Représentations efficaces pour la logique modale S5 *

Alexandre Niveau **Bruno Zanuttini**
GREYC, UMR 6072, UNICAEN/CNRS/ENSICAEN, France
{alexandre.niveau,bruno.zanuttini@unicaen.fr}

Résumé

Cet article examine l'effacité de plusieurs représentations des formules subjectives de la logique modale de la connaissance, S5, et plus généralement des ensembles d'ensembles d'instanciations propositionnelles. L'une des motivations de ce travail est la planification contingente, pour laquelle de nombreuses approches s'appuient sur de telles formules, et pourraient manifestement tirer parti de représentations efficaces. Nous étudions le langage S5-DNF introduit par Bienvenu *et al.*, ainsi qu'une variante naturelle de ce langage, qui utilise des diagrammes de décision binaires au niveau propositionnel. Nous introduisons de plus un autre langage, celui des « diagrammes épistémiques partitionnants », qui fournit des représentations plus compactes. Les trois langages sont comparés d'un point de vue théorique, dans le cadre de la compilation de connaissances, et d'un point de vue expérimental. Cette étude éclaire les avantages et inconvénients de chaque représentation en théorie comme en pratique.

Abstract

We investigate efficient representations of subjective formulas in the modal logic of knowledge, S5, and more generally of sets of sets of propositional assignments. One motivation for this study is contingent planning, for which many approaches use operations on such formulas, and can clearly take advantage of efficient representations. We study the language S5-DNF introduced by Bienvenu *et al.*, and a natural variant of it that uses Binary Decision Diagrams at the propositional level. We also introduce an alternative language, called Epistemic Splitting Diagrams, which provides more compact representations. We compare all three languages from the complexity-theoretic viewpoint of knowledge compilation and also through experiments. Our work sheds light on the pros and cons of each representation in both theory and practice.

*Cet article est la traduction de l'article « Efficient representations for the modal logic S5 », in Proc. 25th International Joint Conference on Artificial Intelligence (IJCAI 2016).

1 Introduction

La logique modale épistémique S5 est la logique de la connaissance mono-agent [6]. Elle permet d'exprimer des formules comme $(\mathrm{K}p \vee \mathrm{K}\bar{p}) \wedge (\neg \mathrm{K}(p \wedge q))$, qui signifie que l'agent sait que p est vrai ou sait que p est faux (c'est-à-dire qu'il connaît la valeur de p), mais ne sait pas que $p \wedge q$ est vrai (soit il sait que $p \wedge q$ est faux, soit il ne connaît pas sa valeur de vérité).

La logique S5 apparaît naturellement dans le cadre de la planification contingente [8, 15, 9, 11, 2], qui vise à calculer un plan menant à un objectif donné en utilisant deux types d'actions : les actions *ontiques* changent l'état du monde de façon non déterministe, tandis que les actions *épistémiques* donnent à l'agent une information sur l'état du monde. Le plan recherché peut contenir des conditions sur les informations reçues via les actions épistémiques.

À tout moment, un agent qui exécute un plan contingent envisage un unique ensemble d'états comme étant les états possibles du monde. Cet ensemble de candidats est appelé *état de croyance* dans la littérature. Par exemple, considérons un agent qui sait que l'état initial satisfait $p \wedge q$, et qui exécute une action ontique *switch*, qu'il sait avoir pour effet d'inverser soit la valeur de p soit celle de q (le choix étant non déterministe). Son état de croyance résultant peut alors être décrit par $(\bar{p} \wedge q) \vee (p \wedge \bar{q})$. S'il exécute alors l'action épistémique *test* indiquant si $p \wedge r$ est vrai, et reçoit comme réponse que c'est effectivement le cas, alors son état de croyance devient $(p \wedge \bar{q} \wedge r)$.

Pendant l'étape de planification, ou lors de la vérification de la validité d'un plan, il est également utile de considérer simultanément l'évolution de plusieurs états de croyance possibles, processus habituellement appelé *progression hors-ligne*. Par exemple, si l'état de croyance initial est $p \wedge q$, alors l'action ontique

switch mène comme précédemment à $(\bar{p} \wedge q) \vee (p \wedge \bar{q})$, mais l'action épistémique *test* peut mener à deux états de croyances différents, en fonction de ce qu'elle renvoie : soit $(p \wedge \bar{q} \wedge r)$, comme précédemment, soit $(\bar{p} \wedge q) \vee (p \wedge \bar{q} \wedge \bar{r})$, si l'agent reçoit l'information $\bar{p} \vee \bar{r}$. Il est donc important, en planification mais aussi pour d'autres applications, de pouvoir manipuler des formules de S5, qui représentent des *ensembles* d'états de croyance. Dans l'exemple précédent, cela permettrait de représenter le résultat de la progression via l'action épistémique par $\mathrm{K}(p \wedge \bar{q} \wedge r) \vee \mathrm{K}\big((\bar{p} \wedge q) \vee (p \wedge \bar{q} \wedge \bar{r})\big)$. Remarquons ici qu'en planification, on préfère généralement utiliser K plutôt que la modalité *only-knowing* [12] : les objectifs étant souvent positifs, il est toujours préférable de connaître davantage. En particulier, il arrive souvent de n'avoir besoin que de formules contenant des connaissances *positives*.

Dans cet article, nous étudions plusieurs manières de représenter les formules (subjectives) de S5, du point de vue de l'efficacité en temps et en espace, dans l'optique de les appliquer à la planification contingente. Nous considérons le langage $\texttt{s-S5-DNF}_{\texttt{DNF,CNF}}$ proposé par Bienvenu *et al.* [1], ainsi que sa variante naturelle $\texttt{s-S5-DNF}_{\texttt{OBDD,OBDD}}$. De plus, nous introduisons une nouvelle représentation (sections 3 et 4), celle des « diagrammes épistémiques partitionnants » (ESD, pour *epistemic splitting diagrams*), qui s'inspirent des diagrammes de décision binaires. Nous examinons ces trois langages du point de vue de la *compilation de connaissances* [5], en comparant leur capacité respective à supporter efficacement *requêtes* et *transformations* (section 5), et à représenter avec *concision* les formules de S5 (section 6). Pour finir, nous présentons quelques résultats expérimentaux, qui confirment en pratique les propriétés des différents langages (section 7). Nos résultats font ressortir des avantages et des inconvénients pour chaque langage ; ils démontrent également que les ESD sont plus compacts que les représentations classiques en ce qui concerne les formules *positives* de S5.

2 Préliminaires

S5 Les concepts de base de la logique propositionnelle sont supposés connus. On considère le langage S5 propositionnel [6], dans lequel les formules sont construites à partir d'un ensemble X d'atomes propositionnels avec les connecteurs usuels $\neg, \vee, \wedge$, auxquels s'ajoute la modalité de connaissance K. Par exemple, $(\mathrm{K}x_1 \wedge \neg\mathrm{K}(x_2 \vee \bar{x}_3)) \vee \neg\mathrm{K}(\bar{x}_1)$ est une formule de S5. On notera $\mathrm{Var}(\Phi)$ l'ensemble des atomes mentionnés dans une formule Φ (c'est-à-dire $\{x_1, x_2, x_3\}$ dans l'exemple précédent). Nous nous focalisons sur les formules *subjectives* de S5, qui permettent d'exprimer des faits sur les connaissances des agents, mais pas sur le véritable état du monde. Ce sont en effet principalement ces formules qui sont nécessaires en planification, et, bien que notre étude puisse s'étendre assez directement à S5 en toute généralité, la longueur et la complexité de l'article s'en verraient substantiellement accrus. Nous ne considérerons donc que des formules dans lesquelles tous les atomes propositionnels se trouvent dans la portée de la modalité K : ainsi, par exemple, on ne rencontrera jamais dans cet article une formule comme $x_1 \wedge \mathrm{K}x_2$, qui n'est *pas* subjective (elle indique que x_1 est objectivement vrai). De plus, étant donné que dans S5, on peut se restreindre sans perte d'expressivité aux formules sans modalités imbriquées, nous utiliserons la définition suivante.

Définition 1. Une *formule S5 subjective* sur X est une combinaison booléenne, par les opérateurs $\neg$, $\vee$ et $\wedge$, d'*atomes épistémiques* de la forme $\mathrm{K}\varphi$, où chaque φ est une formule propositionnelle sur X.

On utilisera des majuscules (resp. minuscules) grecques $\Phi, \Psi, \ldots$ (resp. $\varphi, \psi, \ldots$) pour noter les formules de S5 (resp. les formules propositionnelles). L'axiomatique de S5 fait que les formules subjectives s'interprètent naturellement sur des *structures*, qui sont simplement des sous-ensembles non vides de 2^X, c'est-à-dire des ensembles non vides d'instanciations propositionnelles (on suppose implicitement que les structures portent toujours sur l'ensemble de tous les atomes considérés). Intuitivement, une structure représente un état de croyance, c'est-à-dire un ensemble d'instanciations que l'agent considère comme candidats crédibles à être le véritable état du monde ; une formule de S5 représente un ensemble de telles structures, donc un ensemble d'états de croyance.

La sémantique des formules S5 subjectives se définit inductivement : en premier lieu, on dit qu'une structure M *satisfait* un atome épistémique $\mathrm{K}\varphi$ si toutes les instanciations propositionnelles $m \in M$ satisfont φ (dans la sémantique habituelle des formules propositionnelles). Ensuite, M satisfait $\Phi \wedge \Psi$ (resp. $\Phi \vee \Psi$) si elle satisfait Φ et Ψ (resp. Φ ou Ψ), et enfin elle satisfait $\neg\Phi$ si elle ne satisfait pas Φ. Il est important de noter que M satisfait $\neg\mathrm{K}\varphi$ si elle contient au moins un contre-modèle propositionnel de φ (intuitivement, l'agent ne sait pas φ quand φ est fausse dans au moins un état qui est potentiellement le véritable état du monde), et que ce n'est pas du tout la même chose que satisfaire $\mathrm{K}\neg\varphi$.

On écrit $M \models \Phi$ lorsque M satisfait la formule S5 subjective Φ ; on dit que M est un *modèle* de Φ, et on note $\mathrm{Mod}(\Phi)$ l'ensemble des modèles de Φ. Lorsque l'on a $\mathrm{Mod}(\Phi) = \mathrm{Mod}(\Psi)$, les formules Φ et Ψ représentent le même ensemble d'états de croyance — on dit

qu'elles sont *équivalentes*, et on le note $\Phi \equiv \Psi$. Lorsque l'on a $\mathrm{Mod}(\Phi) \subseteq \mathrm{Mod}(\Psi)$, on dit que Φ *implique* Ψ, et on le note $\Phi \models \Psi$. Une formule est dite *tautologique* si elle est satisfaite par toutes les structures.

Enfin, nous avons mentionné plus haut l'importance en planification des formules (subjectives) *positives*, définies comme les combinaisons d'atomes épistémiques par $\vee, \wedge$. On peut montrer qu'elles sont exactement caractérisées par un ensemble de modèles « clos par sous-ensemble non vide », c'est-à-dire qu'il contient tous les sous-ensembles non vides de ses élements. Nous utiliserons également le fait qu'elles peuvent toujours s'écrire sous la forme $\bigvee_i \mathrm{K}\varphi_i$ [1].

Langages propositionnels Avant de nous intéresser aux représentations efficaces pour les formules S5, rappelons quelques concepts de base sur les langages propositionnels. Remarquons tout d'abord qu'on utilise le terme *langage* pour désigner un ensemble de structures de données muni d'une sémantique (voir Fargier *et al.* [7] pour une définition plus formelle). Une formule propositionnelle (sur X) appartient au langage `NNF` (*negation normal form*, forme normale négative) si c'est une combinaison par $\wedge$ et $\vee$ de *littéraux propositionnels* de la forme x ou $\bar{x}$ ($x \in X$). Les sous-formules identiques sont considérées comme partagées dans les formules `NNF`, qui ne sont donc pas des arbres mais des graphes acycliques orientés (DAG, *directed acyclic graphs*). On définit par conséquent la *taille* $|\varphi|$ d'une formule φ comme son nombre de nœuds. Un *terme* (resp. une *clause*) est une conjonction (resp. une disjonction) de littéraux. Une formule est en forme normale disjonctive (resp. conjonctive) si c'est une disjonction de termes (resp. une conjonction de clauses) ; le langage correspondant est appelé `DNF` (resp. `CNF`). *Conditionner* une formule φ par un littéral ℓ revient, intuitivement, à décider de la valeur de l'atome correspondant ; on peut le faire de façon syntaxique en remplaçant chaque instance de ℓ (resp. $\bar{\ell}$) par $\top$ (resp. $\bot$). Le résultat est noté $\varphi|_\ell$. On utilise également la notation $M|_\ell$ pour désigner la structure obtenue à partir d'une structure M en ne gardant que les instanciations satisfaisant ℓ, puis en retirant ℓ de chacune d'elles.

Un *diagramme de décision binaire* (BDD, *binary decision diagram*) est soit une constante, soit une formule de la forme $(x \wedge N) \vee (\bar{x} \wedge N')$, où N, N' sont des BDD ; cette forme est notée $\mathsf{ite}(x, N, N')$. Un BDD peut être vu comme un DAG dont les nœuds sont étiquetés par des atomes (par exemple, à gauche de la figure 1, le DAG de racine x_1 est un BDD, représentant $x_1 \vee x_3$). Un BDD est dit *ordonné* (OBDD) si les atomes sont rencontrés chacun au plus une fois et toujours dans le même ordre sur tous les chemins de la racine à une feuille. Il est bien connu que toute NNF peut être représentée par un OBDD équivalent (pour n'importe quel ordre des atomes) en utilisant l'*expansion de Shannon* : $\varphi \equiv (x \wedge \varphi|_x) \vee (\bar{x} \wedge \varphi|_{\bar{x}})$.

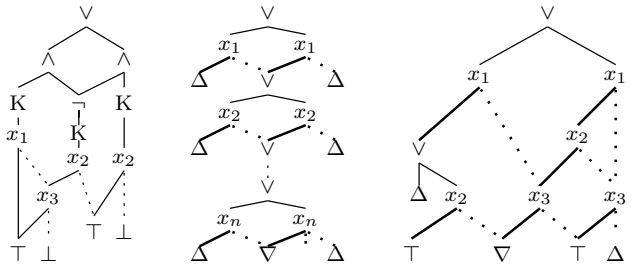


FIGURE 1 – Un EBDD (à gauche) et deux ESD. Les pointillés pointent vers des fils droits des nœuds ite et spl ; les feuilles ne sont dupliquées que pour des raisons de clarté.

S5-DNF Bienvenu *et al.* [1] mènent la première étude de représentations efficaces pour les formules de S5, en introduisant notamment le langage paramétré suivant.

Définition 2. Soient $\mathtt{L}, \mathtt{L}'$ deux sous-langages (propositionnels) de `NNF` ; une formule S5 est dans $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{L},\mathtt{L}'}$ s'il s'agit d'une disjonction de termes, chacun comportant au plus un *littéral épistémique positif* $\mathrm{K}\varphi$ et un nombre arbitraire de *littéraux épistémiques négatifs* $\neg\mathrm{K}\psi_i$, avec φ dans `L` et chaque ψ_i dans $\mathtt{L}'$.

Ils étudient les propriétés de $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{L},\mathtt{L}'}$ dans un cadre général, en fonction des propriétés de `L` et $\mathtt{L}'$. Ils se focalisent en particulier sur une instance spécifique, $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{DNF},\mathtt{CNF}}$ (que nous notons `EDNF`), qui s'avère posséder de bonnes propriétés pour les opérations utilisées en planification. Une autre façon naturelle d'instancier les paramètres est d'utiliser `OBDD` dans les deux cas, ce qui donne le langage $\mathtt{s\text{-}S5\text{-}DNF}_{\mathtt{OBDD},\mathtt{OBDD}}$ (que nous notons `EBDD`) ; Bienvenu *et al.* ne le considèrent pas explicitement, mais la plupart de leurs résultats généraux s'y appliquent. La figure 1 montre (à gauche) un EBDD représentant la formule $(\mathrm{K}(x_1 \vee x_3) \wedge \neg\mathrm{K}(\bar{x}_2 \vee x_3)) \vee (\neg\mathrm{K}(\bar{x}_2 \vee x_3) \wedge \mathrm{K}(x_2))$.

3 Diagrammes épistémiques partitionnants

Nous introduisons à présent un nouveau langage, `ESD`, permettant de représenter des formules S5 subjectives. Comme l'indiquent Bienvenu *et al.* [1, Example 15], l'expansion de Shannon ne peut être utilisée au niveau épistémique dans les formules S5, mais nous introduisons la notion de *split* (*partitionnement*), qui s'inspire de l'expansion de Shannon tout en contournant le problème. Intuitivement, un *split*

sur un atome propositionnel x divise une structure M en deux structures $M|_x$ et $M|_{\bar{x}}$. Par exemple, pour $M_1 = \{x_2x_3, \bar{x}_2\bar{x}_3\}$ et $M_2 = \{x_2x_3\}$, $\mathsf{spl}(x_1, M_1, M_2)$ représente l'union des structures $\{x_1 \,.\, m_1 \mid m_1 \in M_1\}$ et $\{\bar{x}_1 \,.\, m_2 \mid m_2 \in M_2\}$, c'est-à-dire la structure $\{x_1x_2x_3, x_1\bar{x}_2\bar{x}_3, \bar{x}_1x_2x_3\}$.

Nous adaptons les *splits* pour représenter des *ensembles* de structures ayant une forme spécifique : $\mathsf{spl}(x, \mathcal{M}, \mathcal{M}')$ représente l'ensemble de structures $\{\mathsf{spl}(x, M, M') \mid M, M' \in \mathcal{M} \times \mathcal{M}'\}$. Par exemple, si $\mathcal{M}$ contient les structures $M_1 = \{x_2x_3, \bar{x}_2x_3\}$ et $M_2 = \{x_2x_3, \bar{x}_2x_3, \bar{x}_2\bar{x}_3\}$, et si $\mathcal{M}'$ est le singleton contenant la structure $M_3 = \{x_2x_3, \bar{x}_2\bar{x}_3\}$, alors $\mathsf{spl}(x_1, \mathcal{M}, \mathcal{M}')$ représente l'ensemble qui contient les deux structures $\mathsf{spl}(x_1, M_1, M_3)$ et $\mathsf{spl}(x_1, M_2, M_3)$, à savoir l'ensemble contenant $\{x_1x_2x_3, x_1\bar{x}_2x_3, \bar{x}_1x_2x_3, \bar{x}_1\bar{x}_2\bar{x}_3\}$ et $\{x_1x_2x_3, x_1\bar{x}_2x_3, x_1\bar{x}_2\bar{x}_3, \bar{x}_1x_2x_3, \bar{x}_1\bar{x}_2\bar{x}_3\}$.

Cette notion de *split* sur les ensembles de structures constitue la base du langage `ESD`. Cependant, elle n'est pas suffisante pour représenter toute formule subjective — c'est pourquoi `ESD` utilise également le connecteur $\vee$. De plus, les modèles d'une formule S5 subjective étant des structures, c'est-à-dire des ensembles, les constantes de base sont au nombre de quatre : les habituelles $\bot$ et $\top$, respectivement satisfaites par aucune structure et par toutes les structures, et deux nouvelles, Δ, qui n'est satisfaite que par la *structure vide*, et ∇, qui est satisfaite par toutes les structures non vides. Ces deux dernières constantes sont utiles en tant qu'enfants des *splits* ; ainsi, à droite de la figure 1, l'ESD dont la racine est le nœud x_2 en bas à gauche est satisfait par toutes les structures contenant une instanciation satisfaisant $\bar{x}_2$: $\mathsf{spl}(x_2, \top, \nabla) = \{x_2 \,.\, M \cup \bar{x}_2 \,.\, M' \mid M \models \top, M' \models \nabla\} = \{x_2 \,.\, M \cup \bar{x}_2 \,.\, M' \mid M' \neq \emptyset\}$.

La définition suivante formalise tout cela.

Définition 3. Les *diagrammes épistémiques partitionnants* (ESD, *epistemic splitting diagrams*) sont définis inductivement comme suit :

- $\top$, ∇, $\bot$ et Δ sont des ESD ;
- si Φ_1 et Φ_2 sont des ESD et x est un atome propositionnel, alors $\mathsf{spl}(x, \Phi_1, \Phi_2)$ est un ESD ;
- si $\Phi_1, \ldots, \Phi_n$ sont des ESD, alors $\bigvee_{i=1}^{n} \Phi_i$ est un ESD.

La sémantique des ESD est définie comme suit : une structure M *satisfait* un ESD Φ, noté $M \models \Phi$, si l'une des conditions suivantes est remplie :

(i) Φ est Δ (resp. ∇) et M est $\emptyset$ (resp. n'est pas $\emptyset$) ;

(ii) Φ est $\mathsf{spl}(x, \Phi_1, \Phi_2)$ et M vérifie à la fois $M|_x \models \Phi_1$ et $M|_{\bar{x}} \models \Phi_2$;

(iii) Φ est $\bigvee_{i=1}^{n} \Phi_i$ et M satisfait Φ_i pour au moins un $i \in \{1, \ldots, n\}$.

Enfin, M satisfait toujours $\top$ et ne satisfait jamais $\bot$.

Rappelons que les formules de S5 sont interprétées sur l'ensemble des structures *non vides*. Cependant, pour des raisons de simplicité d'exposition, nous autorisons comme on l'a vu les ESD à avoir pour modèle la structure vide (notée $M_\emptyset$). Ce choix est sans réelle conséquence, car la conjonction $\Phi \wedge \nabla$ a exactement les mêmes modèles que Φ, $M_\emptyset$ en moins, et peut être construite efficacement (proposition 14).

La figure 1 donne deux exemples d'ESD. Dans celui de droite, le fils gauche de la racine est satisfait exactement par les structures M telles que (i) $M|_{x_1}$ est vide ou contient une instanciation satisfaisant $\bar{x}_2$, et (ii) $M|_{\bar{x}_1}$ contient une instanciation satisfaisant x_3. Comme le montre la figure, les ESD sont vus comme des DAG, à la façon des OBDD. De plus, nous considérons que les sous-graphes identiques sont systématiquement partagés ; par exemple, dans l'ESD $\mathsf{spl}(x, \Phi, \Psi) \vee \mathsf{spl}(x, \Psi, \Psi')$, Ψ n'est pas dupliqué, il s'agit d'un seul nœud avec deux arcs entrants.

La définition 3 ne restreint pas la syntaxe des ESD, mais il est parfois utile de considérer des ESD *réduits*.

Définition 4. Un ESD Φ est dit *réduit* si aucune des règles suivantes ne s'appliquent à lui :

- simplifier par $(\bot \vee \Phi) \equiv \Phi$, $(\top \vee \Phi) \equiv \top$, $\mathsf{spl}(x, \Delta, \Delta) \equiv \Delta$, $\mathsf{spl}(x, \top, \top) \equiv \top$, $\mathsf{spl}(x, \Phi, \bot) \equiv \mathsf{spl}(x, \bot, \Phi) \equiv \bot$;
- remplacer $\mathsf{spl}(x, \Phi, \Psi_1) \vee \mathsf{spl}(x, \Phi, \Psi_2)$ par $\mathsf{spl}(x, \Phi, \Psi_1 \vee \Psi_2)$, et dualement pour les fils droits ;
- supprimer les doublons parmi les enfants des nœuds $\vee$ et aplanir $(\Phi_1 \vee \cdots \vee (\Psi_1 \vee \cdots \vee \Psi_k) \vee \cdots \vee \Phi_m)$ en $(\Phi_1 \vee \cdots \vee \Psi_1 \vee \cdots \vee \Psi_k \vee \cdots \vee \Phi_m)$, $\bigvee\{\Phi\}$ en Φ, et $\bigvee \emptyset$ en $\bot$;
- remplacer $(\Phi_1 \vee \cdots \vee \Delta \vee \cdots \vee \Phi_k)$ par $(\Phi_1 \vee \cdots \vee \Phi_k)$ si l'un des Φ_i est satisfait par $M_\emptyset$;
- remplacer $(\Phi_1 \vee \cdots \vee \nabla \vee \cdots \vee \Phi_k)$ par $\top$ si l'un des Φ_i est satisfait par $M_\emptyset$, et par ∇ sinon.

On peut voir facilement que toutes ces règles conservent l'équivalence logique, et peuvent être appliquées en temps linéaire. Une autre propriété importante (obtenue par une induction structurelle simple) est que le seul ESD réduit équivalent à $\bot$ (resp. à Δ) est $\bot$ lui-même (resp. Δ lui-même). En revanche, comme pour `EDNF` et `EBDD`, il y a de multiples ESD réduits qui sont équivalents à $\top$ ou ∇. Par exemple (avec un léger abus de notation), $\mathsf{spl}(x, \mathrm{K}y, \Delta) \vee \mathsf{spl}(x, \top, \nabla) \vee \mathsf{spl}(x, \neg\mathrm{K}y, \top)$ est réduit, mais néanmoins logiquement équivalent à la formule $\mathrm{K}(x \wedge y) \vee \neg\mathrm{K}x \vee \neg\mathrm{K}(\bar{x} \vee y)$, qui s'avère tautologique.

Comme pour les OBDD, nous pouvons imposer aux ESD d'être *ordonnés*. Étant donné un ordre total $<$ sur X, un ESD est dit $<$-ordonné si les atomes propositionnels apparaissent de façon strictement croissante suivant $<$ sur tout chemin de la racine à une feuille.

Dans la suite, nous ne considérons que des ESD ordonnés et réduits, et notons ESD le langage correspondant (laissant $<$ implicite). Par exemple, les ESD de la figure 1 sont réduits et ordonnés par $x_1 < \cdots < x_n$.

Les ESD ont de nombreuses caractéristiques communes avec les OBDD ; une différence importante concerne les nœuds *redondants*. Un nœud $\mathsf{ite}(x, \varphi, \varphi)$ peut être éliminé d'un OBDD (remplacé par φ), mais on n'a pas cela dans les ESD : $\mathsf{spl}(x, \Phi, \Phi) \models \Phi$ n'est pas vérifié en général. Par exemple, pour $\Phi \equiv \mathrm{K}y \vee \mathrm{K}\bar{y}$, la structure $M = \{xy, \bar{x}\bar{y}\}$ satisfait $\mathsf{spl}(x, \Phi, \Phi)$ (puisque $M|_x = \{y\}$ satisfait $\mathrm{K}y$ et donc Φ, et $M|_{\bar{x}} = \{\bar{y}\}$ satisfait $\mathrm{K}\bar{y}$ et donc Φ), mais M ne satisfait pas Φ (puisque, contenant $\bar{x}\bar{y}$ elle falsifie $\mathrm{K}y$, et contenant xy elle falsifie $\mathrm{K}\bar{y}$). À cause de cela, pour des raisons d'efficacité, certaines transformations requièrent que les ESD (ordonnés et réduits) aient une forme spécifique.

Définition 5. Soit $<$ un ordre total sur X, avec $x_1 < \cdots < x_n$. Un ESD Φ est dit *explicitement* $<$*-ordonné* s'il est $<$-ordonné et si pour chaque chemin P de la racine à une feuille, l'ensemble des atomes apparaissant le long de P est exactement $\{x_1, x_2, \ldots, x_i\}$ pour un certain $i \in \{1, \ldots, n\}$.

Ainsi, dans l'ESD de droite sur la figure 1, le troisième chemin en partant de la gauche, $(\vee, x_1, \vee, x_2, \nabla)$, est explicitement ordonné pour $x_1 < x_2 < x_3$, mais le quatrième, $(\vee, x_1, x_3, \nabla)$, ne l'est pas (il manque x_2). Cet ESD n'est donc *pas* explicitement ordonné.

Dans toute la suite, nous considérons que tous les OBDD et ESD sont ordonnés suivant le même ordre (implicite) des atomes. En revanche, sauf mention contraire, nous ne considérons pas les ESD comme étant *explicitement* ordonnés.

4 Compilation d'atomes épistémiques en ESD

Il est assez naturel de spécifier les éléments constituant un problème de planification, comme les objectifs ou les états initiaux, sous la forme de formules de S5. Pour les manipuler sous la forme d'ESD, il est donc nécessaire de savoir *compiler* [13] les représentations épistémiques classiques vers le langage ESD. Nous montrons dans cette section comment compiler en ESD des *atomes* épistémiques, c'est-à-dire les blocs de base des représentations classiques ; la section 5 montrera comment les combiner par des opérateurs logiques.

La compilation de formules propositionnelles en OBDD étant une question bien étudiée dans la littérature [4, 14, 10], nous supposons que les atomes épistémiques nous sont donnés sous la forme $\mathrm{K}\varphi$ avec φ un OBDD qui suit l'ordre des atomes désiré pour l'ESD. La construction de l'OBDD φ est difficile, mais une fois ce travail fait, $\mathrm{K}\varphi$ peut être obtenu efficacement.

Proposition 6. *Étant donné un OBDD φ, on peut construire en temps linéaire (resp. quadratique) un ESD ordonné (resp. un ESD explicitement ordonné) logiquement équivalent à $\mathrm{K}\varphi \vee \Delta$ ou à $\neg\mathrm{K}\varphi$.*

Démonstration. Construisons un ESD Φ à partir de φ en remplaçant la feuille $\bot$ par Δ et chaque nœud $\mathsf{ite}(x, \varphi_1, \varphi_2)$ par $\mathsf{spl}(x, \Phi_1, \Phi_2)$, avec Φ_1, Φ_2 obtenus récursivement à partir de φ_1, φ_2. On peut montrer par récurrence que Φ est équivalent à $\mathrm{K}\varphi \vee \Delta$, puisque (i) remplacer $\bot$ par Δ empêche tout contre-modèle de φ d'appartenir à une structure satisfaisant Φ, et (ii) garder $\top$ permet à tout modèle de φ d'appartenir ou non à une structure satisfaisant Φ. Le résultat se déduit alors de $M \models \mathrm{K}\varphi \vee \Delta \iff M \subseteq \mathrm{Mod}(\varphi)$.

Pour $\neg\mathrm{K}\varphi$, on construit un ESD Ψ à partir de φ en remplaçant la feuille $\bot$ par ∇, $\top$ par $\bot$, et tout nœud $\mathsf{ite}(x, \varphi_1, \varphi_2)$ par $\mathsf{spl}(x, \Psi_1, \top) \vee \mathsf{spl}(x, \top, \Psi_2)$, avec Ψ_1, Ψ_2 obtenus récursivement. Une récurrence simple montre qu'une structure M satisfait Ψ exactement lorsqu'elle contient au moins un contre-modèle de φ, d'où l'on déduit $\Psi \equiv \neg\mathrm{K}\varphi$.

Remarquons que les deux constructions ne nécessitent pas que φ soit un OBDD réduit. Cela permet de faire en sorte que l'ESD obtenu soit explicitement ordonné, en appliquant à φ le pré-traitement consistant à remplacer chaque nœud $\mathsf{ite}(x_i, \psi, \cdot)$, avec $\psi = \mathsf{ite}(x_k, \varphi_1, \varphi_2)$, par $\mathsf{ite}(x_i, \mathsf{ite}(x_j, \psi, \psi), \cdot)$, autant de fois que nécessaire, c'est-à-dire tant qu'il existe un j vérifiant $x_i < x_j < x_k$ (et dualement pour l'autre fils). Clairement, ce pré-traitement ne peut augmenter la taille de φ que par un facteur d'au plus $|\mathrm{Var}(\varphi)|$. □

Qui plus est, les ESD ont l'intéressante capacité de pouvoir représenter efficacement les atomes « *only-know* ». Rappelons que $\mathrm{O}\varphi$ est satisfait par une seule structure, à savoir $\mathrm{Mod}(\varphi)$: cela permet de modéliser le fait qu'un agent sait φ, mais rien de plus [12]. Mélanger les modalités K et O nécessite des règles d'inférence spécifiques, tandis que représenter $\mathrm{O}\varphi$ en n'utilisant que des modalités K requiert une conjonction de la forme $\mathrm{K}\varphi \wedge \bigwedge_{m \not\models \varphi} \neg\mathrm{K}\neg m$, qui est de taille exponentielle en général. La possibilité de pouvoir représenter $\mathrm{O}\varphi$ de façon naturelle est donc spécifique aux ESD.

Proposition 7. *Étant donné un OBDD φ, on peut construire en temps quadratique un ESD explicitement ordonné logiquement équivalent à $\mathrm{O}\varphi$.*

Démonstration. On commence par entièrement expliciter φ : on procède comme dans la proposition 6, mais en allant cette fois jusqu'aux feuilles (par exemple, $\mathsf{ite}(x_i, \top, \cdot)$ est récursivement remplacé par

$\mathsf{ite}(x_i, \mathsf{ite}(x_j, \top, \top), \cdot)$ pour tout j vérifiant $x_j > x_i$). On remplace ensuite $\bot$ par Δ, $\top$ par ∇, et chaque nœud $\mathsf{ite}(x, \phi_1, \phi_2)$ par $\mathsf{spl}(x, \Phi_1, \Phi_2)$, avec Φ_1, Φ_2 obtenus récursivement. L'ESD résultant est satisfait exactement par $\mathrm{Mod}(\varphi)$, car Δ empêche les structures de contenir un contre-modèle de φ, tandis que ∇ les oblige à contenir tous ses modèles — la seule structure respectant toutes ces contraintes étant $\mathrm{Mod}(\varphi)$. □

En observant que toute formule S5 subjective Φ est équivalente à $\bigvee_{M \models \Phi} \mathrm{O}\varphi_M$, avec φ_M un OBDD vérifiant $\mathrm{Mod}(\varphi) = M$, la proposition précédente fournit une preuve directe de la complétude des ESD.

Proposition 8. *Le langage* ESD *est* complet *pour S5 subjectif : pour toute formule S5 subjective Φ, il existe un ESD Ψ satisfaisant $\Phi \equiv \Psi$.*

5 Requêtes et transformations

Avec EDNF, EBDD et ESD, nous disposons de trois langages pour représenter des formules S5 subjectives. Avant de comparer leur efficacité respective en termes de compacité des formules (section 6), nous présentons dans cette section leur capacité à supporter des *requêtes* (c'est-à-dire des tâches de raisonnement) et des *transformations*. Les résultats sont résumés dans la table 1, comme le veut la tradition en compilation de connaissances [5], afin de faciliter la comparaison des capacités des trois langages. La plupart des résultats concernant EDNF et EBDD dans la table 1 sont repris de Bienvenu *et al.* [1] ou en découlent directement ; nous nous focalisons ici sur le langage ESD.

Le premier résultat concerne la satisfiabilité et la vérification de modèle ; nous omettons la démonstration, qui est assez directe (pour rappel, nous avons vu en section 3 que les ESD réduits de $\bot$ et Δ sont uniques).

Proposition 9. *Étant donnée une formule Φ dans* EDNF, EBDD *ou* ESD, *on peut décider en temps polynomial si Φ est satisfaisable, et si Φ est satisfaite par une structure M donnée en extension (c'est-à-dire comme un ensemble d'instanciations propositionnelles).*

Proposition 10. *Étant donnée une formule satisfaisable Φ dans* EDNF, EBDD *ou* ESD, *on peut construire en temps polynomial une structure satisfaisant Φ (en particulier, Φ a un modèle de taille polynomiale).*

Démonstration. Pour EDNF (resp. EBDD), on choisit un terme satisfaisable $\mathrm{K}\varphi \wedge \bigwedge_i \neg\mathrm{K}\psi_i$, et on construit M en prenant un modèle de $\varphi \wedge \neg\psi_i$ pour chaque i, ce qui peut être fait en temps polynomial puisque φ est dans DNF et ψ_i dans CNF (resp. puisqu'elles sont toutes deux dans OBDD). Pour ESD, on choisit à chaque nœud $\vee$ un fils satisfaisable, et à chaque nœud $\mathsf{spl}(x, \Phi_1, \Phi_2)$ on construit M à partir des structures $M|_x$ et $M|_{\bar{x}}$ que l'on a obtenues récursivement. □

Examinons à présent les problèmes de vérification de validité et d'implication :

Proposition 11. *Étant donnée une formule Φ de* ESD, *il est* coNP*-difficile de décider si Φ est tautologique, ainsi que de décider si $\mathrm{K}\psi \models \Phi$ est vérifié pour une formule ψ de* NNF *ou* OBDD.

Démonstration. Soit $\phi = \bigwedge_{i \in I} c_i$ une CNF propositionnelle, où chaque c_i est une clause, et soit Φ l'ESD $\Delta \vee \bigvee_{i \in I} \Phi_i$, avec $\Phi_i \equiv \neg\mathrm{K}c_i$ pour tout i. Puisque Φ peut être construit en temps polynomial (prop. 6), et qu'on peut montrer que Φ est tautologique si et seulement si ϕ n'est pas satisfaisable, on obtient le premier résultat. Le second suit, en remarquant que Φ est tautologique si et seulement si $\mathrm{K}\top$ implique Φ. □

Proposition 12. *Pour tout k fixé, étant donné un ESD Φ explicitement ordonné et une disjonction Ψ d'au plus k atomes $\mathrm{K}\varphi_i$ ou $\neg\mathrm{K}\varphi_i$, où les φ_i sont des OBDD, on peut décider $\Phi \models \Psi$ en temps polynomial.*

Démonstration. Puisqu'il est polynomial de construire la négation des atomes de Ψ (prop. 6), et que la conjonction bornée et la satisfaisabilité sont polynomiaux sur ESD (prop. 14 et 9), décider si $\Phi \wedge \neg\Psi$ est insatisfaisable peut être fait en temps polynomial. □

Cependant, nous conjecturons que l'implication clausale *non bornée* (**CE**, *clausal entailment*, c'est-à-dire l'implication d'une disjonction quelconque d'atomes formés sur des OBDD) est difficile sur ESD. On peut également montrer que, comme ESD, EBDD supporte la version bornée de **CE**, en utilisant une preuve similaire à celle de la proposition 12. Notons que EDNF supporte **CE** (non bornée) pour des clauses dont les atomes positifs (resp. négatifs) sont formés sur des CNF (resp. DNF) propositionnelles [1].

Pour finir, nous considérons les combinaisons et transformations de formules.

Proposition 13. *Étant données k formules $\Phi_1, \ldots, \Phi_k$ de* ESD, *on peut construire en temps linéaire un ESD équivalent à $\bigvee_{i=1}^{k} \Phi_i$, mais il n'existe pas toujours d'ESD équivalent à $\bigwedge_{i=1}^{k} \Phi_i$ et de taille polynomiale en $\sum_{i=1}^{k} |\Phi_i|$. De plus, étant donné un ESD Φ, il n'existe pas toujours d'ESD équivalent à $\neg\Phi$ et de taille polynomiale en $|\Phi|$.*

Idée de la preuve. Le résultat est trivial pour la disjonction, l'opérateur $\vee$ étant autorisé dans ESD.

Pour la conjonction, soit Φ_i un ESD équivalent à $\mathrm{K}(x_i \leftrightarrow y_i) \vee \mathrm{K}(\neg(x_i \leftrightarrow y_i))$. Il est clair que $\sum_{i=1}^{k} |\Phi_i|$ est linéaire en k ; mais on peut montrer que le plus

Requête	ESD	EBDD	EDNF
CO	$\surd$ [9]	$\surd$ [B19]	$\surd$ [B19]
VA, **IM**	$\circ$ [11]	$\circ$ [B18]	$\circ$ [B18]
EQ, **SE**	$\circ$ [11]	$\circ$ [B18]	$\circ$ [B18]
$\mathbf{MC}_e$	$\surd$ [9]	$\surd$ [9]	$\surd$ [9]
$\mathbf{B.CE}_{\text{OBDD}}$	$\surd_E$ [12]	$\surd$ [12]	?
$\mathbf{MX}_e$	$\surd$ [10]	$\surd$ [10]	$\surd$ [10]

Transf.	ESD	EBDD	EDNF
$\vee\mathbf{C}$	$\surd$ [13]	$\surd$ [B20]	$\surd$ [B20]
$\wedge\mathbf{BC}$	$\surd_E$ [14]	$\surd$ [B20]	$\surd$ [B20]
$\wedge\mathbf{C}$	$\bullet$ [13]	$\bullet$ [B20]	$\bullet$ [B20]
$\neg\mathbf{C}$	$\bullet$ [13]	$\bullet$ [B20]	$\bullet$ [B20]
FO	$\bullet$ [15]	$\bullet$ [B21]	$\surd$ [B21]
SFO	$\surd_E$ [15]	$\surd$ [B21]	$\surd$ [B21]

TABLE 1 – Complexité des requêtes et transformations ; les noms proviennent de Bienvenu et al. [2010] et de la littérature sur la compilation de connaissances. Les symboles $\surd, \bullet, \circ$ signifient respectivement "polynomial", "non polynomial", et "non polynomial si $\mathsf{P} \neq \mathsf{NP}$" ; $\surd_E$ signifie "polynomial si la formule est explicitement ordonnée" (dans le cas contraire, la question reste ouverte). Les références entre crochets sont vers des propositions dans cet article ou dans celui de Bienvenu et al. [2010].

petit ESD équivalent à $\bigwedge_{i=1}^{k} \Phi_i$ est de taille exponentielle en k. Le résultat sur la négation se déduit des deux autres : puisque l'on a $\bigwedge_{i=1}^{k} \Phi_i \equiv \neg \bigvee_{i=1}^{k} \neg\Phi_i$, si la négation d'une formule était toujours de taille polynomiale, ce serait aussi le cas des conjonctions. □

Il est cependant possible de calculer efficacement la conjonction bornée d'ESD *explicitement ordonnés*.

Proposition 14. *Étant donnés deux ESD explicitement ordonnés Φ_1, Φ_2, on peut construire en temps quadratique un ESD explicitement ordonné équivalent à $\Phi_1 \wedge \Phi_2$.*

Idée de la preuve. On peut établir une procédure suivant les mêmes principes que l'algorithme « *apply* » sur les OBDD [3], en s'appuyant sur les deux règles : (i) $(\Psi_1 \vee \Psi_2) \wedge \Psi \equiv (\Psi_1 \wedge \Psi) \vee (\Psi_2 \wedge \Psi)$ et (ii) $\mathsf{spl}(x, \Psi_1, \Psi_2) \wedge \mathsf{spl}(x, \Psi_3, \Psi_4) \equiv \mathsf{spl}(x, \Psi_1 \wedge \Psi_3, \Psi_2 \wedge \Psi_4)$ (les ESD étant explicitement ordonnés, l'atome x est nécessairement le même des deux côtés). □

La dernière transformation que nous considérons est l'*oubli* (*forgetting*). Étant donnés un atome x et une structure M, $\mathrm{Fo}(x, M)$ est défini comme la structure $M|_x \cup M|_{\bar{x}}$. Pour une formule subjective Φ, $\mathrm{Fo}(x, \Phi)$ est une formule quelconque Ψ satisfaisant $\mathrm{Mod}(\Psi) = \{\mathrm{Fo}(x, M) \mid M \in \mathrm{Mod}(\Phi)\}$; cette définition s'étend naturellement à l'oubli d'*ensembles* d'atomes. L'oubli est une opération importante, utilisée par exemple en planification lors de la progression d'un état de croyance par une action ontique (voir section 7). Il s'avère qu'elle est polynomiale pour les ESD explicitement ordonnés, mais seulement si on ne cherche à oublier qu'un nombre *borné* d'atomes.

Proposition 15. *Étant donné un ESD Φ explicitement ordonné et un atome propositionnel x, on peut construire en temps polynomial un ESD équivalent à $\mathrm{Fo}(x, \Phi)$. Cependant, pour un* ensemble *d'atomes Y, il n'existe pas toujours d'ESD équivalent à $\mathrm{Fo}(Y, \Phi)$ et de taille polynomiale.*

Démonstration. On peut voir facilement que l'opération d'oubli est distributive sur $\vee$. Soit $\Phi = \mathsf{spl}(y, \Phi_1, \Phi_2)$. Pour $x > y$, $\mathsf{spl}(y, \mathrm{Fo}(x, \Phi_1), \mathrm{Fo}(x, \Phi_2))$ convient. Pour $x = y$, on déduit des définitions que $\mathrm{Fo}(x, \Phi)$ est équivalent à $\Psi = \Phi_1 \otimes \Phi_2$, défini par $\mathrm{Mod}(\Psi) = \{M_1 \cup M_2 \mid M_1 \models \Phi_1, M_2 \models \Phi_2\}$. Or, on peut montrer que l'opération binaire $\otimes$ peut être calculée efficacement pour des ESD explicitement ordonnés, avec un algorithme similaire à celui esquissé ci-dessus pour la conjonction binaire. Enfin, on ne peut pas avoir $x < y$ puisque Φ est explicitement ordonné.

Quand au résultat négatif, il se déduit directement du même résultat pour OBDD, en considérant le cas $\mathrm{Fo}(Y, \mathrm{K}\varphi) \equiv \mathrm{K}(\mathrm{Fo}(Y, \varphi))$. □

6 Concision des langages

Nous comparons maintenant les trois langages suivant leur concision, c'est-à-dire leur capacité à représenter des formules S5 de façon compacte.

Définition 16. Un langage $\mathtt{L}_1$ est *au moins aussi concis* qu'un autre langage $\mathtt{L}_2$, ce que l'on note $\mathtt{L}_1 \leq_s \mathtt{L}_2$, si et seulement s'il existe un polynôme P tel que pour toute formule Φ_2 de $\mathtt{L}_2$, il existe une formule équivalente Φ_1 de $\mathtt{L}_1$ telle que $|\Phi_1| \leq P(|\Phi_2|)$.

La relation de concision est un préordre ; on écrit $\mathtt{L}_1 \not\geq_s \mathtt{L}_2$ si $\mathtt{L}_1 \not\leq_s \mathtt{L}_2$ et $\mathtt{L}_1 \not\geq_s \mathtt{L}_2$ sont simultanément vérifiés, c'est-à-dire si les deux langages sont incomparables en termes de concision. La proposition suivante montre que c'est le cas pour deux couples parmi nos trois langages, lorsqu'ils sont restreints aux formules épistémiques positives (on note $\mathtt{L}^+$ le langage $\mathtt{L}$ restreint aux formules positives ; notons qu'il s'agit d'une « restriction sémantique » au sens de Fargier *et al.* [7]).

Proposition 17. *On a* $\mathtt{EDNF}^+ \not\geq_s \mathtt{EBDD}^+$ *et* $\mathtt{EDNF}^+ \not\geq_s \mathtt{ESD}^+$.

Idée de la preuve. Montrons en premier lieu $\mathtt{EBDD}^+ \not\leq_s \mathtt{EDNF}^+$. Soit P un polynôme, et soit $(\varphi_n)_n$ une fa-

mille de formules de DNF telle qu'il n'existe pas de famille $(\psi_n)_n$ d'OBDD vérifiant $\forall n, \psi_n \equiv \varphi_n$ et $|\psi_n| \leq P(|\phi_n|)$ (une telle famille existe puisque l'on a $\texttt{OBDD} \not\geq_s \texttt{DNF}$; voir Darwiche et Marquis [5]). Considérons la famille $(\mathrm{K}\varphi_n)_n$ d'atomes épistémiques, qui est aussi une famille de formules de $\texttt{EDNF}^+$. On peut montrer que la plus petite formule de EBDD représentant $\mathrm{K}\varphi_n$ a la forme $\mathrm{K}\psi_n$, avec ψ_n un OBDD équivalent à φ_n. Puisque par hypothèse ψ_n est nécessairement exponentiellement plus gros que φ_n, on a $\texttt{EBDD}^+ \not\leq_s \texttt{EDNF}^+$.

On peut montrer d'une façon similaire que l'on a $\texttt{EDNF}^+ \not\leq_s \texttt{EBDD}^+$, en utilisant des OBDD dont la représentation en DNF nécessite une taille exponentielle. Finalement, pour ESD, on peut montrer en utilisant la construction de la prop. 6 que le plus petit ESD représentant $\mathrm{K}\varphi$ ($\varphi \in \texttt{OBDD}$) a essentiellement la même taille que φ ; la preuve de $\texttt{EDNF}^+ \not\geq_s \texttt{EBDD}^+$ fonctionne donc également pour $\texttt{EDNF}^+ \not\geq_s \texttt{ESD}^+$. □

On obtient comme corollaire $\texttt{ESD} \not\geq_s \texttt{EDNF}$ (puisque si tout ESD était équivalent à une EDNF de taille polynomiale, ce serait en particulier le cas pour les formules positives, et de même pour le cas réciproque), et on retrouve $\texttt{EBDD} \not\geq_s \texttt{EDNF}$ [1, Prop. 17].

Il nous reste donc à comparer ESD et EBDD. Nous montrons qu'ils sont incomparables dans le cas général, mais que ESD est strictement plus concis pour les formules positives.

Proposition 18. *On a* $\texttt{ESD} \not\leq_s \texttt{EBDD}$.

Idée de la preuve. Soit Φ l'EBDD $\bigwedge_{i=1}^{n} \neg\mathrm{K}(\varphi_i)$, où chaque φ_i est un OBDD représentant $x \leftrightarrow x_i$; supposons que l'ordre des atomes vérifie $\forall i,\ x < x_i$. On peut montrer que le plus petit ESD équivalent à Φ est de la forme $\bigvee_{I \subseteq \{1,\ldots,n\}} \Psi_I$, avec $\Psi_I \equiv \mathsf{spl}(x, \bigwedge_{i \in I} \neg\mathrm{K}\bar{x}_i, \bigwedge_{i \notin I} \neg\mathrm{K}x_i)$, et qu'il est donc exponentiellement plus gros que Φ. □

Proposition 19. $\texttt{ESD}^+$ *est strictement plus concis que* $\texttt{EBDD}^+$ *(et on obtient en corollaire* $\texttt{EBDD} \not\geq_s \texttt{ESD}$*).*

Démonstration. Une formule de $\texttt{EBDD}^+$ est simplement une disjonction d'atomes positifs $\mathrm{K}\phi_i$, où chaque ϕ_i est un OBDD. On obtient $\texttt{ESD}^+ \leq_s \texttt{EBDD}^+$ par la proposition 6 et le fait que ESD autorise le connecteur $\vee$.

Considérons à présent la famille de formules $(\Phi_n)_n$, avec $\Phi_n = \bigwedge_{i=1}^{n}(\mathrm{K}x_i \vee \mathrm{K}\bar{x}_i)$. On peut voir assez facilement que le seul EBDD équivalent à Φ_n est $\bigvee_t(\mathrm{K}t)$, où t parcourt les 2^n termes sur $x_1, \ldots, x_n$, sous forme d'OBDD. Or, remarquons que l'ESD de la figure 1 (au milieu) est équivalent à Φ_n et est de taille linéaire en n ; cela implique $\texttt{ESD}^+ \not\geq_s \texttt{EBDD}^+$. On en déduit $\texttt{ESD} \not\geq_s \texttt{EBDD}$, et, avec la proposition 18, $\texttt{EBDD} \not\geq_s \texttt{ESD}$. □

Il est intéressant de noter que représenter la famille $(\Phi_n)_n$ utilisée dans la preuve précédente dans $\texttt{s-S5-DNF}_{\mathtt{L},\mathtt{L}'}$ nécessite un espace exponentiel pour tout choix de $\mathtt{L}, \mathtt{L}'$, puisque la preuve fonctionne au niveau épistémique. Cela montre que même le langage construit comme l'union de tous les langages $\texttt{s-S5-DNF}_{\mathtt{L},\mathtt{L}'}$ (pour tous les $\mathtt{L}, \mathtt{L}'$) n'est pas au moins aussi concis que ESD pour les formules épistémiques positives. Cette efficacité spatiale ne s'arrête cependant pas aux formules positives : on a vu dans la section 4 que ESD permet de représenter efficacement des atomes de la forme $\mathrm{O}\varphi$, alors que $\texttt{s-S5-DNF}_{\mathtt{L},\mathtt{L}'}$ ne le peut pas (quels que soient $\mathtt{L}, \mathtt{L}'$). C'est donc un exemple de formules *non* positives pour lesquelles ESD est plus concis que le langage $\texttt{s-S5-DNF}_{\mathtt{L},\mathtt{L}'}$ dans son ensemble.

Enfin, le résultat suivant, intuitivement, montre que même si transformer un ESD positif en un EBDD équivalent peut prendre une place exponentielle, ce n'est pas une opération intrinsèquement *difficile*.

Proposition 20. *Il existe un polynôme P, et un algorithme transformant toute formule Φ de* $\texttt{ESD}^+$ *en l'unique formule équivalente Ψ de* $\texttt{EBDD}^+$ *en temps borné par $P(|\Phi|, |\Psi|)$.*

Démonstration. L'algorithme consiste à « faire remonter les disjonctions » dans l'ESD, c'est-à-dire à remplacer de bas en haut chaque nœud $\mathsf{spl}(x, \Phi_1 \vee \Phi_2, \Phi_3)$ par le nœud $\mathsf{spl}(x, \Phi_1, \Phi_3) \vee \mathsf{spl}(x, \Phi_2, \Phi_3)$ (et symétriquement pour les disjonctions dans le fils droit). On peut voir facilement que le processus converge vers un ESD équivalent, dont la structure est exactement celle d'un EBDD, et donc celle de Ψ. Comme la taille de l'ESD ne fait que croître à chaque étape, la procédure est polynomiale en la taille de sa sortie. □

Ce résultat fournit une perspective intéressante sur la relation entre les deux langages : alors que la carte de compilation montre que $\texttt{EBDD}^+$ supporte des requêtes que $\texttt{ESD}^+$ ne supporte pas, cette proposition garantit que $\texttt{ESD}^+$ est en fait seulement *polynomialement* moins efficace que $\texttt{EBDD}^+$ pour ces requêtes. Pour résumer, les formules positives sont généralement plus compactes dans ESD, et si besoin on peut toujours « décompacter » la formule et retrouver un EBDD : ce ne sera pas pire que d'avoir manipulé un EBDD depuis le départ.

7 Expérimentations

Pour étudier le comportement des langages en pratique, nous avons lancé des expérimentations sur des scénarios aléatoires inspirés de la planification. Notre premier groupe d'expériences porte sur les formules positives : il consiste à lancer des *progressions hors-ligne* d'états de croyance par des actions épistémiques $test(\varphi_i)$. Pour chaque expérience, on considère m actions de la forme $test(\varphi_i)$ ($i = 1, \ldots, m$), où φ_i est un

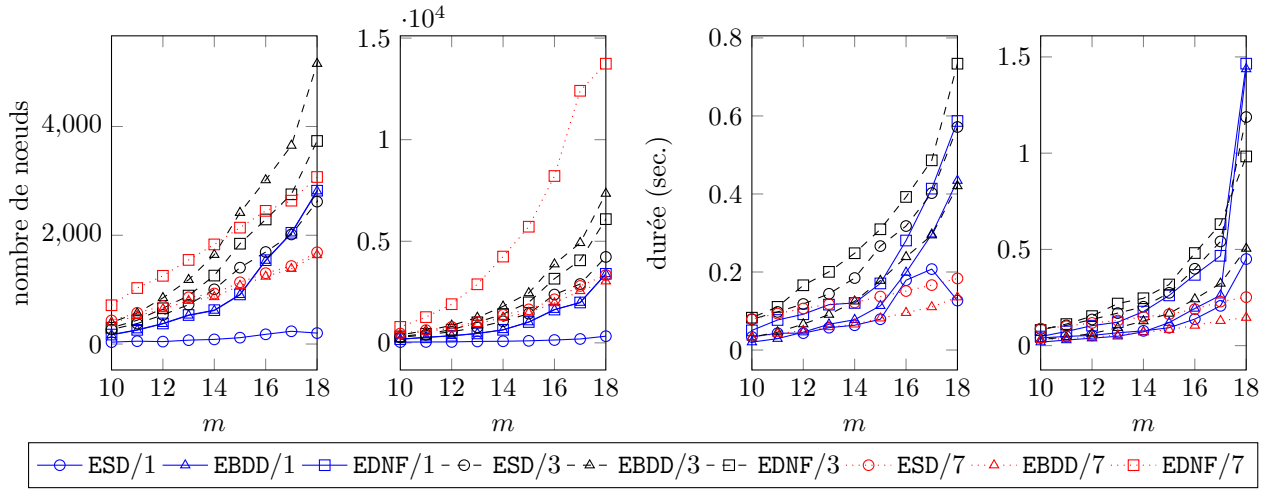


FIGURE 2 – Résultats pour les observations positives. De gauche à droite : taille pour $n = 15$; taille pour $n = 30$; temps pour $n = 15$; temps pour $n = 30$. Chaque courbe correspond à un couple langage/taille du terme. Nous ne rapportons pas le temps de calcul pour `EDNF` avec $t = 7$, qui était très mauvais.

terme satisfaisable de taille (au plus) t tiré aléatoirement [1]. Ensuite, en partant de $\Phi_0 = \top$, on construit itérativement l'ensemble d'états de croyance Φ_i, $i = 1, \ldots, m$, en progressant Φ_{i-1} par $test(\varphi_i)$, c'est-à-dire en calculant $\Phi_{i-1} \wedge (\mathrm{K}\varphi_i \vee \mathrm{K}\neg\varphi_i)$. Par exemple, pour $t = 3$, un terme possible est $x_4 \wedge \bar{x}_1 \wedge x_2$, qui induit une progression par $\mathrm{K}(x_4 \wedge \bar{x}_1 \wedge x_2) \vee \mathrm{K}(\bar{x}_4 \vee x_1 \vee \bar{x}_2)$.

Nous avons lancé des expériences pour un nombre de variables restreint et plus large ($n = 15$ et $n = 30$; pour rappel, il y a 2^{2^n} structures sur n atomes !) avec $t = 1, 3, 7$ comme tailles pour les termes, et un nombre d'actions $m = 1, \ldots, 18$. Pour chaque tuple (n, t, m), nous présentons la moyenne des résultats de 100 expériences. La figure 2 montre les courbes obtenues pour la taille de l'ensemble final d'états de croyance Φ_m et pour le temps pris à le calculer itérativement en partant de Φ_0. Il est clair que `ESD` fournit les représentations les plus compactes, en particulier pour les petits termes : lorsqu'il grossissent (par ex. pour $t = 7$), les observations $\mathrm{K}\varphi_i$ sont plus contraintes, ce qui fait rétrécir l'ensemble d'états de croyances, et masque donc la différence entre `ESD` et `EBDD` (les deux sont efficaces). D'un autre côté, on voit qu'en pratique `EDNF` ne permet *pas* d'obtenir des représentations compactes. Pour les temps de calcul, l'avantage de `ESD` sur `EBDD` et `EDNF` est moins clair ; les gains en compacité pour `ESD` sont contrebalancés par un surcoût calculatoire en pratique (qui vient notamment des opérations de réduction).

Nous avons également expérimenté l'implication : à la fin de chaque expérience nous avons testé $\Phi_m \models (\mathrm{K}x_i \vee \mathrm{K}\neg x_i)$ pour tous les atomes x_i. Les résultats (non reportés pour des raisons d'espace) montrent que les trois langages sont tous très efficaces pour cette opération, même lorsque Φ_m est gros.

Le deuxième groupe d'expériences utilise le même cadre que le premier, mais pour chaque observation $\mathrm{K}\varphi_i$ et $\mathrm{K}\neg\varphi_i$, une polarité est tirée aléatoirement : ainsi, la i-ème action pouvait par exemple donner lieu à une progression par $\mathrm{K}\varphi_i \vee \neg\mathrm{K}\neg\varphi_i$. Si ces « progressions » ne correspondent pas à des actions naturelles, elles nous permettent de tester le comportement des langages pour des opérations entre formules non nécessairement positives. Les résultats (également non reportés) montrent que pour cette application spécifique le langage le plus intéressant est `EBDD`, tant du point de vue de la concision que du temps de calcul. `EDNF` et `ESD` sont tous deux manifestement moins bons, et les EDNF tendent à être plus compactes, mais moins efficaces en temps de calcul que les ESD.

Enfin, nous avons expérimenté l'entremêlement de progressions par $test(\varphi_i)$ (avec observations positives) et de progressions par des actions *ontiques* similaires à des actions STRIPS conditionnelles, comme $\psi = (x_1 \wedge x'_2 \wedge x'_3) \vee (\bar{x}_1 \wedge x'_1 \wedge \bar{x}'_3)$, qui met x_2, x_3 à $\top$ dans les états satisfaisant x_1, et x_1 à $\top$ et x_3 à $\bot$ dans les autres états. Progresser un ensemble d'états de croyance Φ_{i-1} par une action ontique ψ consiste à construire $\Phi_{i-1} \wedge \mathrm{K}\psi$ et à oublier tous les atomes non primés dans le résultat. Nous cherchions ainsi à quantifier l'efficacité de l'oubli pour les trois langages. Les résultats montrent que cette opération est efficace pour chacun d'eux, et nous observons les mêmes tendances que dans le premier groupe d'expériences.

1. En choisissant t littéraux, uniformément et avec remise, et en ne conservant que le premier littéral pour les paires de littéraux opposés ou redondants.

8 Conclusion

Dans cette article, nous avons introduit le langage ESD des diagrammes épistémiques partitionnants pour représenter les formules S5 subjectives. Nous avons étudié ESD et les langages déjà connus $\texttt{s-S5-DNF}_{\texttt{DNF,CNF}}$ et $\texttt{s-S5-DNF}_{\texttt{OBDD,OBDD}}$ (appelés ici EDNF et EBDD), à la fois du point de vue de la compilation de connaissances et par le biais d'expérimentations sur des scénarios aléatoires inspirés de la planification contingente. À notre connaissance, ce travail constitue la première étude empirique sur l'efficacité des représentations pour S5, même s'il existe des travaux en planification traitant des problématiques spécifiques de représentation (voir par exemple Hoffmann et Brafman [9]).

Nos résultats théoriques et expérimentaux sont complémentaires. Sur les formules *positives*, ESD est plus concis que EBDD, tout en supportant à peu près les mêmes requêtes et transformations ; cela est confirmé expérimentalement. D'un autre côté, alors qu'en théorie les deux langages sont incomparables à EDNF pour la concision, on constate que les EDNF obtenues en pratique sont moins compactes. Une autre conclusion des expériences est que les calculs sont plus lourds sur ESD, ce qui peut compenser le gain en concision. Sur les formules générales, les conclusions sont différentes, puisque EBDD s'avère à la fois très concis et efficace.

Notre perspective principale est de revisiter avec des représentations efficaces, en particulier EBDD et ESD, les approches standard de planification s'appuyant (explicitement ou non) sur le raisonnement dans S5.

Références

[1] Meghyn Bienvenu, Hélène Fargier et Pierre Marquis : Knowledge compilation in the modal logic S5. *In Proc. 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[2] Blai Bonet et Hector Geffner : Belief tracking for planning with sensing : Width, complexity and approximations. *J. Artificial Intelligence Research*, 2014.

[3] Randal E. Bryant : Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[4] Randal E. Bryant : Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[5] Adnan Darwiche et Pierre Marquis : A knowledge compilation map. *J. Artificial Intelligence Research*, 17:229–264, 2002.

[6] Ronald Fagin, Joseph Y. Halpern, Yoram Moses et Moshe Y. Vardi : *Reasoning about Knowledge*. MIT Press, 1995.

[7] Hélène Fargier, Pierre Marquis et Alexandre Niveau : Towards a knowledge compilation map for heterogeneous representation languages. *In Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2013.

[8] Andreas Herzig, Jérôme Lang et Pierre Marquis : Action representation and partially observable planning using epistemic logic. *In Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1067–1072, 2003.

[9] Jörg Hoffmann et Ronen I. Brafman : Contingent planning via heuristic forward search with implicit belief states. *In Proc. 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 71–80, 2005.

[10] Jinbo Huang et Adnan Darwiche : DPLL with a trace : From SAT to knowledge compilation. *In Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 156–162, 2005.

[11] Luca Iocchi, Thomas Lukasiewicz, Daniele Nardi et Riccardo Rosati : Reasoning about actions with sensing under qualitative and probabilistic uncertainty. *In Proc. 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 818–822. IOS Press, 2004.

[12] Hector J. Levesque : All I know : A study in autoepistemic logic. *Artificial Intelligence*, 42(2-3):263–309, 1990.

[13] Pierre Marquis : Compile ! *In Proc. 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 4112–4118, 2015.

[14] Christoph Meinel et Thorsten Theobald : *Algorithms and Data Structures in VLSI Design : OBDD — Foundations and Applications*. Springer, 1998.

[15] Ronald P. A. Petrick et Fahiem Bacchus : Extending the knowledge-based approach to planning with incomplete information and sensing. *In Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 2–11, 2004.