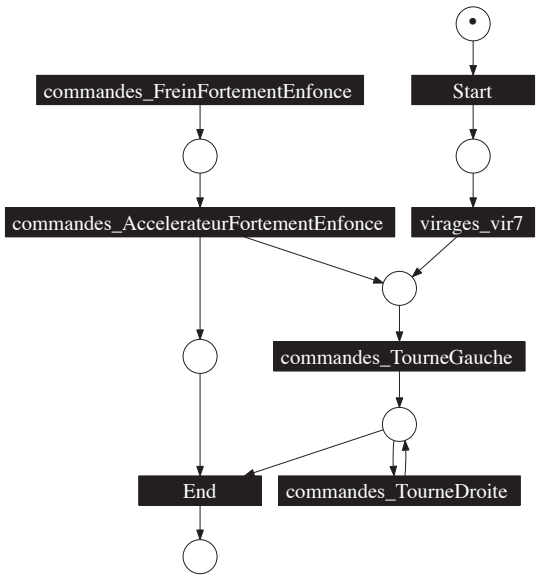
Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir7
Types d'obsels exportés :	
virages_vir7, commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Extremum_Locaux_minVolant, Extremum_Locaux_maxVolant, Top_	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 8
Résultat	
	
Le point de vue de l'analyste	
Le réseau de Petri est découpé en plusieurs morceaux. Nous décidons de limiter encore les types d'*obsel*s exportés. Les « Extremum_Locaux_minVolant » et « Extremum_Locaux_maxVolant » correspondent à des micro-régulations, alors que l'on s'intéresse dans un premier temps à une activité plus macroscopique. Nous décidons de recommencer sans ces types d'*obsel*s.	

TABLE 7.4 – Table de résultat 4 : virages, un premier réseau de Petri à analyser

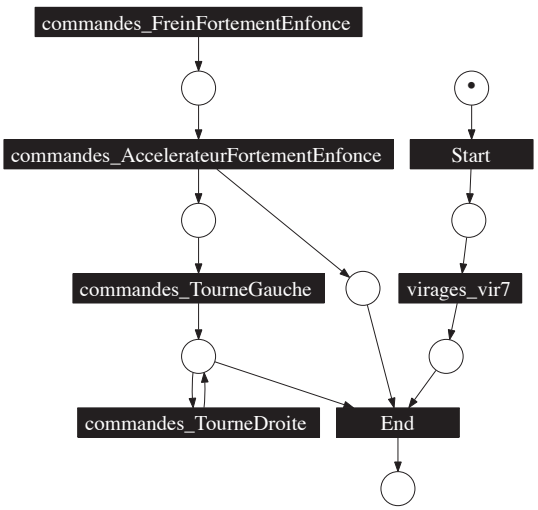
Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir7
Types d'obsels exportés :	
virages_vir7, commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Top_	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 7
Résultat 1 (sans interaction)	
	
Le point de vue de l'analyste	
Le résultat nous semble être suffisamment prometteur pour commencer à interagir au niveau d'AUTOMATA.	

À partir du réseau de Petri présenté à la table 7.4, nous commençons à utiliser l'interaction au niveau d'AUTOMATA. Tout d'abord, l'*obsel* de type virages_vir7 correspond à la situation que l'on exporte. Par définition, le virage se produit en parallèle de tous les autres *obsel*s exportés. Deux types d'*obsel*s a et b qui peuvent apparaître en parallèle, cela se traduit au niveau de la relation de succession $>$ par $a > b$ et $b > a$.

Concrètement dans AUTOMATA : pour tout type d'*obsel*s type : type $>$ virages_vir7 (sauf pour « End » et « virages_vir7 ») et virages_vir7 $>$ type (sauf pour « Start » et « virages_vir7 »). Les exceptions pour « Start » et « End » sont liées à la sémantique de ces deux types d'*obsel*s qui explicitent le début de la situation et sa fin. L'exception pour « virages_vir7 » vient du fait que le virage ne se succède pas à lui-même.

Nous nous servons des filtres intégrés à l'interface pour faciliter ce travail d'édition en n'affichant que les relations impliquant le type d'*obsel*s « virages_vir7 » (table 7.5).

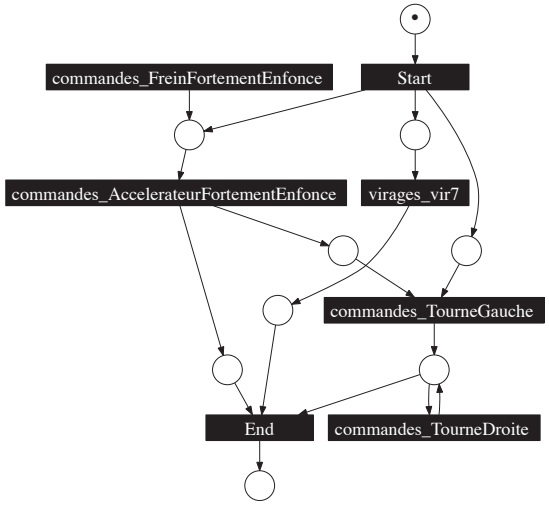
TABLE 7.5 – Table de résultat 5 : virages, première interaction avec AUTOMATA

Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir7
Types d'obsels exportés :	
virages_vir7, commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Top_	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 7
Résultat 2 (avec interaction)	
	
Le point de vue de l'analyste	
La transition « virages_vir7 » est bien parallèle aux autres dans le réseau de Petri construit. On note le fait que la transition « commandes_FreinFortementEnfonce » n'a pas de place d'entrée.	

En regardant les traces correspondantes dans ABSTRACT, nous nous rendons compte que pour une situation donnée, l'export vers AUTOMATA n'exporte que les *obsel*s qui ont débuté pendant la situation. Par exemple, si le conducteur commence à tourner le volant avant le virage et continue de tourner pendant le virage, cet *obsel* ne sera pas exporté correctement. Dans ABSTRACT, nous constatons que cela se produit pour les *obsel*s de type « commandes_AccelerateurFortementEnfonce » et « commandes_TourneGauche ».

Nous essayons donc de compenser ce problème au niveau de l'interaction avec AUTOMATA. Pour cela nous ajoutons manuellement les relations « Start > commandes_AccelerateurFortementEnfonce » et « Start > commandes_TourneGauche ». Là encore, les filtres permettent de faciliter l'interaction (voir résultat 3 table 7.6).

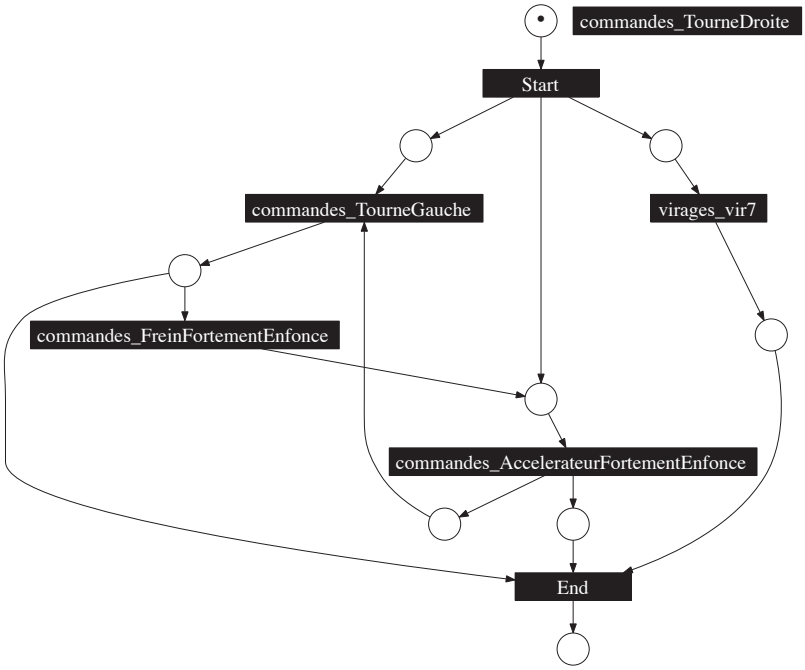
TABLE 7.6 – Table de résultat 6 : virages, deuxième interaction avec AUTOMATA

Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir7
Types d'obsels exportés :	
virages_vir7, commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Top_	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 7
Résultat 3 (avec interaction)	
	
Le point de vue de l'analyste	
La transition « commandes_FreinFortementEnfonce » n'a toujours pas de place d'entrée. Nous étudions les traces pour poursuivre l'analyse.	

Toujours à partir des traces, nous remarquons que certains *obsel*s apparaissent parfois en parallèle : par exemple, les *obsel*s de type « commandes_FreinFortementEnfonce » et « commandes_TourneDroite », mais comme l'export depuis ABSTRACT crée des séquences en ordonnant les *obsel*s par leur date de début, sans aucune indication sur la date de fin de ces *obsel*s, AUTOMATA n'a aucun moyen de détecter que ces événements sont parallèles. Nous décidons d'ajouter cette information dans AUTOMATA via l'interaction.

Nous ajoutons les relations (résultat table 7.6) :
« commandes_FreinFortementEnfonce > commandes_TourneDroite », et
« commandes_TourneGauche > commandes_FreinFortementEnfonce ».

TABLE 7.7 – Table de résultat 7 : virages, troisième interaction avec AUTOMATA

Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir7
Types d'obsels exportés :	
virages_vir7, commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Top_	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 7
Résultat 4 (avec interaction)	
	
Le point de vue de l'analyste	
Le réseau de Petri produit contient un problème : la transition « commandes_TourneDroite » est déconnectée du reste du réseau.	

La transition « commandes_TourneDroite » (table 7.7) est déconnectée du reste du réseau, parce qu'elle correspond à une boucle de longueur 1 (on peut tourner son volant à droite deux fois de suite), mais l'algorithme n'arrive pas à identifier de places auxquelles connecter cette boucle de longueur 1.

Nous décidons d'abandonner cette piste. Nous revenons en arrière au réseau de Petri obtenu ultérieurement (résultat 3 table 7.6). La transition « commandes_FreinFortementEnfonce » n'a pas de place d'entrée. Nous décidons d'essayer de comprendre l'origine de ce problème.

Nous nous rendons compte que le problème posé est en partie lié à la façon dont nous exportons les séquences – nous y reviendrons plus tard – et en partie liée au fait que les *obsel*s que l'on considère correspondent au niveau sensori-moteur de l'activité et sont parallèles et impliqués dans des boucles. Parallèles, parce que le conducteur peut

freiner ou accélérer indépendamment des actions qu'il effectue sur le volant. Bouclées, parce qu'au niveau des actions sur les commandes, l'activité mesurée est une activité de régulation sensori-motrice. Au niveau tactique de l'activité, le conducteur va tourner son volant d'un angle qui lui semble pertinent par rapport au virage dans lequel il s'engage et à la vitesse du véhicule, mais au niveau opérationnel de l'activité, cela se traduit par une régulation plus fine autour de cette valeur cible.

Ainsi, plutôt que d'observer dans les données un seul *obsel* caractérisant le fait que le conducteur tourne son volant à gauche, par exemple, deux *obsel*s de type « commandes_TourneGauche » peuvent se succéder quand le conducteur franchit le seuil d'angle volant, puis rabat légèrement le volant et le tourne à nouveau (voir figure 7.12). Notons que le même problème se pose en général avec les *obsel*s qui sont calculés à partir d'un franchissement de seuil. Le choix d'un paramètre de seuil adapté est donc critique, mais pas seulement. Cela traduit aussi un problème plus fondamental lié à l'activité elle-même. Au niveau de la mesure, un seuil d'angle volant n'aura pas la même

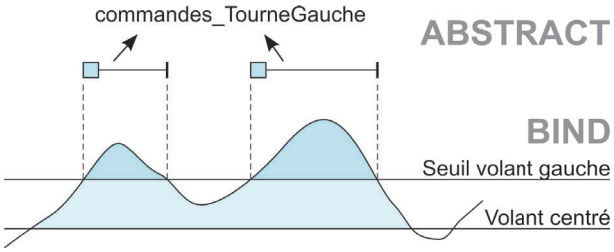
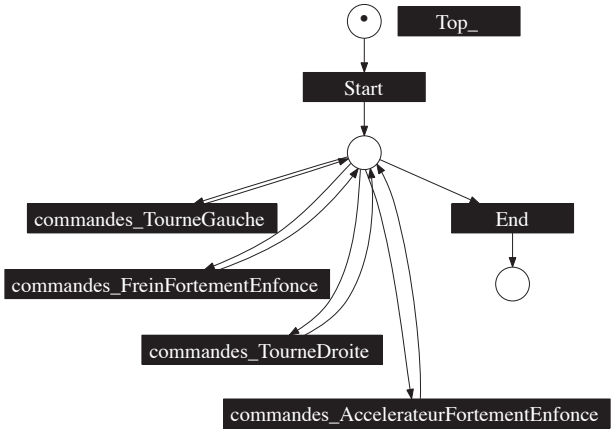


FIGURE 7.12 – Construction des *obsel*s de type « commandes_TourneGauche »

signification en fonction de la vitesse du conducteur (à une vitesse élevée, il suffit de tourner légèrement le volant pour impacter fortement la trajectoire), ni en fonction du virage considéré (en fonction de sa courbure). Au niveau de l'activité, le conducteur ne tourne pas nécessairement le volant de manière fluide et parfaite. Il va réguler son activité et fluctuer autour de l'angle volant cible. Ces fluctuations seront présentes, quelle que soit la façon dont on mesure cet angle volant, car elles sont inhérentes à l'activité de régulation de la position sur la voie.

À cause de ce problème, identifié en prenant du recul sur la signification des données, le risque est d'obtenir avec AUTOMATA un réseau de Petri où toutes les transitions peuvent s'activer en même temps, en parallèle et sous forme de boucles. Cette intuition nous a été confirmée en exécutant l'algorithme $\alpha^{+\prime}$ (algorithme non interactif), sur les séquences de tous les virages de tous les sujets (voir table 7.8). Le réseau de Petri autorise toutes les actions de se dérouler en parallèle, dans n'importe quel ordre et un nombre quelconque de fois. Autrement dit ce modèle est correct, mais trop généralisant et n'a pas d'intérêt pour notre objectif de modélisation de l'activité.

TABLE 7.8 – Table de résultat 8 : virages, le bouclage de l'activité vu avec l'algorithme $\alpha^{+\prime}$

Groupe de traces : Tous les sujets	**Situation exportée :** Virages
Types d'obsels exportés :	
commandes_AccelerateurFortementEnfonce, commandes_FreinFortementEnfonce, commandes_TourneGauche, commandes_TourneDroite, Top_	
Nombre de situations exportées : 152	**Nombre de séquences différentes :** 62
Résultat (algorithme $\alpha^{+\prime}$)	
	
Le point de vue de l'analyste	
Le réseau de Petri autorise toutes les actions à se dérouler en parallèle, dans n'importe quel ordre et un nombre quelconque de fois. Autrement dit ce modèle est juste, mais trop généralisant et n'a pas d'intérêt pour notre objectif de modélisation de l'activité.	

L'autre problème que nous avons identifié, lié à l'export des données depuis ABSTRACT vers AUTOMATA, vient du fait que l'export ne tient pas compte du moment où les *obsel*s se terminent. La figure 7.13 explicite comment les *obsel*s sont exportés. Le premier type d'export est celui utilisé dans tous les exemples précédents et ne tient pas compte de la durée. Le troisième type d'export permet de résoudre le problème.

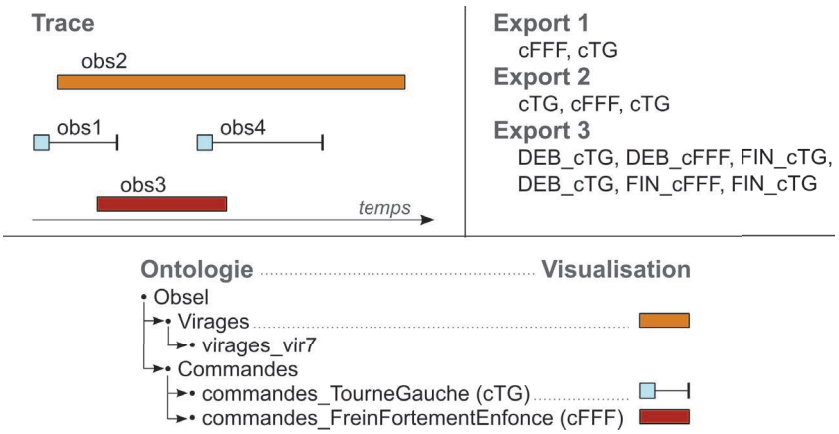


FIGURE 7.13 – Export de séquences depuis ABSTRACT vers AUTOMATA. En haut à gauche, un exemple de trace avec 4 *obsel*s. En bas, un exemple de modèle (présentant la hiérarchie des types d'*obsel*s) et la visualisation correspondante. En haut à droite, différentes façons d'exporter les séquences vers AUTOMATA. Avec le premier type d'export, seuls les *obsel*s débutants pendant la situation à exporter (ici, la situation est définie par l'*obsel* obs2 de type virages_vir7), les *obsel*s exportés sont représentés par leurs types et triés en fonction de leur date de début. Le deuxième type d'export suit le même principe, mais en plus inclus les *obsel*s qui ont débuté avant la situation à exporter et se finissent pendant. Dans le troisième type d'export, comme dans le deuxième, tous les *obsel*s cooccurrents de la situation à exporter sont exportés, cependant, le début et la fin de chaque *obsel* sont différenciés : un préfixe « DEB_ » marque l'occurrence du début d'un *obsel* et « FIN_ » en marque la fin.

Nous avons identifié deux problèmes, l'un technique, l'autre lié à l'activité étudiée. Le problème d'export est résolu dans la suite en utilisant la troisième méthode d'export présentée à la figure 7.13. Pour répondre au problème du parallélisme de l'activité, nous proposons dans un premier temps de nous focaliser sur une seule dimension sensori-motrice à la fois. La suite présente le travail d'analyse que nous avons effectué sur les actions sur le volant.

Réponses aux questions d'évaluation

Est-ce que l'approche permet à un analyste de construire des schémas de conduite ?

Cet exemple n'a pas abouti à la construction d'un schéma de conduite. Le modèle obtenu au final est trop généralisant : toutes les actions peuvent s'effectuer en parallèle (et en boucle) indépendamment les unes des autres. Cependant, cet exemple a permis à l'analyste d'identifier des pistes à explorer pour construire un schéma de conduite.

Est-ce que les réseaux de Petri et l'algorithme $\alpha^{+\prime}$ sont des choix pertinents ?

Pour que l'analyste comprenne l'origine de certains problèmes dans les réseaux de Petri générés, il est nécessaire pour lui de pouvoir faire le lien avec les résultats intermédiaires. Ce lien, nous l'avons fait intellectuellement, par notre connaissance du

fonctionnement de l'algorithme. Par exemple, il s'agit de comprendre pourquoi telle transition n'a pas de place d'entrée ? Ou bien, comment interagir au niveau des relations $>$ et $\triangle$ pour tenir compte du fait que telle transition est parallèle à telle autre ?

Est-ce que l'interaction que nous proposons est pertinente pour l'analyste ?

Pour comprendre le réseau de Petri, il est souvent nécessaire de retourner au niveau d'ABSTRACT pour visualiser les $\mathcal{M}$-Traces. La visualisation permet à l'analyste de savoir quelles sont les situations pertinentes à étudier.

Lorsque l'analyste détecte l'origine d'un problème au niveau du réseau de Petri, il peut parfois interagir dans AUTOMATA, mais il est souvent contraint de revenir plus en amont dans la chaîne de découverte de connaissances. Ce qui confirme la nécessité de disposer d'une chaîne complète où les outils sont interconnectés.

Il serait vraiment utile de pouvoir afficher directement sur le réseau de Petri l'origine (ou l'absence) des places et des arcs du réseau. Il s'agirait de faire le lien entre les données (en termes de relation $>$ et $\triangle$ par exemple) et le réseau de Petri. On aurait envie d'avoir des outils permettant de facilement naviguer du réseau de Petri aux relations intermédiaires (et aussi des relations intermédiaires aux $\mathcal{M}$-Traces).

À quel point les connaissances de l'analyste sont-elles nécessaires ?

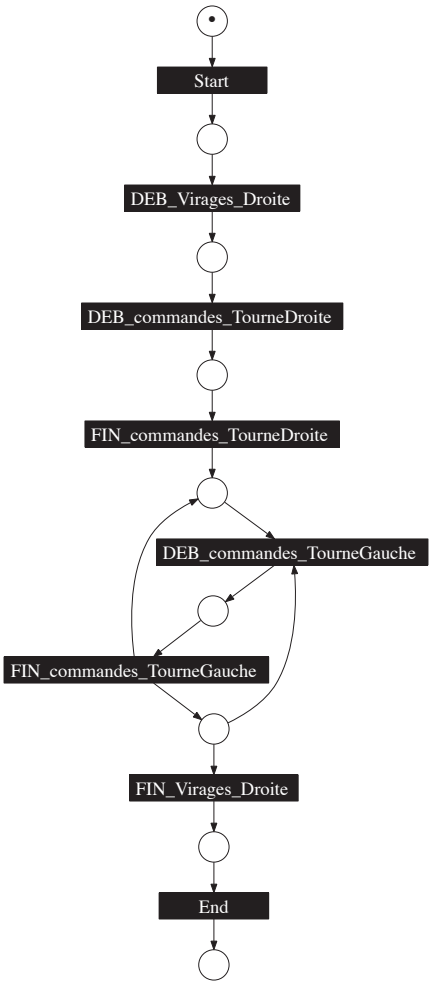
L'analyste détermine quelles sont les données pertinentes. Il détermine quels sont les situations et les types d'*obsel*s pertinents à étudier et à exporter vers AUTOMATA. C'est grâce aux connaissances de l'analyste sur l'activité qu'il a été possible d'identifier le problème de gestion des micro-régulations de l'activité.

7.4.3 Une deuxième analyse focalisée sur les actions sur le volant

À la suite des premiers essais de modélisation avec AUTOMATA, nous avons décidé de réduire le périmètre d'analyse aux actions sur le volant dans les virages. Dans un virage, les actions sur le volant vont évidemment dépendre du sens du virage. Il est donc nécessaire de distinguer les virages à droite des virages à gauche pour pouvoir prendre en compte correctement la signification des commandes sur le volant. Le premier essai est présenté à la table 7.9, il se focalise sur le virage numéro 5 (qui est un virage à droite).

TABLE 7.9 – Table de résultat 9 : actions sur le volant pour un seul virage

Groupe de traces : Tous les sujets	**Situation exportée :** virages_vir5
Types d'obsels exportés :	
Virages_Droite, commandes_TourneGauche, commandes_TourneDroite	
Nombre de situations exportées : 8	**Nombre de séquences différentes :** 2
Résultat	



Le point de vue de l'analyste

Le réseau de Petri est simple et linéaire. Il traduit le fait que le virage numéro 5 a été abordé quasiment de la même manière par tous les sujets. Notons que la structure du réseau de Petri n'est pas satisfaisante au niveau des transitions « DEB_commandes_TourneGauche » et « FIN_commandes_TourneGauche ». Ces transitions sont mortes. En regardant les traces, nous observons que tous les sujets ont d'abord tourné le volant à droite puis à gauche, sauf le sujet 5 qui a tourné le volant à gauche deux fois de suite.

Le réseau de Petri de la table 7.9 montre que sur le virage numéro 5 (un virage à droite), il n'y a pas de grande variabilité entre les sujets au niveau de ces actions sur le volant. Nous essayons de voir s'il est possible de généraliser ce réseau de Petri à tous les virages à droite (table 7.10).

Le résultat de la table 7.10 confirme une fois de plus le constat fait dans la section précédente : pour analyser l'activité au niveau tactique, il faut faire abstraction des micro-régulations opérationnelles. Il s'agit d'un travail d'abstraction des traces à effectuer dans le logiciel ABSTRACT.

Nous proposons de construire deux abstractions. La première vise à coupler l'angle volant avec le sens du virage. Lorsque le conducteur tourne à droite dans un virage à droite, nous créons un *obsel* de type « commande_Braquage_Droite ». Inversement, si le conducteur tourne à gauche dans un virage à droite, nous créons un *obsel* de type commande_ContreBraquage_Gauche. Ces abstractions nous permettent de considérer les braquages (« commande_Braquage_Droite » et « commande_Braquage_Gauche ») et les contre-braquages (« commande_ContreBraquage_Droite » et « commande_ContreBraquage_Gauche ») indépendamment du sens du virage et donc pouvoir effectuer nos traitements sur tous les virages, quel que soit leur sens.

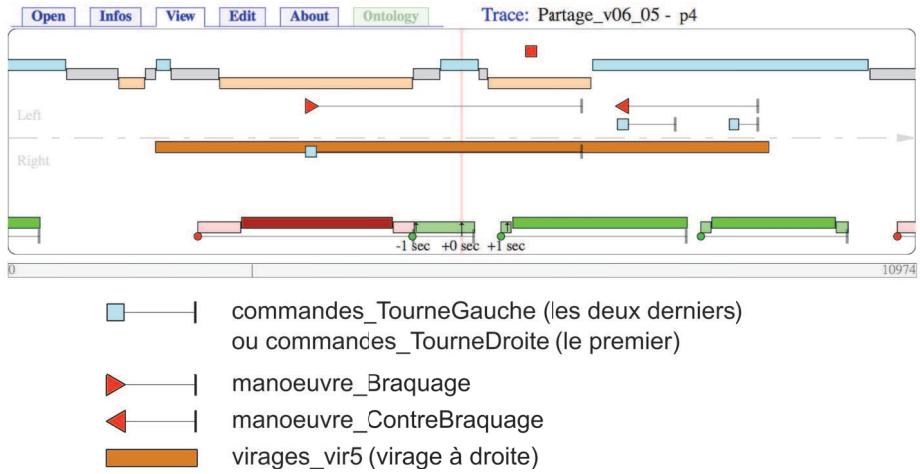
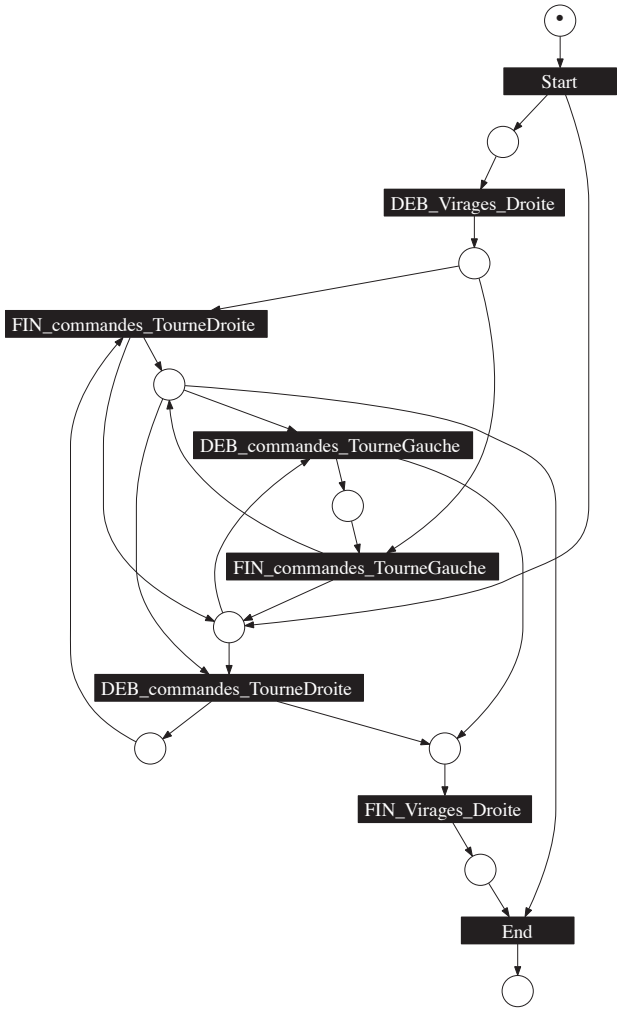


FIGURE 7.14 – Visualisation ABSTRACT du virage 5 du sujet numéro 5. Cette visualisation montre à la fois des *obsel*s provenant de la $\mathcal{M}$-Trace première et des *obsel*s construits par des transformations. Par exemple, l'*obsel* de type « manoeuvre_ContreBraquage » est construit à partir de deux *obsel*s de type « commandes_TourneGauche ».

La deuxième abstraction consiste à agréger les *obsel*s impliqués dans une micro-régulation. Nous avons regroupé les *obsel*s « commande_Braquage_Droite » successifs en un nouvel *obsel* de type « manoeuvre_Braquage » (voir figure 7.14). Le mot « manœuvre » a été choisi pour signifier que ce type d'*obsel*s n'est plus lié à la commande, mais à l'activité tactique.

TABLE 7.10 – Table de résultat 10 : actions sur le volant pour les virages à droite

Groupe de traces : Tous les sujets	**Situation exportée :** Virages_Droite
Types d'obsels exportés :	
Virages_Droite, commandes_TourneGauche, commandes_TourneDroite	
Nombre de situations exportées : 56	**Nombre de séquences différentes :** 16
Résultat	

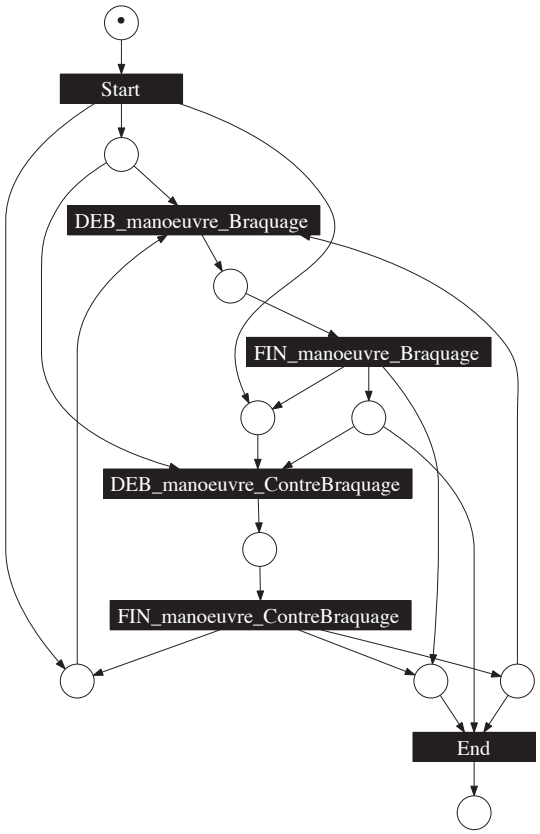


Le point de vue de l'analyste

Le réseau de Petri est confus. Ce n'est pas étonnant, car sur tous les virages à droite (56 virages), nous observons 16 séquences différentes. On observe à nouveau des comportements liés aux micro-régulations de l'angle du volant dont nous avons parlé précédemment : les traces vont parfois contenir plusieurs « commandes_TourneGauche » ou « commandes_TourneDroite » successives. En outre, différentes configurations sont possibles : le conducteur commence parfois à tourner à droite puis à gauche, ou à gauche puis à droite, etc. (bien que nous ne travaillons que sur des virages à droite).

TABLE 7.11 – Table de résultat 11 : actions sur le volant pour les virages simples

Groupe de traces : Tous les sujets	**Situation exportée :** Virages_Simple
Types d'obsels exportés :	
manoeuvre_Braquage, manoeuvre_ContreBraquage	
Nombre de situations exportées : 200	**Nombre de séquences différentes :** 6
Résultat	



Le point de vue de l'analyste
Alors que l'on travaille sur tous les virages (les « Virages_Simple » sont des virages à droite comme à gauche), à l'exception des virages faisant partie de séries, sur les 200 virages considérés, il n'y a plus que 6 séquences différentes. Plus que le réseau de Petri construit, c'est la vue synthétique de ces 6 séquences qui nous a intéressés pour poursuivre l'analyse.

En travaillant sur tous les « Virages_Simple » de tous les sujets, il ne reste que 6 séquences différentes. Le travail d'abstraction a donc porté ses fruits. Les six séquences sont les suivantes (pour simplifier la lecture, nous présentons la séquence sans les préfixes « DEB_ » et « FIN_ ») :

1. Start, manoeuvre_Braquage, manoeuvre_ContreBraquage, End
2. Start, manoeuvre_Braquage, End
3. Start, manoeuvre_ContreBraquage, manoeuvre_Braquage, manoeuvre_ContreBraquage, End
4. Start, manoeuvre_ContreBraquage, manoeuvre_Braquage, End
5. Start, manoeuvre_ContreBraquage, End
6. Start, manoeuvre_Braquage, manoeuvre_ContreBraquage, manoeuvre_Braquage, End

Par rapport aux traces observées dans ABSTRACT, les séquences 5 et 6 sont étranges. En effet, sur les virages observés, le conducteur ne contre-braque pas au milieu du virage. Après une étude plus approfondie, nous notons que cela se produit quand le virage est pris à une vitesse plus lente à cause d'un trafic important. Comme nous l'avons noté précédemment, le seuil d'angle volant qui est utilisé pour produire les *obsel*s de la $\mathcal{M}$-Trace première n'a pas la même signification en fonction de la vitesse du véhicule. Nous décidons donc de poursuivre notre étude sur l'ensemble des virages pris à vitesse normale (cette fois-ci en incluant les virages en série).

Un autre problème, responsable de ces séquences anormales, était lié à la mauvaise caractérisation du virage numéro 17, catégorisé comme virage à droite alors qu'il s'agit d'un virage à gauche. La correction a été faite au niveau d'ABSTRACT.

Le résultat à la table 7.12 confirme cette analyse.

Les cinq séquences obtenues en se focalisant sur les virages pris à vitesse normale (table 7.12) sont les suivantes :

1. Start, manoeuvre_Braquage, End
2. Start, manoeuvre_Braquage, manoeuvre_ContreBraquage, End
3. Start, manoeuvre_ContreBraquage, manoeuvre_Braquage, manoeuvre_Contre-Braquage, End
4. Start, End
5. Start, manoeuvre_ContreBraquage, manoeuvre_Braquage, End

La séquence « Start, End » correspond à des séries de virages sur lesquelles le calcul des *obsel*s de type « manoeuvre_Braquage » et « manoeuvre_ContreBraquage » n'a pas fonctionné. En dehors de ce problème particulier, un braquage apparaît toujours dans un virage. Il peut être précédé et/ou suivi d'un contre-braquage.

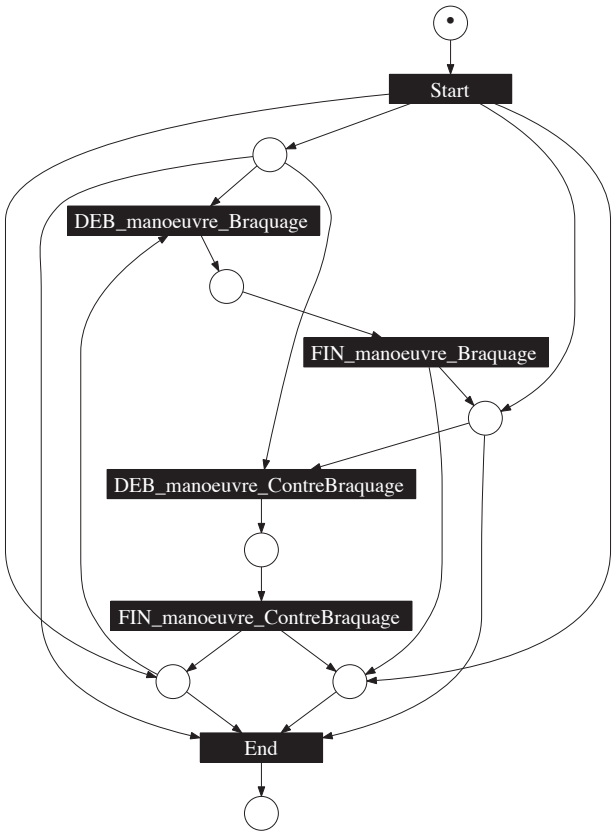
Malgré cette relative simplicité de ce niveau de description, le réseau de Petri construit est confus. En effet, l'algorithme n'est pas capable de distinguer le contre-braquage du début et le contre-braquage de la fin (il faudrait qu'il crée des transitions dupliquées pour cela). Nous proposons de résoudre ce problème en explicitant, dans les traces, le fait que le contre-braquage au début du virage n'a pas la même signification que le contre-braquage en fin de virage.

Nous créons dans ABSTRACT une nouvelle abstraction définissant les nouveaux *obsel*s de type : « schema_tactique_virage_ContreBraquage_preparation_virage », « schema_tactique_virage_Braquage_negociation_virage » et « schema_tactique_virage_ContreBraquage_sortie_virage ».

Le résultat est présenté à la table 7.13.

TABLE 7.12 – Table de résultat 12 : actions sur le volant pour les virages pris à vitesse normale

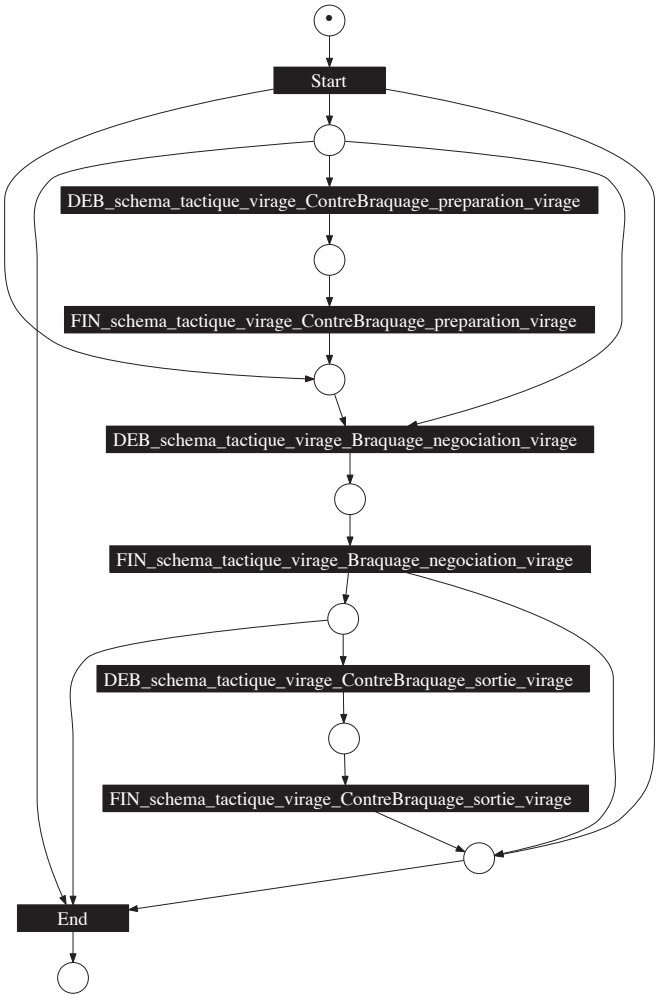
Groupe de traces : Tous les sujets	
Situation exportée : Virages_vitesse_normale	
Types d'obsels exportés :	
manoeuvre_Braquage, manoeuvre_ContreBraquage	
Nombre de situations exportées : 336	**Nombre de séquences différentes :** 5
Résultat	



Le point de vue de l'analyste
Alors que nous avons élargi l'export à un nombre plus grand de virages (336, incluant les virages en série, mais excluant les virages pris trop lentement), nous n'avons plus que 5 différentes séquences d'angle volant, qui sont plus proches de ce que l'on attend de l'activité du conducteur. Par contre, le réseau de Petri produit reste confus.

TABLE 7.13 – Table de résultat 13 : actions sur le volant pour un seul virage, construction d'un schéma tactique

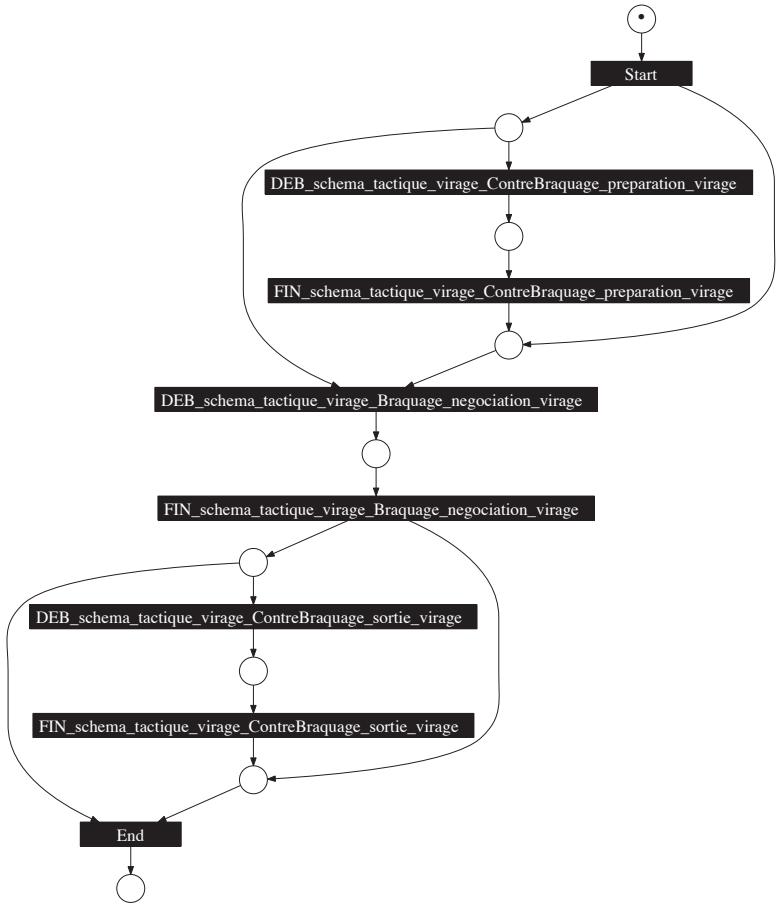
Groupe de traces : Tous les sujets	
Situation exportée : Virages_vitesse_normale	
Types d'obsels exportés :	
schema_tactique_virage_ContreBraquage_preparation_virage, schema_tactique_virage_Braquage_negociation_virage, schema_tactique_virage_ContreBraquage_sortie_virage	
Nombre de situations exportées : 336	**Nombre de séquences différentes :** 5
Résultat	



Le point de vue de l'analyste
Le réseau de Petri est proche d'un schéma tactique (focalisé sur les actions sur le volant) de l'activité. La séquence « Start, End » crée cependant un artefact que nous éliminons en supprimant la relation « Start > End » avec l'interface d'interaction d'AUTOMATA. Le résultat est présenté à la table 7.14.

TABLE 7.14 – Table de résultat 14 : actions sur le volant pour un seul virage et interaction avec AUTOMATA, construction d'un schéma tactique

Groupe de traces : Tous les sujets	
Situation exportée : Virages_vitesse_normale	
Types d'obsels exportés :	
schema_tactique_virage_ContreBraquage_preparation_virage, schema_tactique_virage_Braquage_negociation_virage, schema_tactique_virage_ContreBraquage_sortie_virage	
Nombre de situations exportées : 336	**Nombre de séquences différentes :** 5
Résultat (avec interaction)	



Le point de vue de l'analyste

Nous obtenons notre première esquisse de schéma tactique de l'activité de conduite automobile. Le schéma tactique des actions sur le volant lors d'un virage. Le réseau de Petri n'est pas tout à fait fonctionnel, car l'algorithme ${\alpha_i}^{+\prime}$ n'est pas capable de représenter l'aspect facultatif d'une transition ou d'un groupe de transitions (par ex. « schema_tactique_virage_ContreBraquage_preparation_virage »). Pour rendre compte de transitions facultatives, il faudrait que l'algorithme puisse gérer les transitions cachées. Pour cette analyse, nous nous contentons de ce résultat et nous pouvons ajouter cette transition cachée manuellement (figure 7.15).

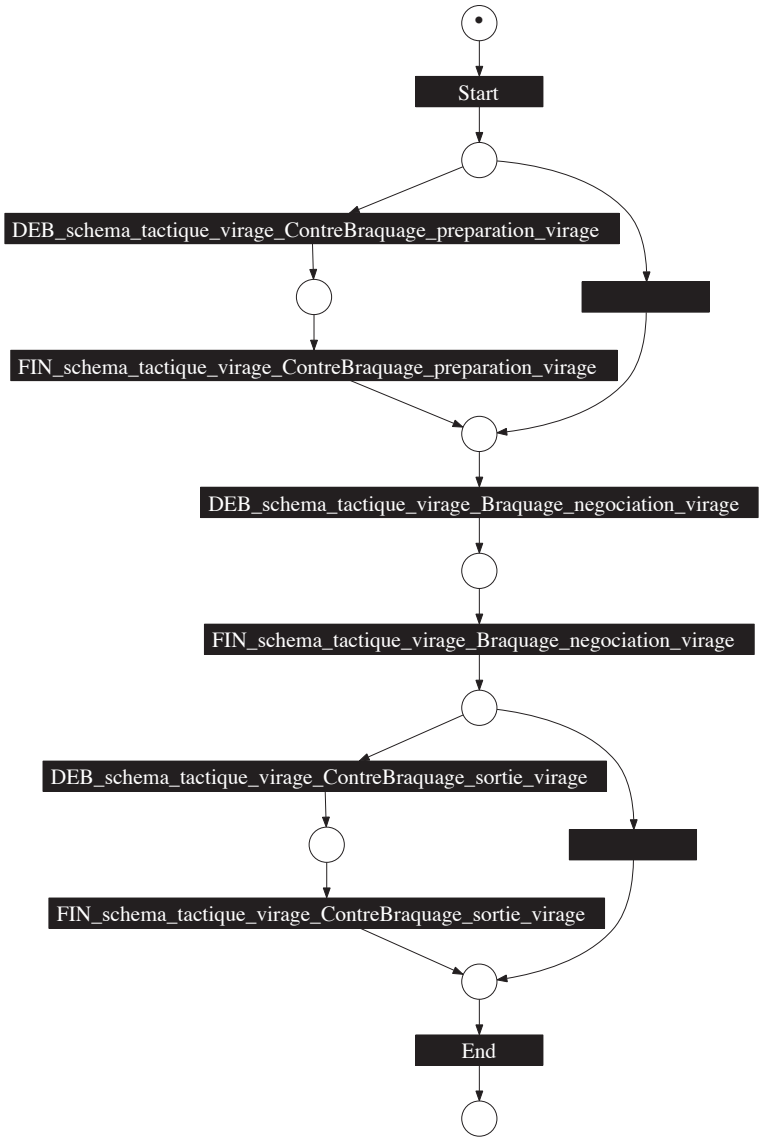


FIGURE 7.15 – Schéma de conduite tactique finalisé manuellement à partir des analyses produites avec AUTOMATA (voir table 7.14).

Réponses aux questions d'évaluation

Est-ce que l'approche permet à un analyste de construire des schémas de conduite ?

À l'issue de cette évaluation, nous avons réussi à construire un réseau de Petri caractérisant un schéma de conduite. À partir d'un réseau de Petri caractérisant l'activité de manière imparfaite, il a été possible de reconstruire (manuellement) un réseau de Petri correct (figure 7.15).

Plus que le résultat final, c'est la démarche de construction de connaissances qui est intéressante pour l'analyste. Les outils de découverte de connaissances ont permis à l'analyste de mieux comprendre la source de problèmes (par exemple certaines données sont relatives aux micro-régulations de l'activité). Ce niveau de l'activité doit être traité différemment pour pouvoir être étudié de manière pertinente.

D'une manière générale, AUTOMATA offre une vue synthétique sur les données qui est très utile à l'analyste et complémentaire des vues plus locales offertes par BIND ou ABSTRACT.

Est-ce que les réseaux de Petri et l'algorithme $\alpha^{+\prime}$ sont des choix pertinents ?

Pour comprendre un réseau de Petri, il est nécessaire d'être formé à sa lecture. Pour comprendre pourquoi certains réseaux de Petri sont mal construits, il faut à la fois accéder aux données et connaître le fonctionnement de l'algorithme.

L'algorithme utilisé n'est pas capable de construire des transitions cachées. C'est pour cette raison que le dernier réseau de Petri produit (table 7.14) n'est pas conforme. L'algorithme n'est pas capable de construire un réseau de Petri où une transition (ou un groupe de transition) est facultative.

Est-ce que l'interaction que nous proposons est pertinente pour l'analyste ?

Encore une fois, il a été nécessaire de revenir au niveau des $\mathcal{M}$-Traces. Le retour au niveau des $\mathcal{M}$-Traces était nécessaire pour deux raisons. D'abord pour visualiser des virages particuliers et comprendre l'origine de certains comportements jugés atypiques lorsqu'ils étaient visualisés hors contexte dans AUTOMATA. Ensuite, pour construire les bons niveaux d'abstractions dans les $\mathcal{M}$-Traces, pour pouvoir poursuivre l'analyse et construire des schémas de conduite pertinents.

La vue synthétique des différentes séquences est très intéressante pour l'analyste. Cependant, l'analyste aimerait que le retour vers les $\mathcal{M}$-Traces soit plus facile. Dans l'état actuel, il n'y a pas de moyens, à partir d'une séquence particulière d'AUTOMATA, de visualiser dans ABSTRACT la (ou les) $\mathcal{M}$-Trace(s) qui en est (sont) à l'origine. Il serait donc nécessaire de fluidifier le retour en arrière (garder le lien avec les $\mathcal{M}$-Traces dans AUTOMATA)

À quel point les connaissances de l'analyste sont-elles nécessaires ?

C'est encore une fois l'analyste, en visualisant les données, qui détecte des erreurs dans les données (virage 17 enregistré comme un virage à droite).

C'est l'analyste qui a proposé des abstractions à mettre en œuvre pour, à partir de traces de micro-régulation de l'activité, construire des traces relevant du niveau

tactique de l'activité. Cette connaissance de l'analyste est ensuite capturée dans les modèles de traces et dans les requêtes de transformation de $\mathcal{M}$-Traces.

7.4.4 Des perspectives d'analyses futures

Nous avons montré sur un exemple relativement simple comment, à partir de traces relatives au niveau opérationnel de l'activité, il est possible de remonter au niveau tactique et construire un premier exemple de schéma de conduite (figure 7.15).

Avec cette étude de cas, nous n'avons fait qu'effleurer le problème de modélisation de schémas de conduite à partir de traces. Cependant, cela nous a déjà permis d'obtenir des retours sur la méthodologie adoptée et plus particulièrement sur l'intérêt d'AUTOMATA.

Nous avons déjà continué l'analyse sur la dimension de la gestion de la vitesse en définissant plusieurs stratégies de décélération de la part du conducteur : relâcher (plus ou moins) l'accélérateur, appuyer (plus ou moins) sur le frein, etc. Cependant cette étude relève pour l'instant de la transformation de traces dans ABSTRACT et il faudra poursuivre encore nos efforts pour pouvoir continuer cette analyse avec AUTOMATA.

CHAPITRE 8

Discussion

Dans cette thèse, nous avons montré comment nous proposons de construire des connaissances à partir de données complexes et dynamiques. Plus qu'une construction, il s'agit d'une co-construction, guidée par un analyste, mêlant les capacités algorithmiques de la machine aux connaissances de l'analyste. Nous avons proposé de réutiliser des algorithmes de construction de modèles et de les adapter pour les rendre interactifs (chapitre 5). La co-construction des connaissances s'inscrit dans le contexte méthodologique plus large que nous avons proposé au chapitre 6.

Dans ce chapitre, nous allons discuter les choix qui ont été effectués dans cette thèse, à la lumière de la mise en œuvre et des premières évaluations qui ont été présentées au chapitre précédent. Cette discussion se découpe en trois parties. Nous commencerons par la discussion des choix algorithmiques faits dans le cadre de cette thèse, de leur pertinence et des perspectives d'évolutions. Dans une deuxième partie, nous discuterons de la méthodologie proposée et de la chaîne d'outils permettant de la mettre en œuvre. Enfin, dans la troisième partie, nous discuterons l'adéquation de notre approche avec l'objectif applicatif et le travail qu'il reste encore à faire pour atteindre cet objectif de modélisation de l'activité de conduite automobile.

8.1 Les choix algorithmiques

8.1.1 Discussion de l'algorithme ${\alpha_i}^{+\prime}$

Le problème des boucles de longueur un

Dans l'évaluation, nous avons constaté à plusieurs reprises que l'algorithme α_i^+ (et sa variante algorithme ${\alpha_i}^{+\prime}$) peine à positionner correctement les boucles de longueur un dans le réseau de Petri. La raison en est relativement simple. L'algorithme cherche à placer la même transition à différents endroits. On peut démontrer que les boucles de longueur un ont un ensemble de places d'entrée égal à leur ensemble de places de sortie. Autrement dit, les boucles de longueur un sont des transitions facultatives qui peuvent se répéter un nombre quelconque de fois. Cette propriété implique que les algorithmes doivent trouver, pour chaque transition impliquée dans une boucle de longueur un, un ensemble de places auquel connecter la transition. Lorsqu'il s'agit d'une action bouclée qui se produit à un moment bien identifié de l'activité, cela ne pose pas de problème. Lorsqu'il s'agit d'une action qui peut s'effectuer à plusieurs moments différents, c'est là que l'algorithme peine à trouver un ensemble de places cohérent.

Il serait possible de contourner cette limitation par l'usage de transitions dupliquées. Plutôt que de vouloir connecter une seule transition correspondant à la boucle de longueur un, il serait possible de considérer qu'en réalité plusieurs transitions (associées au même type d'*obsel*s sont présentes). Nous pensons que ce traitement particulier des boucles de longueur un à l'aide de transitions dupliquées devrait pouvoir s'effectuer avec un posttraitement sur le réseau de Petri résultat.

Le problème des transitions « facultatives »

Les algorithmes que nous utilisons ne sont pas capables de détecter des transitions facultatives. Par exemple, nous avons du corriger manuellement le résultat de la table 7.14 (p. 181). Pour décrire correctement ces transitions facultatives avec un réseau de Petri, il est nécessaire d'avoir recours à des transitions cachées.

Un problème analogue se pose avec les boucles de longueur un. Une transition impliquée dans une boucle de longueur un peut s'exécuter un nombre quelconque de fois, y compris ne pas s'exécuter du tout. Alors que dans certains cas, nous voudrions exprimer le fait que cette transition s'exécute au moins une fois. Là encore, l'utilisation de transitions cachées permettrait de résoudre le problème.

Ainsi, il serait intéressant de s'inspirer des travaux de Wen *et al.* (2007b) sur l'algorithme $\alpha^{\#}$, qui permet de reconstruire des transitions cachées.

Le choix de prétraiter les WF-logs pour ajouter des événements de début et de fin

Dans le chapitre précédent, nous avons explicité un traitement que nous faisons lorsque nous exportons des traces depuis ABSTRACT vers AUTOMATA : nous ajoutons systématiquement un événement de type « Start » au début de chaque trace et un événement de type « End » à la fin de chaque trace.

Ce choix pourrait paraître arbitraire, mais il a plusieurs conséquences positives. Dans le pire des cas (lorsque l'on travaille sur un WF-log complet d'un WF-net structuré et conforme), ces deux événements n'ajoutent pas d'information au réseau de Petri obtenu et ne sont donc pas utiles. Cependant, dans les autres cas, ce traitement offre trois avantages.

Premièrement, ces événements de début et de fin sont équivalents aux transitions cachées de type SIDE ajoutées par l'algorithme $\alpha^{\#}$ (Wen *et al.*, 2007b). Ce prétraitement assure que les cas nécessitant une transition cachée de type SIDE, par exemple lorsque le WF-net débute avec deux transitions parallèles, seront correctement traités.

Deuxièmement, ces transitions de début et de fin permettent de gérer correctement les cas où l'on a des séquences vides (aucun *obsel* n'a été exporté pour une ou plusieurs des séquences).

Enfin, ce prétraitement nous permet, avec l'interaction au niveau de la relation de succession $>$ d'avoir l'équivalent d'une interaction au niveau des ensembles de transitions d'entrée et de sortie (T_I et T_O). En effet, grâce à la transition « Start », au lieu de modifier l'ensemble T_I, il suffit d'agir au niveau de la relation « Start $> x$ » où x appartiendrait à T_I (et inversement, pour l'ensemble T_O, il faudrait modifier la relation « $x >$ End »).

Les choix d'interaction

Nous avons montré au chapitre 5 qu'il existe plusieurs résultats intermédiaires sur lesquels l'analyste pourrait interagir. Nous avons choisi d'interagir seulement au niveau

de la relation de succession $>$ pour l'algorithme α^+ (et de la relation $\triangle$, qui caractérise les boucles de longueur deux, pour l'algorithme α_i^+ et algorithme ${\alpha_i}^{+\prime}$). Avec le prétraitement ajoutant les transitions « Start » et « End », nous avons montré que l'interaction au niveau de la relation $>$ impacte aussi les transitions de début et de fin de manière indirecte. Le choix semble donc d'autant plus justifié.

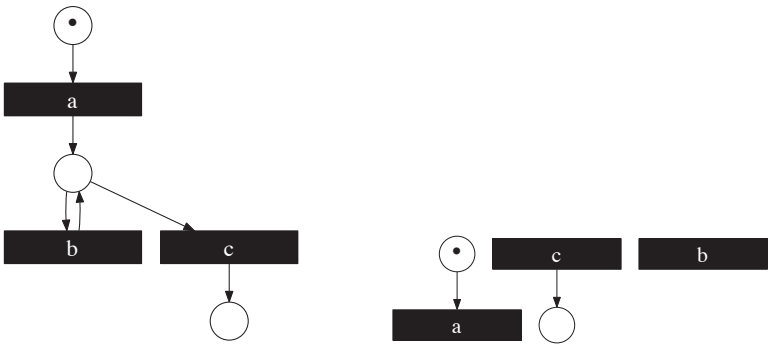
Qu'en est-il de la question de modifier les relations de causalité, d'indépendance et de parallélisme ($\rightarrow$, $\#$ et $||$) plutôt que de la relation $>$? La relation de succession $>$ semble plus facile à interpréter par l'analyste. Surtout, cette relation est directement déduite des traces : elle peut se voir dans les traces. C'est un point fort que nous avons noté lors de notre évaluation. Cependant, lors de l'analyse des données, nous avons parfois voulu matérialiser le fait que deux transitions (a et b) sont parallèles, ce qui est évident avec la relation de parallélisme ($a||b$), mais est plus difficile à exprimer avec la relation de succession ($a > b$ et $b > a$). Pour faciliter le travail de l'analyste, il pourrait être envisageable d'offrir plusieurs modes d'interaction, l'un focalisé sur la relation de succession ($>$) et l'autre sur les relations de causalité, d'indépendance et de parallélisme ($\rightarrow$, $\#$ et $||$). Comme il y a une équivalence entre la définition de la relation de succession et la définition des trois autres relations, il serait souhaitable que l'analyste puisse passer d'un mode d'interaction à l'autre et que le logiciel se charge du maintien de la cohérence entre ces deux types de relations.

La non-équivalence entre l'algorithme α^+ et l'algorithme α_i^+

Nous avons démontré que l'algorithme α_i^+ (et sa variante algorithme ${\alpha_i}^{+\prime}$), tout comme l'algorithme α^+ (et sa variante algorithme $\alpha^{+\prime}$) est capable de reconstruire un WF-net structuré conforme à partir d'un WF-log complet de ce WF-net. L'étude de cas nous montre un cas où les deux algorithmes produisent un résultat différent, ce qui pourrait paraître contradictoire. En réalité, cela souligne une subtilité : l'algorithme α_i^+ n'est équivalent à l'algorithme α^+ que pour un WF-log complet d'un WF-net structuré et conforme. Nous n'avons pas démontré l'équivalence dans les cas où le WF-log n'est pas complet ou dans le cas où le modèle n'est pas un WF-net structuré et conforme. L'exemple présenté à la figure 7.8 (p. 152) illustre un cas où les traces ne forment pas un WF-log complet d'un WF-net structuré et conforme.

Nous savons que les données sur lesquelles nous travaillons sont rarement complètes, c'est une des raisons pour lesquelles nous avons introduit l'interaction. Nous pouvons donc nous poser la question suivante : dans le cas où un WF-log n'est pas complet, quel algorithme produit le résultat le plus pertinent (sans l'ajout des connaissances de l'expert) ? Nous n'avons pas fait suffisamment d'essais pour répondre à cette question, mais nos premiers essais montrent que dans certains cas, l'algorithme original (l'algorithme α^+) produit des résultats plus pertinents sur un WF-log incomplet.

Quelle en est la raison ? La différence principale entre l'algorithme α^+ (original) et l'algorithme α_i^+ que nous proposons vient de la façon de calculer le réseau de Petri privé des boucles de longueur un. Alors que l'algorithme α^+ (et sa variante l'algorithme $\alpha^{+\prime}$) dérive un second WF-log dont les transitions impliquées dans des boucles de longueur un ont été supprimées, l'algorithme α_i^+ (et sa variante l'algorithme ${\alpha_i}^{+\prime}$) effectue tous les calculs à partir du WF-log original. Dans le cas où le WF-log est complet, cela ne change rien au fonctionnement de l'algorithme. Par contre, dans le cas où le WF-log est incomplet, le fait de créer un nouveau WF-log dont les transitions de longueur un ont été supprimées va avoir un impact sur la relation de succession $>$: des successions qui n'ont jamais été observées dans le WF-log original sont créées.



(a) Avec l'algorithme $\alpha^{+\prime}$ (b) Avec l'algorithme ${\alpha_i}^{+\prime}$

FIGURE 8.1 – Réseaux de Petri produits avec l'algorithme $\alpha^{+\prime}$ et l'algorithme ${\alpha_i}^{+\prime}$ (sans que l'analyste n'ajoute de connaissances), à partir du WF-log incomplet constitué de la trace $abbc$.

Par exemple, dans le cas d'un WF-log très simple constitué d'une seule trace « $abbc$ », où l'événement b est impliqué dans une boucle de longueur un, l'algorithme α^+ va créer un deuxième WF-log dans lequel b aura été supprimé : « ac ». L'algorithme α^+ construit donc la relation $a > c$, alors que l'algorithme α_i^+ ne l'ajoute pas, car elle n'est pas observée dans le WF-log original. Le résultat est donc différent (voir figure 8.1). Dans cet exemple, c'est le résultat donné par l'algorithme $\alpha^{+\prime}$ qui semble le plus pertinent.

Il est tout à fait possible avec l'algorithme ${\alpha_i}^{+\prime}$ d'obtenir le même résultat. Il suffit à l'analyste d'ajouter la relation $a > c$. Alors, pourquoi ne pas nous servir du calcul fait par l'algorithme α^+ (ou l'algorithme $\alpha^{+\prime}$) pour ajouter cette relation directement ? Nous ne savons pas s'il est légitime d'inférer cette relation, puisqu'elle n'a pas été observée dans les traces. Cependant, nous pourrions suggérer à l'analyste d'éditer ces relations de succession, qui sont vraisemblables. Ainsi, il s'agirait d'une heuristique permettant de faciliter l'interaction entre l'analyste et l'algorithme.

Définir des heuristiques qui permettent d'assister le travail de l'analyste

Dans le chapitre 5, nous avons suggéré l'idée de développer une sorte de chaînage mixte dans le cadre de l'algorithme α. L'idée étant de chercher des situations dans lesquelles l'algorithme α (ou une de ses variantes) est mis en échec pour proposer à l'analyste des pistes de solution. Autrement dit, il s'agit de découvrir des heuristiques et de s'appuyer sur celles-ci pour proposer des choix à l'analyste. L'heuristique dont nous avons parlé au paragraphe précédent en est un exemple.

Les problèmes liés aux boucles de longueur un ou aux transitions facultatives pourraient aussi être détectés par des heuristiques. De la même manière, il serait pertinent d'étudier d'autres constructions particulières qui se produisent lorsque l'on travaille sur des WF-log incomplets, ou sur des WF-log complets, mais de WF-net qui ne sont pas structurés et conformes, etc. Ces études permettraient de guider l'analyste en lui suggérant un point d'entrée pour initier l'interaction.

Posttraiter le réseau de Petri résultat

Nous avons développé les interactions au niveau des résultats intermédiaires de certains algorithmes de la famille α, mais, comme nous l'avons souligné au chapitre 5, l'interaction pourrait aussi se faire après la construction du réseau de Petri, via des posttraitements.

Certains algorithmes de la littérature s'appuient d'ailleurs sur un posttraitement pour ajouter des informations (ou modifier) le réseau de Petri résultat :

- L'algorithme α^{++} (Wen *et al.*, 2007a), s'appuie sur le résultat de l'algorithme $\alpha^{+\prime}$ et le compare au WF-log pour identifier des constructions de type choix non libre et modifier le réseau de Petri. Il serait pertinent d'utiliser un tel posttraitement dans notre contexte d'analyse : en effet l'activité de conduite contient des choix non libres (un choix futur peut être contraint par des actions passées).
- L'outil EMiT exploite le résultat de l'algorithme α et ajoute au réseau de Petri des contraintes temporelles.

De la même manière, nous pourrions imaginer de mettre en œuvre un tel posttraitement pour, par exemple, construire un réseau de Petri coloré. Les réseaux de Petri colorés ont l'avantage de pouvoir exprimer des contraintes supplémentaires, permettant ainsi de décrire l'activité de manière plus expressive.

8.1.2 Les perspectives d'AUTOMATA

Le logiciel AUTOMATA est un prototype, dont l'objectif est d'expérimenter les propositions développées dans cette thèse. Ces retours permettent d'imaginer de nombreuses pistes d'améliorations pour une version future du logiciel.

Ajouter des heuristiques

Tout d'abord, il serait pertinent d'intégrer les différentes heuristiques dont nous avons discuté jusqu'à présent et les éventuels posttraitements sur le réseau de Petri. Il serait intéressant d'utiliser ces heuristiques et posttraitements pour enrichir les possibilités d'interaction entre l'analyste de l'algorithme. Cela peut se produire de deux manières : l'algorithme suggère à l'analyste d'appliquer l'heuristique, puis il applique l'heuristique sans intervention de l'analyste et produit un résultat ; l'algorithme s'appuie sur l'heuristique pour proposer à l'analyste des changements au niveau des résultats intermédiaires (par exemple) ou du prétraitement des traces, etc. La deuxième solution a l'inconvénient d'être potentiellement plus coûteuse pour l'analyste, mais elle a l'avantage de lui offrir un réel contrôle. Il peut ainsi vérifier que l'heuristique utilisée est pertinente.

Rendre interactive (ou au moins visible) la façon dont le réseau de Petri est calculé

Au-delà de ses dimensions algorithmiques, la mise en œuvre de l'algorithme avec un analyste nous a permis d'imaginer d'autres interactions possibles. À partir du réseau de Petri résultat, il serait intéressant de savoir quels sont les éléments (présents ou absents) des traces qui ont permis d'obtenir ce résultat. Par exemple, pourquoi telle transition est-elle connectée à telle place ? Ou pourquoi telle transition n'a pas de place

d'entrée ? Avec le logiciel actuel, il est nécessaire d'exécuter mentalement l'algorithme pour comprendre comment le réseau de Petri a été construit. C'est d'une part difficile et cela nécessite des compétences non négligeables en mathématiques. Ainsi, s'il était possible d'offrir une interface permettant, au niveau du réseau de Petri d'afficher les éléments qui ont été pris en compte pour construire le réseau de Petri tel qu'il est, cette information serait précieuse pour l'analyste. D'une part elle permettrait à l'analyste de s'approprier plus facilement l'outil et de mieux comprendre l'impact des relations de succession sur le réseau de Petri final. D'autre part, cette interaction permettrait de faciliter l'interaction avec l'analyste, en lui permettant de savoir sur quelles relations de succession il doit agir pour impacter une place donnée du réseau de Petri.

Garder le lien entre les résultats intermédiaires et les $\mathcal{M}$-Traces

De la même manière, pour faciliter l'interaction, il serait souhaitable d'offrir dans AUTOMATA, un moyen de savoir dans quelles séquences une relation de succession particulière a été observée. En particulier, s'il était possible de savoir exactement combien de fois une relation de succession a été observée et sur quelles $\mathcal{M}$-Traces du logiciel ABSTRACT elles ont été observées, l'analyste serait guidé d'autant plus rapidement vers la source d'un éventuel problème. Plus que l'interaction au niveau d'AUTOMATA, c'est l'interaction avec la chaîne de découverte de connaissances qui s'en trouverait améliorée.

Vers une pérennisation des développements

Le prototype AUTOMATA nous a permis de tester l'approche et de proposer des pistes d'amélioration. S'il était question de s'inspirer d'AUTOMATA pour proposer un logiciel mature, quels choix faudrait-il faire ? Nous pensons qu'il y a deux solutions.

Tout d'abord, il existe un framework logiciel dédié à la fouille de workflow : le framework ProM[1]. ProM est un framework open source développé par la communauté de la fouille de workflow[2]. À ce titre, il regroupe, sous forme de modules de nombreux algorithmes de fouille de workflow, ainsi que des outils d'analyse des WF-logs et des WF-nets produits. Ce framework vise à proposer un standard pour les outils de fouille de workflow. Il serait donc tout à fait pertinent d'intégrer AUTOMATA dans ce framework, à la fois pour faire profiter la communauté de la fouille de workflow de nos travaux et pour faire profiter l'analyste des autres outils déjà présents dans le framework ProM. Ne connaissant pas la complexité du framework ProM, il nous est difficile d'évaluer le coût de l'intégration d'AUTOMATA dans le framework ProM.

Une deuxième approche consisterait à intégrer AUTOMATA dans la chaîne d'outils déjà existants : en particulier mieux interconnecter AUTOMATA avec le logiciel ABSTRACT. Le cycle de découverte de connaissances tel que nous le mettons en œuvre nécessite de nombreux retours en arrière, notamment au niveau des $\mathcal{M}$-Traces. L'intégration entre les outils ABSTRACT et AUTOMATA est donc extrêmement importante pour fluidifier le travail de l'analyste. En particulier, il serait pertinent de pouvoir, depuis AUTOMATA accéder aux $\mathcal{M}$-Traces qui sont à l'origine d'un résultat intermédiaire particulier. Mais aussi, dans l'autre sens, de pouvoir directement depuis ABSTRACT lancer le calcul d'un réseau de Petri. Connaissant l'architecture des logiciels BIND et ABSTRACT, nous savons qu'il est possible de mettre en œuvre techniquement cette intégration, notamment entre ABSTRACT et AUTOMATA.

[1] Site Web du projet : http://www.promtools.org/prom6/.
[2] Site web dédié à la fouille de workflow : http://www.processmining.org/

Il s'agit de deux objectifs différents. L'implémentation dans le framework ProM vise la communauté de la fouille de workflow. L'implémentation dans la chaîne de découverte de connaissances composée de BIND et ABSTRACT vise la communauté des analystes de l'activité.

8.2 La méthodologie : retour sur les choix d'ingénierie des connaissances

8.2.1 Qu'apporte AUTOMATA à la chaîne de découverte de connaissances ?

Bien que n'ayant pas été explicitée auparavant, la chaîne de découverte de connaissances allant jusqu'à ABSTRACT existait déjà avant cette thèse. Georgeon (2008) a en effet exploité une méthodologie et des outils similaires pour analyser des traces d'activité de conduite avec le logiciel ABSTRACT.

Apports d'AUTOMATA pour l'analyste

Est-ce pertinent d'ajouter une étape supplémentaire d'analyse ? Qu'est-ce que cela change d'ajouter l'étape AUTOMATA ? D'abord, nous avons constaté que l'approche AUTOMATA vise d'emblée à étudier en parallèle différentes occurrences d'une même activité, alors que l'approche ABSTRACT, orientée sur les traces, se focalise sur l'analyse d'une activité particulière. Cela caractérise selon nous la différence conceptuelle majeure entre ABSTRACT et AUTOMATA. Les $\mathcal{M}$-Traces racontent une histoire singulière, alors que le réseau de Petri synthétise toutes ces histoires en une interprétation particulière. Ainsi AUTOMATA permet réellement d'apporter une nouvelle dimension d'analyse : on passe de l'étude de l'activité phénotypique (les traces) à une étude de l'activité génotypique (un modèle sous forme de réseau de Petri). Le travail effectué par AUTOMATA s'apparente aussi aux travaux sur l'inférence grammaticale, qui consistent à inférer des grammaires à partir de textes (de la Higuera, 2005).

Implications théoriques

Ce saut conceptuel, passant de l'analyse d'un cas singulier à l'analyse d'un ensemble de cas représentant une situation particulière, a un impact sur notre façon de travailler avec ABSTRACT. Nous avons du mettre en place des moyens de travailler sur plusieurs traces en parallèle (import, export, visualisation et transformation des $\mathcal{M}$-Traces). Le travail sur plusieurs traces en parallèle pose des questions théoriques intéressantes. Par exemple, a-t-on le droit de comparer plusieurs $\mathcal{M}$-Traces qui ne partageraient pas le même modèle ? Est-ce que cela aurait du sens ? Comment définir ces ensembles de $\mathcal{M}$-Traces qui partageraient le même modèle ? Ces questions méritent d'être posées dans le cadre de la théorie des traces modélisées, développée par l'équipe SILEX.

De même, il serait pertinent de faire converger la théorie des traces modélisées et la méta-théorie des traces proposée par Deransart (2010). Notre travail de thèse se trouve à l'articulation de ces deux théories.

Après AUTOMATA ?

Maintenant qu'AUTOMATA fait partie de la chaîne de découverte de connaissances, qu'est-ce que cela change pour le futur ? Nous avons déjà mentionné dans cette thèse

différentes motivations : construire des connaissances pour mieux comprendre l'activité étudiée, construire des modèles de l'activité, capables de simulation.

Ainsi, l'étape suivante consiste à utiliser ces connaissances générées. Quel usage pour le chercheur en ergonomie cognitive ou pour l'ergonome ? Nous discuterons plus loin les usages de ces connaissances pour le chercheur en ergonomie cognitive, qui construit un modèle de l'humain, ou pour l'ergonome, qui construit un système d'assistance à la conduite.

Et pour l'ingénieur de la connaissance ? Il serait envisageable d'utiliser des techniques d'identification de motifs pour trouver dans les traces les moments où le modèle a été découvert. Cette identification pourrait être utilisée comme une nouvelle technique de transformation de $\mathcal{M}$-Traces : chaque fois que le modèle est détecté, un nouvel *obsel* serait construit et remplacerait le motif détecté.

8.2.2 Argumenter les connaissances construites

Cette thèse vise à proposer des méthodes de construction de connaissances. Ces connaissances sont développées dans le cadre scientifique de l'analyse de l'activité de conduite automobile. Les algorithmes que nous choisissons pour supporter ce travail de construction de connaissances doivent être déterministes. Une fois les connaissances découvertes, pour pouvoir garantir leur valeur scientifique, il nous semble important de pouvoir reproduire les transformations qui ont permis, à travers les transformations successives de données, de produire des connaissances synthétiques. Il s'agit de pouvoir assurer la réfutabilité des connaissances, proposée par Popper (1979) (voir chapitre 6), que nous proposons d'appliquer au cycle de découverte de connaissances.

BIND : un framework métier

Dans le cycle de découverte de connaissances tel qu'il est mis en œuvre à l'heure actuelle, l'argumentation de la construction des connaissances se répartit à plusieurs niveaux. Au niveau du framework BIND, les connaissances ne sont pas nécessairement explicites. En effet, les transformations appliquées aux données sont enregistrées dans des scripts informatiques. Cependant, le coût d'explicitation des connaissances est rendu plus faible, car il s'agit d'un framework métier et que les méthodes permettant de transformer les données sont d'un niveau d'abstraction suffisamment élevé pour faire sens à un ingénieur voulant comprendre les calculs effectués.

ABSTRACT : les transformations de $\mathcal{M}$-Traces

Au niveau du logiciel ABSTRACT, l'ontologie explicite la sémantique (*observationnelle*, voir 3) des $\mathcal{M}$-Traces. Les transformations de $\mathcal{M}$-Traces contiennent, elles, une sémantique (*interprétationnelle*) décrivant comment reformuler les traces pour, par exemple, inférer des processus qui n'ont pas été directement observés.

Les transformations, dans ABSTRACT, sont des requêtes SPARQL. L'interprétation d'une requête n'est donc pas nécessairement facile. Cependant, dans la pratique, nous constatons que la sémantique des transformations est relativement facile à matérialiser. En effet, le logiciel ABSTRACT et le langage de requêtes SPARQL encouragent l'utilisateur à construire des requêtes unitaires, se focalisant sur la construction d'un type d'*obsel*s à la fois. Une bonne pratique consiste à documenter ces requêtes unitaires. Cette documentation de la requête doit prendre place à la fois au niveau du code source de la requête, et au niveau de l'ontologie de la $\mathcal{M}$-Trace produite : le nouveau type

d'*obsel*s doit être documenté dans l'ontologie et un lien vers la (ou les) requête(s) ayant permis de générer ce type d'*obsel*s permet d'assurer la robustesse de la démarche.

Enfin, ABSTRACT garde la trace des différentes transformations appliquées aux $\mathcal{M}$-Traces via un graphe de transformations[3]. Ce graphe de transformation, combiné à la documentation des requêtes de transformations, permet d'expliciter la façon dont les $\mathcal{M}$-Traces ont été construites. Même dans le cadre de la démarche exploratoire de construction des connaissances que nous adoptons, nous gardons la trace de la construction du résultat.

Cette explicitation de la façon de construire le résultat est essentielle pour deux raisons. Premièrement, elle permet au chercheur de partager avec sa communauté les connaissances construites et la façon dont elles sont construites. Deuxièmement, cette explicitation permet à d'autres chercheurs de vérifier le résultat et de contester (ou approuver) les choix qui ont été faits par l'analyste. Par exemple, d'autres chercheurs pourraient reproduire les analyses sur un nouveau jeu de données.

AUTOMATA : conserver l'histoire des interactions

Au niveau du logiciel AUTOMATA, enfin, les algorithmes que nous avons choisis (algorithmes de la famille α) fonctionnent sur un principe déterministe (contrairement à des algorithmes génétiques par exemple), ce qui garantit la possibilité d'argumenter le résultat à partir des différentes transformations que nous effectuons. Encore faut-il tracer ces transformations. Dans AUTOMATA, nous n'avons pas (encore) mis en œuvre de techniques permettant de garder la trace de la construction du réseau de Petri. Notamment, nous ne gardons pas la trace des interactions entre l'analyste et l'algorithme pour produire un réseau de Petri donné. Cette information manque pour permettre de reproduire (ou comprendre) a posteriori comment un réseau de Petri a été construit. Il est essentiel dans notre démarche de garder l'histoire de ces interactions et de la construction du résultat. Ainsi, pour chaque résultat intermédiaire (relation $>$ ou $\triangle$), il est possible de recalculer comment il a été dérivé des traces, mais en plus, il faudrait savoir si l'analyste a validé, invalidé, ou n'a pas donné d'avis sur ce résultat intermédiaire. Enfin, idéalement, pour comprendre pourquoi l'analyste a validé ou invalidé tel ou tel résultat intermédiaire, il faudrait lui permettre d'annoter ses choix.

Tracer l'activité de l'analyste ?

Pour conserver l'histoire de la découverte de connaissances, il serait pertinent d'instrumenter les outils de découverte de connaissances pour conserver l'histoire de tous les choix effectués. Ce que nous proposons ici, c'est tracer l'activité de découverte de connaissances[4].

Tracer l'activité de découverte de connaissances aurait plusieurs avantages. Premièrement, comme nous l'avons déjà souligné cela permettrait d'expliquer à une autre personne comment tel modèle a été construit, quitte à simplifier l'histoire de la découverte de connaissances en supprimant les pistes explorées qui n'ont pas abouti.

Deuxièmement, cela permettrait à l'analyste de disposer de l'histoire de son activité d'analyste. Mieux encore, s'il peut accéder à la trace de son activité en temps réel, cela lui donnerait un support réflexif de son activité de découverte de connaissances. Des travaux montrent qu'un retour réflexif améliore la métacognition de l'utilisateur

[3] Il s'agit d'un graphe acyclique orienté.

[4] Falzon (1998a) présente l'intérêt de « Tracer et évaluer les activités des ergonomes ».

(ici, l'analyste) et favorise l'appropriation des outils techniques utilisés (Ollagnier-Beldame, 2006).

Enfin, les traces de l'activité de construction de connaissances sont des traces d'activité. Au même titre que les traces de conduite automobile, elles pourraient être exploitées dans le système de découverte de connaissances que nous avons mis en place. Ainsi, il serait possible d'analyser l'activité de l'analyste...

8.3 Analyse de l'activité humaine

8.3.1 Apports de la chaîne de découverte de connaissances pour l'analyste

L'analyse de données complexes telles que les données relatives à l'activité de conduite automobile est une tâche difficile. Elle nécessite, de la part de l'analyste, de poser des questions précises et d'appliquer une méthodologie rigoureuse pour répondre à ces questions. Étudier des schémas de conduite à partir de données issues de capteurs sur un véhicule instrumenté relève de la gageure.

C'est pour cette raison que nous avons développé une approche méthodologique de découverte de connaissances. Cette méthodologie dépasse le cadre de cette thèse. Elle est le fruit d'années de collaborations entre le laboratoire LESCOT, spécialisé dans l'analyse de données de conduite automobile et l'équipe SILEX du laboratoire LIRIS, spécialisé dans l'ingénierie des connaissances.

Nous avons montré que l'approche méthodologique de découverte de connaissances permet techniquement et conceptuellement de construire des schémas de conduite automobile à partir de données collectées sur un véhicule instrumenté. Cette approche pourrait être généralisée à des données issues d'un simulateur de conduite, ou bien d'une flotte de véhicules. Nous pensons même que l'approche méthodologique et une partie des outils (principalement ABSTRACT) pourraient être utilisés pour l'analyse d'activités humaines en général.

Les forces de la chaîne d'outils de découverte de connaissances

Lors de l'exploitation de l'ensemble de la chaîne de découverte de connaissances, nous avons pu constater différents paliers d'appropriation de la chaîne d'outils.

Le premier palier correspond à l'appropriation du framework BIND. L'utilisation de BIND pour l'exécution de scripts d'analyse de données facilite le travail de l'analyste. Les personnes qui ont l'habitude d'utiliser des outils d'analyse ne seront pas surprises par cet usage de BIND. Mais la capacité du framework à concevoir des applications de visualisation synchronisées de courbes de données et de vidéos constitue un premier palier. Pour analyser correctement les données de conduite et bien comprendre les actions du conducteur, les données issues de capteurs sont souvent insuffisantes et il est nécessaire d'accéder à la vidéo. Un tel outil, par ses capacités de visualisation et de synchronisation des données et des vidéos, est précieux pour le travail de l'analyste. Le framework BIND est d'autant plus appréciable qu'il est facile de construire une application de visualisation répondant aux besoins spécifiques d'une analyse.

Le deuxième palier correspond à l'appropriation du logiciel ABSTRACT. Nous avons constaté deux niveaux d'appropriation. Le premier niveau d'appropriation consiste à utiliser ABSTRACT comme un outil de visualisation et de formalisation. L'analyste construit une ontologie (et donc formalise) les concepts qu'il manipule et

importe des $\mathcal{M}$-Traces premières dans ABSTRACT pour les visualiser. Ce premier niveau d'appropriation permet à l'analyste d'exploiter l'outil ABSTRACT comme un support à l'analyse, à la formalisation et à la visualisation.

Le troisième palier d'appropriation vient avec l'utilisation des transformations de $\mathcal{M}$-Traces dans ABSTRACT. Les transformations de $\mathcal{M}$-Traces sont plus difficiles à s'approprier, car elles nécessitent d'utiliser un langage de requête qui peut paraître abscons à l'analyste. Cependant, les requêtes de transformation offrent toute la puissance du calcul symbolique. Elles permettent à l'analyste de construire des analyses nettement plus complexes et poussées, par exemple, en reconstruisant le niveau tactique de l'activité à partir d'observations relevant du niveau opérationnel.

Enfin, le dernier palier d'appropriation correspond à l'usage de l'outil AUTOMATA. L'outil AUTOMATA n'est pas assez mûr pour réellement juger de son appropriation. Nous avons déjà énoncé au début de cette discussion un certain nombre de moyens pour améliorer l'interaction entre l'analyste et AUTOMATA. Une meilleure interaction faciliterait nettement l'appropriation de l'outil. Cependant, est-ce suffisant ? Nous avons constaté que l'utilisation des réseaux de Petri nécessite de former l'analyste à leur lecture. Cela est vrai pour tout formalisme, mais nous avons constaté que la lecture des réseaux de Petri reste moins intuitive que celle d'un automate à états finis, ou même d'un Statechart.

Est-ce qu'il serait souhaitable de changer de formalisme de représentation des connaissances synthétiques ? Nous pensons que non. Nous avons choisi les réseaux de Petri pour leur expressivité et nous ne pensons pas que ce choix soit à remettre en cause. Faut-il former les analystes à la lecture des réseaux de Petri ? C'est certainement une solution, mais limite l'éventuelle diffusion des connaissances construites aux seuls experts connaissant les réseaux de Petri. Nous pensons qu'une solution hybride serait d'offrir des outils de visualisation du résultat avec des annotations ou avec un autre formalisme plus intuitif. Nous pensons par exemple aux travaux visant à convertir certains réseaux de Petri sous forme de Statecharts (Eshuis, 2011).

Vers une adoption généralisée de la chaîne de découverte de connaissances chez les analystes ?

Lors de nos discussions ou collaborations avec différents analystes, nous leur avons présenté les méthodes et outils de découverte de connaissances que nous développons dans cette thèse. Les retours des analystes sont positifs. Ils sont souvent intéressés par la méthode et les outils que nous proposons.

Cependant, nous avons constaté qu'il est parfois difficile d'aller jusqu'à la mise en œuvre pratique de ces outils et méthodes. Pourquoi ? Nous avons identifié trois raisons. Tout d'abord, l'analyste doit utiliser de nouveaux outils, ce qui nécessite un apprentissage. Ensuite, nous pensons qu'il y a un saut conceptuel au niveau de la méthodologie qui est parfois difficile à passer pour l'analyste. Enfin, il nous semble qu'un dernier élément de réponse vient de la difficulté à changer d'outils. Les analystes sont habitués à leurs techniques et outils d'analyse et le changement ne leur semble pas mériter le coût d'apprentissage de nouveaux outils. Ceci est d'autant plus vrai que si l'analyste est familier avec un langage de programmation, il peut se construire ses propres outils d'analyse ad hoc.

L'utilisation de l'approche que nous avons développée a donc un coût humain. Mais notons qu'il est plus facile d'utiliser notre approche dans le cadre d'une nouvelle analyse de données, qu'au cours d'une analyse existante. La raison en est simple : les coûts en termes de conversion de données. En effet, pour une expérimentation,

la quantité de données collectées va de plusieurs giga-octets de données à plusieurs dizaines de giga-octets (et encore plus dans le cadre des études en situation naturelle de conduite, où des centaines de véhicules sont instrumentés). Le coût de conversion des jeux de données dans un format à un autre n'est pas anecdotique. Nous l'avons vu dans le cadre des expérimentations menées à l'Université de Californie, Berkeley, où la conversion des données d'un seul sujet a pris une semaine de temps de calcul. La conversion des données doit donc se faire à la volée, et non pas a posteriori.

La démarche d'analyse que nous proposons dans cette thèse est déjà en cours d'utilisation pour l'analyse de données de conduite automobile dans le cadre du projet Partage. Cette approche sera aussi utilisée dans le cadre d'une thèse en cours au laboratoire LESCOT.

8.3.2 L'art de la découverte de connaissances

Nous l'avons souligné en présentant le cycle de découverte de connaissances, l'analyse de données de conduite automobile n'est jamais le travail isolé d'un seul analyste. C'est le fruit d'une collaboration transdisciplinaire. Par exemple, l'instrumentation du véhicule requiert des compétences en mécanique, électronique, vidéo et informatique. L'analyse des données requiert des compétences en traitement du signal, en informatique, en calcul sur des données cartographiques, en psychologie cognitive et en ergonomie. Un travail en collaboration est essentiel, car la robustesse de l'interprétation faite par l'ergonome dépend de la robustesse du calcul conçu par l'informaticien, qui elle-même dépend de la robustesse de la mesure des capteurs.

La méthodologie que nous avons proposée met l'interaction avec le (ou les) analyste(s) au centre du processus de découverte. D'un point de vue méthodologique, cela peut sembler poser problème : l'analyste ajoute ses connaissances pour découvrir ce qu'il cherche. N'y a-t-il pas un risque, en utilisant cette méthodologie, de construire « n'importe quelle » connaissance ?

Ce risque existe. C'est évident. Il est tout à fait possible de « manipuler » les données pour y trouver ce que l'on y cherche. Mais n'est-ce pas le cas de toute analyse qui manque de rigueur ? La pertinence de la connaissance produite dépend, comme dans toute analyse, de la pertinence de la mesure (incluant la pertinence du protocole expérimental) et de la pertinence de l'analyse et des connaissances ajoutées par l'analyste. La chaîne d'outils que nous proposons offre la possibilité de savoir comment la connaissance a été produite. C'est la clé de la validité d'une analyse : l'analyste peut expliquer comment il a produit les résultats et confronter cette analyse à sa communauté.

L'analyse de données de conduite automobile est complexe. Elle nécessite un grand niveau d'expertise d'acteurs de différents domaines. Ces expertises ne sont pas toujours formalisées et relèvent parfois d'un savoir-faire. Même si de nombreuses études sont faites sur des données de conduite automobile, elles n'exploitent que rarement la richesse de ces données, à cause, peut-être de manque de moyens, mais surtout de la difficulté que cela représente.

Nous espérons que la méthodologie que nous proposons permettra d'expliciter l'expertise (en associant l'art et la rigueur) et fournira aux analystes un cadre les aidant à produire des analyses plus poussées des données de conduite automobile.

8.3.3 Perspectives

Exploitation des connaissances produites

La chaîne de découverte de connaissances offre de nouvelles perspectives d'analyse. Il est donc nécessaire de mettre en œuvre des analyses approfondies telles que le permettent les outils existants. Le travail d'analyse à mettre en œuvre est important. Il est colossal quand il est question de construire des bases de données de schémas de conduite pour alimenter le modèle de simulation cognitive COSMODRIVE.

Avant d'en arriver là, les outils d'analyse permettent à l'ergonome cogniticien d'analyser l'activité de conduite automobile et de mieux la comprendre. L'explicitation des connaissances tout au long du cycle de découverte de connaissances lui fournit un support lui permettant de partager et de publier ces connaissances.

En parallèle, les premiers schémas de conduite pourront être intégrés dans le modèle de simulation cognitive COSMODRIVE pour produire des performances de conduite et les comparer aux performances humaines. La simulation cognitive pourrait ainsi servir d'outil de validation et d'investigation scientifique de l'activité cognitive associée à la conduite automobile.

En vue d'objectifs plus applicatifs, l'ergonome peut aussi exploiter les connaissances sur l'activité, construites avec nos méthodes. D'abord, en comprenant mieux les erreurs liées à la conduite, l'ergonome pourrait proposer des formations destinées aux conducteurs. Ensuite, en connaissant mieux les besoins réels de l'humain, l'ergonome peut concevoir des systèmes d'assistance à la conduite plus pertinents. C'est l'objectif des méthodes de conception anthropocentrées (*Human-Centered Design*). Mieux encore, si l'ergonome dispose d'un modèle de l'humain, pendant les phases amont de la conception, il peut tester le système d'assistance avec le modèle de l'humain. Avec suffisamment de connaissances sur l'activité de conduite, il serait même possible de concevoir des systèmes de copilotage, où l'activité de conduite serait partagée entre l'humain et le véhicule (Bellet *et al.*, 2011a).

Exploitation de l'expertise explicitée

Brachman et Anand (1996) distinguent deux types d'environnements logiciels pour la découverte de connaissances. Le premier environnement, c'est celui auquel on s'est intéressé tout au long de cette thèse : un environnement, très interactif, associé à une découverte itérative et exploratoire de connaissances par un analyste, qui utilise ses propres connaissances, pour guider une découverte qui à son tour va enrichir ses connaissances. Le deuxième type d'environnement est un environnement qui exploite les modèles ou paramètres déjà découverts pour fabriquer une application qui propose d'elle même (avec les modèles découverts en amont) une analyse des données.

Une telle application permettrait d'analyser automatiquement l'activité de conduite automobile. De nouvelles méthodes d'études de l'activité seraient alors rendues possibles. Une première application consiste à analyser l'activité du conducteur pour disposer après l'expérimentation d'un support d'autoconfrontation avec le conducteur. Les logiciels existants permettent déjà la mise en place d'une telle autoconfrontation. Une deuxième application serait d'analyser en temps réel, dans le véhicule instrumenté, l'activité du conducteur et lui fournir un retour sur ce qu'il fait.

Conclusion

Cette thèse s'intègre dans le cadre de l'analyse de l'activité humaine. L'analyse de l'activité passe par un processus d'observation et d'interprétation. Les données sur lesquelles travaille l'analyste ne sont jamais l'activité elle-même, mais des traces d'observation de cette activité. Pour comprendre l'activité sous-jacente à ces traces d'activité, l'analyste doit interpréter ces traces à l'aide de ses connaissances.

Pour mieux comprendre l'activité de conduite automobile, les chercheurs en ergonomie cognitive cherchent à modéliser cette activité, par nature dynamique, complexe et singulière. L'ergonome s'appuie sur des modèles théoriques de la cognition humaine pour comprendre l'activité, mais encore faut-il pouvoir construire et valider ces modèles. Le modèle de simulation cognitive COSMODRIVE, à ce titre, est un outil d'investigation scientifique de l'activité de conduite automobile et de ses composantes cognitives. Mais pour produire une simulation, il faut fournir des connaissances à ce modèle, sous la forme de schémas de conduite.

Les schémas de conduite constituent une connaissance synthétique (génotypique) de l'activité, l'activité dans sa potentialité, contrairement aux traces d'observations, qui enregistrent une histoire particulière (phénotypique) de l'activité telle qu'elle s'est déroulée dans la singularité de la situation. Construire des schémas de conduite revient à construire un modèle synthétique à partir de traces d'observation.

L'objectif applicatif de cette thèse était de concevoir des outils pour un analyste, lui permettant de construire des modèles d'une activité humaine à partir de traces de cette activité et de ses connaissances d'analyste.

Ce travail d'analyse pose de nombreux problèmes d'ingénierie des connaissances. Les deux problèmes majeurs auxquels cette thèse vise à répondre sont : 1) les différents niveaux d'abstraction étudiés et mesurés et 2) la construction de modèles synthétiques (génotypes), à partir de traces d'activité (phénotype).

Alors que l'on cherche à créer des modèles relatifs à l'activité et à la cognition du conducteur, les mesures issues du véhicule instrumenté relèvent plutôt de mesures physiques, par exemple sensori-motrices. Il existe une différence de nature entre le type d'information présente dans les traces et l'information recherchée.

La solution à ce problème nécessite l'utilisation de connaissances expertes venant de l'analyste. Nous proposons d'exploiter le cycle de découverte de connaissances présenté par Fayyad *et al.* (1996b) pour offrir à l'analyste un cadre méthodologique pour analyser ces données. Nous soulignons le rôle central de l'analyste et de ses connaissances dans ce cycle de découverte.

Notre contribution théorique consiste à modifier ce cycle de découverte de connaissances pour permettre à l'analyste d'intégrer explicitement ses connaissances dans le processus de découverte (Mathern *et al.*, 2011a). Pour cela, nous proposons de nous appuyer sur un système à base de connaissances particulier, un SBT. L'utilisation d'un système à base de connaissances offre à l'analyste la possibilité d'expliciter ses connaissances. L'utilisation des traces modélisées offre un cadre théorique permettant de gérer formellement les différents niveaux d'abstraction. Ce cadre offre une solu-

tion méthodologique permettant de transformer des traces pour expliciter des niveaux d'abstraction qui ne sont pas directement observés.

Les différentes étapes du cycle de découverte de connaissances ont été bien étudiées dans la littérature. Mais l'étape d'interprétation (et la construction réelle de connaissances) est exécutée par l'analyste seul. En offrant des outils de construction de modèles synthétiques à partir de traces, nous proposons d'aider l'analyste dans cette phase d'interprétation et de construction des connaissances. Pour pouvoir représenter les connaissances que sont les schémas de conduite, nous avons choisi le formalisme des réseaux de Petri. Les réseaux de Petri sont en effet des modèles synthétiques et sont capables de représenter une activité dynamique et contrainte telle que l'activité de conduite automobile.

Dans la littérature, la fouille de workflow étudie la question de la construction de modèles synthétiques à partir de traces d'une activité. Nous avons présenté une revue de la littérature des algorithmes de fouille de workflow utilisant le formalisme des réseaux de Petri. Mais pour que le modèle produit soit pertinent, ces algorithmes requièrent des jeux de traces vérifiant des contraintes de complétude. Or, nous savons que les jeux de traces de conduite automobile ne sont pas complets. C'est pourquoi, là encore, nous proposons d'introduire l'analyste dans le processus de construction de modèles.

Notre contribution formelle consiste à montrer comment des algorithmes de fouille de workflow peuvent être modifiés pour rendre possible cette interaction avec l'analyste. Nous avons d'abord proposé une version modifiée de l'algorithme α, permettant à l'analyste de valider ou d'invalider des résultats intermédiaires calculés par l'algorithme (Mathern *et al.*, 2010b). Puis, nous avons montré comment généraliser cette interaction à d'autres algorithmes de la famille α. Les algorithmes α_i, α'_i, α_i^+, $\alpha_i^{+\prime}$ sont le fruit de cette contribution.

Enfin, nous avons tenu à évaluer nos deux premières contributions. Cependant, nous avons expliqué pourquoi il est extrêmement difficile d'évaluer des outils génératifs avec les méthodes d'évaluation scientifiques classiques. En effet, en introduisant l'analyste dans l'approche méthodologique que nous avons développée, il n'est plus possible d'évaluer l'outil seul. Ce que l'on évalue, c'est le couplage entre l'analyste et l'outil, dans le cadre de l'exécution d'une tâche particulière.

Pour produire cette évaluation, nous avons développé une implémentation des algorithmes interactifs dans un prototype logiciel que nous avons appelé AUTOMATA. Nous avons eu besoin d'ajouter de nouvelles fonctionnalités au logiciel ABSTRACT (Mathern *et al.*, 2011b) pour autoriser le travail sur plusieurs traces modélisées en parallèle. Lors de cette évaluation, nous avons travaillé en collaboration avec des analystes sur des données de conduite réelles. Nous avons ainsi montré comment un analyste applique la méthodologie et les algorithmes interactifs pour construire des modèles de l'activité de conduite automobile.

Cette évaluation constitue la contribution pragmatique de notre travail. Tout d'abord, cette évaluation a montré que malgré la difficulté du travail d'analyse, l'ensemble de la chaîne de découverte de connaissances pouvait être mis en place sur des données réelles. Ensuite, nous avons montré en quoi l'utilisation des algorithmes interactifs était pertinente pour l'analyste.

Les retours d'évaluation nous ont permis de souligner les limites des algorithmes utilisés et de proposer des pistes d'amélioration. En particulier, vu l'ampleur du travail de l'analyste, il est important de pouvoir assister son travail au maximum. Dans ce but, nous proposons de développer des heuristiques pour alerter l'analyste d'éventuels problèmes dans les données ou pour posttraiter le réseau de Petri produit. Pour améliorer l'interaction et faciliter le flux de travail de l'analyste, nous proposons de

développer des interfaces conservant les liens entre les différentes étapes du processus, par exemple, pour pouvoir visualiser les traces qui ont été utilisées par l'algorithme afin de produire un résultat intermédiaire particulier.

Enfin, ce travail de thèse s'inscrit dans un cadre de recherche plus large. Nous avons vérifié que ce travail de thèse offre à l'analyste des outils lui permettant de construire des modèles de l'activité de conduite automobile à partir de traces de cette activité et de ses connaissances d'analyste. La prochaine étape sera d'exploiter ces outils de découverte de connaissances pour produire des schémas de conduite. Ensuite, ces schémas de conduite pourront être utilisés par les analystes pour publier les connaissances découvertes, ou être intégrés dans un modèle de simulation cognitive pour produire une activité de conduite.

Bibliographie

ALLEN, T. M., LUMENFELD, H. et ALEXANDER, G. J. (1971). Driver information needs. *Highway Research Record*, 366:102–115.

ANASTASI, A. (1976). *Psychological Testing*. Macmillan, fourth édition.

ANDERSON, J. R. (1983). *The Architecture of Cognition*. Harvard University Press.

BACHIMONT, B. (2004a). *Arts et sciences du numérique : Ingénierie des connaissances et critique de la raison computationnelle*. Habilitation à diriger des recherches, Université de Technologie de Compiègne.

BACHIMONT, B. (2004b). Pourquoi n'y a-t-il pas d'expérience en ingénierie des connaissances ? *In Actes de IC 2004*, Lyon.

BANET, A. (2010). *Conscience du risque et attitudes face aux risques chez les motocyclistes*. Thèse de doctorat, Université Lumière Lyon 2.

BELLET, T. (1998). *Modélisation et simulation cognitive de l'opérateur humain : Une application à la conduite automobile*. Thèse de doctorat, Université Paris V.

BELLET, T., BAILLY-ASUNI, B., MAYENOBE, P. et BANET, A. (2009). A theoretical and methodological framework for studying and modelling drivers' mental representations. *Safety Science*, 47(9):1205 – 1221. Research in Ergonomic Psychology in the Transportation Field in France.

BELLET, T., HOC, J.-M., BOVERIE, S. et BOY, G. (2011a). *Human-Computer Interactions in Transport*, chapitre From Human-Machine Interaction to Cooperation : Towards the Integrated Copilot, pages 129–155. ISTE Ltd and John Wiley & Sons, Inc.

BELLET, T., MAYENOBE, P., BORNARD, J.-C., GRUYER, D. et MATHERN, B. (2010). COSMO-SIVIC : a first step towards a virtual platform for human centred design of driving assistances. *In The 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Valenciennes, France.

BELLET, T., MAYENOBE, P., BORNARD, J.-C., PARIS, J.-C., GRUYER, D. et CLAVERIE, B. (2011b). *Human Modelling in Assisted Transportation : Models, Tools and Risk Methods*, chapitre Human Driver Modelling and Simulation into a Virtual Road Environment, pages 251–262. Springer.

BELLET, T., PARIS, J.-C., DELEURENCE, P., BONNARD, A. et MOREAU, F. (2012). Analyse de sorties de voies observées en condition réelles de conduite. Rapport partage l3-5-2, IFSTTAR.

BELLET, T. et TATTEGRAIN-VESTE, H. (2003). COSMODRIVE : un modèle de simulation cognitive du conducteur automobile. *In Formalismes de modélisation pour l'analyse du travail et l'ergonomie*, Le travail humain, chapitre 3, pages 77–110. PUF.

BERGENTHUM, R., DESEL, J., LORENZ, R. et MAUSER, S. (2007). Process mining based on regions of languages. *In* ALONSO, G., DADAM, P. et ROSEMANN, M., éditeurs : *Business Process Management*, volume 4714 de *Lecture Notes in Computer Science*, pages 375–383. Springer Berlin / Heidelberg.

BERTHOLD, M. R., CEBRON, N., DILL, F., GABRIEL, T. R., KÖTTER, T., MEINL, T., OHL, P., SIEB, C., THIEL, K. et WISWEDEL, B. (2008). Knime : The konstanz information miner. *In* PREISACH, C., BURKHARDT, H., SCHMIDT-THIEME, L. et DECKER, R., éditeurs : *Data Analysis, Machine Learning and Applications*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 319–326. Springer Berlin Heidelberg.

BRACHMAN, R. J. et ANAND, T. (1996). *Advances in knowledge discovery and data mining*, chapitre The process of knowlesge discovery in databases : a human-centered approach, pages 37–57. The MIT Press.

BRAMS, G. (1983a). *Réseaux de Petri : Théorie et pratique*, volume 1. Masson. Théorie et analyse.

BRAMS, G. (1983b). *Réseaux de Petri : Théorie et pratique*, volume 2. Masson. Modélisation et applications.

CHARNIAK, E. (1991). Bayesian networks without tears. *AI magazine*, 12(4):50–63.

CRAM, D., MATHERN, B. et MILLE, A. (2011). A complete chronicle discovery approach : application to activity analysis. *Expert Systems*. Wiley Online Library.

DAPZOL, N. (2006). *Analyse des comportements de conduite par les chaînes de Markov cachées et les modèles de ruptures multiphasiques : méthodologie et applications*. Thèse de doctorat, Université Claude Bernard Lyon 1.

de la HIGUERA, C. (2005). A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332 – 1348. Grammatical Inference.

de MEDEIROS, A. K. A. (2006). *Genetic Process Mining*. Thèse de doctorat, Technische Universiteit Eindhoven, Eindhoven, The Netherlands.

de MEDEIROS, A. K. A., van DONGEN, B. F., van der AALST, W. M. P. et WEIJTERS, A. J. M. M. (2004a). Process mining : Extending the α-algorithm to mine short loops. *In Eindhoven University of Technology, Eindhoven*.

de MEDEIROS, A. K. A., van DONGEN, B. F., van der AALST, W. M. P. et WEIJTERS, A. J. M. M. (2005). Process mining for ubiquitous mobile systems : An overview and a concrete algorithm. *In* BARESI, L., DUSTDAR, S., GALL, H. et MATERA, M., éditeurs : *Ubiquitous Mobile Information and Collaboration Systems*, volume 3272 de *Lecture Notes in Computer Science*, pages 151–165. Springer Berlin / Heidelberg.

de MEDEIROS, A. K. A., WEIJTERS, A. J. M. M. et van der AALST, W. M. P. (2004b). Using genetic algorithms to mine process models : Representation, operators and results. Rapport technique, Eindhoven University of Technology, Eindhoven.

de MEDEIROS, A. K. A., WEIJTERS, A. J. M. M. et van der AALST, W. M. P. (2006). Genetic process mining : A basic approach and its challenges. *In* BUSSLER, C. et HALLER, A., éditeurs : *Business Process Management Workshops*, volume 3812 de *Lecture Notes in Computer Science*, pages 203–215. Springer Berlin / Heidelberg.

DERANSART, P. (2010). Conception de traces et applications (vers une méta-théorie des traces). Document de travail. http://hal.inria.fr/inria-00443648.

DESEL, J. et REISIG, W. (1996). The synthesis problem of petri nets. *Acta Informatica*, 33:297–315.

DIEKERT, V. et ROZENBERG, G., éditeurs (1995). *The book of traces*. World scientific.

ELLIS, G. et DIX, A. (2006). An explorative analysis of user evaluation studies in information visualisation. *In Proceedings of the 2006 AVI workshop on BEyond time and errors : novel evaluation methods for information visualization*, BELIV '06, pages 1–7, New York, NY, USA. ACM.

ENDSLEY, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors : The Journal of the Human Factors and Ergonomics Society*, 37(1): 32–64.

ESHUIS, R. (2011). Statechartable petri nets. *Formal Aspects of Computing*, pages 1–23. Online First.

FALZON, P. (1991). *Modèles en analyse du travail*, chapitre Les activités verbales dans le travail, pages 229–252.

FALZON, P. (1993). Médecin, pompier, concepteur : l'activité cognitive de l'ergonome. *Performances Humaines et Techniques*, 66:35–45.

FALZON, P. (1998a). « Qu'est-ce que la recherche en ergonomie ? ». *In Actes du colloque Recherche et Ergonomie*, Toulouse.

FALZON, P. (1998b). *Des évolutions en ergonomie*, chapitre La construction des connaissances en ergonomie : éléments d'épistémologie. Toulouse : Octarès.

FASTENMEIER, W. et GSTALTER, H. (2010). The human reliability approach in road traffic stafety. *In European Conference on Human Centred Design for Intelligent Transport Systems*, Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems, pages 435–444, Berlin, Germany. HUMANIST publications.

FAYYAD, U. M., PIATETSKY-SHAPIRO, G. et SMYTH, P. (1996a). *Advances in knowledge discovery and data mining*, chapitre From Data Mining to Knowledge Discovery : An Overview, pages 1–30. The MIT Press.

FAYYAD, U. M., PIATETSKY-SHAPIRO, G. et SMYTH, P. (1996b). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–54.

FRAWLEY, W. J., PIATETSKY-SHAPIRO, G. et MATHEUS, C. J. (1992). Knowledge discovery in databases : An overview. *AI magazine*, 13(3):57–70.

GANDON, F. (2011). Les questions épistémologiques de la recherche en ingénierie des connaissances. Table ronde Ingénierie des Connaissances 2011 http://fabien-gandon.blogspot.com/2011/05/les-questions-epistemologiques-de-la.html.

GEORGEON, O. (2008). *Analyse de traces d'activité pour la modélisation cognitive : application à la conduite automobile*. Thèse de doctorat, Université Lyon 2.

GEORGEON, O., MILLE, A., BELLET, T., MATHERN, B. et RITTER, F. E. (2011). Supporting activity modelling from activity traces. *Expert Systems*. Wiley Online Library.

GILLET, B. (1987). *Le psychologue et l'ergonomie : traité de psychologie ergonomique*. Collection Psychologie et Pédagogie du Travail. EAP.

GRUDIN, J. (1990). The computer reaches out : the historical continuity of interface design. *In Proceedings of the SIGCHI conference on Human factors in computing systems : Empowering people*, CHI '90, pages 261–268, New York, NY, USA. ACM.

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

HAREL, D. (1987). Statecharts : a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274.

HAREL, D. et KUGLER, H. (2001). Synthesizing state-based object systems from lsc specifications. *In* YU, S. et PAUN, A., éditeurs : *Implementation and Application of Automata*, volume 2088 de *Lecture Notes in Computer Science*, pages 1–33. Springer Berlin / Heidelberg.

HAREL, D., KUGLER, H. et PNUELI, A. (2005). Synthesis revisited : Generating statechart models from scenario-based requirements. *In* KREOWSKI, H.-J., MONTANARI, U., OREJAS, F., ROZENBERG, G. et TAENTZER, G., éditeurs : *Formal Methods in Software and Systems Modeling*, volume 3393 de *Lecture Notes in Computer Science*, pages 309–324. Springer Berlin / Heidelberg.

HENNING, M. J., GEORGEON, O., WYNN, T. et KREMS, J. F. (2008). Modelling driver behaviour in order to infer the intention to change lanes. *In Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems*, pages 113–120, Lyon, France.

HERBST, J. et KARAGIANNIS, D. (2004). Workflow mining with inwolve. *Computers in Industry*, 53(3):245 – 264. Process / Workflow Mining.

HOLLNAGEL, E. (2011). *The Handbook of Human-Machine Interaction*, chapitre The Diminishing Relevance of Human-Machine Interaction, pages 417–429. Ashgate.

HÉTIER, M. (2008). *Analyse et Quantification des Comportements des Conducteurs Automobiles Lors de Phases de Pré-crash : Contribution au Développement d'un Modèle de Détection des Postures de Conduite en Temps Réel*. Thèse de doctorat, Université de Valenciennes et du Hainault-Cambrésis.

JENSEN, K. (1997). A brief introduction to coloured petri nets. *In* BRINKSMA, E., éditeur : *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1217 de *Lecture Notes in Computer Science*, pages 203–208. Springer Berlin / Heidelberg. 10.1007/BFb0035389.

JÜRGENSOHN, T. (2007). *Modelling Driver Behaviour in Automotive Environments*, chapitre Control Theory Models of the Driver, pages 277–292. Springer.

KINDLER, E., RUBIN, V. et SCHÄFER, W. (2006). Process mining and petri net synthesis. *In* EDER, J. et DUSTDAR, S., éditeurs : *Business Process Management Workshops*, volume 4103 de *Lecture Notes in Computer Science*, pages 105–116. Springer Berlin / Heidelberg.

LAFLAQUIÈRE, J. (2009). *Conception de système à base de traces numériques pour les environnements informatiques documentaires*. Thèse de doctorat en informatique, Université de Technologie de Troyes.

LAFLAQUIÈRE, J., PRIÉ, Y. et MILLE, A. (2008). Ingénierie des traces numériques d'interaction comme inscriptions de connaissances. *In Actes des 19es Journées Francophones d'Ingénierie des Connaissances (IC 2008)*.

LEONTIEV, A. N. (1984). *Activité, conscience, personnalité*. Editions du Progrès.

LEPLAT, J. (1990). Relations between task and activitiy : elements for elaborating a framework for error analysis. *Ergonomics*, 33(10/11):1389–1402.

LEPLAT, J. (2003). La modélisation en ergonomie à travers son histoire. *In Formalismes de modélisation pour l'analyse du travail et l'ergonomie*, Le travail humain, chapitre 1, pages 1–26. PUF.

LI, J., LIU, D. et YANG, B. (2007). Process mining : Extending α-algorithm to mine duplicate tasks in process logs. *In* CHANG, K., WANG, W., CHEN, L., ELLIS, C., HSU, C.-H., TSOI, A. et WANG, H., éditeurs : *Advances in Web and Network Technologies, and Information Management*, volume 4537 de *Lecture Notes in Computer Science*, pages 396–407. Springer Berlin / Heidelberg.

MATHERN, B. (2006). Analyser l'activité de conduite automobile : méthodologie et atelier logiciel associé. Rapport de stage de fin d'études ingénieur, École Centrale de Lyon.

MATHERN, B., BELLET, T. et MILLE, A. (2010a). An iterative approach to develop a cognitive model of the driver for human centred design of ITS. *In European Conference on Human Centred Design for Intelligent Transport Systems*, Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems, pages 85–95, Berlin, Germany. HUMANIST publications.

MATHERN, B., MILLE, A. et BELLET, T. (2010b). An interactive method to discover a petri net model of an activity. Rapport scientifique RR-LIRIS-2010-026, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon.

MATHERN, B., MILLE, A. et BELLET, T. (2011a). Rendre interactive la découverte d'automates à partir de traces d'activités. *In 7e Plateforme AFIA*, pages 689–704, Chambéry, France. Best paper of the conference IC2011.

MATHERN, B., MILLE, A., BELLET, T., SHLADOVER, S. E. et NOWAKOWSKI, C. (2011b). Adapting LESCOT data-analysis tools to PATH data format : A first step for further collaboration. Rapport technique, PATH, University of California, Berkeley.

MCCULLOCH, W. S. et PITTS, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5:115–133.

MCKNIGHT, A. J. et ADAMS, B. B. (1970). Driver education task analysis. volume i : Task descriptions. Rapport technique, Human Resources Research Organization International, Alexandria.

MICHON, J. A. (1985). *Human behavior and traffic safety*, chapitre A critical view of driver behavior models : What do we know, what should we do ?, pages 485–520. Plenum Press, New York.

MINSKY, M. (1975). *The Psychology of Computer Vision*, chapitre A framework for representing knowledge, pages 211–277. McGraw-Hill.

MINSKY, M. et PAPERT, S. (1969). *Perceptrons : An Introduction to Computational Geometry*. MIT Press.

MURATA, T. (1989). Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

NEISSER, U. (1976). *Cognition and Reality : principles and implications of cognitive psychology*. W.H.Freeman & Co Ltd.

NOWAKOWSKI, C., O'CONNELL, J., SHLADOVER, S. E. et CODY, D. (2010). Cooperative adaptive cruise control : Driver acceptance of following gap settings less than one second. *In Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 54, pages 2033–2037.

NÄÄTÄNEN, R. et SUMMALA, H. (1974). A model for the role of motivational factors in drivers' decision-making. *Accident Analysis & Prevention*, 6(3-4):243–261.

OLLAGNIER-BELDAME, M. (2006). *Traces d'interactions et processus cognitifs en activité conjointe : Le cas d'une co-rédaction médiée par un artefact numérique*. Thèse de doctorat, Université Lumière Lyon 2.

PETERSON, J. L. (1977). Petri nets. *ACM Comput. Surv.*, 9:223–252.

PETIT ROBERT (2010). Le nouveau Petit Robert : Dictionnaire alphabétique et analogique de la langue française.

PIAGET, J. (1936). *Naissance de l'intelligence chez l'enfant*. Delachaux & Niestlé.

POPPER, K. (1979). *La connaissance objective : Une approche évolutionniste*. Flamarion.

RASKIN, A. (2011). You are solving the wrong problem. last accessed on August 2011 http://www.azarask.in/blog/post/the-wrong-problem/.

RASMUSSEN, J. (1986). *Information Processing and Human-Machine Interaction : An Approach to Cognitive Engineering*. Elsevier Science Inc.

ROSCH, E. (1975). Cognitive reference points. *Cognitive Psychology*, 7(4):532–547.

ROZINAT, A. et van der AALST, W. M. P. (2006). Conformance testing : Measuring the fit and appropriateness of event logs and process models. *In* BUSSLER, C. et HALLER, A., éditeurs : *Business Process Management Workshops*, volume 3812 de *Lecture Notes in Computer Science*, pages 163–176. Springer Berlin / Heidelberg.

RUSSEL, S. et PETER, N. (2006). *Intelligence Artificielle*. Pearson Education, deuxième édition.

SCHNEIDER, W. et SHIFFRIN, R. M. (1977). Controlled and automatic human information processing : I. detection, search, and attention. *Psychological Review*, 84(1):1–66.

SETTOUTI, L. S. (2011). *Systèmes à Base de Traces Modélisées : Modèles et Langages pour l'exploitation des traces d'Interactions*. Thèse de doctorat, Université Claude Bernard Lyon 1.

SETTOUTI, L. S., PRIÉ, Y., CHAMPIN, P.-A., MARTY, J.-C. et MILLE, A. (2009). A trace-based systems framework : Models, languages and semantics. Rapport de recherche, Université Claude Bernard Lyon 1. Work in progress.

SHLADOVER, S. E., LU, X.-Y., CODY, D., NOWAKOWSKI, C., QIU, Z. T., CHOW, A., O'CONNELL, J., NIENHUIS, J. et SU, D. (2010a). Development and evaluation of selected mobility applications for vii. California path research report, California PATH Program, Institute of Transportation Studies, University of California, Berkeley.

SHLADOVER, S. E., NOWAKOWSKI, C., O'CONNELL, J. et CODY, D. (2010b). Cooperative adaptive cruise control : Driver selection of car-following gaps. *In 17th ITS World Congress, Busan, 2010 : Proceedings*.

SPERANDIO, J.-C. (2003). Modèles et formalismes, ou le fond et la forme. *In Formalismes de modélisation pour l'analyse du travail et l'ergonomie*, Le travail humain, chapitre 2, pages 27–75. PUF.

STANTON, N. A., SALMON, P. M., WALKER, G. H. et JENKINS, D. (2009). Genotype and phenotype schemata as models of situation awareness in dynamic command and control teams. *International Journal of Industrial Ergonomics*, 39:480–489.

THOMAS, J. J. et COOK, K. A., éditeurs (2005). *Illuminating the Path : The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center.

van der AALST, W. M. P. (1998a). The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66.

van der AALST, W. M. P. (1998b). *Information and Process Integration in Enterprises : Rethinking Documents*, volume 428 de *The Kluwer International Series in Engineering and Computer Science*, chapitre Chapter 10 : Three Good reasons for Using a Petri-net-based Workflow Management System, pages 161–182. Kluwer Academic Publishers.

van der AALST, W. M. P. (2004). Business process management demystified : A tutorial on models, systems and standards for workflow management. *In* DESEL, J., REISIG, W. et ROZENBERG, G., éditeurs : *Lectures on Concurrency and Petri Nets*,

volume 3098 de *Lecture Notes in Computer Science*, pages 21–58. Springer Berlin / Heidelberg.

van der AALST, W. M. P., de MEDEIROS, A. K. A. et WEIJTERS, A. J. M. M. (2005). Genetic process mining. *In* CIARDO, G. et DARONDEAU, P., éditeurs : *Applications and Theory of Petri Nets 2005*, volume 3536 de *Lecture Notes in Computer Science*, pages 985–985. Springer Berlin / Heidelberg.

van der AALST, W. M. P., de MEDEIROS, A. K. A. et WEIJTERS, A. J. M. M. (2006). Process equivalence : Comparing two process models based on observed behavior. *In* DUSTDAR, S., FIADEIRO, J. et SHETH, A., éditeurs : *Business Process Management*, volume 4102 de *Lecture Notes in Computer Science*, pages 129–144. Springer Berlin / Heidelberg.

van der AALST, W. M. P. et GÜNTHER, C. W. (2007). Finding structure in unstructured processes : The case for process mining. *In Application of Concurrency to System Design, 2007. ACSD 2007. Seventh International Conference on*, pages 3–12. IEEE.

van der AALST, W. M. P. et van DONGEN, B. F. (2002). Discovering workflow performance models from timed logs. *In* HAN, Y., TAI, S. et WIKARSKI, D., éditeurs : *Engineering and Deployment of Cooperative Information Systems*, volume 2480 de *Lecture Notes in Computer Science*, pages 107–110. Springer Berlin / Heidelberg.

van der AALST, W. M. P., van DONGEN, B. F., HERBST, J., MĂRUŞTER, L., SCHIMM, G. et WEIJTERS, A. J. M. M. (2003). Workflow mining : A survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267.

van der AALST, W. M. P. et WEIJTERS, A. J. M. M. (2004). Process mining : a research agenda. *Computers in Industry*, 53(3):231–244. Process / Workflow Mining.

van der AALST, W. M. P., WEIJTERS, A. J. M. M. et MĂRUŞTER, L. (2002). Workflow mining : Which processes can be rediscovered ? Rapport technique, Eindhoven University of Technology.

van der AALST, W. M. P., WEIJTERS, A. J. M. M. et MĂRUŞTER, L. (2004). Workflow mining : Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1128–1142.

van der MOLEN, H. H. et BÖTTICHER, A. M. T. (1988). A hierarchical risk model for traffic participants. *Ergonomics*, 31(4):537–555.

van der WERF, J., van DONGEN, B. F., HURKENS, C. et SEREBRENIK, A. (2008). Process discovery using integer linear programming. *In* van HEE, K. et VALK, R., éditeurs : *Applications and Theory of Petri Nets*, volume 5062 de *Lecture Notes in Computer Science*, pages 368–387. Springer Berlin / Heidelberg.

van DONGEN, B. F., de MEDEIROS, A. K. A. et WEN, L. (2009). Process mining : Overview and outlook of petri net discovery algorithms. *In* JENSEN, K. et van der AALST, W. M. P., éditeurs : *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 de *Lecture Notes in Computer Science*, pages 225–242. Springer Berlin / Heidelberg.

VERMERSCH, P. (1994). *L'entretien d'explicitation : en formation initiale et en formation continue*. ESF éditeur.

WEIJTERS, A. J. M. M. et van der AALST, W. M. P. (2003). Rediscovering workflow models from event-based data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162.

WEIJTERS, A. J. M. M., van der AALST, W. M. P. et de MEDEIROS, A. K. A. (2006). Process mining with the HeuristicsMiner algorithm. BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven.

WEN, L., van der AALST, W. M. P., WANG, J. et SUN, J. (2007a). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15:145–180.

WEN, L., WANG, J. et SUN, J. (2007b). Mining invisible tasks from event logs. *In* DONG, G., LIN, X., WANG, W., YANG, Y. et YU, J., éditeurs : *Advances in Data and Web Management*, volume 4505 de *Lecture Notes in Computer Science*, pages 358–365. Springer Berlin / Heidelberg.

WEN, L., WANG, J., van der AALST, W. M. P., HUANG, B. et SUN, J. (2009). A novel approach for process mining based on event types. *Journal of Intelligent Information Systems*, 32:163–190.

WEN, L., WANG, J., van der AALST, W. M. P., WANG, Z. et SUN, J. (2004). A novel approach for process mining based on event types. Rapport technique, Eindhoven University of Technology.

WILDE, G. J. S. (1982). Critical issues in risk homeostasis theory. *Risk Analysis*, 2(4):249–258.

Publications

Revues internationales

- CRAM, D., MATHERN, B. et MILLE, A. (2011). A complete chronicle discovery approach : application to activity analysis. *Expert Systems*. Wiley Online Library.
- GEORGEON, O., MILLE, A., BELLET, T., MATHERN, B. et RITTER, F. E. (2011). Supporting activity modelling from activity traces. *Expert Systems*. Wiley Online Library.

Conférences internationales avec comité de relecture

- MATHERN, B., BELLET, T. et MILLE, A. (2010a). An iterative approach to develop a cognitive model of the driver for human centred design of ITS. *In European Conference on Human Centred Design for Intelligent Transport Systems*, pages 85–95, Berlin, Germany. HUMANIST publications.
- BELLET, T., MAYENOBE, P., BORNARD, J.-C., GRUYER, D. et MATHERN, B. (2010). COSMO-SIVIC : a first step towards a virtual platform for human centred design of driving assistances. *In The 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Valenciennes, France.

Conférences nationales avec comité de relecture

- MATHERN, B., MILLE, A. et BELLET, T. (2011a). Rendre interactive la découverte d'automates à partir de traces d'activités. *In 7e Plateforme AFIA*, pages 689–704, Chambéry, France. Meilleur papier de la conférence IC2011.

Rapports

- MATHERN, B., MILLE, A., BELLET, T., SHLADOVER, S. E. et NOWAKOWSKI, C. (2011b). Adapting LESCOT data-analysis tools to PATH data format : A first step for further collaboration. Rapport technique, PATH, University of California, Berkeley.
- MATHERN, B., MILLE, A. et BELLET, T. (2010b). An interactive method to discover a petri net model of an activity. Rapport scientifique, LIRIS, Université de Lyon.
- MATHERN, B. (2006). Analyser l'activité de conduite automobile : méthodologie et atelier logiciel associé. Rapport de stage de fin d'études ingénieur, École Centrale de Lyon.

Index des auteurs

Annexes

ANNEXE A

Définitions

A.1 Notations

$[\![i;j]\!]$	Soit deux entiers i et j, $[\![i;j]\!] = \{i, i+1, \ldots, j-1, j\}$ décrit l'ensemble des entiers de i à j inclus.
$t \in \sigma$	Soit σ une trace d'un WF-log. Soit t un type d'événement. La notation $t \in \sigma$ signifie : la trace σ contient un événement de type t. Autrement dit, $\sigma = t_1 ... t_n \wedge \exists i \in [\![1;n]\!]$ tq $t_i = t$.
$A \setminus B$	Soit A et B deux ensembles. $A \setminus B$ est l'ensemble « A privé de B » : $A \setminus B = \{x \in A \mid x \notin B\}$.
$=$	Le signe $=$ désigne une égalité.
$\equiv$	Le signe $\equiv$ désigne une équivalence.
$:=$	Le signe $:=$ désigne une affectation de variable (au sens informatique) $a := b$ signifie que l'on affecte la valeur de la variable b à la variable a.
$\bullet x$	Voir définition 18 p. 220.
$x\bullet$	Voir définition 18 p. 220.

A.2 PT-net

Les définitions suivantes (à l'exception de celle de « multiensemble ») sont reprises et traduites de l'article de van der Aalst *et al.* (2004).

Définition 16 (multiensemble) *Un* multiensemble M *(aussi appelé* sac*) de support* T *est une fonction qui à chaque élément* t *de* T *associe un entier naturel* $M(t)$.
On utilise les crochets pour énumérer un multiensemble, par exemple le multiensemble $[a^2, b, c^3]$ *de support* $\{a, b, c, d\}$ *représente le multiensemble contenant deux* a*, un* b*, trois* c *et zéro fois* d*. Les opérations de somme (*$X+Y$*), de soustraction (*$X-Y$*), la présence d'un élément dans un multiensemble (*$x \in X$*) et la notion de sous-multiensemble (*$X \leq Y$*) sont définies de manière directe.*

Intuitivement, un multiensemble est un ensemble pouvant contenir plusieurs fois le même élément. Le nombre $M(t)$ représente le nombre d'éléments t présents dans le multiensemble M.

Définition 17 (PT-net) *Un réseau Place/Transition, ou plus simplement un PT-net, est un triplet* (P, T, F) *tel quel :*

1. *P est un ensemble fini de* places*,*
2. *T est un ensemble fini de* transitions*, disjoint de l'ensemble de places ($P \cap T = \emptyset$), et*
3. *$F \subseteq (P \times T) \cup (T \times P)$ est un ensemble d'arcs orientés.*

Un PT-net marqué *est un couple (N, s), tel que $N = (P, T, F)$ est un PT-net et s est un multiensemble de support P appelé le* marquage *du réseau. L'ensemble de tous les PT-net marqués est noté $\mathcal{N}$.*

Définition 18 (Nœud, nœud d'entrée, nœud de sortie) *Soit $N = (P, T, F)$ un PT-net. On appelle* noeud *tout élément de $P \cup T$. Un nœud est donc soit une place, soit une transition.*
On dit qu'un nœud x est un noeud d'entrée *d'un autre nœud y si et seulement si il existe un arc orienté entre x et y (c.-à-d. $(x, y) \in F$). Un nœud x est un* nœud de sortie *du nœud y si et seulement si $(y, x) \in F$. Par extension, on parlera de places d'entrée (ou de sortie) d'une transition et de transitions d'entrée (ou de sortie) d'une place.*
On note ${}^{N}\bullet x$ l'ensemble des nœuds d'entrée de x : ${}^{N}\bullet x = \{y | (y, x) \in F\}$. On note $x^{N}\bullet$ l'ensemble des nœuds de sortie de x : $x^{N}\bullet = \{y | (x, y) \in F\}$. Dans la suite du document, le symbole N sera omis quand le contexte permettra de déduire de quel PT-net il est question.

Définition 19 (Règle de franchissement) *Soit $(N = (P, T, F), s)$ un WF-net marqué. Une transition $t \in T$ est dite* franchissable*, noté $(N, s)[t\rangle$, si et seulement si $\bullet t \leq s$. La* règle de franchissement *$\_[\_\rangle\_ \subseteq \mathcal{N} \times T \times \mathcal{N}$ est la plus petite relation satisfaisant pour tout $(N = (P, T, F), s) \in \mathcal{N}$ et pour tout $t \in T$, $(N, s)[t\rangle(N, s - \bullet t + t\bullet)$.*

Définition 20 (Marquage accessible) *Soit (N, s_0) un PT-net marqué de $\mathcal{N}$. Un marquage s est* accessible *depuis le marquage initial s_0 si et seulement si il existe une séquence de transitions franchissables dont le franchissement conduit du marquage s_0 au marquage s. L'ensemble des marquages accessible de (N, s_0) est notée $[N, s_0\rangle$.*

Définition 21 (Séquence) *Soit A un alphabet. Une* séquence de longueur n*, pour un entier naturel non nul n, et de support l'alphabet A est une fonction $\sigma : [\![1; n]\!] \to A$. Une séquence de longueur nulle est appelée séquence vide et s'écrit ε. Pour faciliter l'écriture, on écrira $\sigma = t_1 t_2 \dots t_n$ la séquence de longueur n telle que pour tout $i \in [\![1; n]\!], t_i \in T$ et $\sigma(i) = t_i$. L'ensemble de toutes les séquences de longueur arbitraires d'un alphabet A est noté A^*.*

Définition 22 (Séquence de franchissement) *Soit $(N = (P, T, F), s_0)$ un PT-net marqué. Une séquence $\sigma \in T^*$ est appelée* séquence de franchissement *de (N, s_0) si et seulement si il existe un entier naturel $n \in \mathbb{N}$, des marquages $s_1, \dots, s_n$ et des transitions $t_1, \dots, t_n \in T$ tels que $\sigma = t_1 \dots t_n$ et, pour chaque i tel que $i \in [\![0; n-1]\!]$, $(N, s_i)[t_{i+1}\rangle$ et $s_{i+1} = s_i - \bullet t_{i+1} + t_{i+1}\bullet$. (Notons que $n = 0$ implique que $\sigma = \varepsilon$ et que ε est une séquence de franchissement de (N, s_0).) La séquence σ est dite* franchissable *à partir du marquage s_0, ce qui est noté $(N, s_0)[\sigma\rangle$. Le franchissement de la séquence σ produit le marquage s_n, ce qui est noté $(N, s_0)[\sigma\rangle(N, s_n)$.*

Définition 23 (Connexité) *Un réseau $N = (P, T, F)$ est* faiblement connecté *si et seulement si, pour chaque paire de nœuds x et y de $P \cup T$, on a $x(F \cup F^{-1})^* y$, où R^{-1} est la relation inverse de R et R^* est la fermeture réflexive et transitive de la relation R. Le réseau N est* fortement connecté *si et seulement si pour toute paire de nœuds x et y on a xF^*y.*

Définition 24 (Réseau borné, réseau sain) *Un réseau marqué $(N = (P, T, F), s)$ est* borné *si et seulement si l'ensemble des marquages accessibles $[N, s\rangle$ est fini. Un réseau est* sain *si et seulement si, pour tout marquage $s' \in [N, s\rangle$ et pour toute place $p \in P$, $s'(p) \leq 1$. Notons qu'un réseau sain est borné.*

Définition 25 (Transitions mortes, réseau vivant) *Soit $(N = (P, T, F), s)$ un PT-net marqué. Une transition $t \in T$ est* morte *dans (N, s) si et seulement si il n'y a pas de marquage accessible $s' \in [N, s\rangle$ tel que $(N, s')[t\rangle$. Le réseau (N, s) est* vivant *si et seulement si pour chaque marquage accessible $s' \in [N, s\rangle$ et pour tout $t \in T$, il y a un marquage accessible $s'' \in [N, s'\rangle$ telle que $(N, s'')[t\rangle$. Notons qu'un réseau vivant ne contient pas de transitions mortes.*

A.3 WF-net et WF-log

Les définitions suivantes sont reprises et traduites de l'article de van der Aalst *et al.* (2004).

Définition 26 (Workflow nets) *Soient $N = (P, T, F)$ un PT-net et $\bar{t}$ un nouvel identifiant non inclut dans $P \cup T$. N est un worflow net (WF-net) si et seulement si :*

1. *initialisation : P contient une place d'entrée i telle que $\bullet i = \emptyset$,*
2. *terminaison : P contient une place de sortie o telle que $o\bullet = \emptyset$, et*
3. *connexité : $\bar{N} = (P, T \cup \{\bar{t}\}, F \cup \{(o, \bar{t}), (\bar{t}, i)\})$ est fortement connecté.*

Définition 27 (Conforme) *Soit $N = (P, T, F)$ un WF-net, soient i sa place d'entrée et o sa place de sortie. N est* conforme *si et seulement s'il possède les propriétés suivantes :*[1]

1. *sain : $(N, [i])$ est sain,*
2. *terminaison propre : pour tout marquage $s \in [N, [i]\rangle$, $o \in s$ implique $s = [o]$;*
3. *terminaison accessible : pour tout marquage $s \in [N, [i]\rangle$, $[o] \in [N, s\rangle$, et*
4. *absence de transitions mortes : $(N, [i])$ ne contient pas de transition morte.*

L'ensemble de tous les WF-net conformes est noté $\mathcal{W}$.

Définition 28 (Trace de workflow, log de workflow) *Soit T un ensemble de types de transition. $\sigma \in T^*$ est une* trace de workflow *et $W \in \mathcal{P}(T^*)$ est un* log de workflow *(ou WF-log).*[2]

[1] Le mot « conforme » est la traduction que nous proposons pour *sound*. La propriété de conformité vise à vérifier que le réseau est un modèle acceptable ou correct du processus analysé du point de vue d'un analyste.

[2] Pour un ensemble A, $\mathcal{P}(A)$ désigne l'ensemble de tous les ensembles inclus dans A : $\mathcal{P}(A) = \{B | B \subseteq A\}$. Autrement dit $W \in \mathcal{P}(T^*)$ signifie $W \subseteq T^*$.

Définition 29 (Relations d'ordre basées sur les logs) *Soit W un WF-log de support T, c'est-à-dire $W \in \mathcal{P}(T^*)$. Soient $a, b \in T$, on a :*

- $a >_W b$ *si et seulement s'il existe une trace $\sigma = t_1 \dots t_n$ et un entier $i \in [\![1; n-1]\!]$ tels que $\sigma \in W$ et $t_i = a$ et $t_{i+1} = b$,*
- $a \rightarrow_W b$ *si et seulement si $a >_W b$ et $b \not>_W a$,*
- $a \#_W b$ *si et seulement si $a \not>_W b$ et $b \not>_W a$, et*
- $a ||_W b$ *si et seulement si $a >_W b$ et $b >_W a$.*

Définition 30 ($\in, first, last$) *Soient un ensemble A, un élément $a \in A$ et une séquence non vide $\sigma = a_1 \dots a_n \in A^*$ de support A et de longueur n ($n \geq 1$). On définit $\in$, $first$ et $last$ de la manière suivante :*

1. *$a \in \sigma$ si et seulement si $a \in \{a_1, \dots, a_n\}$,*
2. *$first(\sigma) = a_1$, et*
3. *$last(\sigma) = a_n$.*

Définition 31 (WF-log complet) *Soit $N = (P, T, F)$ un WF-net conforme, c'est-à-dire $N \in \mathcal{W}$. W est un* WF-log *de N si et seulement si $W \in \mathcal{P}(T^*)$ et pour chaque trace $\sigma \in W$ il existe une séquence de franchissement de N commençant avec le marquage $[i]$ et finissant avec le marquage $[o]$, autrement dit, $(N, [i])[\sigma\rangle(N, [o])$. W est un WF-log* complet *de N si et seulement si 1) pour tout WF-log W' de N : $>_{W'} \subseteq >_W$, et 2) pour tout $t \in T$ il y a une trace $\sigma \in W$ telle que $t \in \sigma$.*

Définition 32 (Capacité à redécouvrir) *Soit $N = (P, T, F)$ un WF-net conforme, c'est-à-dire $N \in \mathcal{W}$ et soit α un algorithme qui associe un WF-log de N à un WF-net conforme, c'est-à-dire $\alpha : \mathcal{P}(T^* \rightarrow \mathcal{W})$. Si pour tout WF-log complet W de N l'algorithme renvoie le WF-net N (au renommage des places près), on dit que α est capable de* redécouvrir *N.*

Définition 33 (Place implicite) *Soit $N = (P, T, F)$ un PT-net avec le marquage initial s. Une place $p \in P$ est dite* implicite *dans (N,s) si et seulement si pour tout marquage accessible $s' \in [N, s\rangle$ et pour toute transition $t \in p\bullet$, on a $s' \geq \bullet t \setminus \{p\} \Rightarrow s' \geq \bullet t$.*

Définition 34 (WF-net structuré) *Un WF-net $N = (P, T, F)$ est un WF-net* structuré *si et seulement si :*

1. *Pour tout $p \in P$ et $t \in T$ tels que $(p, t) \in F$: $|p \bullet| > 1$ implique $|\bullet t| = 1$.*
2. *Pour tout $p \in P$ et $t \in T$ tels que $(p, t) \in F$: $|\bullet t| > 1$ implique $|\bullet p| = 1$.*
3. *N ne contient pas de places implicites.*

ANNEXE B

Démonstrations

Théorème 2 (Équivalence entre relation $\triangle_W$ et relation $\diamond_W$) *Soit W un WF-log complet (au sens de l'algorithme α^+, de Medeiros* et al.*, 2004a.) d'un WF-net structuré et conforme sans boucles de longueur un. On a :*

$$a \triangle b \Rightarrow b \triangle a$$

Corollaire 1 *Soit W un WF-log complet (au sens de l'algorithme α^+, de Medeiros* et al.*, 2004a.) d'un WF-net structuré et conforme sans boucles de longueur un. On a :*

$$a \triangle b \Leftrightarrow b \triangle a \Leftrightarrow a \diamond b$$

Une fois le théorème 2 démontré, le corollaire 1 se déduit de la définition de la relation losange (p. 94) : $a \diamond b \Leftrightarrow (a \triangle b$ et $b \triangle a)$.

Preuve. Pour démontrer le théorème 2, nous nous appuyons sur le théorème suivant proposé par de Medeiros *et al.* (2004a, théorème 3.4 p. 11), dont nous reprenons la définition ci-dessous.

Théorème 3 *Soit $N = (P, T, F)$ un WF-net structuré et conforme sans boucles de longueur un. Soit W un WF-log complet (au sens de l'algorithme α^+) de N. Pour tout $a, b \in T$, tels que $\bullet a \cap b\bullet \neq \emptyset$ et $a \bullet \cap \bullet b \neq \emptyset$, $a \triangle b$ implique $b \triangle a$*

À l'aide de ce théorème, Il nous suffit de montrer que $a \triangle b$ implique ($\bullet a \cap b\bullet \neq \emptyset$ et $a \bullet \cap \bullet b \neq \emptyset$), pour prouver que $a \triangle b$ implique $b \triangle a$.

Soit $a, b \in T$ deux transitions du WF-net N telles que $a \triangle b$. $a \triangle b$ signifie qu'il existe une trace $\sigma \in W$ telle que $\sigma = \ldots aba \ldots$. Comme le réseau N ne contient pas de boucles de longueur un, la transition a n'est pas impliquée dans une boucle de longueur un. Donc $\bullet a \nsubseteq a\bullet$. Or, comme N est WF-net, la propriété de connexité nous assure que $a\bullet \neq \emptyset$. Donc il existe une place p_1 telle que $p_1 \in \bullet a$ et $p_1 \notin a\bullet$.

Montrons que $p_1 \in \bullet a \cap b\bullet$ Notons σ_1 et σ_2 les séquences d'événements telles que $\sigma = \sigma_1 aba \sigma_2$. Le franchissement de la séquence σ_1 depuis le marquage initial marque la place p_1, car a est franchissable. Autrement dit $(N, [i])[\sigma_1\rangle$ marque p_1. Comme $p_1 \notin a\bullet$, la place p_1 n'est plus marquée après le franchissement de la transition a (séquence $\sigma_1 a$). Comme la séquence $\sigma_1 ab$ rend la transition a à nouveau franchissable, on en déduit que le franchissement de b marque p_1. Donc $p_1 \in b\bullet$. Donc $p_1 \in \bullet a \cap b\bullet$. Donc $\bullet a \cap b\bullet \neq \emptyset$

Maintenant, nous allons montrer par l'absurde qu'il existe $p_2 \in a\bullet \cap \bullet b \neq \emptyset$. Nous avons montré que $a\bullet \neq \emptyset$ (propriété de connexité). S'il n'existe pas de place p_2 telle que $p_2 \in a\bullet \cap \bullet b$, alors quelque soit $p \in a\bullet$, le franchissement de la séquence $\sigma_1 a$ marque p. Comme $p \notin \bullet b$, la place p est toujours marquée après le franchissement de la séquence $\sigma_1 ab$, qui rend pourtant a franchissable. Le franchissement de a (séquence $\sigma_1 aba$) ajoute donc un deuxième jeton à la place p, ce qui est en contradiction avec la propriété de conformité du réseau N (N est sain). Donc il existe $p_2 \in a\bullet \cap \bullet b$. Donc $a\bullet \cap \bullet b \neq \emptyset$.

Nous avons donc montré que $a \triangle b \Rightarrow (\bullet a \cap b\bullet \neq \emptyset$ et $a\bullet \cap \bullet b \neq \emptyset)$. Donc, avec le théorème 3, nous en déduisons que $a \triangle b \Rightarrow b \triangle a$.

Théorème 4 (Équivalence entre l'algorithme α_i^+ et l'algorithme α^+)
L'algorithme α_i^+ est équivalent à l'algorithme α^+. C'est-à-dire que pour tout WF-log complet W d'un WF-net structuré et conforme, $\alpha_i^+(W) = \alpha^+(W)$.

Preuve. Pour montrer l'équivalence entre l'algorithme α^+ et l'algorithme α_i^+, nous reproduisons ici la définition de l'algorithme α^+. Cette définition est issue des publications de de Medeiros *et al.* (2004a, 2005). Nous en proposons une traduction en français qui explicite les étapes de calcul des relations $>_W$, $\triangle_W$, $\rightarrow_W$ et $\#_W$ (à l'étape 4 et 8c'). Cette écriture prend en compte le théorème 2 que nous venons de démontrer (à l'étape 8c'). En outre, cette version corrige deux erreurs présentes dans la définition originale.[1]

Définition 35 (algorithme α^+)

1. $T_{log} = \{t \in T | \ \exists_{\sigma \in W}[t \in \sigma]\}$,
2. $L1L = \{t \in T_{log} | \exists \sigma = t_1 t_2 ... t_n \in W, i \in [\![1; n-1]\!] : t = t_i \wedge t = t_{i+1}\}$
3. $T' = T_{log} \setminus L1L$
4. $F_{L1L} = \emptyset$

4.5 $\forall a, b \in T(W), a >_W b \Leftrightarrow \exists \sigma = t_1 \dots t_n \in W, i \in [\![1; n-1]\!] : t_i = a$ *et* $t_{i+1} = b$,

5. *Pour chaque $t \in L1L$ calculer :*
   a) $A = \{a \in T' | a >_W t\}$
   b) $B = \{b \in T' | t >_W b\}$
   c) $F_{L1L} := F_{L1L} \cup \{(t, p(A \setminus B, B \setminus A)), (p(A \setminus B, B \setminus A), t)\}$
6. $W^{-L1L} = \emptyset$
7. *Pour chaque $\sigma \in W$ calculer :*
   a) $\sigma' = \sigma$
   b) *Pour chaque $t \in L1L$ calculer :*
      i. $\sigma' := eliminateTask(\sigma', t)$
   c) $W^{-L1L} := W^{-L1L} \cup \sigma'$

[1] À l'étape 2, l'indice sur les i, pour le calcul de $L1L$.
À l'étape 5, le nom de la transition (« a » au lieu de « b ») pour le calcul de l'ensemble B, utilisé ultérieurement pour calculer F_{L1L}.

8. *a)* $T(W^{-L1L}) = \{t | \exists \sigma \in W^{-L1L},\ t \in \sigma\}$,

   b) $T_I(W^{-L1L}) = \{t \mid \exists \sigma \in W^{-L1L},\ t = \mathit{first}(\sigma)\}$,

   c) $T_O(W^{-L1L}) = \{t \mid \exists \sigma \in W^{-L1L},\ t = \mathit{last}(\sigma)\}$,

   (c') *i.* $\forall a, b \in T(W^{-L1L}), a >_{W^{-L1L}} b \Leftrightarrow \exists \sigma \in W^{-L1L} : \sigma = t_1 \dots t_n, i \in [\![1; n-1]\!], t_i = a$ *et* $t_{i+1} = b$,

   ii. $\forall a, b \in T(W^{-L1L}), a \triangle_{W^{-L1L}} b \Leftrightarrow \exists \sigma \in W^{-L1L} : \sigma = t_1 \dots t_n, i \in [\![1; n-2]\!], t_i = t_{i+2} = a$ *et* $t_{i+1} = b$,

   iii. $\forall a, b \in T(W^{-L1L}), a \rightarrow_{W^{-L1L}} b \Leftrightarrow (a >_{W^{-L1L}} b \wedge b \not>_{W^{-L1L}} a) \vee a \triangle_{W^{-L1L}} b \vee b \triangle_{W^{-L1L}} a$,

   iv. $\forall a, b \in T(W^{-L1L}), a \#_{W^{-L1L}} b \Leftrightarrow a \not>_{W^{-L1L}} b$ *et* $b \not>_{W^{-L1L}} a$,

   d) $X(W^{-L1L}) = \{(A, B) \mid A \subseteq T(W^{-L1L}) \wedge B \subseteq T(W^{-L1L}) \wedge \forall (a, b) \in A \times B : a \rightarrow_{W^{-L1L}} b \wedge \forall a_1, a_2 \in A : a_1 \#_{W^{-L1L}} a_2 \wedge \forall b_1, b_2 \in B : b_1 \#_{W^{-L1L}} b_2\}$,

   e) $Y(W^{-L1L}) = \{(A, B) \in X(W^{-L1L}) \mid \forall (A', B') \in X(W^{-L1L}) : A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$,

   f) $P(W^{-L1L}) = \{p(A, B) \mid (A, B) \in Y(W^{-L1L})\} \cup \{i_w, o_w\}$,

   g) $F(W^{-L1L}) = \{(a, p(A, B)) \mid (A, B) \in Y(W^{-L1L}) \wedge a \in A\} \cup \{(p(A, B), b) \mid (A, B) \in Y(W^{-L1L}) \wedge b \in B\} \cup \{(i_w, t) \mid t \in T_I(W^{-L1L})\} \cup \{(t, o_w) \mid t \in T_O(W^{-L1L})\}$,

   h) $\alpha'(W^{-L1L}) = (P(W^{-L1L}), T(W^{-L1L}), F(W^{-L1L}))$.

   $(P_{W^{-L1L}}, T_{W^{-L1L}}, F_{W^{-L1L}}) := \alpha'(W^{-L1L})$

9. $P_W = P_{W^{-L1L}}$

10. $T_W = T_{W^{-L1L}} \cup L1L$

11. $F_W = F_{W^{-L1L}} \cup F_{L1L}$

12. $\alpha^+ = (P_W, T_W, F_W)$

Nous définissons l'algorithme α_i^+ en nous appuyant sur les fonctions *analyseLogs()*, *buildDependencyRelation()*, *buildPlaces()* et *postprocess_L1L()* définies au chapitre 5. La figure B.2 montre comment il est possible de reformuler l'algorithme α^+ en utilisant ces fonctions. L'algorithme α_i^+ (figure B.3) est une simplification de l'écriture de l'algorithme α^+ qui permet en outre d'économiser le calcul du WF-log W^{-L1L} (noté W' sur la figure B.2).

Les définitions formelles de l'algorithme α_i^+ et des fonctions *analyseLogs()*, *buildDependencyRelation()*, *buildPlaces()* et *postprocess_L1L()* sont données au chapitre 5.

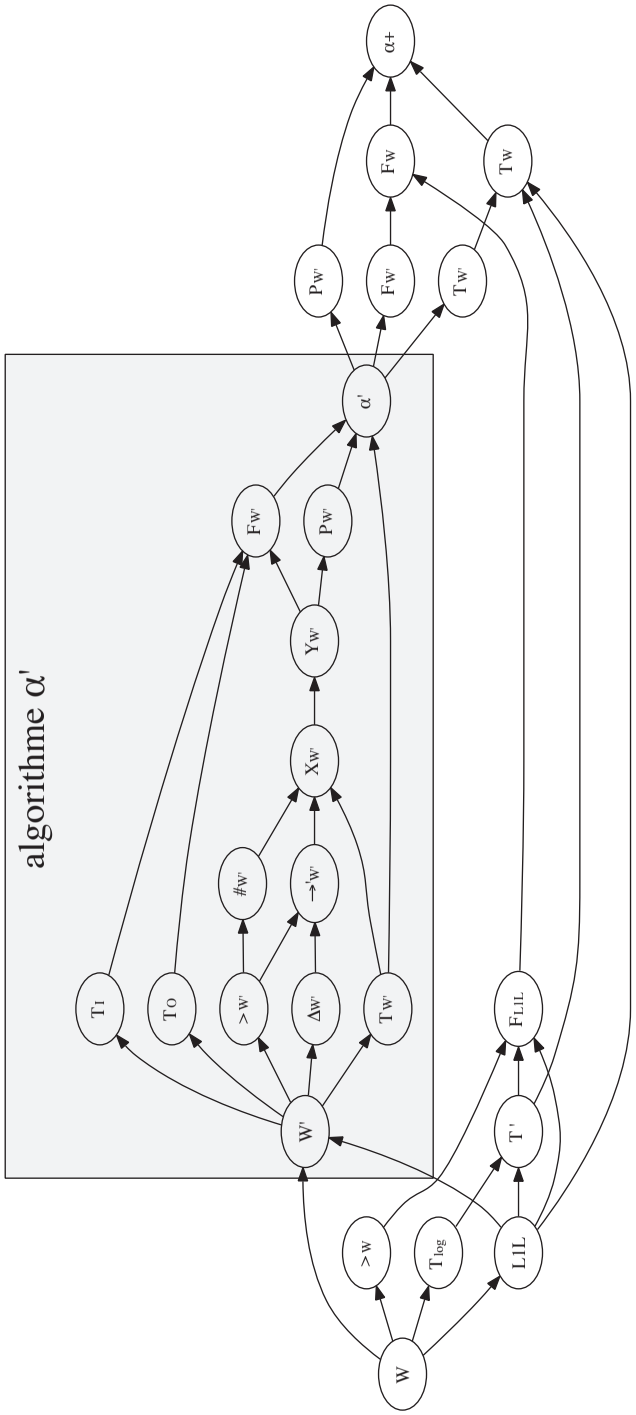


FIGURE B.1 – Enchaînement des étapes de l'algorithme α^+. Ce schéma reprend les étapes de l'algorithme α^+ (en explicitant l'algorithme α') et montre les interdépendances entre ces étapes.

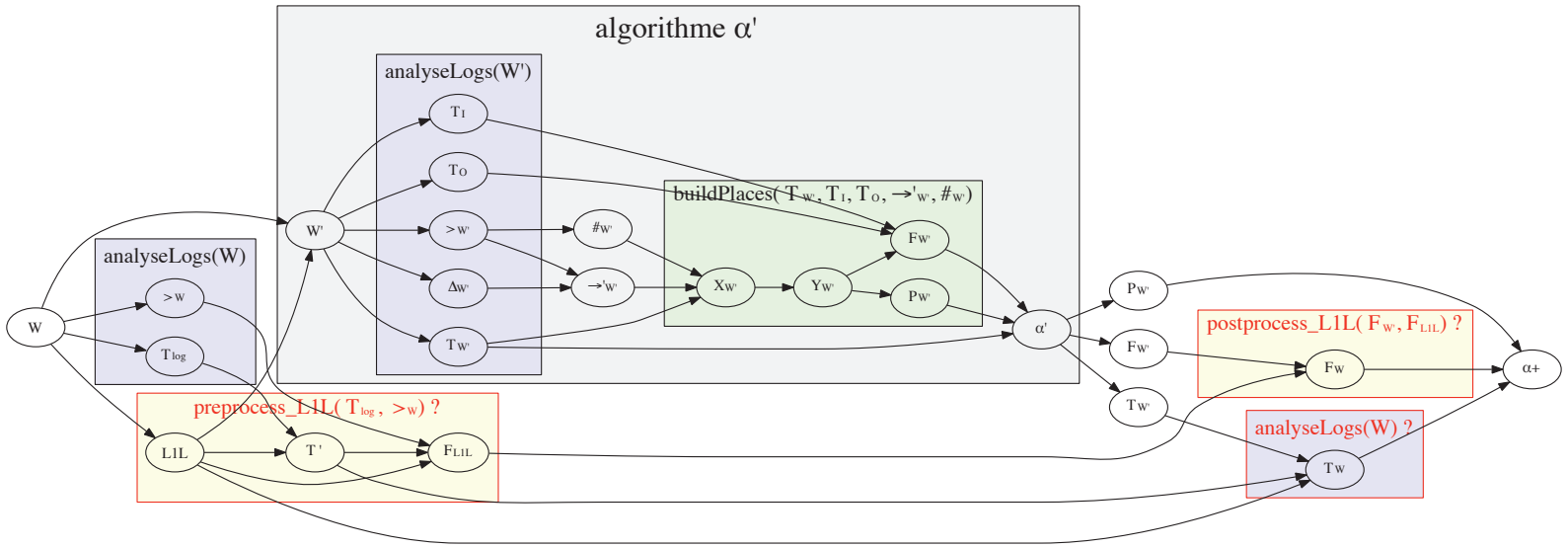


FIGURE B.2 – Enchaînement des étapes de l'algorithme α^+. Ce schéma reprend les étapes de l'algorithme α^+ en suggérant des regroupements possibles utilisant le même principe de reformulation que pour l'algorithme α. Les groupes notés en rouges sont des intuitions à démontrer ultérieurement.

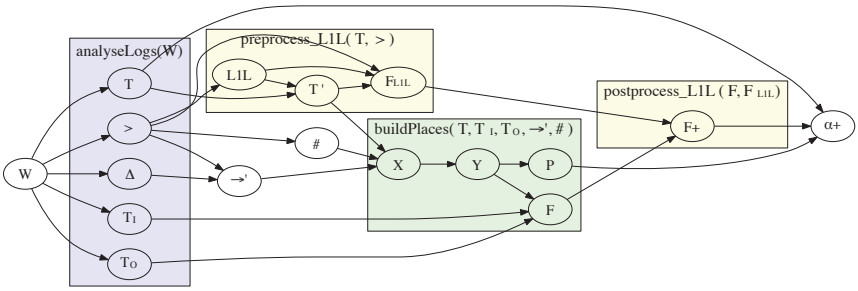


FIGURE B.3 – Enchaînement des étapes de l'algorithme α_i^+. Ce schéma reprend les étapes de l'algorithme α_i^+ en détaillant les sous-étapes contenues dans chaque fonction.

Pour démontrer l'équivalence entre l'algorithme α_i^+ et l'algorithme α^+ sur un WF-log complet d'un WF-net complet et conforme, nous devons vérifier l'équivalence des formules suivantes[2].

$$L1L \equiv \{t \in T_{log} | t >_W t\} \tag{B.1}$$

$$T' \equiv T_{W^{-L1L}} \tag{B.2}$$

$$T_{log} \equiv T_W \tag{B.3}$$

$$T_I(W) \equiv T_I(W^{-L1L}) \tag{B.4}$$

$$T_O(W) \equiv T_O(W^{-L1L}) \tag{B.5}$$

$$\forall a, b \in T', (a >_W b \equiv a >_{W^{-L1L}} b) \tag{B.6}$$

$$\forall a, b \in T', (a \triangle_W b \equiv a \triangle_{W^{-L1L}} b) \tag{B.7}$$

La propriété B.3 montre la redondance du calcul de T_W. Les propriétés B.2, B.4, B.5, B.6 et B.7 nous montrent qu'il n'est pas nécessaire de calculer le WF-log W^{-L1L}, ni de calculer deux fois la relation de succession $>$. Enfin, la propriété B.1 permet d'exprimer autrement l'ensemble $L1L$ et ainsi exploiter la relation $>$ comme résultat intermédiaire pour inférer la présence de boucles de longueur un.

Preuve de B.1 : $L1L \equiv \{t \in T_{log} | t >_W t\}$

Par définition, $L1L = \{t \in T_{log} | \exists \sigma = t_1 t_2 ... t_n \in W, i \in [\![1; n-1]\!] : t = t_i \wedge t = t_{i+1}\}$, ce qui est équivalent à $L1L = \{t \in T_{log} | t >_W t\}$, par définition de la relation $>_W$.

Preuve de B.2 : $T' \equiv T_{W^{-L1L}}$

Si $t \in T'$, alors $t \in T_{log} \wedge t \notin L1L$. $t \in T_{log}$ signifie qu'il existe une trace $\sigma \in W$ telle que $t \in \sigma$. À l'étape 7 de l'algorithme α^+, la transition t n'est jamais supprimée de σ car $t \notin L1L$. Donc il existe une trace $\sigma' \in W^{-L1L}$ telle que $t \in \sigma$. Donc $T' \subseteq T_{W^{-L1L}}$.

[2]Nous utilisons ici les notations de l'algorithme α^+.

Si $t \in T_{W^{-L1L}}$ alors il existe une trace $\sigma' \in W^{-L1L}$ telle que $t \in \sigma'$. Or, par construction des traces σ' de W^{-L1L}, cela signifie qu'il existe une trace $\sigma \in W$ à partir de laquelle σ' a été dérivée et $t \in \sigma$. Donc $T_{W^{-L1L}} \subseteq T'$.
Donc $T' = T_{W^{-L1L}}$.

Preuve de B.3 : $T_{log} \equiv T_W$

Par définition $T_W = T_{W^{-L1L}} \cup L1L$, or $T' = T_{W^{-L1L}}$, $T' = T_{log} \setminus L1L$, et par construction $L1L \subseteq T_{log}$, donc $T_{log} = T_W$.

Preuve de B.4 : $T_I(W^{-L1L}) \equiv \{t \mid \exists \sigma \in W,\ t = \mathit{first}(\sigma)\}$

Montrons que pour un WF-log W d'un WF-net conforme, $T_I(W) = T_I(W^{-L1L})$.

Soit t_i une transition de $T_I(W)$. Elle apparaît en premier dans une des traces donc elle est franchissable à partir du marquage initial : $\bullet t_i \subseteq \{i_w\}$. Avec la propriété de connexité d'un WF-log, $\bullet t_i = \{i_w\}$. Or, par définition des WF-net, $\bullet i_w = \emptyset$, donc t_i ne peut être franchie qu'une seule fois. Donc $t_i \notin L1L$, t_i n'est pas impliquée dans une boucle de longueur un.

La construction du WF-log W^{-L1L} ne fait que supprimer des transitions de $L1L$, donc pour toute trace $\sigma \in W$ et pour sa trace dérivée $\sigma' \in W^{-L1L}$, on a $\mathit{first}(\sigma) = \mathit{first}(\sigma')$. Donc $T_I(W) = T_I(W^{-L1L})$

Preuve de B.5 : $T_O(W) \equiv T_O(W^{-L1L})$

Par un raisonnement similaire à la démonstration de B.4, montrons que pour un WF-log W d'un WF-net conforme, $T_O(W) = T_O(W^{-L1L})$.

Soit t_o une transition de $T_O(W)$. Elle apparaît en dernier dans une des traces donc elle marque la place de sortie et seulement la place de sortie (propriété de terminaison propre d'un WF-net conforme) : $t_o\bullet = \{o_w\}$. Or, par définition des WF-net, $o_w\bullet = \emptyset$, donc t_o ne peut être franchie qu'une seule fois. Donc $t_o \notin L1L$, t_o n'est pas impliquée dans une boucle de longueur un.

La construction du WF-log W^{-L1L} ne fait que supprimer des transitions de $L1L$, donc pour toute trace $\sigma \in W$ et pour sa trace dérivée $\sigma' \in W^{-L1L}$, on a $\mathit{last}(\sigma) = \mathit{last}(\sigma')$. Donc $T_O(W) = T_O(W^{-L1L})$

Preuve de B.6 : $\forall a, b \in T', (a >_W b \equiv a >_{W^{-L1L}} b)$

Pour un WF-log complet W d'un WF-net N, montrons que $\forall a, b \in T', a >_W b = a >_{W^{-L1L}} b$.

Pour toute transition $t \in L1L$, il existe une trace $\sigma = t_1 ... t_n$ et $i \in [\![1; n-1]\!]$, tels que $t_i = t_{i+1} = t$. On a $\bullet t \subseteq t\bullet$, car le franchissement de t_i rend t_{i+1} franchissable et que le réseau N est 1-borné (les places de $\bullet t$ ne peuvent contenir plus d'un jeton). De la même manière, $t\bullet \subseteq \bullet t$, car le réseau N est 1-borné et que les places de $t\bullet$ ne peuvent contenir plus d'un jeton. Donc $\bullet t = t\bullet$. Donc le franchissement de la transition $t \in L1L$ ne modifie par le marquage du réseau : la transition t est facultative. Donc si on enlève les transitions $L1L$ des traces de W, le WF-log obtenu est toujours un WF-log de N. Donc le WF-log W^{-L1L} est un WF-log de N.

Par définition de la complétude, pour tout WF-log W' de N, $>_W \subseteq >_{W'}$. Donc $>_{W^{-L1L}} \subseteq >_W$. Donc pour toutes transitions a et b de T', $a >_{W^{-L1L}} b \Rightarrow a >_W b$.

Inversement si on a a et b deux transitions de T' telles que $a >_W b$, il existe une trace $\sigma = t_1 ... t_n \in W$ telle que $\exists i \in [\![1; n-1]\!] : t_i = a$ et $t_{i+1} = b$. Comme

a et b n'appartiennent pas à $L1L$, la trace dérivée $\sigma' = t'_1...t'_{n'} \in W^{-L1L}$ telle que $\exists j \in [\![1; n'-1]\!] : t'_j = a$ et $t'_{j+1} = b$. Donc $a >_{W^{-L1L}} b$.

Donc $\forall a, b \in T', a >_W b = a >_{W^{-L1L}} b$.

Preuve de B.7 : $\forall a, b \in T', (a \triangle_W b \equiv a \triangle_{W^{-L1L}} b)$

Par un raisonnement similaire à la démonstration de B.6, montrons que pour un WF-log complet W d'un WF-net N, $\forall a, b \in T', a \triangle_W b = a \triangle_{W^{-L1L}} b$.

Nous avons déjà montré dans la démonstration de B.6 que W^{-L1L} est un WF-log de N. Par la définition de complétude, pour tout WF-log W' de N, $\triangle_W \subseteq \triangle_{W'}$. Donc $\triangle_{W^{-L1L}} \subseteq \triangle_W$. Donc pour toutes transitions a et b de T', $a \triangle_{W^{-L1L}} b \Rightarrow a \triangle_W b$.

Inversement si on a a et b deux transitions de T' telles que $a \triangle_W b$, il existe une trace $\sigma = t_1...t_n \in W$ telle que $\exists i \in [\![1; n-2]\!] : t_i = t_{i+2} = a$ et $t_{i+1} = b$. Comme a et b n'appartiennent pas à $L1L$, la trace dérivée $\sigma' = t'_1...t'_{n'} \in W^{-L1L}$ telle que $\exists j \in [\![1; n'-2]\!] : t'_j = t_{j+2} = a$ et $t'_{j+1} = b$. Donc $a \triangle_{W^{-L1L}} b$.

Donc $\forall a, b \in T', a \triangle_W b = a \triangle_{W^{-L1L}} b$.

Définition 36 (algorithme α_i^+) *En détaillant toutes les fonctions, l'algorithme α_i^+ s'écrit de la manière suivante :*

- $(T, T_I, T_O, >, \triangle) := analyseLogs(W)$:

1. $T = \{t | \ \exists \sigma \in W,\ t \in \sigma\}$,
2. $T_I = \{t \mid \exists \sigma \in W,\ t = first(\sigma)\}$,
3. $T_O = \{t \mid \exists \sigma \in W,\ t = last(\sigma)\}$,
4. $\forall a, b \in T, a > b \Leftrightarrow \exists \sigma = t_1 \dots t_n \in W, i \in [\![1; n-1]\!] : t_i = a$ *et* $t_{i+1} = b$,
5. $\forall a, b \in T, a \triangle b \Leftrightarrow \exists \sigma = t_1 \dots t_n \in W, i \in [\![1; n-2]\!] : t_i = t_{i+2} = a$ *et* $t_{i+1} = b$,

- $(T', F_{L1L}) := preprocess\_L1L(T, >)$:

6. $L1L = \{t \in T | t > t\}$,
7. $T' = T \setminus L1L$
8. $F_{L1L} = \emptyset$
9. *Pour chaque* $t \in L1L$ *faire :*
   a) $A = \{a \in T' | a > t\}$
   b) $B = \{b \in T' | t > b\}$
   c) $F_{L1L} := F_{L1L} \cup \{(t, p(A \setminus B, B \setminus A)), (p(A \setminus B, B \setminus A), t)\}$

- $\rightarrow' := buildDependencyRelation(T, >, \triangle)$:

10. $\forall a, b \in T, a \rightarrow b \Leftrightarrow (a > b \wedge b \not> a) \vee a \triangle b \vee b \triangle a$,

- $\# := buildUnrelatedRelation(T, >)$:

11. $\forall a, b \in T, a \# b \Leftrightarrow a \not> b$ *et* $b \not> a$,

- $(P, F) := buildPlaces(T', T_I, T_O, \rightarrow', \#)$:

12. $X = \{(A, B) \mid A \subseteq T' \ \wedge \ B \subseteq T' \ \wedge \ \forall (a, b) \in A \times B : a \rightarrow b \ \wedge \ \forall a_1, a_2 \in A : a_1 \# a_2 \ \wedge \ \forall b_1, b_2 \in B : b_1 \# b_2\}$,
13. $Y = \{(A, B) \in X \mid \forall (A', B') \in X : A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$,
14. $P = \{p(A, B) \mid (A, B) \in Y\} \cup \{i_w, o_w\}$,
15. $F = \{(a, p(A, B)) \mid (A, B) \in Y \wedge a \in A\} \cup \{(p(A, B), b) \mid (A, B) \in Y' \wedge b \in B\} \cup \{(i_w, t) \mid t \in T_I\} \cup \{(t, o_w) \mid t \in T_O\}$,

- $F^+ := postprocess\_L1L(F, F_{L1L})$:

16. $F^+ = F \cup F_{L1L}$

- $\alpha_i^+ = (P, T, F^+)$.