Vers une carte de compilation pour des langages de représentation hétérogènes *

Hélène Fargier[1] **Pierre Marquis**[2] **Alexandre Niveau**[2]

[1] IRIT, Université Paul Sabatier, F-31062 Toulouse Cedex 9
[2] CRIL, Université d'Artois, F-62307 Lens Cedex

fargier@irit.fr marquis@cril.fr niveau@cril.fr

Résumé

La carte de compilation introduite par Darwiche et Marquis s'appuie sur divers concepts (principalement ceux de requête, transformation, expressivité et concision) pour comparer la relative adéquation des langages de représentation à certains problèmes d'IA. Cependant, ce cadre est limité à la comparaison de langages interprétés de manière homogène (les formules sont interprétées comme des fonctions booléennes). Cela empêche la comparaison formelle entre des langages pourtant essentiellement proches, tels que ceux des OBDDs, MDDs et ADDs. Pour combler cette lacune, cet article présente un cadre généralisé dans lequel la comparaison formelle de langages de représentation hétérogènes devient faisable. En particulier, il explique comment les notions-clefs de requête, transformation, expressivité et concision peuvent s'adapter au formalisme généralisé.

Abstract

The knowledge compilation map introduced by Darwiche and Marquis takes advantage of a number of concepts (mainly queries, transformations, expressiveness, and succinctness) to compare the relative adequacy of representation languages to some AI problems. However, the framework is limited to the comparison of languages that are interpreted in a homogeneous way (formulæ are interpreted as Boolean functions). This prevents one from comparing, on a formal basis, languages that are close in essence, such as OBDD, MDD, and ADD. To fill the gap, we present a generalized framework into which comparing formally heterogeneous representation languages becomes feasible. In particular, we explain how the key notions of queries and transformations, expressiveness, and succinctness can be lifted to the generalized setting.

1 Introduction

Il existe des myriades de langages de représentation, ayant des capacités différentes ; chacun d'eux est adapté à certaines applications, mais pas à d'autres — il n'y a pas de « meilleur langage ». Choisir un bon langage pour une application donnée est donc un problème fondamental en IA. Levesque et Brachman [14] ont montré que ce problème revient à un compromis entre l'efficacité (calculatoire) et l'expressivité ; pour les langages de même expressivité, le compromis est entre efficacité et concision [12]. La carte de compilation [6] s'appuie sur ces aspects pour comparer des langages représentant des fonctions booléennes.

Cependant, il y a peu de travaux sur la comparaison de langages de représentation « hétérogènes », basés sur des domaines d'interprétation distincts. Il est bien connu que les diagrammes de décision multivalués ordonnés (OMDDs) [18] peuvent être transformés en temps polynomial en diagrammes de décision binaires ordonnés (OBDDs) [5], en utilisant par exemple un encodage logarithmique (voir la figure 1) ; ces deux langages sont considérés comme étant plus ou moins équivalents. Malgré tout, cette « équivalence » ne peut pas être établie de manière formelle dans les cadres existants, à cause de leur spécificité, et son utilisation dans les démonstrations requiert une grande prudence ; on aimerait pouvoir l'utiliser comme une brique élémentaire pour inférer des résultats de manière automatique. Il est à noter que lorsque la notion de « langage

*Cet article est la traduction de « Towards a Knowledge Compilation Map for Heterogeneous Representation Languages », paru dans les actes d'IJCAI 2013 [10]. Les travaux ont été partiellement financés par l'ANR dans le cadre du projet BR4CP (ANR-11-BS02-008).

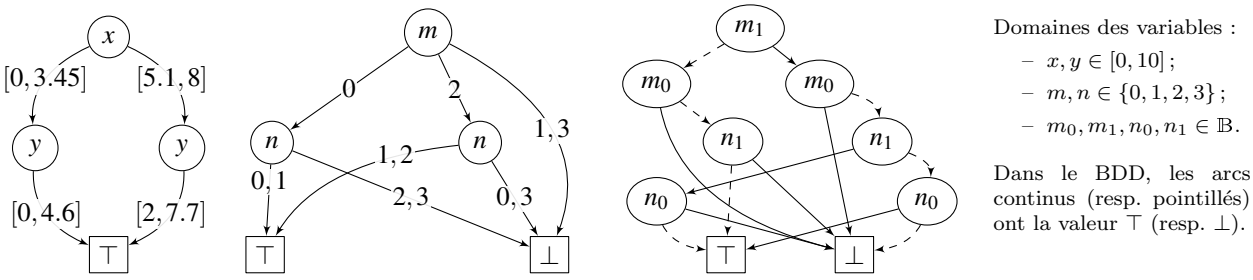

FIGURE 1 – Un automate à intervalles ordonné (OIA), un diagramme de décision multivalué ordonné (OMDD) et un diagramme de décision binaire ordonné (OBDD). L'OMDD est une discrétisation de l'OIA, via les partitions $\{[0, 3.45], (3.45, 5.1), [5.1, 8], (8, 10]\}$ pour x et $\{[0, 2), [2, 4.6], (4.6, 7.7], (7.7, 10]\}$ pour y. L'OBDD est obtenu à partir de l'OMDD par encodage logarithmique.

de représentation » est utilisée au sens large, comme dans les ouvrages de synthèse sur la représentation des connaissances ou l'IA [4, 19, 17], elle est seulement décrite de manière informelle ; mais dès lors qu'une définition formelle devient nécessaire, sa portée est limitée aux structures spécifiques dont il est question.

Cet article propose un cadre formel généralisé pour les langages de représentation, comprenant une définition de l'équivalence adaptée aux langages hétérogènes, avec l'objectif à long terme de les faire rentrer dans une unique carte de compilation. Il montre comment plusieurs propriétés classiques des cartes spécifiques peuvent être adaptées à ce cadre général. Après une définition formelle, dans la section 2, de ce qu'est un « langage de représentation » et d'importantes notions liées, les concepts usuels servant à comparer les langages dans la carte de compilation sont généralisés aux langages hétérogènes : la section 3 traite de l'efficacité représentationnelle (expressivité, concision, traduisibilité polynomiale), et la section 4 de l'efficacité calculatoire (requêtes et transformations, notamment). La section 5 conclut l'article en présentant une utilisation du cadre en pratique, sur le cas d'étude des MDDs et BDDs.

2 Langage de représentation

2.1 Définitions

La carte de compilation est construite sur la notion de *langage*, qui n'était pas définie dans l'article originel [6], ayant implicitement le sens classique de langage *formel* — un ensemble de mots sur un alphabet. Cette acception de « langage » ne se réfère qu'à la syntaxe ; la sémantique est donnée par une *fonction d'interprétation* implicite (celle des formules propositionnelles). Il y a également une hiérarchie naturelle sur les langages, celle de l'*inclusion* : un sous-langage, ou fragment, est simplement un sous-ensemble d'un langage.

Dans ce cadre classique, la plupart des aspects liés aux langages sont donc *implicites* — fonction d'interprétation et hiérarchie, mais aussi complétude et domaines des variables. Cette limitation empêche l'application directe de ce formalisme à un cadre plus général. Considérons la suite de bits 10101010 : interprétée comme la représentation en base 2 d'un entier naturel, elle correspond à 170 ; comme le codage « signe-module » d'un entier relatif, à -42 ; comme un nombre réel en virgule fixe, à 10.625 ; etc. Ces langages ont la même syntaxe, mais des fonctions d'interprétation différentes. Notre définition d'un langage de représentation développe celle de Fargier et Marquis [9], en ce qu'elle utilise une sémantique explicite ; mais la restriction aux langages représentant des fonctions booléennes est relâchée, ce qui permet de considérer des langages représentant potentiellement n'importe quels « objets ». Dans ce but, on considère un univers du discours $\mathfrak{U}$, contenant au moins tous les objets que nous voulons représenter, en particulier les nombres réels et naturels et les fonctions booléennes [1]. On utilise également un alphabet générique Σ, supposé infini dénombrable ; on appelle *formule* un mot sur cet alphabet, c'est-à-dire un élément de Σ^* [2]. Au plus haut niveau d'abstraction, un langage de représentation est une *relation* liant formules et objets de l'univers.

Définition 2.1 (Langage de représentation)**.** Un *langage de représentation* est un couple $\mathtt{L} = \langle \Phi_\mathtt{L}, \mathcal{I}_\mathtt{L} \rangle$, où $\Phi_\mathtt{L} \subseteq \Sigma^*$ est un ensemble de formules appelé *syntaxe* de $\mathtt{L}$, et $\mathcal{I}_\mathtt{L}$, appelé *sémantique* de $\mathtt{L}$, est une relation fonctionnelle $\mathcal{I}_\mathtt{L} \subseteq \Sigma^* \times \mathfrak{U}$ définie au moins sur toutes les formules de $\Phi_\mathtt{L}$.

La sémantique d'un langage est une façon d'inter-

1. On note $\mathfrak{B}$ l'ensemble des fonctions booléennes sur des variables booléennes, et $\mathfrak{B}_\mathbb{N}$ et $\mathfrak{B}_\mathbb{R}$ les ensembles des fonctions booléennes sur des variables respectivement entières et réelles.

2. On utilise l'étoile de Kleene pour noter un produit cartésien non borné : $S^* = \bigcup_{i \in \mathbb{N}} S^i$ (on considère les mots comme des tuples de symboles de Σ).

préter les symboles de Σ ; on peut la voir comme une application [3] de $\Phi_{\mathtt{L}}$ dans $\mathfrak{U}$, associant à toute formule φ dans la syntaxe de L (appelée une L-*représentation*) son interprétation $[\![\varphi]\!]_{\mathtt{L}}$. Par exemple, le langage de la logique propositionnelle peut être défini comme $\mathtt{PROP} = \langle \Phi_{\mathtt{PROP}}, \mathcal{I}_{\mathtt{PROP}} \rangle$, avec $\Phi_{\mathtt{PROP}}$ l'ensemble des formules bien formées (sur des opérateurs restreints à $\neg$, $\vee$ et $\wedge$) et $\mathcal{I}_{\mathtt{PROP}}$ la fonction d'interprétation récursive habituelle, qui associe à toute formule φ la fonction booléenne $[\![\varphi]\!]_{\mathtt{PROP}} \in \mathfrak{B}$ correspondante. Cette sémantique est utilisée dans de nombreux langages, tels que $\mathtt{CNF} = \langle \Phi_{\mathtt{CNF}}, \mathcal{I}_{\mathtt{PROP}} \rangle$ (avec $\Phi_{\mathtt{CNF}}$ l'ensemble des formules en forme normale conjonctive) ou $\mathtt{HORN\text{-}C} = \langle \Phi_{\mathtt{HORN\text{-}C}}, \mathcal{I}_{\mathtt{PROP}} \rangle$, (avec $\Phi_{\mathtt{HORN\text{-}C}}$ l'ensemble des Horn-CNFs). Citons également BDD, MDD et IA, les langages des diagrammes de décision binaires et multivalués et des automates à intervalles [15], leurs versions ordonnées OBDD, OMDD et OIA (voir la figure 1), et le langage des diagrammes de décision algébriques, ADD [2]. Pour des raisons de concision, l'article se focalise sur ces langages (en particulier, la relation entre les familles de MDD et de BDD est présentée en section 5), mais le cadre est beaucoup plus général [4].

Remarquons qu'il peut y avoir des éléments $\omega \in \mathfrak{U}$ que $\mathcal{I}_{\mathtt{L}}$ n'associe à *aucune* formule, qu'elle soit ou non dans $\Phi_{\mathtt{L}}$. Ces éléments sont ainsi complètement indépendants du langage ; par exemple, $\mathcal{I}_{\mathtt{PROP}}$ ignore tout objet qui n'est pas une fonction booléenne. Les éléments de $\mathfrak{U}$ que $\mathcal{I}_{\mathtt{L}}$ associe à au moins une formule sont appelés L-*interprétations*.

Définition 2.2 (Espace d'interprétation)**.** L'*espace d'interprétation* d'un langage de représentation L, noté $\Omega_{\mathtt{L}}$, est le codomaine de $\mathcal{I}_{\mathtt{L}}$, c'est-à-dire que $\Omega_{\mathtt{L}} = \{\, \omega \in \mathfrak{U} \mid \exists \varphi \in \Sigma^*, \varphi\, \mathcal{I}_{\mathtt{L}}\, \omega \,\}$.

Puisque CNF et HORN-C ont la même sémantique, ils ont aussi le même espace d'interprétation, en l'occurrence $\mathfrak{B}$. Il est à noter qu'il n'y a aucune garantie que tout élément de l'espace d'interprétation d'un langage L ait une *représentation* dans $\Phi_{\mathtt{L}}$; ainsi, la syntaxe de HORN-C n'est *pas assez expressive* pour couvrir toutes les capacités d'interprétation de sa sémantique — le langage est *incomplet*.

3. La définition 2.1 utilise une relation fonctionnelle (c'est-à-dire *many-to-one*) plutôt qu'une fonction, pour simplifier les notations ultérieures. Notons que nous adoptons cette hypothèse de *many-to-one* pour simplifier le propos, mais que le cadre pourrait être adapté pour prendre en compte des sémantiques ambiguës (qui peuvent être utiles, un bon exemple étant les langages naturels).

4. Il inclut notamment VNNF [7] et des langages hors de la portée habituelle de la compilation de connaissances, comme les différentes sortes de réseaux de contraintes (discrets ou continus, pondérés, flous, etc.) [16], les *quadtrees* [11], les $R^\star$*-trees* [3], et avec des espaces d'interprétation appropriés, les logiques d'ordre supérieur, les ontologies, les langages de programmation, etc.

Définition 2.3 (Complétude)**.** Un langage L est *complet* si et seulement si $\forall \omega \in \Omega_{\mathtt{L}}, \exists \varphi \in \Phi_{\mathtt{L}}, [\![\varphi]\!]_{\mathtt{L}} = \omega$.

2.2 Sous-langages

Un langage de représentation n'étant plus simplement un ensemble de formules, la notion de *sous-langage* ne peut plus être basée uniquement sur une restriction de syntaxe, comme dans la carte de compilation classique.

Définition 2.4 (Sous-langage)**.** Soient L et L′ deux langages de représentation ; L′ est un *sous-langage* de L, ce que l'on note $\mathtt{L}' \subseteq \mathtt{L}$, si et seulement si $\Phi_{\mathtt{L}'} \subseteq \Phi_{\mathtt{L}}$ et $\mathcal{I}_{\mathtt{L}'} \subseteq \mathcal{I}_{\mathtt{L}}$. De plus, si $\mathtt{L}' \subseteq \mathtt{L}$ et $\mathcal{I}_{\mathtt{L}'} = \mathcal{I}_{\mathtt{L}}$, L′ est appelé *fragment* de L.

La condition selon laquelle les formules du sous-langage doivent respecter la syntaxe du langage parent est toujours valable ; mais elles doivent également respecter sa sémantique (ce qui était implicite dans le cadre classique). Remarquons que la définition d'un langage de représentation permet de distinguer deux sortes de hiérarchies sur les langages, celle des *fragments*, qui classe des langages ayant une même sémantique, et celle plus générale des *sous-langages*, dans laquelle même des langages hétérogènes peuvent être comparés : ainsi, $\mathtt{OBDD} \subseteq \mathtt{OMDD} \subseteq \mathtt{OIA}$, puisque les OBDDs sont des OMDDs particuliers (restreints aux variables booléennes), et que les OMDDs sont des OIAs particuliers (restreints aux variables discrètes). Cette propriété naturelle ne peut pas être établie formellement dans la carte de compilation classique.

Il est intéressant de noter que, contrairement à l'habitude, l'incomplétude n'est pas héritée par les sous-langages. Ainsi, OIA n'est pas complet (les étiquettes des arcs sont restreints à des intervalles fermés), mais OBDD l'est, alors que $\mathtt{OBDD} \subseteq \mathtt{OIA}$. Cela vient du fait que la complétude est relative à l'espace d'interprétation, qui n'est pas nécessairement partagé par un sous-langage. La propriété attendue est en fait vérifiée sur les fragments, dont la définition est plus restrictive.

Proposition 2.5. *Soit* L *un langage, et* L′ *un fragment de* L*. Si* L′ *est complet, alors* L *l'est aussi.*

Démonstration. Le fait que L′ soit complet signifie que toute interprétation dans $\Omega_{\mathtt{L}'}$ a une L′-représentation. Par définition d'un fragment, l'espace d'interprétation de L est le même que celui de L′, et toute L′-représentation est aussi une L-représentation. Conséquemment, tout élément dans l'espace d'interprétation de L a une L-représentation. □

2.3 Opérations sur les langages

Dans la carte de compilation classique, la hiérarchie des fragments est induite par un ensemble de propriétés syntaxiques sur les langages : en ne considérant que

les BDDs *read-once*, on obtient le langage FBDD, en ajoutant la contrainte d'ordonnancement, on obtient OBDD, etc. On peut voir ces propriétés comme des *restrictions syntaxiques* ; elles peuvent réduire l'expressivité, puisque l'espace d'interprétation ne change pas. D'un autre côté, la hiérarchie des diagrammes de décision au sens large (OBDD $\subseteq$ OMDD) ne peut être générée en se contentant de restreindre la syntaxe, des interprétations devant également être retirées : on ne peut la construire que via des *restrictions sémantiques*.

Définition 2.6 (Restrictions de langages)**.** La *restriction (syntaxique)* d'un langage $\mathtt{L}$ à un ensemble de formules $\Phi \subseteq \Sigma^*$ est le fragment $\mathtt{L}|_\Phi = \langle \Phi_\mathtt{L} \cap \Phi, \mathcal{I}_\mathtt{L} \rangle$.

La *restriction (sémantique)* d'un langage $\mathtt{L}$ à un espace d'interprétation $\Omega \subseteq \mathfrak{U}$ est le sous-langage $\mathtt{L}|^\Omega = \langle \{ \varphi \in \Phi_\mathtt{L} \mid [\![\varphi]\!]_\mathtt{L} \in \Omega \}, \mathcal{I}_\mathtt{L} \cap (\Sigma^* \times \Omega) \rangle$.

Ainsi, HORN-C peut être défini comme la restriction syntaxique de CNF aux Horn-CNFs, et BDD comme la restriction sémantique de MDD aux fonctions sur des variables booléennes. Une autre façon de construire un langage est d'en *combiner* d'autres.

Définition 2.7 (Union et intersection)**.** L'*union* de deux languages $\mathtt{L}$ et $\mathtt{L}'$ est $\mathtt{L} \cup \mathtt{L}' = \langle \Phi_\mathtt{L} \cup \Phi_{\mathtt{L}'}, \mathcal{I} \cup \mathcal{I}' \rangle$, et leur *intersection* est $\mathtt{L} \cap \mathtt{L}' = \langle \Phi_\mathtt{L} \cap \Phi_{\mathtt{L}'}, \mathcal{I} \cap \mathcal{I}' \rangle$.

On peut alors écrire BDD = MDD$\cap$ADD, pour exprimer le fait que les BDDs sont à la fois les MDDs sur des variables booléennes et les ADDs ayant deux feuilles.

Il est à présent possible de définir les concepts de la carte de compilation, d'une part l'expressivité, la concision et la traduisibilité polynomiale, et d'autre part la satisfaction de requêtes et transformations.

3 Efficacité représentationnelle

Dans la carte de compilation classique, il est difficile de comparer des langages d'expressivité différente, la concision étant fortement liée à l'expressivité ; une expressivité moindre implique une concision moindre par définition. Les langages sur des espaces d'interprétation différents ne sont *a fortiori* pas comparés dans la carte. Il existe cependant des travaux portant sur la traduction entre langages hétérogènes [20, 13], et il serait intéressant d'incorporer les résultats connus dans la carte. Ceci appelle une définition plus générale de l'expressivité, de la concision et de la traduisibilité polynomiale, qui ne requerrait pas l'égalité des interprétations, mais seulement leur équivalence modulo une *correspondance sémantique*, c'est-à-dire une relation liant deux espaces d'interprétation hétérogènes — induisant une traduction entre les formules.

Définition 3.1 (Correspondance sémantique, traduction)**.** Une *correspondance sémantique* entre deux espaces d'interprétation Ω_1 et Ω_2 est un sous-ensemble de $\Omega_1 \times \Omega_2$. On appelle *traduction de* $\mathtt{L}_1$ *vers* $\mathtt{L}_2$ une correspondance sémantique $\mathcal{T} \subseteq \Omega_{\mathtt{L}_1} \times \Omega_{\mathtt{L}_2}$.

Parmi les traductions bien connues, citons les divers encodages de réseaux de contraintes discrets en formules propositionnelles [20], comme les encodages direct $\mathcal{T}_{\text{dir}}$ et logarithmique $\mathcal{T}_{\log}$, qui associent à toute fonction dans $\mathfrak{B}_\mathbb{N}$ une fonction dans $\mathfrak{B}$; ou encore la discrétisation $\mathcal{T}_{\text{discr}}$, liant un réseau de contraintes continu à un réseau de contraintes discret correspondant. On utilise également la traduction générique d'identité Id, qui désignera abusivement toute relation de type $\{\langle \omega, \omega \rangle : \omega \in \Omega\}$ avec $\Omega \subseteq \mathfrak{U}$.

3.1 Expressivité et concision

Un premier critère pour la comparaison de langages de représentation est leur expressivité relative [12, 8]. Quand ils ont la même sémantique, l'expressivité évalue leur capacité à représenter des objets. Cependant, puisque l'on cherche à comparer des langages ayant des sémantiques différentes, l'expressivité est définie comme dépendant d'une correspondance sémantique. C'est également le cas pour la concision relative [12, 6], qui évalue la capacité des langages à représenter les objets *de manière compacte*.

Définition 3.2 (Expressivité, concision)**.** Soient $\mathtt{L}_1$ et $\mathtt{L}_2$ deux langages, et $\mathcal{T}$ une traduction de $\mathtt{L}_1$ vers $\mathtt{L}_2$.

$\mathtt{L}_2$ est *au moins aussi expressif* que $\mathtt{L}_1$ modulo $\mathcal{T}$, ce que l'on note [5] $\mathtt{L}_1 \geq_e^\mathcal{T} \mathtt{L}_2$, si et seulement si pour toute $\mathtt{L}_1$-représentation φ_1, il existe une $\mathtt{L}_2$-représentation φ_2 telle que $[\![\varphi_1]\!]_{\mathtt{L}_1} \; \mathcal{T} \; [\![\varphi_2]\!]_{\mathtt{L}_2}$.

$\mathtt{L}_2$ est *au moins aussi concis* que $\mathtt{L}_1$ modulo $\mathcal{T}$, ce que l'on note $\mathtt{L}_1 \geq_s^\mathcal{T} \mathtt{L}_2$, si et seulement s'il existe un polynôme $P(\cdot)$ tel que pour toute $\mathtt{L}_1$-représentation φ_1, il existe une $\mathtt{L}_2$-représentation φ_2 telle que $|\varphi_2| \leq P(|\varphi_1|)$ et $[\![\varphi_1]\!]_{\mathtt{L}_1} \; \mathcal{T} \; [\![\varphi_2]\!]_{\mathtt{L}_2}$.

Pour qu'une traduction soit adaptée à la comparaison entre deux langages, il est nécessaire qu'elle les rende comparables sur le plan de l'expressivité. La condition n'est manifestement pas suffisante : pour deux langages $\mathtt{L}_1$ et $\mathtt{L}_2$, la correspondance sémantique triviale $\mathcal{T} = \Omega_{\mathtt{L}_1} \times \Omega_{\mathtt{L}_2}$ vérifie $\mathtt{L}_1 \geq_e^\mathcal{T} \mathtt{L}_2$, mais cela ne permet d'inférer aucun résultat sur $\mathtt{L}_1$ et $\mathtt{L}_2$. Des conditions suffisantes seront étudiées en section 4.

5. On utilisera également les notations suivantes : (i) $\mathtt{L}_1 \leq_e^\mathcal{T} \mathtt{L}_2$ signifie $\mathtt{L}_2 \geq_e^{\mathcal{T}^{-1}} \mathtt{L}_1$, (ii) $\mathtt{L}_1 \sim_e^\mathcal{T} \mathtt{L}_2$ signifie que $\mathtt{L}_1 \geq_e^\mathcal{T} \mathtt{L}_2$ et $\mathtt{L}_1 \leq_e^\mathcal{T} \mathtt{L}_2$, et (iii) $\mathtt{L}_1 >_e^\mathcal{T} \mathtt{L}_2$ signifie que $\mathtt{L}_1 \geq_e^\mathcal{T} \mathtt{L}_2$ mais que $\mathtt{L}_1 \not\leq_e^\mathcal{T} \mathtt{L}_2$. Ces notations s'appliquent, *mutatis mutandis*, à la concision et la traduisibilité polynomiale.

Comme dans le cadre classique de la compilation de connaissances, la concision est un raffinement de l'expressivité : $\mathtt{L} \geq_s^{\mathcal{T}} \mathtt{L}' \implies \mathtt{L} \geq_e^{\mathcal{T}} \mathtt{L}'$. Cependant, alors que cela implique que la relation classique de concision, qui correspond à $\geq_s^{\mathrm{Id}}$, n'est pas très informative quand on l'applique à des langages hétérogènes, notre généralisation ne souffre pas de ce défaut : il est possible d'écrire formellement que $\mathtt{MDD} \not\geq_s^{\mathcal{T}_{\mathrm{dir}}} \mathtt{CNF}$, par exemple. Cela permet notamment d'appliquer les notations de la compilation de connaissances à n'importe quel langage de représentation : par exemple, si l'on note $\mathtt{RADIX}_n$ le langage des entiers naturels représentés en base n, il est bien connu que $\mathtt{RADIX}_1 >_s^{\mathrm{Id}} \mathtt{RADIX}_2$, et que pour tout $n > 1$, $\mathtt{RADIX}_n \sim_s^{\mathrm{Id}} \mathtt{RADIX}_2$.

Qui plus est, on peut utiliser des traductions spécifiques pour comparer la concision de langages qui, bien que portant sur le même espace d'interprétation, sont d'expressivité différente. Par exemple, les langages HORN-C et $\mathtt{KROM\text{-}C} = \langle \Phi_{\mathtt{KROM\text{-}C}}, \mathcal{I}_{\mathtt{PROP}} \rangle$, où $\Phi_{\mathtt{KROM\text{-}C}}$ est l'ensemble des formules en 2-CNF, sont incomparables selon $\geq_e^{\mathrm{Id}}$, et donc selon la concision classique $\geq_s^{\mathrm{Id}}$; il peut pourtant être intéressant de savoir lequel est le plus concis si l'on se restreint aux fonctions booléennes que tous deux peuvent représenter. Il est possible d'exprimer cela dans le cadre généralisé, en définissant une correspondance sémantique ad hoc $\mathcal{T}$, pour laquelle $\mathtt{HORN\text{-}C} \geq_s^{\mathcal{T}} \mathtt{KROM\text{-}C}$ est vrai si et seulement si toute HORN-C-représentation est soit non représentable dans KROM-C, soit représentable en espace polynomial [6].

3.2 Traduisibilité polynomiale

La concision requiert l'existence d'une traduction de taille polynomiale, mais pas l'existence d'un algorithme construisant cette traduction, sans même parler de son efficacité. Le dernier raffinement de l'expressivité est la traduisibilité polynomiale [9].

Définition 3.3 (Traduisibilité polynomiale)**.** Soient $\mathtt{L}_1$ et $\mathtt{L}_2$ deux langages de représentation, et $\mathcal{T}$ une traduction de $\mathtt{L}_1$ vers $\mathtt{L}_2$; $\mathtt{L}_1$ est *polynomialement traduisible* vers $\mathtt{L}_2$ modulo $\mathcal{T}$, ce que l'on note $\mathtt{L}_1 \geq_p^{\mathcal{T}} \mathtt{L}_2$, si et seulement s'il existe un algorithme associant en temps polynomial à toute $\mathtt{L}_1$-représentation φ_1 une $\mathtt{L}_2$-représentation φ_2 telle que $[\![\varphi_1]\!]_{\mathtt{L}_1} \; \mathcal{T} \; [\![\varphi_2]\!]_{\mathtt{L}_2}$. De plus, si la sortie est garantie d'être *au plus* polynomialement plus *petite* que l'entrée, c'est-à-dire, s'il existe un polynôme $P(\cdot)$ tel que pour tout φ_1, toutes les sorties φ_2 vérifient $|\varphi_1| \leq P(|\varphi_2|)$, alors la traduction est dite *stable*, et l'on note $\mathtt{L}_1 \geq_{p\sim}^{\mathcal{T}} \mathtt{L}_2$.

Il vient, par exemple, que $\mathtt{OMDD} \geq_{p\sim}^{\mathcal{T}_{\mathrm{dir}}} \mathtt{OBDD}$ et $\mathtt{OMDD} \geq_{p\sim}^{\mathcal{T}_{\log}} \mathtt{OBDD}$ [18] (par exemple, en figure 1, l'OBDD résulte d'un encodage logarithmique de l'OMDD). La condition de stabilité n'est généralement pas difficile à remplir ; elle permet notamment d'écarter les cas particuliers où la formule traduite est exponentiellement plus petite que l'originale.

Dans le cadre classique de la compilation, la traduisibilité polynomiale correspond à $\geq_p^{\mathrm{Id}}$, et a des conséquences importantes sur la satisfaction de requêtes et transformations. Par exemple, en notant MODS la restriction de PROP aux DNFs *smooth* et déterministes [6], le fait que $\mathtt{MODS} \geq_p^{\mathrm{Id}} \mathtt{OBDD}$ (c'est-à-dire que les MODS-représentations sont transformables en des OBDDs équivalents en temps polynomial) implique que MODS satisfait toutes les requêtes satisfaites par OBDD ; et puisque $\mathtt{NNF} \sim_p^{\mathrm{Id}} \mathtt{PROP}$ (les formules propositionnelles sur les opérateurs $\neg$, $\vee$ et $\wedge$ peuvent être mises en forme normale négative en temps polynomial), NNF et PROP satisfont exactement le même ensemble de requêtes et transformations. Dans ce dernier cas, les deux langages sont par conséquent considérés comme strictement équivalents. Cependant, passer par une correspondance sémantique ne permet pas de conserver ces propriétés, car les requêtes et transformations qui s'appliquent sur un langage ne s'appliquent pas nécessairement à un autre. Les OIAs peuvent ainsi être discrétisés en OMDDs (par exemple, en figure 1, l'OMDD est une discrétisation de l'OIA), c'est-à-dire que $\mathtt{OIA} \geq_p^{\mathcal{T}_{\mathrm{discr}}} \mathtt{OMDD}$, mais tandis qu'OMDD satisfait la requête d'énumération de modèles, les OIAs ont généralement un nombre infini de modèles. On étudiera des conditions permettant ce type d'inférence en section 4.

3.3 Propriétés des relations de comparaison

On utilise $\geq$ pour désigner une des trois relations définies, $\geq_e$, $\geq_s$ ou $\geq_p$. En passant par la correspondance sémantique Id, on retrouve la définition originale de chaque relation ; $\geq^{\mathrm{Id}}$ est seulement adaptée aux langages homogènes, mais a l'avantage d'être un préordre. Ce n'est pas le cas pour une $\mathcal{T}$ quelconque, simplement parce que les espaces d'interprétation peuvent être différents (auquel cas $\geq^{\mathcal{T}}$ ne peut notamment pas être réflexive). Cependant, quand $\mathcal{T}$ est une endorelation, $\geq^{\mathcal{T}}$ hérite certaines de ses propriétés.

Proposition 3.4. *Soient $\Omega \subseteq \mathfrak{U}$ et $\mathcal{T} \subseteq \Omega^2$. Si $\mathcal{T}$ est réflexive (resp. transitive), alors $\geq^{\mathcal{T}}$ est aussi réflexive (resp. transitive).*

Démonstration. Supposons que $\mathcal{T}$ soit réflexive ; soit L un langage de représentation avec $\Omega_{\mathtt{L}} = \Omega$. Considérons une L-représentation φ ; il existe une L-représentation φ' telle que $[\![\varphi]\!]_{\mathtt{L}} \; \mathcal{T} \; [\![\varphi']\!]_{\mathtt{L}}$ (il suffit de prendre $\varphi' = \varphi$), par conséquent $\mathtt{L} \geq_e^{\mathcal{T}} \mathtt{L}$.

Supposons que $\mathcal{T}$ soit transitive ; soient $\mathtt{L}_1$, $\mathtt{L}_2$ et $\mathtt{L}_3$ trois langages de représentation d'espace d'interprétation Ω, tels

6. On peut définir $\mathcal{T}$ comme l'ensemble des couples de fonctions booléennes $\langle f, g \rangle \in \mathfrak{B}^2$ pour lesquels si g a une KROM-C-représentation, alors $f = g$.

que $\mathtt{L}_1 \geq_e^{\mathcal{T}} \mathtt{L}_2$ et $\mathtt{L}_2 \geq_e^{\mathcal{T}} \mathtt{L}_3$. Soit φ_1 une $\mathtt{L}_1$-représentation. Par définition, il existe une $\mathtt{L}_2$-représentation φ_2 telle que $[\![\varphi_1]\!]_{\mathtt{L}_1} \ \mathcal{T} \ [\![\varphi_2]\!]_{\mathtt{L}_2}$, et une $\mathtt{L}_3$-représentation φ_3 telle que $[\![\varphi_2]\!]_{\mathtt{L}_2} \ \mathcal{T} \ [\![\varphi_3]\!]_{\mathtt{L}_3}$. Puisque $\mathcal{T}$ est transitive, $[\![\varphi_1]\!]_{\mathtt{L}_1} \ \mathcal{T} \ [\![\varphi_3]\!]_{\mathtt{L}_3}$, et donc $\mathtt{L}_1 \geq_e^{\mathcal{T}} \mathtt{L}_3$.

La même preuve est adaptable pour $\geq_s$ et $\geq_p$, la composée de deux polynômes étant un polynôme. □

De telles correspondances sémantiques peuvent être utilisées pour comparer des langages de même espace d'interprétation, comme illustré avec HORN-C et KROM-C. Quand $\mathcal{T}$ n'a aucune propriété notable, la proposition suivante exprime malgré tout une « pseudo-transitivité ».

Proposition 3.5. *Si, pour des langages de représentation* $\mathtt{L}_1$, $\mathtt{L}_2$ *et* $\mathtt{L}_3$, *et des correspondances sémantiques* $\mathcal{T}$ *et* $\mathcal{T}'$, *il est vérifié que* $\mathtt{L}_1 \geq^{\mathcal{T}} \mathtt{L}_2$ *et que* $\mathtt{L}_2 \geq^{\mathcal{T}'} \mathtt{L}_3$, *alors* $\mathtt{L}_1 \geq^{\mathcal{T}' \circ \mathcal{T}} \mathtt{L}_3$ *(où* $\circ$ *désigne la composition de relations).*

Démonstration. Puisque $\mathtt{L}_1 \geq_e^{\mathcal{T}} \mathtt{L}_2$, pour toute $\mathtt{L}_1$-représentation φ_1, il existe une $\mathtt{L}_2$-représentation φ_2 telle que $[\![\varphi_1]\!]_{\mathtt{L}_1} \ \mathcal{T} \ [\![\varphi_2]\!]_{\mathtt{L}_2}$. Mais puisque $\mathtt{L}_2 \geq_e^{\mathcal{T}'} \mathtt{L}_3$, il existe une $\mathtt{L}_3$-représentation φ_3 telle que $[\![\varphi_2]\!]_{\mathtt{L}_2} \ \mathcal{T}' \ [\![\varphi_3]\!]_{\mathtt{L}_3}$. Par définition de la composition de relations, $[\![\varphi_1]\!]_{\mathtt{L}_1} \ (\mathcal{T}' \circ \mathcal{T}) \ [\![\varphi_3]\!]_{\mathtt{L}_3}$, d'où le résultat.

La preuve est similaire pour $\geq_s$ et $\geq_p$, une nouvelle fois en se basant sur le fait que composer deux polynômes donne un polynôme. □

Cette proposition a un corollaire utile, qui permet d'étendre les résultats de concision d'une hiérarchie de fragments à une autre, à condition qu'une traduction polynomiale « réversible » existe entre elles (voir la section 5 pour un exemple d'utilisation).

Corollaire 3.6. *Soient* $\mathtt{L}_1$ *et* $\mathtt{L}_2$ *(resp.* $\mathtt{L}'_1$ *et* $\mathtt{L}'_2$*) deux langages d'espace d'interprétation* Ω *(resp.* Ω'*). S'il existe une correspondance sémantique bijective* $\mathcal{T}$ *entre* Ω *et* Ω' *telle que* $\mathtt{L}_1 \leq^{\mathcal{T}} \mathtt{L}'_1$ *et* $\mathtt{L}_2 \geq^{\mathcal{T}} \mathtt{L}'_2$, *alors* $\mathtt{L}_1 \geq^{\mathrm{Id}} \mathtt{L}_2 \implies \mathtt{L}'_1 \geq^{\mathrm{Id}} \mathtt{L}'_2$.

Démonstration. Supposons que $\mathtt{L}_1 \leq^{\mathcal{T}} \mathtt{L}'_1$ (c'est-à-dire, $\mathtt{L}'_1 \geq^{\mathcal{T}^{-1}} \mathtt{L}_1$), que $\mathtt{L}_2 \geq^{\mathcal{T}} \mathtt{L}'_2$ et que $\mathtt{L}_1 \geq^{\mathrm{Id}} \mathtt{L}_2$. Par la proposition 3.5, les première et troisième équations impliquent que $\mathtt{L}'_1 \geq^{\mathrm{Id} \circ \mathcal{T}^{-1}} \mathtt{L}_2$, ce qui est équivalent à $\mathtt{L}'_1 \geq^{\mathcal{T}^{-1}} \mathtt{L}_2$, par définition de Id et de la composition de relations. De nouveau, on peut utiliser la proposition 3.5 pour inférer (grâce à la deuxième équation) que $\mathtt{L}'_1 \geq^{\mathcal{T} \circ \mathcal{T}^{-1}} \mathtt{L}'_2$. Puisque $\mathcal{T}$ est bijective, cela revient à $\mathtt{L}'_1 \geq^{\mathrm{Id}} \mathtt{L}'_2$. □

4 Efficacité calculatoire

On utilise les langages de représentation pour résoudre des problèmes portant sur les objets qu'ils représentent. On trouve généralement une solution d'un problème grâce à un algorithme, c'est-à-dire une suite d'opérations sur les objets en question ; l'implantation pratique de l'algorithme dépend des langages choisis.

4.1 Opérations

Définition 4.1 (Opération sémantique)**.** Une *opération sémantique* sur un univers $\Omega \subseteq \mathfrak{U}$ est une relation totale par morceaux [7] $\rho \subseteq \Omega^\kappa \times \mathcal{P} \times \mathcal{A}$, où $\mathcal{P} \subseteq \mathfrak{U}$ et $\mathcal{A} \subseteq \mathfrak{U}$ sont des ensembles contenant respectivement des *paramètres* et des *réponses*, et où soit $\kappa \in \mathbb{N}$ (l'opération sémantique étant alors *bornée*), soit $\kappa = *$ (l'opération sémantique étant alors *non bornée*).

Donnons quelques exemples basés sur la carte de compilation, avec Ω l'ensemble des fonctions booléennes sur des variables réelles : $\Omega = \mathfrak{B}_{\mathbb{R}}$. L'opération associée à la transformation de « conditionnement » est la fonction $\rho_{\mathbf{CD}}$ de $\Omega \times (\mathbb{R})^*$ dans Ω, définie par $\rho_{\mathbf{CD}} \colon \langle f, \vec{x} \rangle \mapsto f|_{\vec{x}}$ ($\rho_{\mathbf{CD}}$ n'étant définie que sur les couples $\langle f, \vec{x} \rangle$ tels que f est définie sur $\vec{x}$). Ici $\kappa = 1$, $\mathcal{P} = (\mathbb{R})^*$ est l'ensemble de toutes les instanciations, et $\mathcal{A} = \Omega$ est l'ensemble des fonctions booléennes. L'opération associée à la requête d'« extraction de modèle » n'est pas une fonction, mais une relation $\rho_{\mathbf{MX}}$, qui associe à toute fonction booléenne f l'ensemble de *tous* ses modèles ; ici $\kappa = 1$, $\mathcal{A} = (\mathbb{R})^*$ et $\mathcal{P}$ est ignoré [8]. L'opération associée à la transformation de « conjonction » est l'application $\rho_{\wedge\mathbf{C}} \colon \langle f_1, \ldots, f_{n_\kappa} \rangle \mapsto \bigwedge_{i=1}^{n_\kappa} f_i$ pour tout $n_\kappa \in \mathbb{N}$. Ici, l'opération est non bornée : $\kappa = *$.

La complexité d'une opération sémantique $\rho \subseteq \Omega^\kappa \times \mathcal{P} \times \mathcal{A}$ dépend des langages de représentation considérés. Le plus important est le langage « d'entrée », c'est-à-dire le langage servant à représenter les éléments de Ω. Les opérations sémantiques peuvent être beaucoup plus génériques que nécessaire : les opérations présentées précédemment s'appliquent à toute fonction booléenne sur des variables réelles, mais peuvent aussi être utilisées, de manière plus restreinte, pour comparer des fonctions booléennes sur des variables booléennes. Une opération sémantique peut en effet être *appliquée* à un sous-ensemble Ω' de son univers : on note $\rho|^{\Omega'}$ la plus grande (pour l'inclusion) opération sémantique vérifiant $\rho|^{\Omega'} \subseteq \rho \cap (\Omega'^\kappa \times \mathcal{P} \times \mathcal{A})$, et $\mathcal{P}_{\Omega'}$ et $\mathcal{A}_{\Omega'}$ les ensembles de paramètres et de réponses de $\rho|^{\Omega'}$ — desquels tous les éléments « superflus » ont été retirés. On dira qu'une opération sémantique sur $\Omega \subseteq \mathfrak{U}$ est *applicable* à un langage $\mathtt{L}$ quand $\Omega_{\mathtt{L}} \subseteq \Omega$: ainsi, $\rho_{\mathbf{CD}}$ est applicable à CNF, et alors $\mathcal{P}_{\Omega_{\mathtt{CNF}}}$ est l'ensemble

7. Une relation $\mathcal{R} \subseteq D_1 \times \cdots \times D_n$ est totale par morceaux si et seulement si $\forall i \in \{1, \ldots, n\}, \forall d_i \in D_i, \exists \langle r_1, \ldots, r_n \rangle \in \mathcal{R}$, $r_i = d_i$. Moins formellement, cela signifie que sa projection sur chacune de ses dimensions est totale.

8. Quand une opération sémantique ne nécessite aucun paramètre, on omet $\mathcal{P}$ par simplicité, considérant implicitement que c'est un singleton.

des instanciations booléennes et $\mathcal{A}_{\Omega_{\text{CNF}}}$ l'ensemble des fonctions booléennes sur des variables booléennes. On utilise cette notion d'application pour définir les opérations syntaxiques associées aux opérations sémantiques.

Définition 4.2 (Opération)**.** Une *opération (syntaxique)* est un tuple $\mathcal{O} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$ tel que (i) ρ est une opération sémantique : $\rho \subseteq \Omega^\kappa \times \mathcal{P} \times \mathcal{A}$; (ii) $\mathtt{L}$ est un langage de représentation auquel ρ est applicable : $\Omega_\mathtt{L} \subseteq \Omega$; (iii) $\mathtt{PRM}$ est un langage de représentation couvrant tous les paramètres compatibles avec $\mathtt{L}$: $\mathcal{P}_{\Omega_\mathtt{L}} \subseteq \Omega_\mathtt{PRM}$; (iv) $\mathtt{ANS}$ est un langage de représentation couvrant toutes les réponses compatibles avec $\mathtt{L}$: $\mathcal{A}_{\Omega_\mathtt{L}} \subseteq \Omega_\mathtt{ANS}$.

Insistons sur la manière dont le choix du langage de représentation principal dans l'opération syntaxique restreint l'opération sémantique. Quand on choisit $\mathtt{L}$, on écarte tous les éléments du domaine de ρ qui ne sont pas des $\mathtt{L}$-interprétations (cela sera apparent dans le lemme 4.4). Les langages dans une opération induisent une version syntaxique de l'opération sémantique sous-jacente.

Définition 4.3 (Complexité d'une opération)**.** Soit $\mathcal{O} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$ une opération syntaxique, avec $\rho \subseteq \Omega^\kappa \times \mathcal{P} \times \mathcal{A}$. Le problème associé à $\mathcal{O}$ est le suivant : pour tout tuple $\langle \varphi_1, \ldots, \varphi_{n_\kappa}, \pi \rangle$ de formules de $\Phi^\kappa_\mathtt{L} \times \Phi_\mathtt{PRM}$, construire une formule $\alpha \in \Phi_\mathtt{ANS}$ telle que $\langle [\![\varphi_1]\!]_\mathtt{L}, \ldots, [\![\varphi_{n_\kappa}]\!]_\mathtt{L}, [\![\pi]\!]_\mathtt{PRM}, [\![\alpha]\!]_\mathtt{ANS} \rangle \in \rho$, s'il en existe une. La complexité de $\mathcal{O}$ est celle de son problème associé.

C'est la complexité de ce problème syntaxique qui définit la complexité de l'opération correspondante. Ainsi, « l'opération $\mathcal{O}$ est en temps polynomial » signifie qu'il existe un algorithme résolvant en temps polynomial le problème associé à $\mathcal{O}$. Une opération est donc une façon de spécifier un problème ; les *problèmes de décision* classiques de la théorie de la calculabilité constituent un cas particulier d'opérations. On peut modéliser un problème de décision par $\mathcal{O} = \langle \rho_\Phi, \Sigma^*, \mathtt{bool} \rangle$, où ρ_Φ est la fonction indicatrice d'un langage formel Φ, et $\mathtt{bool}$ le langage de représentation associant $\bot$ au symbole « 0 » et $\top$ au symbole « 1 ». Plus généralement, les opérations permettent d'exprimer des *problèmes fonctionnels* sans en perdre la sémantique. Avant de passer aux opérations particulières que sont les requêtes et les transformations, le lemme suivant illustre les liens entre le langage d'entrée et les autres éléments d'une opération.

Lemme 4.4 (Application d'une opération)**.** *Soit $\mathcal{O} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$ une opération, avec $\rho \subseteq \Omega^\kappa \times \mathcal{P} \times \mathcal{A}$. Le problème associé à $\mathcal{O}$ ne change pas si l'on remplace ρ par $\rho|^{\Omega_\mathtt{L}}$, et/ou $\mathtt{PRM}$ par $\mathtt{PRM}|^{\mathcal{P}_{\Omega_\mathtt{L}}}$, et/ou $\mathtt{ANS}$ par $\mathtt{ANS}|^{\mathcal{A}_{\Omega_\mathtt{L}}}$.*

Démonstration. Le problème associé à $\mathcal{O}$ est le suivant : étant donné un tuple $\vec{\varphi} = \langle \varphi_1, \ldots, \varphi_{n_\kappa}, \pi \rangle$ dans $\Phi^\kappa_\mathtt{L} \times \Phi_\mathtt{PRM}$, construire une $\mathtt{ANS}$-représentation α d'un élément de[9] $\rho([\![\varphi_1]\!]_\mathtt{L}, \ldots, [\![\varphi_{n_\kappa}]\!]_\mathtt{L}, [\![\pi]\!]_\mathtt{PRM})$, s'il en existe un.

La relation ρ est un ensemble de tuples « sémantiques » $\langle \omega_1, \ldots, \omega_{n_\kappa}, p, a \rangle$. Les tuples pour lesquels l'un des ω_i n'est pas représentable dans $\mathtt{L}$ ne sont pas considérés. Le problème reste donc le même si l'on remplace ρ par n'importe quelle opération sémantique $\rho' \subseteq \rho \cap (\Omega^\kappa_\mathtt{L} \times \mathcal{P} \times \mathcal{A})$, et notamment par $\rho|^{\Omega_\mathtt{L}}$. On a donc montré que $\mathcal{O}$ est identique à $\mathcal{O}' = \langle \rho|^{\Omega_\mathtt{L}}, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$.

Si dans le tuple $\vec{\varphi}$, la $\mathtt{PRM}$-représentation π n'est pas dans $\Phi_{\mathtt{PRM}|^{\mathcal{P}_{\Omega_\mathtt{L}}}}$, cela signifie que son interprétation p n'est pas dans $\mathcal{P}_{\Omega_\mathtt{L}}$, et donc aucun tuple sémantique dans $\rho|^{\Omega_\mathtt{L}}$ ne contient p ; le problème associé à $\mathcal{O}'$ et $\mathcal{O}$ n'est pas défini sur ce tuple $\vec{\varphi}$. Par conséquent, il est possible de supprimer du problème ces $\mathtt{PRM}$-représentations inutiles : $\mathcal{O}$ et $\mathcal{O}'$ restent identiques si l'on y remplace $\mathtt{PRM}$ par $\mathtt{PRM}|^{\mathcal{P}_{\Omega_\mathtt{L}}}$.

Enfin, l'interprétation de la réponse $\alpha \in \Phi_\mathtt{ANS}$ est nécessairement dans $\mathcal{A}_{\Omega_\mathtt{L}}$, par définition. Toutes les réponses sont donc des représentations du langage $\mathtt{ANS}|^{\mathcal{A}_{\Omega_\mathtt{L}}}$, ce qui implique que l'opération ne change pas quand $\mathtt{ANS}$ est remplacé par $\mathtt{ANS}|^{\mathcal{A}_{\Omega_\mathtt{L}}}$. □

4.2 Requêtes et transformations

On retrouve les requêtes et les transformations en tant qu'opérations particulières en fixant certains éléments, à savoir les langages et la complexité.

Définition 4.5 (Requête, transformation)**.** Une *requête* (resp. une *transformation*) est un tuple $\mathbf{Q} = \langle \rho, \mathtt{PRM}, \mathtt{ANS} \rangle$ (resp. $\mathbf{T} = \langle \rho, \mathtt{PRM} \rangle$) tel que pour tout langage de représentation $\mathtt{L}$ auquel ρ est applicable, $\mathcal{O}_{\mathbf{Q},\mathtt{L}} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{ANS} \rangle$ (resp. $\mathcal{O}_{\mathbf{T},\mathtt{L}} = \langle \rho, \mathtt{L}, \mathtt{PRM}, \mathtt{L} \rangle$) est une opération. Le langage $\mathtt{L}$ *satisfait* $\mathbf{Q}$ (resp. $\mathbf{T}$) si et seulement si $\mathcal{O}_{\mathbf{Q},\mathtt{L}}$ est en temps polynomial en son entrée et sa sortie (resp. $\mathcal{O}_{\mathbf{T},\mathtt{L}}$ est en temps polynomial).

Pour les langages représentant des fonctions booléennes, en notant $\mathtt{term}$ le langage des conjonctions d'atomes de la forme $[x = n]$ (où $x \in \Sigma$ est une variable et $n \in \mathbb{R}$), la requête d'extraction de modèle peut être définie comme $\mathbf{MX} = \langle \rho_\mathbf{MX}, \mathtt{term} \rangle$, la transformation de conditionnement par $\mathbf{CD} = \langle \rho_\mathbf{CD}, \mathtt{term} \rangle$, et la transformation de conjonction par $\wedge\mathbf{C} = \langle \rho_{\wedge\mathbf{C}} \rangle$. Ces opérations s'appliquent à tout langage représentant des fonctions booléennes, qu'il soit ou non restreint à un sous-ensemble de cet espace — par exemple aux fonctions sur des variables booléennes ou entières.

La satisfaction de certaines requêtes et transformations est un indicateur des capacités absolues et relatives des langages de représentation à permettre une

9. Dans cette preuve et les suivantes, on utilisera parfois la notation suivante : pour une relation $\mathcal{R} \subseteq D_1 \times \cdots \times D_n \times E$, où $n \in \mathbb{N}$, on notera $\mathcal{R}(d_1, \ldots, d_n)$ l'ensemble $\{ e \in E \mid \langle d_1, \ldots, d_n, e \rangle \in \mathcal{R} \}$; c'est-à-dire qu'on considère la relation $\mathcal{R}$ comme une fonction de $D_1 \times \cdots \times D_n$ dans 2^E.

manipulation efficace des objets. Avec cette définition, le concept s'applique universellement à tout langage de représentation. Insistons cependant sur le fait que la liste des requêtes et transformations utiles pour évaluer les langages d'une hiérarchie donnée *dépend* en général de l'espace d'interprétation. Il est néanmoins parfois possible d'inférer des résultats sur les requêtes et transformations dans une hiérarchie à partir d'une autre ; par exemple, la satisfaction de la plupart des requêtes et transformations pour les langages de la famille de MDD peut être déduite de résultats sur la famille de BDD, ces deux familles étant polynomialement équivalentes modulo une correspondance sémantique « adaptée ». Examinons maintenant des conditions suffisantes permettant ce type d'inférence.

4.3 Opérations et traduction polynomiale

Dans le cas général, on voudrait pouvoir déduire qu'une opération $\mathcal{O}$ est polynomiale du fait qu'une autre, vers laquelle $\mathcal{O}$ peut être traduite, est polynomiale. Commençons par définir la notion de traduction entre opérations sémantiques.

Définition 4.6 (Traduction entre opérations sémantiques)**.** Soit ρ_1 et ρ_2 deux opérations sémantiques : $\rho_1 \subseteq \Omega_1^\kappa \times \mathcal{P}_1 \times \mathcal{A}_1$ et $\rho_2 \subseteq \Omega_2^\kappa \times \mathcal{P}_2 \times \mathcal{A}_2$. Une *traduction entre opérations sémantiques* de ρ_1 vers ρ_2 est un triplet $\langle \mathcal{T}_\Omega, \mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{A} \rangle$, où (i) $\mathcal{T}_\Omega \subseteq \Omega_1 \times \Omega_2$, (ii) $\mathcal{T}_\mathcal{P} \subseteq \mathcal{P}_1 \times \mathcal{P}_2$, et (iii) $\mathcal{T}_\mathcal{A} \subseteq \mathcal{A}_1 \times \mathcal{A}_2$, qui vérifie $\rho_1 = \mathcal{T}_\mathcal{A}^{-1} \circ \rho_2 \circ (\mathcal{T}_\Omega^\kappa \cdot \mathcal{T}_\mathcal{P})$, où $\circ$ (resp. $\cdot$) désigne la composition (resp. le produit) de relations.

On commence à voir en quoi certaines correspondances sémantiques sont plus utiles que d'autres : par exemple, la correspondance sémantique triviale $\mathcal{T}{=}\ \Omega_1 \times \Omega_2$ a peu de chances d'être utilisée dans une traduction entre opérations sémantiques. L'intérêt d'avoir $\mathtt{L}_1 \geq_p^\mathcal{T} \mathtt{L}_2$ dépend en fait de la manière dont $\mathcal{T}$ est liée aux opérations considérées. De plus, on ne peut rien déduire si la traduction des paramètres et réponses est difficile — d'où le besoin d'une notion de traduction polynomiale entre opérations syntaxiques.

Définition 4.7 (Traduction polynomiale entre opérations)**.** Soient $\mathcal{O}_1 = \langle \rho_1, \mathtt{L}_1, \mathtt{PRM}_1, \mathtt{ANS}_1 \rangle$ et $\mathcal{O}_2 = \langle \rho_2, \mathtt{L}_2, \mathtt{PRM}_2, \mathtt{ANS}_2 \rangle$ deux opérations. On dit que $\mathcal{O}_1$ est *polynomialement traduisible* vers $\mathcal{O}_2$ si et seulement s'il existe une traduction entre opérations sémantiques $\langle \mathcal{T}_\Omega, \mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{A} \rangle$ de ρ_1 vers ρ_2 vérifiant (i) $\mathtt{L}_1 \geq_p^{\mathcal{T}_\Omega} \mathtt{L}_2$, (ii) $\mathtt{PRM}_1 \geq_p^{\mathcal{T}_\mathcal{P}} \mathtt{PRM}_2$, et (iii) $\mathtt{ANS}_1 \leq_p^{\mathcal{T}_\mathcal{A}} \mathtt{ANS}_2$. Si la traduction polynomiale entre les langages de réponse est stable, c'est-à-dire si $\mathtt{ANS}_1 \leq_{p\sim}^{\mathcal{T}_\mathcal{A}} \mathtt{ANS}_2$, alors la traduction entre opérations est dite *answer-stable*.

Cette notion inclut les réductions polynomiales bien connues de la théorie de la complexité : un problème, c'est-à-dire une opération $\mathcal{O}_1 = \langle \rho_{\Phi_1}, \Sigma_1^*, \mathtt{bool} \rangle$, se réduit polynomialement à un autre, c'est-à-dire à une opération $\mathcal{O}_2 = \langle \rho_{\Phi_2}, \Sigma_2^*, \mathtt{bool} \rangle$, si et seulement si $\mathcal{O}_1$ est polynomialement traduisible vers $\mathcal{O}_2$. De la même manière que pour les réductions entre problèmes, la complexité d'une opération dépend de celles des opérations vers lesquelles elle peut être polynomialement traduite.

Théorème 4.8. *Soit $\mathcal{O}_1$ et $\mathcal{O}_2$ deux opérations telles que $\mathcal{O}_1$ est polynomialement traduisible vers $\mathcal{O}_2$. Si $\mathcal{O}_2$ est en temps polynomial, alors $\mathcal{O}_1$ l'est également. Quand la traduction est* answer-stable*, si $\mathcal{O}_2$ est en temps polynomial en son entrée et sa sortie, alors $\mathcal{O}_1$ l'est également.*

Démonstration. Notons $\mathcal{O}_1 = \langle \rho_1, \mathtt{L}_1, \mathtt{PRM}_1, \mathtt{ANS}_1 \rangle$ et $\mathcal{O}_2 = \langle \rho_2, \mathtt{L}_2, \mathtt{PRM}_2, \mathtt{ANS}_2 \rangle$, et supposons que $\langle \mathcal{T}_\Omega, \mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{A} \rangle$ est une traduction sémantique de ρ_1 vers ρ_2 vérifiant les conditions de traduisibilité polynomiale.

Considérons l'algorithme suivant, qui prend en entrée un tuple de formules $\vec{\varphi} = \langle \varphi_1, \ldots, \varphi_{n_\kappa}, \pi \rangle \in \Phi_{\mathtt{L}_1}^\kappa \times \Phi_{\mathtt{PRM}_1}$. Il construit un tuple $\vec{\varphi}' = \langle \varphi'_1, \ldots, \varphi'_{n_\kappa}, \pi' \rangle \in \Phi_{\mathtt{L}_2}^\kappa \times \Phi_{\mathtt{PRM}_2}$, tel que $\forall i \in \{1, \ldots, n_\kappa\}, [\![\varphi_i]\!]_{\mathtt{L}_1}\ \mathcal{T}_\Omega\ [\![\varphi'_i]\!]_{\mathtt{L}_2}$, et $[\![\pi]\!]_{\mathtt{PRM}_1}\ \mathcal{T}_\mathcal{P}\ [\![\pi']\!]_{\mathtt{PRM}_2}$. La construction de chaque formule du tuple peut être faite en temps polynomial, par hypothèse (déf. 4.7) ; la construction du tuple entier est donc polynomiale.

L'algorithme applique ensuite au tuple la procédure pour $\mathcal{O}_2$: il construit une $\mathtt{ANS}_2$-représentation α', telle que $[\![\alpha']\!]_{\mathtt{ANS}_2} \in \rho_2([\![\varphi'_1]\!]_{\mathtt{L}_2}, \ldots, [\![\varphi'_{n_\kappa}]\!]_{\mathtt{L}_2}, [\![\pi']\!]_{\mathtt{PRM}_2})$. Enfin, notre algorithme construit une $\mathtt{ANS}_1$-représentation α vérifiant $[\![\alpha]\!]_{\mathtt{ANS}_1}\ \mathcal{T}_\mathcal{A}\ [\![\alpha']\!]_{\mathtt{ANS}_2}$, cette étape étant faite en temps polynomial, par hypothèse (déf. 4.7).

Vérifions d'abord que α est une réponse correcte pour $\mathcal{O}_1$. Par hypothèse (déf. 4.6), $\rho_1 = \mathcal{T}_\mathcal{A}^{-1} \circ \rho_2 \circ (\mathcal{T}_\Omega^\kappa \cdot \mathcal{T}_\mathcal{P})$, c'est-à-dire que

$$\forall \vec{\omega} \in \Omega_1^\kappa \times \mathcal{P}_1, \forall a \in \mathcal{A}_1,$$
$$a \in \rho_1(\vec{\omega}) \iff$$
$$\exists \vec{\omega}' \in \Omega_2^\kappa \times \mathcal{P}_2, \exists a' \in \mathcal{A}_2, \begin{cases} & \vec{\omega}' \in (\mathcal{T}_\Omega^\kappa \cdot \mathcal{T}_\mathcal{P})(\vec{\omega}) \\ \wedge & a' \in \rho_2(\vec{\omega}') \\ \wedge & a' \in \mathcal{T}_\mathcal{A}(a)). \end{cases}$$

Ici, $\vec{\omega} = \langle [\![\varphi_1]\!]_{\mathtt{L}_1}, \ldots, [\![\varphi_{n_\kappa}]\!]_{\mathtt{L}_1}, [\![\pi]\!]_{\mathtt{PRM}_1} \rangle$ et $a = [\![\alpha]\!]_{\mathtt{ANS}_1}$. Nous avons construit un $\vec{\omega}'$ et un a' vérifiant toutes les conditions : prendre $\vec{\omega}' = \langle [\![\varphi'_1]\!]_{\mathtt{L}_2}, \ldots, [\![\varphi'_{n_\kappa}]\!]_{\mathtt{L}_2}, [\![\pi']\!]_{\mathtt{PRM}_2} \rangle$ et $a' = [\![\alpha']\!]_{\mathtt{ANS}_2}$. Ainsi, $[\![\alpha]\!]_{\mathtt{ANS}_1} \in \rho_1(\vec{\omega})$; α est donc une réponse correcte pour $\mathcal{O}_1$.

Passons à présent à la complexité. Toutes les étapes de notre algorithme sont en temps polynomial, excepté l'application de la procédure pour $\mathcal{O}_2$. Si elle est polynomiale, alors de manière claire, ayant été obtenu par composition de trois procédures polynomiales, notre algorithme est en temps polynomial. Si elle est en temps polynomial en son entrée et sa sortie, alors cela signifie que α' a été obtenu en temps $P(\sum_{i=1}^n |\varphi_i| + |\pi|, |\alpha'|)$ (avec P un polynôme fixé). Si la traduction entre les langages de réponse est stable, il existe, par la définition 3.3, un certain polynôme P' tel que

$|\alpha'| \leq P'(|\alpha|)$, donc α' a été obtenu en temps borné par un polynôme de $\sum_{i=1}^{n}|\varphi_i| + |\pi|$ et $|\alpha|$, et puisque l'étape de traduction des réponses est polynomiale, l'algorithme entier est bien polynomial en son entrée et sa sortie. □

Par application de ce théorème, $\langle \rho_{\mathbf{MX}}, \texttt{OMDD}, \texttt{term}\rangle$, $\langle \rho_{\mathbf{CD}}, \texttt{MDD}, \texttt{term}, \texttt{MDD}\rangle$, et $\langle \rho_{\wedge\mathbf{C}}, \texttt{MDD}, \texttt{MDD}\rangle$ sont traitables, en se basant sur le fait que les opérations correspondantes pour les langages de la famille de BDD le sont. Cependant, le fait que OBDD satisfasse **SFO** (l'*oubli* d'une unique variable) n'implique pas que OMDD satisfait **SFO**, car oublier une variable multivaluée revient à oublier un nombre non borné de variables booléennes, ce qui est NP-difficile sur OBDD. De la même façon, les correspondances sémantiques qui ne maintiennent pas le nombre de modèles ne peuvent pas être utilisées pour déduire que OMDD satisfait **CT** du fait que c'est le cas pour OBDD. À noter que la condition d'*answer-stability* est nécessaire à la deuxième partie du théorème, car la complexité temporelle de la procédure toute entière dépend de la taille de la réponse à $\mathcal{O}_2$, qui peut être exponentielle en celle de son entrée.

Les déductions permises par ce théorème peuvent relier des opérations très disparates. Cependant, dans le contexte d'une carte de compilation, le cadre est généralement plus restreint : typiquement, on veut exploiter une correspondance sémantique entre deux espaces d'interprétation, qui induit une traduction polynomiale entre les langages des deux hiérarchies, pour faire des déductions du genre « si L satisfait telle requête, alors L′ la satisfait aussi » ; c'est-à-dire qu'on s'intéresse à *une unique* opération sémantique, applicable aux deux langages. De fait, cela peut être réalisé grâce à un corollaire du théorème 4.8, tant que l'on ne considère que des requêtes et transformations *adaptées* à la correspondance sémantique en question.

Définition 4.9 ($\mathcal{T}$-adaptation)**.** Soit $\mathcal{T}$ une correspondance sémantique entre des univers $\Omega_1 \subseteq \mathfrak{U}$ et $\Omega_2 \subseteq \mathfrak{U}$. Une requête $\mathbf{Q} = \langle \rho, \texttt{PRM}, \texttt{ANS}\rangle$ est $\mathcal{T}$*-adaptée* si et seulement si : (i) ρ est applicable à Ω_1 et Ω_2 ; (ii) il existe deux correspondances sémantiques $\mathcal{T}_\mathcal{P}$ et $\mathcal{T}_\mathcal{A}$ telles que $\langle \mathcal{T}, \mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{A}\rangle$ est une traduction de $\rho|^{\Omega_1}$ vers $\rho|^{\Omega_2}$; (iii) $\texttt{PRM}|^{\mathcal{P}_{\Omega_1}} \geq_p^{\mathcal{T}_\mathcal{P}} \texttt{PRM}|^{\mathcal{P}_{\Omega_2}}$; (iv) $\texttt{ANS}|^{\mathcal{A}_{\Omega_1}} \leq_{p\sim}^{\mathcal{T}_\mathcal{A}} \texttt{ANS}|^{\mathcal{A}_{\Omega_2}}$. Une transformation $\mathbf{T} = \langle \rho, \texttt{PRM}\rangle$ est $\mathcal{T}$*-adaptée* si et seulement si elle vérifie les conditions i-iii, avec $\mathcal{T}_\mathcal{A}$=$\mathcal{T}$ (la dernière condition ne s'applique pas : il n'y a pas de langage ANS dans une transformation).

La plupart des requêtes et transformations de la carte de compilation classique sont adaptées aux encodages direct et logarithmique, y compris **MX**, **CD**, $\wedge$**C** et presque toutes celles considérées par Darwiche et Marquis [6] (**SFO** exceptée, pour les raisons données précédemment). Cela est particulièrement intéressant au regard du résultat suivant.

Théorème 4.10. *Soient* $\mathtt{L}_1$ *et* $\mathtt{L}_2$ *deux langages, et soit* $\mathcal{T}$ *une traduction de* $\mathtt{L}_1$ *vers* $\mathtt{L}_2$*. Si* $\mathtt{L}_1 \geq_p^{\mathcal{T}} \mathtt{L}_2$*, alors toute requête* $\mathcal{T}$*-adaptée satisfaite par* $\mathtt{L}_2$ *l'est aussi par* $\mathtt{L}_1$*. Si* $\mathtt{L}_1 \sim_p^{\mathcal{T}} \mathtt{L}_2$*, alors toute transformation* $\mathcal{T}$*-adaptée satisfaite par* $\mathtt{L}_2$ *l'est aussi par* $\mathtt{L}_1$*.*

Démonstration. Supposons que $\mathtt{L}_1 \geq_p^{\mathcal{T}} \mathtt{L}_2$. Soit $\mathbf{Q} = \langle \rho, \texttt{PRM}, \texttt{ANS}\rangle$ une requête $\mathcal{T}$-adaptée, et supposons que $\mathtt{L}_2$ satisfasse $\mathbf{Q}$: cela signifie (déf. 4.5) que l'opération $\mathcal{O}_{\mathbf{Q},\mathtt{L}_2} = \langle \rho, \mathtt{L}_2, \texttt{PRM}, \texttt{ANS}\rangle$ est polynomiale en son entrée et sa sortie.

Par le lemme 4.4, il vient que $\mathcal{O}_{\mathbf{Q},\mathtt{L}_2} = \langle \rho|^{\Omega_{\mathtt{L}_2}}, \mathtt{L}_2, \texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_2}}}, \texttt{ANS}|^{\mathcal{A}_{\Omega_{\mathtt{L}_2}}}\rangle$, et $\mathcal{O}_{\mathbf{Q},\mathtt{L}_1} = \langle \rho|^{\Omega_{\mathtt{L}_1}}, \mathtt{L}_1, \texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_1}}}, \texttt{ANS}|^{\mathcal{A}_{\Omega_{\mathtt{L}_1}}}\rangle$ — on sait que $\mathcal{O}$ est applicable à $\mathtt{L}_1$ grâce au premier point de la définition d'une requête $\mathcal{T}$-adaptée (déf. 4.9) : ici, $\Omega_1 = \Omega_{\mathtt{L}_1}$ et $\Omega_2 = \Omega_{\mathtt{L}_2}$.

Grâce aux autres points de cette définition, on sait qu'il existe une traduction entre opérations sémantiques $\langle \mathcal{T}, \mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{A}\rangle$ de $\rho|^{\Omega_{\mathtt{L}_1}}$ vers $\rho|^{\Omega_{\mathtt{L}_2}}$, telle que (i) $\texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_1}}} \geq_p^{\mathcal{T}_\mathcal{P}} \texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_2}}}$; et (ii) $\texttt{ANS}|^{\mathcal{A}_{\Omega_{\mathtt{L}_1}}} \leq_{p\sim}^{\mathcal{T}_\mathcal{A}} \texttt{ANS}|^{\mathcal{A}_{\Omega_{\mathtt{L}_2}}}$. Puisque, par hypothèse, $\mathtt{L}_1 \geq_p^{\mathcal{T}} \mathtt{L}_2$, cela signifie que $\langle \mathcal{T}, \mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{A}\rangle$ est une traduction polynomiale *answer-stable* de $\mathcal{O}_{\mathbf{Q},\mathtt{L}_1}$ vers $\mathcal{O}_{\mathbf{Q},\mathtt{L}_2}$. Comme nous avons pris l'hypothèse que $\mathcal{O}_{\mathbf{Q},\mathtt{L}_2}$ est polynomiale en son entrée et sa sortie, on peut déduire du théorème 4.8 que $\mathcal{O}_{\mathbf{Q},\mathtt{L}_1}$ l'est aussi : $\mathtt{L}_1$ satisfait $\mathbf{Q}$, et le premier point du théorème est démontré.

Supposons que $\mathtt{L}_1 \sim_p^{\mathcal{T}} \mathtt{L}_2$, et considérons $\mathbf{T} = \langle \rho, \texttt{PRM}\rangle$, une transformation $\mathcal{T}$-adaptée et satisfaite par $\mathtt{L}_2$. Par la définition 4.5, cela signifie que l'opération $\mathcal{O}_{\mathbf{T},\mathtt{L}_2} = \langle \rho, \mathtt{L}_2, \texttt{PRM}, \mathtt{L}_2\rangle$ est polynomiale. Une fois encore, le lemme 4.4 implique que $\mathcal{O}_{\mathbf{T},\mathtt{L}_2} = \langle \rho|^{\Omega_{\mathtt{L}_2}}, \mathtt{L}_2, \texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_2}}}, \mathtt{L}_2\rangle$ et que $\mathcal{O}_{\mathbf{T},\mathtt{L}_1} = \langle \rho|^{\Omega_{\mathtt{L}_1}}, \mathtt{L}_1, \texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_1}}}, \mathtt{L}_1\rangle$.

Comme $\mathbf{T}$ est $\mathcal{T}$-adaptée, par la définition 4.9, il existe une traduction entre opérations sémantiques $\langle \mathcal{T}, \mathcal{T}_\mathcal{P}, \mathcal{T}\rangle$ de $\rho|^{\Omega_{\mathtt{L}_1}}$ vers $\rho|^{\Omega_{\mathtt{L}_2}}$, telle que $\texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_1}}} \geq_p^{\mathcal{T}_\mathcal{P}} \texttt{PRM}|^{\mathcal{P}_{\Omega_{\mathtt{L}_2}}}$. Puisque, par hypothèse, $\mathtt{L}_1 \geq_p^{\mathcal{T}} \mathtt{L}_2$ et $\mathtt{L}_1 \leq_p^{\mathcal{T}} \mathtt{L}_2$, $\langle \mathcal{T}, \mathcal{T}_\mathcal{P}, \mathcal{T}\rangle$ est une traduction polynomiale de $\mathcal{O}_{\mathbf{T},\mathtt{L}_1}$ vers $\mathcal{O}_{\mathbf{T},\mathtt{L}_2}$. On peut de nouveau appliquer le théorème 4.8 : $\mathcal{O}_{\mathbf{T},\mathtt{L}_2}$ est polynomiale, donc $\mathcal{O}_{\mathbf{T},\mathtt{L}_1}$ est polynomiale : $\mathtt{L}_1$ satisfait $\mathbf{T}$. □

Cela permet d'étendre automatiquement aux MDDs la plupart des résultats connus sur les BDDs, et plus généralement, d'étendre la plupart des résultats de la carte de compilation aux langages sur des variables non booléennes. Notons cependant que les résultats ne sont pas exactement les mêmes, puisqu'ils dépendent de l'adaptation de chaque requête et transformation à la correspondance sémantique considérée ; ainsi, OMDD ne satisfait pas **SFO** [1], bien que OBDD la satisfasse.

5 Conclusion

Cet article a présenté un cadre pour la comparaison de langages de représentation. Tout en faisant aussi peu d'hypothèses que possible sur ce qui constitue un langage de représentation, il a montré comment les concepts usuels de la carte de compilation pouvaient être adaptés à ce cadre plus large, permettant ainsi

la comparaison de langages hétérogènes selon leur efficacité représentationnelle et leurs capacités opérationnelles, et l'extension de résultats connus dans le cas booléen à de nombreux langages, portant par exemple sur des variables continues ou discrètes, ou encore à valuation non booléenne.

Pour illustrer ce dernier point, prenons l'exemple simple de la famille des « MDDs bornés » : on définit le langage k-MDD comme la restriction de MDD aux variables de domaines de cardinalité k, et ses fragments k-FMDD, k-OMDD, et k-OMDD$_<$ comme sa restriction aux diagrammes respectivement *read-once*, ordonnés, et <-ordonnés [6].

Proposition 5.1. *Il vient que k-MDD $<_s$ k-FMDD $<_s$ k-OMDD $<_s$ k-OMDD$_<$, et chacune des requêtes et transformations considérées par Darwiche et Marquis [6] est satisfaite par k-MDD (resp. k-FMDD, k-OMDD, k-OMDD$_<$) si et seulement si elle l'est par* BDD *(resp.* FBDD, OBDD, OBDD$_<$*).*

Cette proposition vient directement du fait que k-MDD $\sim_p^{\mathcal{T}_k}$ BDD, k-FMDD $\sim_p^{\mathcal{T}_k}$ FBDD, k-OMDD $\sim_p^{\mathcal{T}_k}$ OBDD, et k-OMDD$_<$ $\sim_p^{\mathcal{T}_k}$ OBDD$_<$, en notant $\mathcal{T}_k$ la restriction de $\mathcal{T}_{\text{dir}}$ aux variables de domaine de cardinalité k, en appliquant respectivement le corollaire 3.6 ($\mathcal{T}_k$ est bijective) et le théorème 4.10 (toutes les requêtes et transformations considérées sont $\mathcal{T}_k$-adaptées).

Il s'agit d'un premier pas vers une carte de compilation généralisée, dans laquelle les hiérarchies de langages hétérogènes seraient présentées de façon unifiée.

Références

[1] 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.

[2] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2/3) :171–206, 1997.

[3] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree : An efficient and robust access method for points and rectangles. In *SIGMOD Conference*, pages 322–331, 1990.

[4] Ronald J. Brachman and Hector J. Levesque. *Knowledge Representation and Reasoning.* Elsevier, 2004.

[5] Randall E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comp.*, 35(8) :677–691, 1986.

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

[7] Hélène Fargier and Pierre Marquis. On valued negation normal form formulas. In *IJCAI*, pages 360–365, 2007.

[8] Hélène Fargier and Pierre Marquis. Extending the knowledge compilation map : Krom, Horn, affine and beyond. In *AAAI*, pages 442–447, 2008.

[9] Hélène Fargier and Pierre Marquis. Knowledge compilation properties of trees-of-BDDs, revisited. In *IJCAI*, pages 772–777, 2009.

[10] Hélène Fargier, Pierre Marquis, and Alexandre Niveau. Towards a knowledge compilation map for heterogeneous representation language. In *IJCAI*, 2013.

[11] Raphael A. Finkel and Jon Louis Bentley. Quad trees : A data structure for retrieval on composite keys. *Acta Inf.*, 4 :1–9, 1974.

[12] Goran Gogic, Henry A. Kautz, Christos H. Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *IJCAI*, pages 862–869, 1995.

[13] Georg Gottlob. Translating default logic into standard autoepistemic logic. *Journal of the ACM*, 42(4) :711–740, 1995.

[14] Hector J. Levesque and Ronald J. Brachman. A fundamental tradeoff in knowledge representation and reasoning (revised version). In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 41–70. Kaufmann, Los Altos, CA, 1985.

[15] Alexandre Niveau, Hélène Fargier, Cédric Pralet, and Gérard Verfaillie. Knowledge compilation using interval automata and applications to planning. In *ECAI*, pages 459–464, 2010.

[16] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming.* Elsevier, 2006.

[17] Stuart J. Russell and Peter Norvig. *Artificial Intelligence — A Modern Approach (3. internat. ed.).* Pearson Education, 2010.

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

[19] Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors. *Handbook of Knowledge Representation*, volume 1. Elsevier Science, 2008.

[20] Toby Walsh. SAT v CSP. In *CP*, pages 441–456, 2000.