Compilation de CSPs : carte de complexité des MDDs non-déterministes

Jérôme Amilhastre, Hélène Fargier, Alexandre Niveau, Cédric Pralet

Compilation de CSPs : carte de complexité des MDDs non-déterministes *

Jérôme Amilhastre[1] **Hélène Fargier**[2] **Alexandre Niveau**[3] **Cédric Pralet**[4]

[1] Cameleon Software, Le Galilée, 185 rue Galilée, F-31670 Labège
[2] IRIT, Université Paul Sabatier, F-31062 Toulouse Cedex 9
[3] CRIL, Université d'Artois, F-62307 Lens Cedex
[4] ONERA—The French Aerospace Lab, F-31055, Toulouse

jamilhastre@cameleon-software.com fargier@irit.fr niveau@cril.fr cpralet@onera.fr

Résumé

Les CSPs fournissent un cadre puissant pour la représentation de problèmes très divers. La difficulté est que la plupart des requêtes associées aux CSPs sont NP-difficiles, mais doivent dans certains contextes être traitées « en ligne ». C'est pour cette raison que les diagrammes de décision multivalués (MDDs) ont été proposés pour la compilation de CSPs.

Cet article dresse une carte de compilation des MDDs, dans l'esprit de la carte de la famille des NNFs de Darwiche et Marquis, en analysant les MDDs selon leur compacité et les requêtes et transformations qu'ils supportent en temps polynomial. Les MDDs déterministes et ordonnés généralisant les diagrammes de décision binaire ordonnés à des variables non-booléennes, le fait que leurs propriétés soient similaires n'est pas surprenant. Cependant, notre étude met en avant l'intérêt des MDDs ordonnés *non-déterministes* : restreint aux variables booléennes, ce fragment est strictement plus compact que ceux des OBDDs et des DNFs, et admet des performances proches de celles des DNNFs. La comparaison aux MDDs classiques montre que relâcher la contrainte du déterminisme améliore la compacité et permet à plus de transformations d'être supportées en temps polynomial. Des expériences sur des problèmes aléatoires confirment le gain en compacité.

Abstract

CSPs offer a powerful framework for representing a great variety of problems. The difficulty is that most of the requests associated with CSPs are NP-hard. As these requests must be addressed online, Multivalued Decision Diagrams (MDDs) have been proposed as a way to compile CSPs.

In the present paper, we draw a compilation map of MDDs, in the spirit of the NNF compilation map, analyzing MDDs according to their succinctness and to their polytime transformations and queries. Deterministic ordered MDDs are a generalization of ordered binary decision diagrams to non-Boolean domains : unsurprisingly, they have similar capabilities. More interestingly, our study puts forward the interest of *non-deterministic* ordered MDDs : when restricted to Boolean domains, this fragment captures OBDD and DNF as proper subsets and has performances close to those of DNNF. The comparison to classical, deterministic MDDs shows that relaxing the determinism requirement leads to an increase in succinctness and allows more transformations to be satisfied in polytime (typically, the disjunctive ones). Experiments on random problems confirm the gain in succinctness.

*Cet article est la traduction de « Compiling CSPs: A Complexity Map of (Non-Deterministic) Multivalued Decision Diagrams », paru dans les actes d'ICTAI 2012 [2]. Les travaux ont été partiellement financés par l'ANR dans le cadre du projet BR4CP (ANR-11-BS02-008).

1 Introduction

Le cadre très puissant des problèmes de satisfaction de contraintes (CSPs) permet de représenter des problèmes très variés. Un CSP peut faire l'objet de requêtes de différentes sortes, telles que la classique extraction d'une solution, mais aussi l'exigence de cohérence des domaines, l'ajout dynamique de nouvelles contraintes, le comptage de solutions, et même des combinaisons de ces requêtes. Ainsi, la résolution interactive d'un problème de configuration de produit se ramène à l'ajout successif de contraintes unaires tout en maintenant la cohérence forte.

La plupart de ces requêtes sont NP-difficiles ; elles doivent pourtant être parfois effectuées en ligne. Pour

résoudre cette contradiction, une possibilité consiste à représenter l'ensemble des solutions du CSP par un diagramme de décision multivalué (MDD) [14, 10, 5], c'est-à-dire un graphe dont chaque nœud est étiqueté par une variable et chaque arc représente l'instanciation d'une variable. Dans de tels diagrammes, chaque chemin depuis la racine jusqu'au puits représente une solution du CSP. Sur un MDD, diverses opérations — dont celles précédemment citées — peuvent être effectuées en temps polynomial en la taille du diagramme ; cette taille peut théoriquement être exponentiellement plus grande que celle du CSP d'origine, mais reste raisonnable en pratique dans beaucoup d'applications. En effet, les MDDs tirent parti de leur structure de graphe et de l'interchangeabilité (conditionnelle) des valeurs du CSP pour gagner de l'espace en fusionnant les sous-problèmes identiques. Les diagrammes de décision ont été utilisés dans des contextes variés, comme la configuration de produit [4], les systèmes de recommandation [6], ou, dans leur forme booléenne originelle, pour la planification [8, 9] et le diagnostic [13]. À notre connaissance, ces applications considèrent toujours des MDDs *déterministes*, c'est-à-dire que les étiquettes des arcs sortant d'un même nœud sont mutuellement exclusives. Cela implique que chaque solution est représentée exactement une fois dans le graphe ; cependant, cette contrainte est superflue pour beaucoup d'opérations. Il semble donc intéressant de considérer des structures *non-déterministes*.

Pour évaluer les avantages et les inconvénients du déterminisme par rapport au non-déterminisme dans les MDDs, nous proposons d'en dresser une carte de compilation, dans l'esprit de la carte des NNFs [7]. Une carte de compilation permet d'identifier les langages les plus concis qui supportent en temps polynomial les opérations nécessaires à une application donnée. Pour construire une telle carte, nous conduisons une analyse générale de la complexité des MDDs sur un ensemble d'opérations, utilisées dans des problèmes de raisonnement ou de décision.

La section suivante présente le langage `MDD` et ses sous-langages, comme ceux des MDDs déterministes ou ordonnés. Dans la section 3, nous étudions le cas des domaines booléens, pour situer `MDD` dans la carte de `NNF` : nous montrons qu'au-delà de `dOMDD`, qui correspond plus ou moins à `OBDD`, la famille de `MDD` inclut le langage des diagrammes de décision non-déterministes (`OMDD`) qui généralise à la fois `OBDD` et `DNF`. La section 4 est dédiée à la carte de `MDD`, avec une analyse de la concision des différents sous-langages de la famille de `MDD`, ainsi que l'analyse de la complexité de plusieurs requêtes et transformations. La section 5 présente ensuite quelques résultats expérimentaux sur la compacité relative des MDDs déterministes et non-déterministes. Enfin, les preuves les plus significatives sont rassemblées en annexe.

2 Diagrammes de décision multivalués

Un problème de satisfaction de contraintes $P = \langle X, C\rangle$ consiste en un ensemble fini de variables $X = \{x_1, \ldots, x_n\}$ possédant un domaine fini de valeurs, et un ensemble fini de contraintes C qui spécifient les combinaisons de valeurs autorisées pour certains sous-ensembles de variables. Pour $y \in X$, $\mathrm{Dom}(y)$ désigne le domaine de y. Pour $Y = \{y_1, \ldots, y_k\} \subseteq X$, $\mathrm{Dom}(Y)$ représente $\mathrm{Dom}(y_1) \times \cdots \times \mathrm{Dom}(y_k)$, et $\vec{y}$ désigne une Y-*instanciation* des variables de Y, c'est-à-dire $\vec{y} \in \mathrm{Dom}(Y)$. Quand $Z \cap Y = \varnothing$, $\vec{z}\,.\,\vec{y}$ est la *concaténation* de $\vec{z}$ et $\vec{y}$. Enfin, $\vec{y}|_{y_i}$ désigne la valeur assignée à y_i dans $\vec{y}$.

L'ensemble de solutions $\mathrm{Sol}(P)$ d'un CSP $P = \langle X, C\rangle$ est l'ensemble des éléments de $\mathrm{Dom}(X)$ qui satisfont toutes les contraintes dans C. On peut représenter $\mathrm{Sol}(P)$ par un diagramme de décision multivalué sur X [14, 5].

Définition 1 (MDDs)**.** Un *diagramme de décision multivalué* (MDD) sur un ensemble X de variables à domaines finis, est un graphe acyclique orienté $\varphi = \langle \mathcal{N}, \mathcal{E}\rangle$ où $\mathcal{N}$ est un ensemble de nœuds contenant au plus une racine (notée $\mathrm{Root}(\varphi)$) et au plus une feuille (appelée *puits* et notée $\mathrm{Sink}(\varphi)$).

Excepté le puits, chaque nœud $N \in \mathcal{N}$ est étiqueté par une variable $\mathrm{Var}(N)$ de X, et chaque arc $E \in \mathcal{E}$ sortant de N est étiqueté par une valeur appartenant au domaine de $\mathrm{Var}(N)$, notée [1] $\mathrm{Lbl}(E)$.

La taille de φ, notée $\|\varphi\|$, est égale au nombre de ses nœuds et arcs, plus les cardinalités des domaines des variables de X.

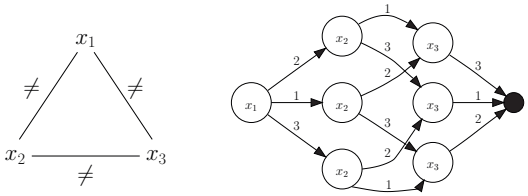

FIGURE 1 – Le problème « alldiff » de coloriage sur une clique (3 variables, 3 couleurs) et un dOMDD représentant son ensemble de solutions ($x_1 < x_2 < x_3$).

Chaque solution de P est représentée par un chemin depuis la racine du MDD jusqu'à son puits (voir par

1. On note également $\mathrm{Out}(N)$ (resp. $\mathrm{In}(N)$) l'ensemble des arcs sortants (resp. entrants) de N. Un arc $E \in \mathcal{E}$ est souvent désigné par le triplet $\langle N, N', a\rangle$ constitué par son nœud d'origine N, noté $\mathrm{Src}(E)$, son nœud de destination N', noté $\mathrm{Dest}(E)$, et sa valeur associée $a = \mathrm{Lbl}(E)$.

exemple la figure 1). Pour vérifier si une instanciation $\vec{x} = \langle a_1, \ldots, a_n \rangle \in \mathrm{Dom}(X)$ appartient à $\mathrm{Sol}(P)$, il suffit de parcourir le MDD de la racine au puits, et pour chaque nœud N étiqueté par la variable x_i, de suivre l'arc étiqueté a_i. Si un tel chemin existe, alors $\vec{x}$ appartient à $\mathrm{Sol}(P)$. Dans le cas contraire — si le puits ne peut être atteint — alors $\vec{x}$ n'est pas solution de P.

Au-delà du cadre CSP, un MDD peut représenter n'importe quel sous-ensemble de $\mathrm{Dom}(X)$ — ou, de manière équivalente, n'importe quelle fonction booléenne sur $\mathrm{Dom}(X)$.

Définition 2 (Sémantique des MDDs)**.** Un MDD φ sur X représente une fonction $\mathrm{I}(\varphi)$ de $\mathrm{Dom}(X)$ vers $\{\top, \bot\}$, appelée son *interprétation* et définie comme suit : pour toute X-instanciation $\vec{x}$, $\mathrm{I}(\varphi)(\vec{x}) = \top$ si et seulement s'il existe un chemin p de la racine au puits de φ tel que pour tout arc $E = \langle N, N', a \rangle$ sur p, $\vec{x}|_{\mathrm{Var}(N)} = a$.

On dit que $\vec{x}$ est un modèle de φ si $\mathrm{I}(\varphi)(\vec{x}) = \top$; $\mathrm{Mod}(\varphi)$ désigne l'ensemble des modèles de φ. Un MDD φ est dit *équivalent* à un autre MDD ψ (ce que l'on note $\varphi \equiv \psi$) si et seulement si $\mathrm{Mod}(\varphi) = \mathrm{Mod}(\psi)$.

Les travaux existants relatifs à la compilation de CSPs en diagrammes de décision supposent que ceux-ci sont déterministes et ordonnés [14, 5].

Définition 3 (MDD déterministe)**.** Un nœud N d'un MDD est *déterministe* si et seulement si les valeurs étiquetant les arcs sortant de N sont distinctes deux à deux. Un *MDD déterministe* (dMDD) est un MDD ne contenant que des nœuds déterministes.

Définition 4 (MDD ordonné)**.** Soit $<$ un ordre total sur X. Un MDD est dit *ordonné suivant* $<$ si et seulement si tout couple de nœuds $\langle N, M \rangle$ tel que N est un ancêtre de M vérifie $\mathrm{Var}(N) < \mathrm{Var}(M)$.

Dans les sections suivantes, nous étudions l'influence du déterminisme sur (i) la concision relative des formes compilées, et (ii) les performances de la forme compilée pour différentes opérations. Pour ce faire, six langages sont considérés.

Définition 5 (La famille de MDD)**.** On définit les langages suivants :

- MDD est le langage[2] de tous les diagrammes de décision multivalués sur X ;
- dMDD est le langage de tous les MDDs déterministes ;
- $\mathtt{OMDD}_<$ est le langage de tous les MDDs ordonnés suivant $<$ (ces MDDs pouvant être non-déterministes) ;
- OMDD est l'union de tous les langages $\mathtt{OMDD}_<$;
- $\mathtt{dOMDD}_<$ est le langage de tous les MDDs déterministes et ordonnés suivant $<$;
- dOMDD est l'union de tous les langages $\mathtt{dOMDD}_<$.

Il est clair que $\mathtt{dOMDD}_< \subseteq \mathtt{dOMDD} \subseteq \mathtt{dMDD} \subseteq \mathtt{MDD}$, que $\mathtt{dOMDD}_< \subseteq \mathtt{OMDD}_< \subseteq \mathtt{OMDD} \subseteq \mathtt{MDD}$, et que $\mathtt{dOMDD} \subseteq \mathtt{OMDD}$.

De la même manière que pour les diagrammes de décision binaires, nous considérons que les MDDs sont *réduits*, c'est-à-dire que (i) les nœuds isomorphes (étiquetés par la même variable et pointant vers les mêmes fils pour les mêmes valeurs) ont été fusionnés, (ii) les nœuds redondants (ayant un unique fils et un arc sortant par valeur dans le domaine de la variable) ont été éliminés, et (iii) les nœuds sans successeur (sauf le puits) ont été retirés. Supposer les MDDs réduits est « indolore », car la réduction peut être effectuée en temps polynomial en la taille du graphe.

Proposition 6 (Réduction)**.** *Soit* L *un langage parmi* $\{\mathtt{dOMDD}_<, \mathtt{OMDD}_<, \mathtt{dOMDD}, \mathtt{OMDD}, \mathtt{dMDD}, \mathtt{MDD}\}$*. Il existe un algorithme polynomial transformant tout* φ *de* L *en un graphe réduit équivalent* φ' *de* L *vérifiant* $\|\varphi'\| \leq \|\varphi\|$*.*

3 Les diagrammes de décision dans la carte de NNF

Pour souligner la relation entre les MDDs et la carte de compilation de NNF [7], considérons le cas booléen. Pour tout sous-langage L de MDD, on note $\mathtt{L}^{\mathfrak{B}}$ le sous-langage de L obtenu en le restreignant aux variables booléennes. Les langages $\mathtt{dMDD}^{\mathfrak{B}}$, $\mathtt{dOMDD}^{\mathfrak{B}}$ et $\mathtt{dOMDD}^{\mathfrak{B}}_<$ correspondent respectivement à BDD, OBDD et $\mathtt{OBDD}_<$, moyennant une transformation linéaire.

Définition 7. Un sous-langage $\mathtt{L}_2$ de MDD est *polynomialement traduisible* (resp. *linéairement traduisible*) en un autre sous-langage $\mathtt{L}_1$ de MDD, ce que l'on note $\mathtt{L}_1 \leq_{\mathcal{P}} \mathtt{L}_2$ (resp. $\mathtt{L}_1 \leq_{\mathcal{L}} \mathtt{L}_2$), si et seulement s'il existe un algorithme $A_{\mathtt{L}_2 \mapsto \mathtt{L}_1}$ associant en temps polynomial (resp. linéaire) à tout élément de $\mathtt{L}_2$ un élément de $\mathtt{L}_1$ équivalent.

Quand les traductions $A_{\mathtt{L}_1 \mapsto \mathtt{L}_2}$ et $A_{\mathtt{L}_2 \mapsto \mathtt{L}_1}$ sont stables (c'est-à-dire quand $A_{\mathtt{L}_1 \mapsto \mathtt{L}_2} = A^{-1}_{\mathtt{L}_2 \mapsto \mathtt{L}_1}$) et linéaires, on dit que $\mathtt{L}_1$ et $\mathtt{L}_2$ sont linéairement équivalents, et on note $\mathtt{L}_1 \equiv_{\mathcal{L}} \mathtt{L}_2$.

Définition 8 (Concision)**.** Un sous-langage $\mathtt{L}_1$ de MDD est *au moins aussi concis* qu'un autre sous-langage $\mathtt{L}_2$ de MDD (ce que l'on note $\mathtt{L}_1 \leq_s \mathtt{L}_2$) si et seulement s'il existe un polynôme $P(\cdot)$ tel que pour tout élément

2. Un *langage* est un ensemble de graphes, muni d'une fonction d'interprétation et d'une fonction de taille. Il faudrait écrire $\mathtt{MDD}_X$, mais nous omettons l'indice car il n'y a jamais d'ambiguïté sur X. Un élément du langage MDD est un MDD (noter la différence de police).

φ de $\mathtt{L}_2$, il existe un élément équivalent ψ de $\mathtt{L}_1$ qui vérifie $\|\psi\| \leq P(\|\varphi\|)$.

La relation $\leq_s$ est un préordre (partiel). On note $\sim_s$ sa partie symétrique, et $<_s$ sa partie asymétrique. Il est clair que $\mathtt{L}_1 \leq_{\mathcal{L}} \mathtt{L}_2 \Rightarrow \mathtt{L}_1 \leq_{\mathcal{P}} \mathtt{L}_2 \Rightarrow \mathtt{L}_1 \leq_s \mathtt{L}_2$.

Les langages BDD et $\mathtt{dMDD}^{\mathfrak{B}}$ sont linéairement équivalents : pour transformer un BDD en un MDD booléen déterministe, il suffit de retirer la feuille $\bot$ et d'appliquer la réduction. Pour transformer un MDD booléen déterministe en BDD, il suffit de lui adjoindre une feuille $\bot$, et d'ajouter à tout nœud n'ayant qu'un seul arc sortant E un deuxième arc sortant, pointant vers la feuille $\bot$ et étiqueté 0 si $\mathrm{Lbl}(E) = 1$, et 1 sinon.

Proposition 9. $\mathtt{dMDD}^{\mathfrak{B}} \equiv_{\mathcal{L}} \mathtt{BDD}$, $\mathtt{dOMDD}^{\mathfrak{B}} \equiv_{\mathcal{L}} \mathtt{OBDD}$ *et* $\mathtt{dOMDD}^{\mathfrak{B}}_{<} \equiv_{\mathcal{L}} \mathtt{OBDD}_{<}$.

Les diagrammes de décision multivalués capturent également des fragments hors de la famille de BDD. Ainsi, les DNFs peuvent être transformés en OMDDs en temps linéaire, bien que certaines DNFs ne puissent pas être représentés par des OMDDs de taille polynomiale. Cela implique que OMDD est strictement plus concis que DNF — on peut le voir comme un « sur-ensemble » strict de ce fragment.

Proposition 10. $\mathtt{OMDD}^{\mathfrak{B}}_{<} <_s \mathtt{DNF}$.

Enfin, $\mathtt{dOMDD}^{\mathfrak{B}} \subseteq \mathtt{d\text{-}DNNF}$ et $\mathtt{OMDD}^{\mathfrak{B}} \subseteq \mathtt{DNNF}$; de plus, $\mathtt{d\text{-}DNNF} <_s \mathtt{dOMDD}^{\mathfrak{B}}$ (puisque $\mathtt{dOMDD}^{\mathfrak{B}} \equiv_{\mathcal{L}} \mathtt{OBDD}$). La figure 2 résume nos résultats : $\mathtt{OMDD}^{\mathfrak{B}}$ apparaît comme un nouveau fragment de la carte de concision de NNF, situé en-dessous de DNNF et au-dessus de DNF et OBDD. La question de la concision relative de $\mathtt{OMDD}^{\mathfrak{B}}$ et DNNF, et plus généralement l'extension des diagrammes multivalués à des structures de type DNNF, est laissée à des travaux ultérieurs.

4 Carte de compilation de MDD

4.1 Concision

Les résultats de notre analyse de concision sont présentés dans le tableau 1 (voir aussi la figure 2 pour le cas booléen).

Proposition 11. *Les résultats du tableau 1 sont démontrés.*

Certains de ces résultats ne sont guère surprenants, et viennent directement du fait que dOMDD et $\mathtt{dOMDD}_{<}$ deviennent OBDD et $\mathtt{OBDD}_{<}$ quand les domaines sont booléens : $\mathtt{dOMDD}_{<} \not\leq_s \mathtt{dOMDD}$ est une conséquence directe de $\mathtt{OBDD}_{<} \not\leq_s \mathtt{OBDD}$. Certains autres résultats sont dérivés de la carte de NNF de manière moins immédiate. Par exemple, $\mathtt{dOMDD} \not\leq_s \mathtt{OMDD}$ (et donc $\mathtt{OMDD} <_s \mathtt{dOMDD}$) vient du fait que $\mathtt{OMDD}^{\mathfrak{B}} \leq_{\mathcal{P}} \mathtt{DNF}$ et $\mathtt{OBDD} \sim_s \mathtt{dOMDD}^{\mathfrak{B}}$: si $\mathtt{dOMDD} \leq_s \mathtt{OMDD}$ était vrai, on pourrait en déduire que $\mathtt{OBDD} \leq_s \mathtt{DNF}$, ce qui est faux [7].

L	MDD	dMDD	OMDD	dOMDD	$\mathtt{OMDD}_{<}$	$\mathtt{dOMDD}_{<}$
MDD	$\leq_s$	$\leq_s$	$\leq_s$	$\leq_s$	$\leq_s$	$\leq_s$
dMDD	?	$\leq_s$	?	$\leq_s$	?	$\leq_s$
OMDD	$\not\leq_s$	$\not\leq_s$	$\leq_s$	$\leq_s$	$\leq_s$	$\leq_s$
dOMDD	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\leq_s$	$\not\leq_s$	$\leq_s$
$\mathtt{OMDD}_{<}$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\leq_s$	$\leq_s$
$\mathtt{dOMDD}_{<}$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\leq_s$

TABLE 1 – Résultats sur la concision.

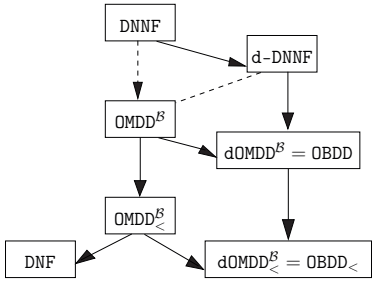

FIGURE 2 – $\mathtt{OMDD}^{\mathfrak{B}}$ et ses sous-langages dans la carte de concision de DNNF. Un arc $\mathtt{L}_1 \to \mathtt{L}_2$ indique que $\mathtt{L}_1$ est strictement plus concis que $\mathtt{L}_2$. Les arcs en pointillés indiquent des résultats incomplets.

D'autres résultats peuvent être obtenus indépendamment. Par exemple, pour prouver que $\mathtt{OMDD} \not\leq_s \mathtt{MDD}$, on peut passer par le problème de la n-coloration d'une clique de n nœuds. On peut montrer que l'ensemble de solutions S de ce problème peut être représenté par un MDD de taille polynomiale en n ; cependant, tout dOMDD ou OMDD représentant S est de taille exponentielle en n. La figure 1 donne une idée de l'explosion en espace sur une petite instance.

On montre que $\mathtt{dOMDD}_{<} \not\leq_s \mathtt{OMDD}_{<}$ en considérant un autre CSP, celui du problème de la n-coloration d'un graphe en étoile avec n nœuds (voir la figure 3 pour un exemple où $n = 3$). Soit x_1 la variable centrale, et soit l'ordre de variables $x_n < \cdots < x_2 < x_1$: le $\mathrm{dOMDD}_{<}$ représentant ce problème contient au moins 2^n nœuds et $n \cdot 2^{n-1}$ arcs, tandis qu'on peut montrer que ce CSP admet une représentation en $\mathrm{OMDD}_{<}$ (suivant le même ordre $<$) dont la taille est polynomiale en n.

4.2 Requêtes et transformations

Comme l'ont souligné Darwiche et Marquis [7], évaluer l'adéquation d'un langage-cible de compilation à une application donnée revient à comparer sa concision à l'ensemble d'opérations qu'il supporte en temps

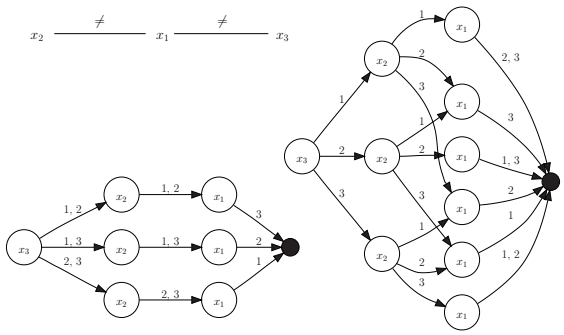

FIGURE 3 – Le problème de la coloration sur un graphe en étoile (3 variables, 3 couleurs), ainsi qu'un OMDD et un dOMDD représentant son ensemble de solutions (pour l'ordre $x_3 < x_2 < x_1$).

polynomial. Les opérations identifiées par Darwiche et Marquis sont orientées vers des applications de gestion de connaissances, mais la plupart d'entre elles ont également du sens dans certaines applications visées par le cadre CSP. Nous enrichissons cet ensemble de quelques nouvelles opérations, nécessaires pour des applications plus orientées décision.

- Les opérations les plus courantes sont celles consistant à tester la cohérence du CSP (on la note **CO**), à extraire une ou toutes les solutions (on note ces requêtes **MX** et **ME**), et à compter (**CT**) le nombre de solutions.
- La requête d'« extraction de contexte » (**CX**) vise à fournir à l'utilisateur toutes les valeurs possibles d'une variable donnée.
- L'opération de « conditionnement » (**CD**) assigne des valeurs à certaines variables ; plus généralement, la « restriction à un terme » (**TR**) restreint les valeurs possibles de certaines variables à un sous-ensemble de leur domaine. Les opérations **TR**, **CD** et **CX** sont souvent utilisées en séquence dans la résolution interactive de CSPs, au cours de laquelle l'utilisateur examine itérativement les valeurs possibles de certaines variables puis restreint leur domaine selon ses préférences [4].
- L'« implication clausale » (**CE**) provient de problèmes de raisonnement : elle consiste à déterminer si toutes les solutions d'un problème satisfont une disjonction de conditions élémentaires (contraintes unaires). Les problèmes de raisonnement en IA nécessitent également d'autres opérations, comme **VA** (la formule est-elle valide ?) et $\neg$**C** (construire la négation d'une formule).
- L'équivalence (**EQ**) et l'implication (**IM**) viennent du *model-checking*. Elles peuvent être utiles pour des problèmes de modélisation en CSP : **IM** revient à vérifier si un ensemble de contraintes est une relaxation sémantique d'un autre (c'est-à-dire, si les instanciations satisfaisant le second satisfont aussi le premier) ; **EQ** est le test d'équivalence.
- Les applications de modélisation interactive peuvent aussi nécessiter la manipulation de contraintes compilées, par exemple leur conjonction ($\wedge$**C**), leur disjonction ($\vee$**C**) ou leur négation ($\neg$**C**), de manière à construire des contraintes composées à partir de quelques opérateurs. Le processus de modélisation peut encore s'appuyer sur d'autres opérations pour vérifier la contrainte résultante, par exemple en la projetant (via **CX**) sur une de ses variables.
- L'opération d'« oubli » (**FO**) permet d'éliminer des variables (intermédiaires) du problème — elle revient à une projection existentielle. Son opération duale, l'« enforcement » (**EN**), effectue l'élimination universelle de variables. Ces deux opérations sont notamment utilisées dans la planification basée sur la compilation, comme le paradigme de *planning as model-checking* [8].

Toutes ces opérations sont souvent appliquées en séquence. Le conditionnement (**CD**) suivi d'une extraction de modèle (**MX**) peut s'avérer utile pour la décision dans l'incertain : si on dispose de la forme compilée d'une politique de décision π, associant des décisions à chaque état possible, alors **CD** permet de conditionner π suivant l'état courant, et **MX** de récupérer une décision appropriée pour cet état. Les applications de diagnostic en ligne constituent un autre exemple : une fois que le modèle (compilé) du système a été conditionné par les observations avec **CD**, **CX** peut fournir les différents modes d'échec de chaque composant. Des hypothèses d'échec plus complexes peuvent être testées via l'implication clausale (**CE**).

Plus généralement, les opérations susmentionnées peuvent être divisées en deux catégories, celle des transformations et celle des requêtes. Les transformations prennent en entrée un problème (compilé) et en retournent un autre (par exemple, conditionner un CSP par une instanciation produit un CSP avec moins de variables). Les requêtes ne modifient pas le problème, mais répondent simplement à une question (**CO**, **CT** et **CE** font ainsi partie des requêtes).

Formalisons à présent ces opérations ; on les définit comme des propriétés qu'un langage de compilation peut ou non satisfaire. Les requêtes et transformations qui sont des généralisations directes du cas booléen [7] ne sont pas définies explicitement.

Définition 12 (Requêtes)**.** Soit `L` un sous-langage de `MDD`.

L	CO	VA	MC	CE	IM	EQ	SE	MX	CX	CT	ME
MDD	○	○	√	○	○	○	○	○	○	○	○
dMDD	○	○	√	○	○	○	○	○	○	○	○
OMDD	√	○	√	√	○	○	○	√	√	○	√
dOMDD	√	√	√	√	√	√	○	√	√	√	√
$\text{OMDD}_<$	√	○	√	√	○	○	○	√	√	○	√
$\text{dOMDD}_<$	√	√	√	√	√	√	√	√	√	√	√
DNF	√	○	√	√	○	○	○	√	√	○	√
DNNF	√	○	√	√	○	○	○	√	√	○	√
d-DNNF	√	√	√	√	√	?	○	√	√	√	√
OBDD	√	√	√	√	√	√	○	√	√	√	√
$\text{OBDD}_<$	√	√	√	√	√	√	√	√	√	√	√

L	CD	TR	FO	SFO	EN	SEN	$\vee$C	$\vee$BC	$\wedge$C	$\wedge$BC	$\neg$C
MDD	√	○	○	√	○	√	√	√	√	√	?
dMDD	√	○	○	√	○	√	√	√	√	√	√
OMDD	√	√	√	√	○	○	[○]	[○]	○	○	○
dOMDD	√	•	•	•	•	•	•	○	•	○	√
$\text{OMDD}_<$	√	√	√	√	○	○	√	√	○	√	○
$\text{dOMDD}_<$	√	•	•	•	•	•	•	√	•	√	√
DNF	√	√	√	√	○	√	√	√	•	√	•
DNNF	√	√	√	√	○	○	√	√	○	○	○
d-DNNF	√	√	○	○	○	○	○	○	○	○	?
OBDD	√	•	•	√	•	√	•	○	•	○	√
$\text{OBDD}_<$	√	•	•	√	•	√	•	√	•	√	√

TABLE 2 – Résultats sur les requêtes et les transformations ; √ signifie « satisfait », • signifie « ne satisfait pas », et ○ signifie « ne satisfait pas, sauf si P = NP ». Les crochets [·] désignent une conjecture. La plupart des résultats pour DNF, DNNF, d-DNNF, OBDD et $\text{OBDD}_<$ viennent de la carte de NNF et sont donnés à titre de comparaison.

- L satisfait **CE** (resp. **IM**) si et seulement s'il existe un polynôme $P(\cdot,\cdot)$ et un algorithme associant à tout MDD φ dans L, tout ensemble de variables $\{y_1,\ldots,y_k\} \subseteq \text{Var}(\varphi)$ et toute suite $(A_1,\ldots,A_k)$ d'ensembles finis d'entiers, 1 si $\text{I}(\varphi) \models [y_1 \in A_1] \vee \cdots \vee [y_k \in A_k]$ (resp. $[y_1 \in A_1] \wedge \cdots \wedge [y_k \in A_k] \models \text{I}(\varphi)$), et 0 sinon, en un temps borné par $P(\|\varphi\|, n_A)$, où $n_A = \max_{1\leq i\leq k}|A_i|$.
- L satisfait **MC** si et seulement s'il existe un algorithme polynomial associant à tout MDD φ dans L et toute $\text{Var}(\varphi)$-instanciation $\vec{x}$, 1 si $\vec{x}$ est un modèle de φ, et 0 sinon.
- L satisfait **MX** si et seulement s'il existe un algorithme polynomial associant à tout MDD φ dans L un modèle de φ s'il en existe un, et s'arrêtant sans rien retourner dans le cas contraire.
- L satisfait **CX** si et seulement s'il existe un algorithme polynomial associant à tout φ dans L et tout $y \in \text{Var}(\varphi)$, l'ensemble de toutes les valeurs prises par y dans au moins un modèle de φ.

Avant de définir les transformations, présentons les opérations sémantiques sur lesquelles elles sont basées.

Définition 13. Soient I et J les fonctions d'interprétation, sur Var(I) et Var(J), de deux MDDs.

- La projection existentielle de I sur $Y \subseteq \text{Var(I)}$ est la fonction $\text{I}^{\downarrow Y}$ sur les variables de Y définie comme suit : $\text{I}^{\downarrow Y}(\vec{y}) = \top$ si et seulement s'il existe une Z-instanciation $\vec{z}$ (avec $Z = \text{Var(I)} \setminus Y$) telle que $\text{I}(\vec{z}\,.\,\vec{y}) = \top$. L'opération d'« oubli » est son dual : $\text{Forget}(\text{I}, Y) = \text{I}^{\downarrow \text{Var(I)}\setminus Y}$.
- La projection universelle de I sur $Y \subseteq \text{Var(I)}$ est la fonction $\text{I}^{\Downarrow Y}$ sur les variables de Y définie comme suit : $\text{I}^{\Downarrow Y}(\vec{y}) = \top$ si et seulement si pour toute Z-instanciation $\vec{z}$ (avec $Z = \text{Var(I)} \setminus Y$), $\text{I}(\vec{z}\,.\,\vec{y}) = \top$. L'opération d'« enforcement » est sa duale : $\text{Ensure}(\text{I}, Y) = \text{I}^{\Downarrow \text{Var(I)}\setminus Y}$.
- La restriction de I à J, notée $\text{I}|_{\text{J}}$, est définie par $\text{I}|_{\text{J}} = \text{Forget}(\text{I} \wedge \text{J}, \text{Var(J)})$.
- Pour une instanciation $\vec{y}$ d'un ensemble de variables $Y \subseteq \text{Var(I)}$, le conditionnement de I par $\vec{y}$ est la fonction $\text{I}|_{\vec{y}}$ sur les variables de $Z = \text{Var(I)} \setminus Y$ définie par $\text{I}|_{\vec{y}}(\vec{z}) = \text{I}(\vec{y}\,.\,\vec{z})$.

Définition 14 (Transformations)**.** Soit L un sous-langage de MDD.

- L satisfait **FO** (resp. **EN**) si et seulement s'il existe un algorithme polynomial associant à tout MDD φ dans L et tout $Y \subseteq \text{Var}(\varphi)$, un MDD φ' dans L vérifiant $\text{I}(\varphi') = \text{Forget}(\text{I}(\varphi), Y)$ (resp. $\text{I}(\varphi') = \text{Ensure}(\text{I}(\varphi), Y)$).
- L satisfait **SFO** (resp. **SEN**) si et seulement s'il satisfait **FO** (resp. **EN**) pour une seule variable (c'est-à-dire quand $|Y| = 1$).
- L satisfait **TR** si et seulement s'il existe un polynôme $P(\cdot,\cdot)$ et un algorithme associant à tout MDD φ dans L, tout ensemble de variables $\{y_1,\ldots,y_k\} \subseteq \text{Var}(\varphi)$ et toute suite $(A_1,\ldots,A_k)$ d'ensembles finis d'entiers, un MDD φ' dans L vérifiant $\text{I}(\varphi') = \text{I}(\varphi)|_{[y_1\in A_1]\wedge\cdots\wedge[y_k\in A_k]}$ en temps borné par $P(\|\varphi\|, n_A)$, où $n_A = \max_{1\leq i\leq k}|A_i|$.
- L satisfait **CD** si et seulement s'il existe un algorithme polynomial associant à tout MDD φ dans L et toute instanciation $\vec{y}$ de $Y \subseteq \text{Var}(\varphi)$, un MDD φ' dans L vérifiant $\text{I}(\varphi') = \text{I}(\varphi)|_{\vec{y}}$.

Les résultats de notre analyse de complexité de ces opérations sont présentés dans le tableau 2.

Proposition 15. *Les résultats du tableau 2 sont démontrés.*

En ce qui concerne dMDD, dOMDD et $\texttt{dOMDD}_<$, les résultats sont généralement connus ou viennent directement de travaux antérieurs [11, 10, 7]. Les dOMDDs ont presque les mêmes capacités que les OBDDs, ce qui n'est pas surprenant vu leur forte proximité. Cependant, il est notable que l'oubli et l'enforcement simples (**SFO**, **SEN**), bien que satisfaits par OBDD, ne le sont pas par dOMDD ; cela est dû au fait que les tailles des domaines ne sont pas bornées.

La proposition 15 met en avant l'attractivité des OMDDs *non-déterministes* au niveau des transformations : $\texttt{OMDD}_<$ satisfait les mêmes transformations que DNF à part **SEN**, et plus de transformations que $\texttt{dOMDD}_<$ à part la négation, tout en étant strictement plus concis que ces deux langages.

En ce qui concerne les performances sur les requêtes, OMDD et $\texttt{OMDD}_<$ sont moins intéressants que leur version déterministe ; malgré tout, on ne perd « que » **CT**, **VA**, **EQ**, **IM** et **SE** en relâchant le déterminisme (à noter que les mêmes requêtes sont perdues quand on passe de DNNF à d-DNNF). Ces langages gardent donc tout leur intérêt pour de nombreuses applications ne s'appuyant pas sur ces requêtes, comme la planification, où l'on a besoin de tester la cohérence, d'oublier des variables et d'extraire des modèles, ou la configuration interactive, qui nécessite essentiellement le conditionnement et l'extraction de contexte.

5 Résultats expérimentaux

Le tableau 3 rapporte quelques résultats [1] obtenus sur des CSPs générés aléatoirement contenant 15 variables de domaines de taille 6. Pour chaque ligne, 50 problèmes aléatoires ont été générés ; pour chaque problème, l'ordre des variables utilisé dans les formes compilées a été construit grâce à l'heuristique MCSInv [12], qui choisit itérativement la variable connectée au plus grand nombre de variables restantes dans le graphe de contraintes. Le compilateur de dOMDDs utilisé suit l'approche *bottom-up* [14] : chaque contrainte est compilée en un dOMDD, et les dOMDDs élémentaires ainsi obtenus sont alors combinés par conjonction. La compilation continue par l'application d'opérations de compactage, plus puissantes que la fusion des nœuds isomorphes, mais ne préservant pas le déterminisme [3] ; on obtient finalement un OMDD.

Il apparaît que l'intérêt des structures non-déterministes ne se limite pas à quelques problèmes bien spécifiques comme ceux utilisés dans les preuves : même sur des CSPs aléatoires, autoriser des opérations de compactage non-déterministes peut parfois diviser par 2 ou plus le nombre de nœuds dans le graphe.

%T	%C	#SOL	#N OMDD	#N dOMDD
70	10	290 888 073	80	81
	20	136 056 826	1338	1558
	30	5 006 576	5662	8132
	40	95 131	3315	5005
	50	2367	737	897
80	10	391 615 179	79	80
	20	1 581 648 506	2572	2932
	30	189 551 100	12 223	16 370
	40	11 557 737	20 501	35 486
	50	1 035 884	13 815	25 240
	60	70 185	5776	9253
	70	4662	1719	2246
	80	229	54	401

TABLE 3 – Résultats pour des CSPs binaires générés aléatoirement (15 variables, domaines de taille 6). %T désigne le pourcentage de tuples satisfaisant chaque contrainte, %C la densité du graphe de contraintes, #SOL le nombre de solutions du CSP, et #N le nombre de nœuds dans la forme compilée.

6 Conclusion

Autant les résultats théoriques de complexité que les expérimentations montrent que relâcher la contrainte du déterminisme dans les diagrammes de décision ordonnés peut améliorer la concision. Relâcher le déterminisme permet également à plus de transformations d'être traitables en temps polynomial : typiquement, toutes les transformations satisfaites par DNF le sont aussi par $\texttt{OMDD}_<$ (sauf **SEN**, qui dépend de la taille du domaine). Cela inclut l'oubli et la disjonction, qui ne sont pas satisfaites par les langages déterministes. Le prix à payer quand on s'affranchit du déterminisme est la perte de la transformation de négation et des requêtes de comptage, validité, équivalence et implication (il est à noter que les mêmes opérations sont perdues quand on relâche le déterminisme sur les DNNFs). Par conséquent, $\texttt{OMDD}_<$ est particulièrement intéressant pour les applications s'appuyant sur des transformations ($\neg$**C** exceptée) et des requêtes basiques de cohérence (**CO**, **MX**, **ME**, **CX**), comme c'est le cas en planification et en configuration interactive. D'un point de vue théorique, il est établi que, restreint à des domaines booléens, $\texttt{OMDD}_<$ est un nouveau fragment dans la carte de NNF, situé sous DNNF et au-dessus de DNF et OBDD. De plus, $\texttt{OMDD}_<$ satisfait plus de requêtes et de transformations que DNNF.

Une prochaine étape est l'introduction de nœuds « et » décomposables dans le cadre MDD. Cela permettrait de capturer les graphes AND/OR, et de définir de nouveaux fragments, tels les graphes AND/OR non-déterministes.

Références

[1] Jérôme Amilhastre. *Représentation par automate d'ensemble de solutions de problèmes de satisfaction de contraintes*. PhD thesis, Université Montpellier II, 1999.

[2] Jérôme Amilhastre, Hélène Fargier, Alexandre Niveau, and Cédric Pralet. Compiling CSPs : A complexity map of (non-deterministic) multivalued decision diagrams. In *ICTAI*, pages 1–8, 2012.

[3] Jérôme Amilhastre, Philippe Janssen, and Marie-Catherine Vilarem. FA minimisation heuristics for a class of finite languages. In *International Workshop on Implementing Automata (WIA)*, pages 1–12, 1999.

[4] Jérôme Amilhastre, Hélène Fargier, and Pierre Marquis. Consistency restoration and explanations in dynamic CSPs — application to configuration. *Artif. Intell. J.*, 135(1–2) :199–234, 2002.

[5] Henrik Reif Andersen, Tarik Hadzic, John N. Hooker, and Peter Tiedemann. A constraint store based on multivalued decision diagrams. In *CP*, pages 118–132, 2007.

[6] Hadrien Cambazard, Tarik Hadzic, and Barry O'Sullivan. Knowledge compilation for itemset mining. In *ECAI*, pages 1109–1110, 2010.

[7] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 17 :229–264, 2002.

[8] Fausto Giunchiglia and Paolo Traverso. Planning as model checking. In *ECP*, pages 1–20, 1999.

[9] Jesse Hoey, Robert St-Aubin, Alan J. Hu, and Craig Boutilier. SPUDD : Stochastic planning using decision diagrams. In *UAI*, pages 279–288, 1999.

[10] Timothy Kam, Tiziano Villa, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli. Multivalued Decision Diagrams : Theory and Applications. *Multiple-Valued Logic*, 4(1–2) :9–62, 1998.

[11] Arvind Srinivasan, Timothy Kam, Sharad Malik, and Robert K. Brayton. Algorithms for discrete function manipulation. In *ICCAD*, pages 92–95, November 1990.

[12] Robert Endre Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. on Computing*, 13(3) :566–579, 1984.

[13] Pietro Torasso and Gianluca Torta. Model-based diagnosis through OBDD compilation : A complexity analysis. In *Reasoning, Action and Interaction in AI Theories and Systems*, pages 287–305, 2006.

[14] Nageshwara Rao Vempaty. Solving constraint satisfaction problems using finite state automata. In *AAAI*, pages 453–458, 1992.

A Preuves

Cette annexe rassemble les preuves les plus significatives. Les autres sont soit directes, soit héritées du cadre des BDDs ; elles se trouvent dans l'article original en anglais [2], disponible en ligne à l'adresse `ftp://ftp.irit.fr/IRIT/ADRIA/PapersFargier/ICTAI12-AFNP-long.pdf`.

Comme c'est souvent le cas dans les cartes de compilation, de nombreuses preuves tirent parti du fait que tous les langages présentés permettent de représenter termes et clauses booléennes en temps et en espace linéaires. La preuve est similaire au cas booléen [7].

Lemme 16. *Étant donné un ordre $<$ sur les variables, tout terme et toute clause de la logique propositionelle peut être mis sous la forme d'un dOMDD$^{\mathfrak{B}}_{<}$ en temps linéaire.*

A.1 Preuves de concision (proposition 11)

• $\texttt{OMDD}_{<} \not\leq_s \texttt{dOMDD}$. On considère le CSP Π portant sur $2n$ variables booléennes partitionnées en deux ensembles $Y = \{y_1, \ldots, y_n\}$ et $Z = \{z_1, \ldots, z_n\}$, ayant pour ensemble de contraintes $C = \{y_i = z_i, 1 \leq i \leq n\}$. Soient les deux ordres $y_1 <^a z_1 <^a y_2 <^a z_2 <^a \cdots <^a y_n <^a z_n$ et $y_1 <^b y_2 <^b \cdots <^b y_n <^b z_1 <^b \cdots <^b z_n$.

Le CSP Π a une représentation en $\texttt{dOMDD}_{<^a}$ de taille polynomiale en n, que l'on note φ_a. Montrons que la taille de tout $\text{OMDD}_{<^b}$ représentant Sol(Π) est exponentielle en n.

Soit φ_b un $\text{OMDD}_{<^b}$ représentant Sol(Π). On considère un arc $E = \langle N, N', v\rangle$ de φ_b correspondant à l'instanciation d'une variable de Y (c'est-à-dire un arc dans la première moitié du graphe). Soit y_i la variable étiquetant N, et soit p le chemin $\langle N, N', v\rangle \rightsquigarrow \text{Sink}(\varphi)$. Sur ce chemin, la valeur de z_i doit être v ; par conséquent, φ_b ne peut pas contenir d'arc $\langle N'', N', u\rangle$ avec $\text{Var}(N'') = y_i$ et $u \neq v$. Pour cette raison, deux chemins de la racine à N' sont soit disjoints, soit associés à la même instanciation de $\{y_1, \ldots, y_i\}$. Chacune des instanciations possibles des variables de Y est représentée par (au moins) un chemin menant à un nœud z_1 distinct. Les instanciations des variables de Y n'étant pas contraintes, φ_b contient donc au moins 2^n nœuds au niveau z_1.

Au final, φ_a est un $\text{dOMDD}_{<^a}$ n'ayant aucune représentation en $\text{OMDD}_{<^b}$ de taille polynomiale.

• $\texttt{OMDD} \not\leq_s \texttt{dMDD}$. On considère le problème de la n-coloration d'une clique de n nœuds (ou, de manière équivalente, la contrainte « alldiff » sur n variables — voir la figure 1). Ce problème n'a aucune représentation de taille polynomiale en OMDD ; en effet, de manière similaire à la preuve précédente, on peut montrer qu'au niveau i, il doit y avoir autant de nœuds que de possibilités de choisir i valeurs dans $\{1, \ldots, n\}$ (s'il y en a moins, on peut construire un chemin sur lequel la même valeur est assignée à deux variables différentes, une fois avant le niveau i, et une fois après). Le nombre total de nœuds est donc supérieur à $\sum_{i=1}^{n} \binom{n}{i} = 2^n$.

Cependant, ce problème peut facilement être exprimé comme un dMDD de taille polynomiale, puisque chaque contrainte $x_i \neq x_j$ a une représentation en dMDD comportant $n(n-1)$ arcs, et $\texttt{dMDD}$ satisfait $\wedge\mathbf{C}$ (voir la proposition 15).

• $\texttt{dOMDD} \not\leq_s \texttt{OMDD}_<$. La proposition 10 établit que $\texttt{OMDD}_<^{\mathfrak{B}} \leq_s \texttt{DNF}$. S'il était vrai que $\texttt{dOMDD} \leq_s \texttt{OMDD}_<$, et donc que $\texttt{dOMDD}^{\mathfrak{B}} \leq_s \texttt{OMDD}_<^{\mathfrak{B}}$, alors puisque $\texttt{OBDD} \equiv_{\mathcal{L}} \texttt{dOMDD}^{\mathfrak{B}}$, on aurait $\texttt{OBDD} \leq_s \texttt{DNF}$ — ce qui est faux [7].

• Tous les autres résultats sont obtenus par transitivité et inclusion entre langages.

A.2 Preuves des requêtes (proposition 15)

• $\texttt{MDD}$ **et ses sous-langages satisfont MC.** Tous ces langages satisfaisant **CD** (voir la section A.3), on peut vérifier en temps polynomial si $\vec{x}$ est un modèle de φ en conditionnant le MDD par $\vec{x}$; en effet, on obtient soit le MDD vide (l'instanciation n'est alors pas un modèle) soit le MDD puits (alors $\vec{x}$ est un modèle). Par conséquent, tous ces langages satisfont **MC**.

• **Requêtes sur** $\texttt{dMDD}$ **et** $\texttt{MDD}$. $\texttt{BDD}$ est un sous-langage de $\texttt{dMDD}$ et $\texttt{MDD}$, donc $\texttt{dMDD}$ et $\texttt{MDD}$ ne satisfont pas les autres requêtes considérées, sauf si $\mathsf{P} = \mathsf{NP}$.

• $\texttt{OMDD}$ **et** $\texttt{OMDD}_<$ **ne satisfont pas VA, IM, EQ, SE, CT.** Le lemme 16 établit que tout terme peut être traduit dans $\texttt{OMDD}_<^{\mathfrak{B}}$ en temps linéaire, et $\texttt{OMDD}_<^{\mathfrak{B}}$ satisfait $\vee\mathbf{C}$ (proposition 15) ; ainsi $\texttt{OMDD}_<^{\mathfrak{B}} \leq_{\mathcal{L}} \texttt{DNF}$, donc $\texttt{OMDD}^{\mathfrak{B}} \leq_{\mathcal{L}} \texttt{DNF}$ (par inclusion), d'où le résultat, puisque $\texttt{DNF}$ ne satisfait pas **VA**, **IM**, **EQ**, **SE** ou **CT** sauf si $\mathsf{P} = \mathsf{NP}$.

• $\texttt{OMDD}$ **et** $\texttt{OMDD}_<$ **satisfont CO.** En effet, le seul OMDD réduit incohérent est l'OMDD vide.

• $\texttt{OMDD}$ **et** $\texttt{OMDD}_<$ **satisfont CE.** Vérifier si φ implique $[x_1 \in A_1] \vee \cdots \vee [x_k \in A_k]$ revient à vérifier si $\varphi \wedge [x_1 \notin A_1] \wedge \cdots \wedge [x_k \notin A_k]$ est incohérent. Comme (i) $[x \notin A] \equiv [x \in \mathrm{Dom}(x) \setminus A]$, (ii) un « terme » peut être représenté en temps polynomial en un OMDD suivant le même ordre de variables que φ, et (iii) $\texttt{OMDD}_<$ satisfait $\wedge\mathbf{BC}$ (voir plus loin), $\texttt{OMDD}$ satisfait **CE**.

Algorithme 1 Conditionnement d'un MDD φ par une instanciation $\vec{y}$ de $Y \subseteq \mathrm{Var}(\varphi)$.

```
Numéroter les nœuds de φ suivant < (si pertinent)
pour tout E = ⟨N, N', v⟩ tel que Var(N) ∈ Y faire
  si v ≠ y⃗|_{Var(N)} alors
    Retirer E de φ : Out(N) := Out(N) \ {E}
  sinon
    Étiqueter E avec ⊤
pour tout x_i ∈ Y, i allant de 1 à n faire
  pour tout N tel que Var(N) = x_i faire
    si N est racine alors
      Créer une nouvelle racine N' qui est la disjonction de tous les fils de N (si φ est déterministe, il n'y en a qu'un)
    sinon
      I := In(N), O := Out(N), E_new := ∅
      pour tout E_I = ⟨N_I, N, a⟩ ∈ I faire
        pour tout E_O = ⟨N, N_O, ⊤⟩ ∈ O faire
          Ajouter un arc E = ⟨N_I, N_O, a⟩ à E_new
      Retirer N, I et O du graphe
      Ajouter E_new au graphe
```

• $\texttt{OMDD}$ **et** $\texttt{OMDD}_<$ **satisfont ME, MX, CX.** Cela vient du fait que $\texttt{OMDD}$ satisfait **CD** et **CO**. Énumérer les modèles peut être fait en explorant l'arbre entier des instanciations, comme dans le cas booléen [7] (vérifier la cohérence après chaque branchement évite d'avoir à revenir en arrière — la taille de l'arbre de recherche est donc polynomiale en la taille de la sortie). L'extraction de modèle est un cas particulier de **ME**, et l'extraction de contexte peut se faire en vérifiant, pour chaque variable x_i et chaque valeur v de son domaine, si $\varphi|_{[x_i=v]}$ est cohérent.

A.3 Preuves des transformations (proposition 15)

Plusieurs preuves s'appuient sur le fait qu'un MDD peut être conditionné en temps polynomial, grâce à l'algorithme 1, qui effectue un conditionnement syntaxique de φ par une instanciation $\vec{y}$ de $Y \subseteq X$. On note $\varphi|_{\vec{y}}$ le MDD construit par cet algorithme.

• $\texttt{MDD}$ **et ses sous-langages satisfont CD.** On peut montrer que l'algorithme 1 :

- préserve le déterminisme et l'ordre des variables ;
- est en temps polynomial ;
- est tel que pour tout MDD φ, $\mathrm{I}(\varphi|_{\vec{y}}) = \mathrm{I}(\varphi)|_{\vec{y}}$.

Seule la preuve de complexité est détaillée ici ; on montre que l'algorithme 1 est en temps polynomial. En effet, la première boucle visite chaque arc au plus une fois. La seconde boucle, quant à elle, visite chaque nœud N une fois, retire $|\mathrm{In}(N)| + |\mathrm{Out}(N)|$ arcs et en

Algorithme 2 Étant donné un dMDD φ, construit un dMDD Compl(φ).

1: Soit ψ le MDD puits
2: **pour tout** nœud N de φ, ordonné du puits à la racine, en excluant le puits **faire**
3: Créer un nœud N' de même variable x que N
4: $U := \{v \in \mathrm{Dom}(x) : \forall E \in \mathrm{Out}(N), \mathrm{Lbl}(E) \neq v\}$
5: **pour tout** $v \in U$ **faire**
6: Ajouter à N' un arc sortant $\langle N', \mathrm{Sink}(\psi), v\rangle$
7: **pour tout** arc $\langle N, D, v\rangle$ sortant de N **faire**
8: **si** D correspond à un nœud D' de ψ **alors**
9: Ajouter à N' un arc sortant $\langle N', D', v\rangle$
10: **si** N' a au moins un arc sortant **alors**
11: Ajouter N' à ψ
12: **renvoyer** ψ

crée au plus $|\mathrm{In}(N)| \times |\mathrm{Out}(N)|$, étiquetés seulement avec des valeurs mentionnées dans φ : le nombre d'arcs sortants est donc borné par $|\mathrm{Edges}(\varphi)| \times |\mathrm{Nodes}(\varphi)|$ (au pire cas, il y a un arc pour chaque valeur et chaque nœud). La complexité de l'opération est donc polynomiale ; et puisqu'aucun nœud n'est créé, c'est aussi le cas pour la procédure toute entière.

• **dMDD, dOMDD et dOMDD$_<$ satisfont ¬C.** On peut montrer (on omet ici la preuve du dernier point) que l'algorithme 2 :

– préserve le déterminisme et l'ordre des variables ;
– est polynomial en $\|\varphi\|$;
– est tel que pour tout dMDD φ, Compl(φ) $\equiv \neg\varphi$.

Ordre : Cela vient du fait que (i) aucun « nouveau » nœud n'est ajouté, et (ii) les seuls arcs ajoutés pointent vers le puits.

Déterminisme : Cela vient du fait qu'aucun « nouveau » nœud n'est ajouté, et que les seuls arcs ajoutés sont disjoints avec les autres arcs qui sortent de leur nœud d'origine.

Complexité : Chaque nœud N est rencontré une fois, et pour chacun, toutes les valeurs de $\mathrm{Dom}(\mathrm{Var}(N))$ sont considérées ; on ajoute au plus un arc par valeur. La complexité temporelle et spatiale est donc en $\mathrm{O}(d \times |\mathrm{Nodes}(\varphi)|)$, où $d = \max_{x \in X} |\mathrm{Dom}(x)|$, et est donc polynomiale en $\|\varphi\|$, puisque d et $|\mathrm{Nodes}(\varphi)|$ sont tous deux bornés par $\|\varphi\|$.

• **OMDD$_<$ et OMDD ne satisfont pas ¬C.** La négation d'une CNF est une DNF, que l'on peut traduire en un OMDD suivant n'importe quel ordre en temps linéaire (proposition 10). Ainsi, si OMDD$_<$ ou OMDD satisfaisait ¬**C**, on pourrait transformer toute CNF en un OMDD équivalent en temps polynomial ; comme OMDD$_<$ et OMDD satisfont **CO**, on aurait alors un algorithme polynomial décidant si une CNF est cohérente, ce qui est impossible sauf si P = NP.

• **OMDD$_<$ satisfait ∨C.** Pour disjoindre k OMDDs $\varphi_1, \ldots, \varphi_k$ de même ordre $<$, il faut s'assurer que les racines sont étiquetées par la même variable (sinon, il suffit d'en ajouter une nouvelle, avec un arc par valeur dans le domaine, pointant vers l'ancienne racine), puis fusionner toutes les racines en une.

• **Oubli et enforcement.** Excepté les trois preuves suivantes, les résultats peuvent être adaptés du cas booléen, en utilisant par exemple une décomposition de Shannon pour **SEN** et **SFO**, et les relations entre transformations pour **CO** ou **VA**.

• **OMDD$_<$ et OMDD satisfont FO, SFO.** L'algorithme 1 peut être modifié pour effectuer un oubli syntaxique des variables de $Y \subseteq X$ dans φ au lieu d'un conditionnement : à la place des instructions des lignes 3 à 6, il suffit d'étiqueter E par $\top$. L'algorithme modifié remplace alors, pour chaque N tel que $\mathrm{Var}(N) \in Y$, tout chemin $\langle N_1, N, u\rangle \rightsquigarrow \langle N, N_2, \top\rangle$ par un raccourci $\langle N_1, N_2, u\rangle$. La preuve est similaire à celle de **CD**. Il est cependant à noter que cet algorithme ne préserve pas le déterminisme.

• **dOMDD et dOMDD$_<$ ne satisfont pas SFO, FO.** Supposons que dOMDD (resp. dOMDD$_<$) satisfasse **SFO**. On pourrait alors obtenir en temps polynomial un dOMDD (resp. un dOMDD$_<$) représentant la disjonction d'un nombre arbitraire de dOMDDs$_<$, en les joignant grâce à une nouvelle racine étiquetée par une nouvelle variable, puis en oubliant cette variable. Comme tout terme a une représentation polynomiale dans dOMDD$_<$ (lemme 16), on pourrait construire un dOMDD (resp. dOMDD$_<$) de taille polynomiale équivalent à n'importe quelle DNF. Cependant, c'est impossible, puisque OBDD $\not\leq_s$ DNF (resp. OBDD$_<$ $\not\leq_s$ DNF).

• **OMDD et OMDD$_<$ ne satisfont pas SEN.** La même technique permet de montrer que si OMDD ou OMDD$_<$ satisfaisait **SEN**, on pourrait traduire toute CNF vers un de ces langages en temps polynomial, et donc vérifier sa cohérence en temps polynomial, ce qui est impossible sauf si P = NP.

• **MDD, dMDD, dOMDD, et dOMDD$_<$ ne satisfont pas TR.** La satisfaction de **TR** implique celle de **FO** (l'oubli est une restriction à $[y_1 \in \mathrm{Dom}(y_1)] \wedge \cdots \wedge [y_k \in \mathrm{Dom}(y_k)]$), donc MDD et dMDD ne satisfont pas **TR** sauf si P = NP, et dOMDD et dOMDD$_<$ ne satisfont pas **TR**.

• **OMDD et OMDD$_<$ satisfont TR.** Tout « terme » peut être traduit en temps polynomial vers un OMDD de n'importe quel ordre, et OMDD$_<$ satisfait ∧**BC** et **FO** ; il satisfait donc **TR**. Tout OMDD étant un élément de OMDD$_<$ pour un certain $<$, OMDD satisfait également **TR**.