Compte-rendu intermédiaire

Projet ANR-18-CE40-0011

PING/ACK

A IDENTIFICATION

Acronyme du projet	PING/ACK
Titre du projet	*Preprocessing Information for Nontrivial Goals / Advanced Compilation of Knowledge*
Coordinateur du projet (société/organisme)	Pierre Marquis CRIL, Univ Artois & CNRS
Date de début du projet Date de fin du projet	20 mars 2019 20 septembre 2023 (compte-tenu des 6 mois de prolongation attribués à cause de la pandémie)
Labels et correspondants des pôles de compétitivité (pôle, nom et courriel du corresp.)	sans objet
Site web du projet, le cas échéant	http://www.cril.univ-artois.fr/pingack/

Rédacteur de ce rapport	
Civilité, prénom, nom	Pierre Marquis
Téléphone	03 21 79 17 86
Courriel	marquis@cril.univ-artois.fr
Date de rédaction	mars 2021
Période faisant l'objet du rapport d'activité	mars 2019 – mars 2021

B LIVRABLES ET JALONS

Au delà des notes rédigées au fur et à mesure de l'avancement du projet et servant de supports aux interactions entre partenaires (cf. C.2) mais aussi des publications déjà obtenues (cf. D1), deux rapports ont été produits ; il s'agit de synthèses de travaux existants sur deux thèmes, au cœur de PING/ACK :

- *KC maps of preference languages*
- *Parameterized algorithms and lower bounds for classical problems and well-established languages*

C RAPPORT D'AVANCEMENT

C.1 OBJECTIFS INITIAUX DU PROJET

Concevoir des algorithmes assurant des temps de réponse rapides est un problème fondamental en informatique. Son importance est cruciale lorsque ces algorithmes sont utilisés dans des applications nécessitant des interactions fréquentes avec les humains, comme on en trouve souvent dans la vie quotidienne (par exemple, lors de l'utilisation d'applications sur le Web ou

via un smartphone). En outre, dans de nombreuses applications, une grande partie des informations données au début ou échangées avec l'utilisateur se présentent sous la forme de «connaissances» sur la base desquelles des raisonnements ou des processus décisionnels s'appuient. Malheureusement, les tâches basées sur les connaissances sont typiquement NP-difficiles, ce qui implique que, dans le cas général, elles souffrent de limitations de calcul intrinsèques ne permettant pas de garantir des temps de réponse acceptables (sauf si P = NP).

La compilation de connaissances est une approche permettant de contourner ces limitations en pré-traitant, lors d'une phase hors ligne, une partie des connaissances disponibles (la partie F dite "fixe"). Deux problématiques principales ont été étudiées à ce sujet au cours des vingt dernières années. D'une part, développer un modèle de compilabilité permettant de décider si une tâche d'intérêt est compilable ou non (c'est-à-dire réalisable efficacement quand une représentation compilée de F de taille polynomiale a été calculée au préalable). D'autre part, établir des cartes de compilation de connaissances pour choisir un langage de représentation L dans lequel F devrait être compilée. Il s'agit d'un problème de décision multicritères, le choix de L dépendant à la fois de son efficacité spatiale (c'est-à-dire de sa capacité relative à coder des informations en utilisant peu d'espace mémoire) et de son efficacité temporelle (c'est-à-dire de la complexité temporelle des sous-tâches élémentaires du problème étudié). Cependant, le modèle actuel de compilabilité présente l'inconvénient majeur de classer comme non-compilables de nombreuses tâches pour lesquelles la compilation des connaissances apparaît comme une approche intéressante en pratique. Un tel écart entre la théorie et la pratique vient du fait que le cadre de compilabilité se concentre sur le cas le plus défavorable et n'est pas suffisamment précis pour prendre en compte des caractéristiques spécifiques des entrées. De plus, l'expressivité des langages L qui ont été utilisés jusqu'à présent dans les cartes de compilation de connaissances est encore trop limitée ou insuffisante pour certaines applications nécessitant des constructions plus sophistiquées.

L'objectif de PING / ACK est de s'attaquer aux manques des travaux existants en matière de compilation de connaissances, en élargissant son champ d'application, à la fois du côté théorique et du côté pratique. Nous prévoyons de définir de nouvelles cartes de compilation de connaissances adaptées à des langages de représentation plus expressifs que ceux existants et d'améliorer l'applicabilité de la compilation de connaissances. Notre objectif est aussi de préciser le concept de compilabilité, en élaborant des cartes plus fines (qui ne seraient pas centrées sur les scénarios les plus défavorables) et en explorant d'autres approches permettant d'améliorer les avantages offerts par la compilation des connaissances. Sont attendus des résultats de caractérisation de différents types (expressivité, concision, complexité), mais aussi de nouveaux concepts et langages, de nouveaux algorithmes (compilateurs et raisonneurs), qui seront mis en œuvre et évalués expérimentalement sur des problèmes variés (recommandation, configuration robotique et, dans une perspective plus risquée, bioinformatique).

C.2 TRAVAUX EFFECTUES ET RESULTATS ATTEINTS SUR LA PERIODE CONCERNEE

Le projet a démarré le 20 mars 2019, le kick-off meeting s'est tenu à Lens et a réuni l'ensemble des partenaires.

Dans le cadre de son WP5, un site web dédié au projet a été créé au lancement du projet (http://www.cril.univ-artois.fr/pingack/), ainsi qu'une liste de diffusion regroupant les participants. Un système de notes de travail, accessibles en interne, a été mis en place pour faciliter les échanges à distance entre les partenaires. Plusieurs notes (8) ont ainsi été rédigées, portant sur des aspects variés du projet.

Le recrutement du premier doctorant financé par le projet, Alexis de Colnet, impliqué dans les

WPs 2 et 3, a pu être opéré en temps voulu, au 1er septembre 2019. Le recrutement du second doctorant, Sergej Scheck, a été un peu plus problématique (Sergej a été recruté au 6 janvier 2020).

Des résultats intéressants, en phase avec nos attentes, ont déjà pu être obtenus en dépit des problèmes rencontrés (cf. C3). Les publications internationales réalisées ont été publiées dans des supports très reconnus.

C.3 DIFFICULTES RENCONTREES ET SOLUTIONS

Le projet a éprouvé plusieurs difficultés, d'ordre budgétaire et surtout organisationnel.

Le budget finalement alloué au projet est, en effet, moindre que celui initialement demandé : le financement prévu pour recruter le post-doctorant demandé n'a pas été alloué, conduisant à une baisse de budget de l'ordre de 13% et une implication (en personne.mois) plus réduite que prévu lorsque le projet a été bâti. Pour cette raison, il faudra vraisemblablement supprimer des tâches du WP 4, auquel le post-doctorant était censé participer.

Par ailleurs, lors du montage du projet, un financement de thèse avait été demandé et obtenu, à hauteur de 94320 € pour trois ans. Un doctorant a pu commencer au 1er septembre 2019 au CRIL en bénéficiant de ce support. Or, le CRIL est soumis à une délégation globale de gestion au profit du CNRS, et à compter du 1er septembre, une hausse significative (+20%) des salaires des doctorants CNRS a été décidée : la masse salariale a monté à 117000 €, soit un surcoût de 22680 € par rapport à ce qui était prévu initialement. Il nous a donc fallu ponctionner la part restante du budget reçu par le CRIL pour couvrir ce surcoût : cette part étant de 50500 €, il restait 50500 - 22680 = 27820 € pour fonctionner (soit seulement 55% de ce qui avait été demandé au départ). L'ANR a été prévenue immédiatement (avril 2019) du problème. Depuis lors, une enveloppe exceptionnelle de 10 000 € dédiée au projet a pu être dégagée par le CNRS. Néanmoins, la part manquante obligera le partenaire CRIL à limiter les réunions de travail.

Une autre difficulté rencontrée est liée au recrutement du second doctorant, Sergej Scheck, financé par le projet et impliqué dans les WPs 1 et 4. Ce recrutement a été plus long que prévu initialement. La pression du marché est aujourd'hui très importante chez les ingénieurs / masters possédant des compétences fortes en IA et de ce fait, le recrutement de candidats de très bon niveau est devenu difficile. Cette difficulté est maintenant derrière nous.

Une troisième difficulté, plus récente, mais avec un impact bien plus grand sur le déroulement du projet, tient évidemment à la crise liée à la pandémie COVID-19. Au delà du surplus d'activités engendrées hors projet par la gestion à mettre en place (plan de continuité des activités, tant en enseignement qu'en recherche), les mesures sanitaires actées par la puissance publique (du confinement à respecter jusqu'à l'interdiction de se réunir à plus de 6 personnes dans les locaux de recherche, qui s'applique aujourd'hui) ne nous ont pas permis de nous réunir comme attendu pour avancer dans le développement du projet et travailler conjointement sur la rédaction des livrables. La réunion prévue à Bordeaux en mars 2020 pour discuter avec les partenaires impliqués dans la RoboCup (qui a été reportée) n'a pas pu avoir lieu. La seconde vague de la pandémie, la difficulté de se déplacer et la contrainte imposée par certains établissements gestionnaires interdisant la tenue de séminaires dans leurs locaux, ne sont pas propices à l'organisation de réunions en présentiel, qui est pourtant indispensable pour collaborer efficacement. Pour pallier cette difficulté, et à titre de pis-aller, nous avons mis en place en mars 2021 un système de réunions virtuelles en attendant de pouvoir nous retrouver pour travailler par groupes devant un tableau comme nous en avons l'habitude.

C.4 FAITS ET RESULTATS MARQUANTS

Un aspect marquant de la production scientifique réalisée jusqu'ici est la variété des sujets abordés mais aussi de la nature de la production, qui va d'articles décrivant des résultats à visée théorique jusqu'à des développements logiciels (conception et évaluation de programmes). Cette variété était espérée et est bien au rendez-vous. Elle est décrite plus en détail dans les annexes à ce document (section E).

C.5 TRAVAUX SPECIFIQUES AUX ENTREPRISES (LE CAS ECHEANT)

Sans objet pour ce projet.

C.6 REUNIONS DU CONSORTIUM (PROJETS COLLABORATIFS)

Date	Lieu	Partenaires présents	Thème de la réunion
20/03/2019	Lens	CRIL, GREYC, IRIT	Rappel du projet et de ses WPs, discussion sur l'organisation des activités
02/07/2019	Toulouse	CRIL, GREYC, IRIT	Point sur les activités réalisées
18/12/2019	Arras	Tous	Point sur les activités réalisées

Pour ce qui est des réunions en mode présentiel, on notera aussi une visite de travail de 3 jours d'Alexis de Colnet au GREYC en septembre 2019.

C.7 COMMENTAIRES LIBRES

Commentaires du coordinateur

Le projet se déroule du mieux possible, mais clairement pas dans des conditions idéales, compte tenu des difficultés – pour ne pas écrire la quasi-impossibilité - qu'il y a à se réunir physiquement depuis mars 2020 à cause de la situation sanitaire, mais aussi des grèves des transports publics qui n'ont pas facilité les déplacements durant la période automne 2019 – hiver 2020.

Des résultats intéressants ont été obtenus, mais plutôt dans le cadre de collaborations intra-laboratoires qu'inter-laboratoires. Plusieurs publications dans des revues internationales ou conférences internationales de rang A* sont issues des travaux réalisés. D'autres articles et communications sont actuellement en cours d'évaluation.

A cause des difficultés liées au contexte, les publications réalisées jusqu'ici n'ont impliqué qu'un seul partenaire et notre objectif est de faire croître les publications intra-laboratoires lors de la seconde moitié du projet (de mars 2021 à septembre 2023). La vaccination, quand elle concernera assez de monde, devrait nous permettre de retrouver un mode de fonctionnement normal, plus confortable et plus efficace.

Le renfort d'un doctorant supplémentaire à l'IRIT (sans coût supplémentaire pour le projet) devrait permettre d'avancer plus vite sur les aspects liés à la compilation de connaissances temporelles. Enfin, du côté des applications (WP4), une inflexion des activités vers l'utilisation de la compilation de connaissances pour l'IA explicable (sujet très porteur aujourd'hui et étudié par plusieurs participants à PING/ACK) pourra être envisagée si les difficultés posées par les autres applications prévues ne sont pas surmontées.

Commentaires des autres partenaires

Pas de commentaires spécifiques.

Question(s) posée(s) à l'ANR

Pas de questions particulières.

D VALORISATION ET IMPACT DU PROJET DEPUIS LE DEBUT

D.1 PUBLICATIONS ET COMMUNICATIONS

Voici les articles et communications déjà acceptés (plusieurs autres publications sont actuellement en cours de soumission à des revues et conférences). Les revues et conférences internationales du meilleur rang (A* ou A selon les classement ERA et CORE : http://portal.core.edu.au/) sont visées. On gardera en tête qu'en IA, la pratique de publication est de diffuser ses résultats plutôt dans des actes de conférence que dans des revues.

Liste des publications		
International	**Revues à comité de lecture**	1. J.-M. Lagniez, E. Lonca, and P. Marquis. Definability for Model Counting Artificial Intelligence 281: 103229 (2020). 2. B. Zanuttini, J. Lang, A. Saffidine, and F. Schwarzentruber. Knowledge-based programs as succinct policies for partially observable domains. Artificial Intelligence 288: 103365 (2020).
	Communications (conférence)	1. E. Grandjean and Th. Grente. Descriptive complexity for minimal time of cellular automata. Proc. of LICS'19, 1-13. 2. A. de Colnet, and S. Mengel Lower Bounds for Approximate Knowledge Compilation Proc. of IJCAI-PRICAI'20, 1834-1840. 3. A. de Colnet A lower bound on DNNF encodings of pseudo-Boolean constraints Proc. of SAT'20, 312-321. 4. J. Clément and A. Genitrini. Binary Decision Diagrams: from Tree Compaction to Sampling. Proc. of LATIN'20, 571-583. 5. H. Fargier, and J. Mengin An knowledge compilation map for conditional preference statements-based languages Proc. of AAMAS'21, à paraître. 6. F. Capelli, J.-M. Lagniez, and P. Marquis Certifying Top-Down Decision-DNNF Compilers Proc. of AAAI'21, à paraître. 7. S. Scheck, A. Niveau and B. Zanuttini Knowledge Compilation for Nondeterministic Action Languages Proc. of ICAPS'21, à paraître.
National	**Communications (conférence)**	1. S. Scheck, A. Niveau, and B. Zanuttini Knowledge Compilation for Action Languages Actes des 15es Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA'20). 2. H. Fargier, and J. Mengin An incomplete knowledge compilation map for conditional preference statements-based languages Actes des 14es Journées d'Intelligence Artificielle Fondamentale (JIAF'20).

D.2 AUTRES ELEMENTS DE VALORISATION

Rien à signaler.

D.3 POLES DE COMPETITIVITE (PROJET LABELLISES)

Sans objet pour ce projet.

D.4 PERSONNELS RECRUTES EN CDD (HORS STAGIAIRES)

Avant le recrutement sur le projet				Recrutement sur le projet				
Nom et prénom	Sexe H/F	Adresse email (1)	Dernier diplôme obtenu	Lieu d'études (France, UE, hors UE)	Expérience prof. antérieure (ans)	Partenaire ayant embauché la personne	Poste dans le projet (2)	Date de recrutement
DE COLNET Alexis	H	decolnet@cril.fr	Diplôme d'ingénieur + master	France et Singapour	-	CRIL	doctorant	01/09/19
SCHECK Sergej	H	sergej.scheck@unicaen.fr	Master	Allemagne	-	GREYC	doctorant	06/01/20

D.5 ÉTAT FINANCIER

Nom du partenaire	Crédits consommés (en %)	Commentaire éventuel
CRIL	15,17% (fonctionnement et missions) 56,53% (personnel)	Nous avons eu très peu de frais de mission jusqu'ici dans la mesure où deux des réunions qui se sont tenues (sur les trois) ont eu lieu à Lens ou à côté (Arras), et que la réunion à Toulouse s'est tenue à l'occasion de la plate-forme AFIA à laquelle plusieurs d'entre nous participions et bénéficions d'autres soutiens financiers. Compte-tenu des soucis de budget éprouvés et décrits en section C.3, ces « économies » en début de projet nous permettront de disposer des fonds nécessaires pour nous réunir le plus longtemps possible ensuite.
GREYC	12,23% (fonctionnement et missions) 39,81% (investissement) 33,62% (personnel)	
IRIT	21% (fonctionnement et missions)	
LaBRI	-	Pas d'information disponible à ce jour

E ANNEXES

Nous présentons dans la suite de façon synthétique les problématiques abordées et les résultats obtenus. Dans un souci de clarté, l'organisation choisie pour cette présentation est par WP (pour les WPs 1 à 4 -- le WP5, lié à la coordination et la dissémination, n'appelant pas de description scientifique), même si les choix d'assignation qui ont été faits sont parfois arbitraires et artificiels, les frontières entre WPs étant (fort heureusement !) perméables.

E.1 WP1

Carte de compilation pour les préférences. Les préférences conditionnelles ont été utilisées pour représenter de manière compacte des préférences sur des domaines combinatoires. Elles sont au cœur des CP-nets et de leurs généralisations, mais aussi des arbres de préférences lexicographiques. Plusieurs travaux ont abordé la complexité de certaines requêtes (optimisation, dominance notamment). Nous nous sommes attachés à étendre certains de ces résultats, et étudier d'autres requêtes et certaines transformations qui ont pas été abordées jusqu'à présent, comme l'équivalence, les coupures, les marginalisations, contribuant ainsi à une carte de compilation pour les langages de représentation des préférences à base de préférence conditionnelles.

Un article décrivant les résultats déjà obtenus a été rédigé. Il a été accepté aux Journées d'Intelligence Artificielle Fondamentale (JIAF'20). Un autre article a été accepté récemment à la conférence internationale de rang A* AAMAS'21.

Carte de compilation pour les langages d'action. Un travail sur l'efficacité temporelle et spatiale des requêtes et transformations liées à la représentation d'actions a été entrepris. Les logiciels ne planification utilisent souvent deux sortes de représentation pour les actions : l'une, souvent basée sur le langage PDDL, est utilisée pour spécifier les problèmes en entrée — c'est un langage relativement intuitif à manipuler pour les humains. Cependant, avant de travailler, le planificateur va généralement transformer les actions en une représentation interne plus efficace et plus pratique à manipuler. La façon dont les actions sont représentées est donc d'une grande importance pour les planificateurs, et disposer d'une carte de compilation comparant l'efficacité spatiale et temporelle des différents langages existants serait très intéressant.

Le travail mené s'est intéressé à plusieurs langages pour la représentation d'actions non-déterministes (le cas déterministe présentant peu de défis) : PDDL, dans une version non-déterministe et propositionnelle, une version non-déterministe de STRIPS conditionnel, les théories d'actions en NNF, et DL-PPA. Nous avons montré un paysage très riche en termes de concision relative des langages et de leur capacité à traiter les requêtes au cœur de la planification : déterminer si un état est successeur d'un autre, si une action est applicable dans un état, si une succession d'action assure la satisfaction d'une formule logique. Les résultats ainsi obtenus ont été présentés aux journées francophones JFPDA 2020, et sont acceptés à la conférence internationale de rang A* ICAPS'21.

Dans un autre travail, nous nous sommes intéressés aux langages d'actions du point de vue de leur capacité à représenter efficacement la persistance des variables non affectées par l'action (problème du cadre). Nous proposons d'augmenter le langage des théories d'actions en NNF par un opérateur syntaxique imposant la persistance de certaines variables, et étudions deux versions de cet opérateur : une version « syntaxique » considérant comme affectées les variables apparaissant explicitement dans une formule, et une version « sémantique », inspirée du raisonnement par circonscription, imposant des changements minimaux sur ces variables. L'originalité première de cette proposition est que, contrairement au raisonnement par circonscription, par exemple, l'opérateur peut apparaître à tout niveau d'imbrication dans le formule. Nous montrons ainsi que la version sémantique permet de gagner en concision au sens de la carte de compilation, tandis que la version syntaxique peut être éliminée en temps polynomial. L'adjonction de ce dernier fournit

donc un langage plus riche pour spécifier des actions, sans nécessiter le développement d'algorithmes de planification dédiés. Nous étudions également la complexité de répondre aux requêtes naturelles de la planification pour les langages ainsi obtenus. Ces résultats sont soumis à la conférence internationale de rang A* IJCAI'21.

Compilation pour la représentation de politiques d'actions pour la planification contingente. On s'intéresse à des domaines dans lesquels un agent a à sa disposition un ensemble d'actions pour réaliser un but, mais où l'observabilité est partielle : l'agent ne connaît pas exactement l'état de l'environnement à un instant donné, il a seulement des croyances sur cet état, résultant de croyances initiales et d'observations (imparfaites) apportées par ces actions. Il s'agit alors de calculer des politiques, qui sont des objets donnant l'action optimale à effectuer dans tout état de croyance atteignable ou, de façon équivalente, pour toute séquence d'observations effectuées. De telles politiques contiennent donc de nombreux branchements, et leur représentation croît typiquement exponentiellement avec la profondeur (ou l'horizon) de la politique. C'est pourquoi nous avons étudié la représentation de politiques par des programmes à base de connaissances, dont nous avons montré qu'ils fournissent une représentation possiblement exponentiellement plus succincte que toute représentation standard, la contrepartie étant une difficulté algorithmique (modérément) plus importante pour retrouver les actions à effectuer au moment d'exécuter la politique. Ces résultats ont fait l'objet d'un article paru dans la revue internationale de rang A* *Artificial Intelligence*.

Contraintes pseudo-booléennes et DNNF. Les contraintes pseudo-booléennes (PB) sont des inégalités ou équations linéaires sur les littéraux de variables booléennes régulièrement utilisées pour représenter des problèmes en IA. Pour déterminer une affectation de variables satisfaisant un ensemble de contraintes PB, une méthode consiste à coder ces contraintes en une formule CNF agissant sur un sur-ensemble des variables, laquelle est ensuite confiée à un solveur SAT. Pour que la CNF soit traitée efficacement par le solveur, celle-ci doit être de taille raisonnable et avoir certaines garanties vis-à-vis des algorithmes implémentés par le solveur, notamment ceux basés sur la propagation unitaire. Afin d'obtenir des garanties supérieures à celles des codages actuels, une approche consiste à compiler individuellement les contraintes PB dans des formes de type BDD ou DNNF, puis à générer des CNF depuis les formes compilées par des transformations de types Tseitin. L'instance SAT est la conjonction des CNF ainsi obtenues.

Nous avons mis en évidence les limites de cette approche quant à la taille de l'instance SAT générée. La CNF pour une contrainte PB est au moins aussi large que la forme compilée intermédiaire, et il a déjà été montré l'existence de contraintes PB dont les représentations par des OBBD sont de taille au moins exponentielle vis-à-vis du nombre de variables. En réponse à cette borne inférieure, d'aucun pourrait envisager une compilation vers des DNNF, lesquelles sont strictement plus succinctes que les OBDD. Mais nous avons montré que la borne s'étend aux DNNF : nous avons prouvé l'existence de contraintes PB dont les représentations DNNF sont de taille exponentielle. Ce résultat renforce les études sur la séparation entre les systèmes de contraintes PB et les formes compilées usuelles. Ces études montrent notamment qu'il existe des systèmes de contraintes dont les représentations DNNF sont de taille plus que polynomiale, nous avons donc généralisé ce résultat pour des contraintes individuelles.

Une publication décrivant ce travail est parue dans les actes de la conférence internationale de rang A SAT'20.

Sum Product Networks. Nous avons étudié des modèles de représentation des connaissances au-delà du cadre logique. Les circuits SPN (*Sum Product Networks*) sont des représentations de modèles probabilistes ou plus généralement de fonctions réelles positives. Ce sont des circuits manipulant des valeurs réelles, dont les portes correspondent à des opérateurs + ou x, et dont les

entrées sont des constantes ou des variables indicatrices, voire des variables aléatoires suivant des distributions usuelles (par exemple gaussiennes) pour des SPN probabilistes.

Comme pour les circuits booléens largement étudiés en compilation de connaissances, il est possible d'identifier des propriétés structurelles des SPN et de les regrouper en classes selon ces propriétés. Par exemple les SPN décomposables sont ceux dont chaque porte x manipule des sous-circuits agissant sur des ensembles disjoints de variables, les SPN monotones ne manipulent aucune constante négative, etc. L'idée derrière cette catégorisation est d'identifier des classes de SPN pour lesquelles certaines requêtes - le plus souvent de nature probabiliste, comme des calculs de marginales ou de maximums a posteriori - sont traitables en temps polynomial en la taille du circuit. Se pose alors la question des relations de compacité entre ces classes de SPN : pour deux classes données, est-il toujours possible de passer d'un SPN de la première classe à un SPN équivalent dans la seconde sans que la taille n'explose, voire, cette réécriture d'une classe à l'autre peut-elle diminuer la taille d'un facteur exponentiel ? Si oui, selon les requêtes à traiter, utiliser un SPN de la seconde classe sera plus avantageux.

Notre travail apporte des premiers éléments de réponse : par le biais de réductions vers des classes de circuits booléens, nous avons dressé une carte de compacité complète des classes les plus étudiées de SPN monotones. Nos réductions ne s'appliquent pas aux classes de SPN non monotones, c'est-à-dire les SPN manipulant des valeurs négatives, mais nous avons étendu notre étude à ces autres SPN, motivés par l'observation qu'introduire des négations induit un gain en espace parfois exponentiel, sans impacter pour autant la nature des requêtes traitables en temps polynomial. En nous inspirant des méthodes existantes pour analyser la taille de circuits booléens, nous avons développé des techniques pour prouver des bornes inférieures sur la taille de circuits SPN selon leurs propriétés et avons appliqué ces techniques pour obtenir des bornes exponentielles pour des SPN non monotones.

Ordonnancement de lignes de production. Nous avons conduit une étude d'identification et de formalisation des requêtes qui se posent dans divers problèmes de d'ordonnancement sous incertitude de lignes de production (réaction à des aléas). En parallèle, nous avons testé une approche de compilation naïve (par discrétisation du temps) des problèmes d'ordonnancement de type RCPSP, qui montre que même en utilisant les compilateurs les plus performants à ce jour (identifiés par la pré-étude mentionnée ci-après – WP4), les problèmes compilables restent de taille restreinte (30 tâches au plus).

Jeux bayésiens. Nous avons aussi entamé une analyse de complexité des problèmes de recherche d'équilibres mixtes dans les jeux bayésiens succincts, montrant que le langage des jeux polymatriciels / jeux hypergraphiques permettent une augmentation de la compacité des jeux décrits, sans apporter d'augmentation de la complexité théorique.

Distance entre représentations. Pour enrichir les applications possibles de la compilation de connaissances, il est intéressant d'identifier des requêtes et transformations supplémentaires qui sont difficiles en général, mais deviennent traitables lorsque tout ou partie des informations qu'elles mobilisent ont été compilées.

Parmi les requêtes d'intérêt figure celle du calcul de distance entre représentations. En effet, dans de nombreux problèmes d'IA, on évalue la singularité d'une situation comme sa distance à une situation normale, de référence. Ceci peut servir par exemple à définir les diagnostics préférés d'un système (ceux qui sont les plus proches d'une situation de référence où aucun composant du système n'est en panne), à définir des préférences (plus une alternative s'écarte du meilleur choix, moins elle est préférée), à définir des opérateurs de révision de croyances comme l'opérateur de Dalal (qui préfère parmi les modèles de la formule de changement ceux qui sont à une distance de Hamming minimale de la base de croyances initiale).

Dans l'étude que nous avons réalisée, nous nous sommes focalisés sur le calcul de la distance de Hamming (possiblement pondérée) entre représentations propositionnelles cohérentes. Il s'agit de déterminer le nombre de bits à inverser au minimum pour passer d'un modèle d'une des deux représentations à un modèle de l'autre. Nous avons considéré plusieurs langages de compilation de la littérature, comme DNF, OBDD, FBDD, d-DNNF, SDNNF, DNNF. Nous avons identifié la complexité du calcul de la distance de Hamming entre deux formules prises dans ces langages. Si le problème est NP-difficile en général, quelques cas traitables ont été mis en évidence. Ainsi, si l'une des deux formules est en DNF, le problème est traitable. Par ailleurs, si les deux formules sont des formules structurées ayant la même structure (des OBDDs reposant sur le même ordre ou plus généralement, des SDNNFs reposant sur le même vtree), le calcul de la distance de Hamming entre ces formules est réalisable en temps polynomial.

Apprentissage de DNNF. De par leur interprétabilité et leurs capacités explicatives, il est naturel de s'intéresser à l'apprentissage de circuits DNNF. Notamment, à la question de savoir comment induire à partir d'un ensemble de données binaires, chacune étiquetée comme positive ou négative, un modèle sous forme DNNF capable de classifier précisément E de futures données, non étiquetées ? Cette tâche de classification s'appelle l'apprentissage de concepts et fait intervenir deux composants : la classe de concept C et l'ensemble d'entraînement E.

Les deux principaux problèmes reliés à l'apprentissage de concept sont le problème de cohérence (existe- t-il au moins un concept dans C qui classifie correctement tous les exemples de E ?) et le problème d'optimisation (trouver un concept dans C qui minimise le nombre d'erreurs de classification dans E).

L'apprentissage de concepts logiques a été très étudié dans la communauté COLT (*Computational Learning Theory*) et les résultats sont bien souvent négatifs. En particulier, lorsque C est la classe des arbres de décision ou bien la classe des formules de type DNF (Forme Normale Disjonctive), le problème de cohérence est NP-complet, et le problème d'optimisation (NP-dur) n'est pas approximable en temps polynomial avec un facteur constant [Feldman et al., 2009]. De tels résultats nous incitent naturellement à explorer des approches heuristiques pour apprendre des concepts logiques. Dans cette perspective, Bessiere et al. (2009) et Narodytska et al. (2018) se sont intéressés à déterminer un encodage SAT du problème de cohérence lorsque C est la classe des arbres de décision (avec un nombre de nœuds fixé). Notamment, les résultats empiriques de [Narodytska et al., 2018] ont montré qu'en résolvant le problème de cohérence via un solveur SAT, il est possible d'apprendre des arbres de décision optimaux sur des données de taille réelle.

Dans ce contexte, un de nos objectifs est d'adopter une approche similaire pour la classe C des concepts DNNF de taille ou de profondeur fixée. Les problèmes de cohérence et d'optimisation sont respectivement encodés en SAT et partial MAXSAT. Comme les DNNF sont construites sur des graphes orientés sans circuit (DAG) avec la propriété de décomposabilité sur les nœuds AND, notre encodage SAT pour DNNF est différent des encodages SAT proposés pour les arbres de décision. Notre travail actuel consiste à optimiser cet encodage SAT et de tester l'apprentissage de DNNF sur des données de taille réelle.

Bessiere, C., Hebrard, E., and O'Sullivan, B. (2009). Minimising decision tree size as combinatorial optimisation. In Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming, CP'09, page 173–187. Springer-Verlag.

Feldman, V., Gopalan, P., Khot, S., and Ponnuswami, A. K. (2009). On agnostic learning of parities, monomials, and halfspaces. SIAM Journal of Computing, 39(2):606–645.

Narodytska, N., Ignatiev, A., Pereira, F., and Marques-Silva, J. (2018). Learning optimal decision trees with sat. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18), IJCAI'18, page 1362–1368. AAAI Press.

Abduction. Le raisonnement abductif constitue un sujet d'étude important en intelligence artificielle. Dans un cadre logique il s'agit de déterminer une conjonction d'hypothèses (prise dans un vocabulaire particulier), qui ajoutée à la théorie du domaine dont on dispose, permet de rendre compte d'un ensemble de manifestations observées (i.e., de les déduire logiquement). Pour éviter de devoir considérer des explications triviales, on demande que les explications produites soient cohérentes avec la théorie du domaine dont on dispose. En logique propositionnelle, le problème de déterminer si une explication abductive existe est difficile (au deuxième niveau de la hiérarchie polynomiale). Imposer des restrictions sur la traitabilité de la théorie du domaine pour le problème de la déduction clausale (en supposant par exemple qu'elle est donnée sous forme normale disjonctive, sous forme d'un diagramme de décision binaire ordonné ou encore par une formule de Horn sous forme normale conjonctive) fait baisser la complexité du problème d'un niveau, mais ne suffit pas, sous les hypothèses usuelles de la théorie de la complexité, à rendre le problème traitable. Pour cela, des restrictions beaucoup plus drastiques doivent être imposées à la théorie du domaine.

De façon alternative, nous avons étudié la possibilité d'obtenir des restrictions traitables au problème d'existence d'explications abductives rendant compte d'un ensemble de manifestations lorsque la théorie du domaine a été compilée et que l'on considère certains paramètres. Au delà du problème de l'existence d'explications abductives, nous avons considéré le problème (plus général) de leur énumération. Plutôt que de nous focaliser sur un langage de compilation particulier, nous avons identifié des conditions (en terme de requêtes et transformations offertes par le langage de compilation, dans l'esprit de la carte de compilation de Darwiche et Marquis) sous lesquelles le problème de l'existence ou de l'énumération d'explications abductives est traitable pour certains paramètres. Dans cette étude, nous avons considéré comme paramètres le nombre d'hypothèses possibles, le nombre de non-hypothèses possibles et le nombre de manifestations à expliquer. Pour chacun des cas, des résultats de traitabilité ont pu être identifiés.

E.2 WP2

Réfutations et DNNF. L'étude de la complexité de preuves a pour objectif l'obtention de bornes inférieures sur des réfutations de formules propositionnelles dans des systèmes de preuves. Pour certains systèmes et certaines classes de formules, la taille des réfutations est directement liée à la difficulté de représenter certaines fonctions (en lien avec les formules) dans des langages spécifiques, qui parfois sont des langages de compilation. Nous utilisons des techniques initialement mises au point pour étudier ces langages dans le cadre de la compilation de connaissance, afin répondre à des questions sur les systèmes de preuves.

Un des systèmes de preuve les plus étudiés est le système basé sur la règle de résolution pour des clauses. L'intérêt que suscite ce système de preuve s'explique, entre autre, par son utilité pour l'analyse des solveurs SAT basés sur l'approche CDCL. La difficulté à montrer des bornes inférieures sur la longueur de réfutations par résolution a conduit la communauté à étudier des variantes plus contraintes de ces réfutations, notamment la réfutation par résolution dite «régulière» (*regular resolution refutation*).

Nous avons étudié ce type de réfutations pour des formules de Tseitin. Ces formules, qui encodent des systèmes d'équations linéaires dont la structure est décrite par un graphe, ont été largement étudiées pour la réfutation dans de nombreux systèmes de preuve. Nos travaux ont abouti à une caractérisation des formules de Tseitin non satisfiables, dont la réfutation régulière par résolution

ne nécessite qu'un nombre polynomial de clauses. Notre contribution est la preuve que les plus petites réfutations de ce type pour les formules de Tseitin sont de longueur polynomiale si et seulement si la «*treewidth*» (ou largeur d'arbre) des graphes qui en décrivent la structure est en O(log(n)), où n désigne le nombre de nœuds du graphe. L'originalité de ces travaux réside dans le lien qui y est fait entre la longueur de la plus petite réfutation régulière par résolution de formules de Tseitin non satisfiables et la taille des représentations DNNF de formules de Tseitin satisfiables travaillant sur les mêmes graphes. Nous montrons que si ces représentations DNNF sont de taille au moins exponentielle alors il est en est de même pour la longueur des réfutations étudiées, puis nous prouvons que les DNNF des formules de Tseitin sont effectivement exponentielles en la *treewidth* du graphe de la formule.

Définition et étude de systèmes de preuve pour les formules booléennes quantifiées (QBF). Nous avons proposé et étudié des systèmes de preuve pour les formules booléennes quantifiées (QBF) opérant sur des diagrammes de décision binaire ordonnés (OBDD). Ces systèmes capturent le comportement des solveurs QBF à base d'élimination des quantifications. En tant que tels, ils permettent des preuves qui sont courtes en terme de largeur de chemin bornée et de complexité de quantification. Nous avons obtenu des séparations exponentielles des longueurs des preuves en comparaison aux systèmes de preuve standard à base de formules sous forme normale conjonctive (CNF), de type QU-Resolution ou IR-Calc. Nous avons également développé une technique de borne inférieure pour les systèmes de preuve QBF, basée sur l'extraction de stratégies, qui permet de généraliser les bornes connues, issues de la théorie de la complexité de communication. Cette technique nous permet de dériver de meilleures bornes inférieures pour les systèmes de preuve QBF, qui sont indépendantes de l'ordre des variables utilisé pour définir les diagrammes de décision et qui restent valides même quand le système de preuve est équipé d'un oracle NP.

Compilation ascendante (*bottom-up*). L'approche bottom-up pour la compilation de CNF est un paradigme qui consiste à compiler en premier chaque clause individuellement dans le langage cible, puis à construire, par des opérations «simples», des circuits compilés à partir de ceux précédemment obtenus jusqu'à l'obtention d'une forme compilée équivalente à la formule CNF. La compilation bottom-up passe donc par une série de formes compilées intermédiaires avant d'obtenir celle de la CNF. En pratique, l'implémentation les opérations «simples» nécessite que les formes compilées en opérande partagent une certaine structure. Ainsi les algorithmes de compilation bottom-up sont essentiellement utilisés pour des langages structurés tels qu'OBDD ou SDD.

Nous nous sommes penchés sur la pertinence de l'approche bottom-up pour des classes de CNF dont nous connaissons l'existence de formes compilées très compactes, notamment des CNF non satisfiables. Pour nos classes de CNF non satisfiables, nous avons montré que l'approche bottom-up utilise des formes compilées intermédiaires de taille exponentielle et prend donc un temps exponentiel pour construire une représentation finale qui se résume pourtant à un seul nœud. La compilation bottom-up de formules non satisfiables agissant comme un programme de réfutation, nous avons choisi des classes de formules connues comme difficiles à réfuter dans de nombreux systèmes de preuves. Les formules en question sont les formules de Tseitin qui, pour rappel, encodent des systèmes d'équations linéaires dont la structure est décrite par un graphe.

Nous avons étudié la réfutation de ces formules par des compilations bottom-up dont les formes intermédiaires sont des DNNF structurées par des v-tree (des arbres de variables) et dont les opérations sont le changement de v-tree d'une DNNF structurée et la conjonction de deux DNNF respectant des v-tree compatibles. Ce système relativement simple englobe déjà les compilations bottom-up utilisées en pratique vers les langages SDD et OBDD. Nos avons montré l'existence de classes de formules de Tseitin non satisfiables dont les réfutations par compilation utilisent des DNNF de taille exponentielle en la *treewidth* des graphes correspondant aux formules.

E.3 WP3

Approximation de représentations par des d-DNNFs. Dans de nombreuses applications de la compilation de connaissances, la connaissance disponible constitue un modèle approché de la réalité. Ainsi, les modèles probabilistes - par exemple des réseaux bayésiens - construits par apprentissage sont nécessairement imparfaits. Dans ce contexte, on peut envisager de compiler des approximations des modèles pour obtenir des formes approchées plus compactes. Une forme approchée reste une approximation de la réalité et, moyennant la possibilité de contrôler l'erreur d'approximation, de précision comparable à celle de la forme exacte. La recherche en compilation de connaissances approchée s'articule autour de deux axes. Le premier est la caractérisation de classes de fonctions dont les représentations dans un langage donné sont larges mais ont des approximations exponentiellement plus compactes. Le deuxième est la recherche de bornes inférieures pour des classes de fonctions dont les représentations exactes comme approchées sont de taille exponentielle.

Le travail que nous avons réalisé s'inscrit dans le deuxième axe. Nous nous sommes concentrés sur les représentations d-DNNF, lesquelles sont d'intérêt pour les modèles probabilistes puisqu'elles permettent le comptage (potentiellement pondéré) de modèles, auquel l'inférence probabiliste se réduit. Nous avons raffiné la notion d'approximation jusqu'alors considérée dans la littérature, que nous appelons approximation faible et qui, parce qu'elle autorise l'approximation des fonctions de peu de modèles par la fonction nulle, est inadaptée à nos besoins. Nous avons défini une approximation forte qui s'affranchit de ce problème et offre des garanties vis-à-vis du comptage de modèles. Nous avons ensuite prouvé les limites des approximations faible et forte pour la compilation en d-DNNF. Nous avons d'abord exhibé des classes de fonctions dont les d-DNNF exactes et les d-DNNF de faibles approximations sont nécessairement de taille exponentielle. Puis nous avons mis en évidence des classes de fonctions, dont il existe des approximations faibles triviales et sans intérêt (typiquement par la fonction nulle) mais dont les fortes approximations n'ont que des d-DNNF de taille exponentielles.

Une publication décrivant ce travail est parue dans les actes de la conférence internationale de rang A* IJCAI-PRICAI'20.

Compilation approchée. Nous avons étudié dans quelle mesure le pré-processeur B+E pour le comptage de modèles, que nous avons développé précédemment (communication à IJCAI'16), pouvait être utile à la compilation approchée de formules propositionnelles. Par compilation approchée, il faut entendre ici compilation restreinte à un sous-alphabet du vocabulaire propositionnel de la formule considérée. Il est attendu que la forme compilée approchée à calculer constitue une conséquence logique (*upper approximation*) de la formule de départ qui doit être logiquement équivalente à la projection de cette formule d'entrée sur le sous-alphabet considéré. De ce fait, si la forme compilée obtenue ne permet pas de répondre exactement à toutes les requêtes d'interrogation formulées vis-à-vis de la formule initiale, elle préserve les réponses portant sur le sous-alphabet choisi. Cette condition suffit dans un bon nombre d'applications (les variables propositionnelles introduites à des fins d'encodage n'ont pas de contreparties dans l'ontologie du domaine et peuvent être laissées de côté).

Nous avons montré que le pré-processeur B+E peut être utilisé en amont d'un compilateur exact ou approché en préservant la condition énoncée, à condition de contraindre la bipartition des variables qu'il calcule à inclure toutes les variables d'intérêt dans l'ensemble des variables d'entrée. Nous avons réalisé des expérimentations pour évaluer les bénéfices pouvant être obtenus en utilisant B+E en amont des compilateurs exacts c2d et d4, disponibles en ligne. Nos expérimentations ont porté sur des instances du problème de planification classique en horizon

fini. 530 instances issues de différentes familles ont été considérées. Le vocabulaire d'intérêt est réduit aux variables propositionnelles codant les fluents à l'instant initial de l'horizon, à l'instant final de l'horizon, et les variables d'action à tout instant de l'horizon. Grâce à ces expérimentations, nous avons montré qu'en pratique, les temps de compilation et les tailles des formes compilées pouvaient être réduits (parfois drastiquement) en utilisant B+E.

Le travail réalisé a donné lieu à une publication dans la revue internationale de rang A* *Artificial Intelligence* en 2020.

Opérations arithmétiques en temps constant. Comme pour les algorithmes usuels, la conception et l'analyse fine des algorithmes de comptage, d'énumération, et, plus largement, celles des algorithmes de compilation s'effectuent dans le modèle RAM. Dans le *meilleur des cas*, un algorithme de compilation pré-calcule d'abord, *en temps linéaire*, à partir d'une donnée initiale, un index (souvent un ensemble de tables), à partir duquel il peut répondre, *en temps constant,* à une certaine question, sur toute donnée complémentaire (de taille fixée), fournie à la volée. Les opérations élémentaires sur les structures de données (arbres, circuits, etc.) et les opérations arithmétiques usuelles sont les instructions de base des algorithmes. Implémenter ces opérations de base de façon optimale est donc un enjeu fondamental pour une algorithmique efficace et, singulièrement, pour la compilation de connaissances. A ce sujet, nous avons entrepris une étude systématique des opérations arithmétiques calculables en temps constant dans le modèle RAM munie de la seule addition et où les contenus des registres sont bornés à O(N), où N est la taille de l'entrée (= le nombre de registres où elle est stockée). C'est le modèle minimal, sachant que l'allocation de mémoire et l'adressage des tableaux nécessite à l'évidence l'addition. Il était connu depuis les années 90 que ce modèle permet d'implémenter en temps constant, en plus de l'addition, la soustraction et la multiplication d'entiers « polynomiaux », c'est-à-dire inférieurs à N^d, pour une constante d ; ceci à partir de tables, pré-calculées en mémoire et temps linéaire, c'est-à-dire O(N). Depuis plus de vingt ans, la dernière des quatre opérations arithmétiques usuelles, la division, semblait inaccessible, sauf de façon partielle.

Depuis juin 2019, suite à un premier algorithme obtenu par Louis Jachiet en 2017 (symptomatiquement motivé par la résolution efficace d'un problème de compilation de connaissances), nous avons entrepris une étude systématique et exhaustive des opérations arithmétiques calculables en temps constant dans le modèle RAM défini ci-dessus : nous avons non seulement établi que la division portant sur des opérandes polynomiaux se calcule en temps constant — toujours à partir de tables pré-calculées en temps linéaire —, donc résolu complètement le problème des quatre opérations arithmétiques usuelles dans le modèle RAM, mais, en combinant ces opérations — et le rôle de la division s'avère ici fondamental —, nous avons établi que d'autres opérations naturelles, toujours portant sur des opérandes entiers polynomiaux et dans les mêmes conditions, s'implémentent aussi en temps constant: racine carrée entière et ses généralisations (partie entière de la racine c-ième, pour une constante entière c), partie entière du logarithmique de x en base y, où x, y sont les opérandes, etc. Les méthodes de tous ces algorithmes fonctionnant en temps constant sont des méthodes d'approximation, dont la méthode classique de Newton, adaptées aux entiers. Sur ces résultats, nous préparons actuellement un article qui sera soumis au journal *Computational Complexity*.

A la suite de ce travail, il nous paraît intéressant, d'une part, de constituer un inventaire systématique des algorithmes disponibles implémentant *en temps constant* les opérations élémentaires sur les structures de données utilisées en compilation de connaissances (arbres, circuits), d'autre part, de concevoir de nouveaux algorithmes — ou de montrer leur impossibilité — quand nous ne disposons pas de tels algorithmes. Il est clair que de tels algorithmes pourront utiliser «sans surcoût » (= en temps constant) les opérations arithmétiques étudiées précédemment.

Génération de ROBDDs à taille fixée. Une question liée à plusieurs problématiques du WP (compilation approchée, complexité en moyenne) est celle de la génération (aléatoire ou exhaustive) de formes compilées. Nous avons travaillé sur cette question en ce qui concerne les ROBDD (pour *Reduced Ordered Binary Decision Diagrams*), une des structures de graphes acycliques permettant d'encoder une fonction booléenne. Grâce à une nouvelle approche de la construction des ROBDD, nous sommes en mesure de les spécifier via une construction non ambiguë, qui donne une formule de récurrence permettant de compter les structures en fonction de leur taille. Cette récurrence constructive est la base de l'algorithme de génération à taille fixée que nous avons introduit. Il s'agit d'un algorithme de déclassement (*unranking* en anglais) qui permet aussi bien la génération exhaustive que la génération aléatoire. L'approche a été validée pour les ROBDDs possédant jusqu'à 9 variables là où les travaux précédents sur ce même type d'approche s'arrêter à 4 variables. Une communication a été acceptée à la conférence internationale de rang B LATIN'20 (et présentée en janvier 2021) sur ce travail. Depuis, les algorithmes de comptage et de génération aléatoire ont été fortement améliorés en considérant une récurrence de type double et un parcours adapté de la structure d'arbre sous-jacente aux ROBDDs. Cela permet de gagner un ordre de grandeur sur la complexité en temps. Un article de revue est en cours de rédaction.

Compilation de programme logique en programme parallèle. Si le modèle de référence du calcul séquentiel est le modèle RAM, les automates cellulaires constituent le modèle de référence pour le calcul parallèle et local. Le travail dont on fait état ici est une forme de compilation pour les programmes parallèles. Il s'agit de résoudre un problème à partir de sa définition inductive en logique, en l'occurrence une logique de Horn. On a exhibé des logiques de Horn qui caractérisent exactement la classe des langages reconnaissables en temps minimal (appelé temps réel) sur automates cellulaires.

L'intérêt de ces résultats est qu'ils permettent, à partir de la définition inductive d'un problème en logique — un travail « facile » de programmation logique — d'en déduire, par un processus automatique (une forme de compilation) un programme d'automate cellulaire qui décide le problème considéré, ceci en temps minimal. Ces résultats sont d'autant plus intéressants que la conception de programmes parallèles est connue pour être une tâche difficile.

L'aspect pratique de la méthode de programmation ainsi introduite est validé par le fait que nous l'avons appliquée à un échantillon représentatif des problèmes de la littérature connus pour être décidables en temps minimal sur automates cellulaires, re-démontrant ainsi de façon simple et courte — grâce à la logique et à sa compilation — que tous ces problèmes ont une telle complexité minimale. Ces travaux ont fait l'objet de trois articles, l'un publié à la conférence internationale de rang A* LICS'19 et deux actuellement soumis, un à la revue *Theoretical Computer Science* et l'autre à l'atelier Automata'21.

E.4 WP4

Evaluation de compilateurs existants. Nous avons installé et expérimenté plusieurs compilateurs « sur étagère » (D4, Dsharp, cnf2obdd, cnf2eadt, cudd, sdd, saladd, cn2mddg) sur un ensemble de benchmarks et de *case studies* commun pour évaluer et comparer leurs performances en pratique. Des expérimentations ont été lancées en novembre 2020, s'appuyant sur un protocole de configuration de produits ; nous les complétons actuellement avec d'autres expérimentations s'appuyant cette fois sur un protocole de diagnostic de pannes.

Compilation certifiée. L'utilisation croissante de la compilation de connaissances dans divers cadres (configuration de produits assemblés, provenance dans les bases de données, méthodes formelles, IA explicable, etc.) rend impérieux le développement d'outils permettant de vérifier les formes compilées produites. Il s'agit non seulement de pouvoir s'assurer que les résultats fournis par les compilateurs sont bien ceux que l'on attend, mais aussi de pouvoir contrôler que les outils

de raisonnement s'appuyant sur les formes compilées sont eux aussi corrects. Par exemple, dans une application dont les requêtes en ligne nécessitent de devoir incorporer des contraintes simples (mais ne se réduisant pas seulement à un conditionnement) avant de produire une solution disponible, on pourra opter pour le langage de compilation d-SDNNF qui offre la conjonction bornée : pour s'assurer qu'une requête est traitée correctement, il faut non seulement être sûr que la forme compilée initiale est correcte, mais que chaque ajout de contrainte conduit à une forme compilée qui l'est encore.

Aucun outil n'existe à ce jour pour réaliser de telles tâches de vérification efficacement (i.e., en temps polynomial) et vraisemblablement, un tel outil n'existera jamais. En effet, quand les contraintes sont exprimées en CNF, le test d'équivalence entre une représentation en d-SDNNF et une représentation en CNF est coNP-complet. Pour utiliser la terminologie de la carte de compilation, on peut dire que d-SDNNF ne vérifie pas la propriété de CNF-certifiabilité, qui consiste à pouvoir décider efficacement si la forme compilée considérée est ou n'est pas équivalente à une CNF donnée. Pour assurer cette propriété, il importe donc de définir de nouveaux langages de compilation, qui bien sûr devront offrir les propriétés attendues des formes compilées (le test de cohérence, le conditionnement, etc.) et garantir en plus la CNF-certifiabilité.

Nous nous sommes attachés à la réalisation de cet objectif et, à cet effet, introduit de nouveaux langages de compilation dans la famille DNNF, appelés langages des DNNF décorées. Intuitivement, il s'agit de mémoriser de l'information supplémentaire dans les nœuds d'une représentation DNNF « classique » de façon à offrir la propriété de CNF-certifiabilité. La décoration ajoutée joue le rôle d'un certificat permettant une vérification en temps polynomial. Comme une formule CNF incohérente possède une représentation DNNF réduite à un seul nœud, ce certificat a dans le pire des cas une taille exponentielle dans la taille de la représentation DNNF « classique » associée, et il en va ainsi de tout certificat possible sauf si NP = coNP. En conséquence, le langage des DNNF décorées est strictement moins succinct que le langage des DNNF « classique ».

Nous avons aussi montré comment transformer un compilateur DNNF top-down « classique » en compilateur top-down produisant des formes compilées décorées. Nous avons appliqué les règles de transformation introduites au compilateur Decision-DNNF d4 pour produire un compilateur générant des formes décorées et réalisé un large ensemble d'expérimentations permettant de comparer les temps de compilation et les tailles de forme compilées, selon que des formes « classiques » ou décorées sont considérées.

Une publication décrivant ce travail a été accepté à la conférence internationale de rang A* AAAI'21.

Nouvelles approches de gestion de cache. L'utilisation en pratique de la compilation pour traiter des instances de taille de plus en plus grande et de structure de plus en plus complexe requiert le développement de compilateurs de plus en plus performants. La mémoire consommée lors de la phase de compilation apparaît bien souvent en pratique comme une ressource critique. Pour maintenir à un niveau acceptable cette ressource, nous avons conçu et évalué un schéma de mise en cache amélioré de formules CNF et une stratégie de gestion améliorée d'un tel cache. Ces stratégies peuvent être exploités au sein de tout compilateur DNNF descendant (top-down) ou encore de compteurs de modèles procédant de façon similaire. Le schéma de mise en cache proposé consiste à stocker pour chaque entrée (une formule CNF étant donnée une affectation partielle des variables) l'ensemble correspondant de variables et de clauses, à l'exception des clauses qui sont satisfaites ou non raccourcies lorsqu'elles sont conditionnées par l'affectation partielle courante. La stratégie de gestion du cache comprend une stratégie de nettoyage du cache, basée non seulement sur l'âge des entrées, mais également sur la proportion d'entrées de même taille qui ont conduit à des résultats positifs. Il inclut également une stratégie d'insertion dans le

cache, qui vise à économiser la mémoire en évitant de stocker dans le cache chaque formule CNF rencontrée lors de la recherche. Nous avons prouvé la correction du schéma de mise en cache amélioré que nous avons proposé. Nous avons également réalisé une évaluation empirique montrant les bénéfices fournis en pratique par le schéma de mise en cache et la stratégie de gestion de cache proposés lorsqu'ils sont utilisés dans un solveur #SAT descendant. Sur le jeu de données formé par les 400 instances CNF considérées pour évaluer les performances des compteurs modèles (éventuellement pondérés) lors du premier concours international de comptage de modèles qui s'est tenu en 2020 (voir https://mccompetition.org/), les gains obtenus avec notre compteur / compilateur d4 muni de la version optimisée du cache ont ainsi dépassé 10% d'instances résolues supplémentaires.