are obtained thanks to a simple leaf-merging procedure), and thus satisfies each of the $\mathbf{MX}$, $\mathbf{ME}$, $\mathbf{CT}$, and $\mathbf{CO}$ queries.

Combination, variable elimination, marginalization. None of the VDD languages considered in this paper satisfies the unbounded combination or unbounded variable elimination transformations; this result is not very surprising, observing that (i) unbounded disjunctive combination ($\vee\mathbf{C}$) and forgetting ($\mathbf{FO}$) are not satisfied by OBDD; (ii) any OBDD formula can be viewed as an ADD formula, and thus be translated in polynomial time into an $\texttt{SLDD}_+$ (resp. $\texttt{SLDD}_\times$, AADD) formula; and (iii) the disjunction of several OBDD formulæ amounts to cutting their $+$-combination at the minimal level ($\mathbf{CUT}_{\min}$ is satisfied): by construction, the countermodels of the disjunction are the $\vec{x}$ such that $\varphi(\vec{x}) = 0$, so the resulting formula can be viewed as an OBDD formula the negation of which is equivalent to the disjunction of the original formulæ). The other proofs are based on similar arguments. Note that contrary to the OBDD case, even $\odot$-eliminating a single variable is hard (roughly speaking, the $\odot$-combination of two formulæ can be built by $\odot$-eliminating an additional variable).

Proposition 4.8. *For any* $\texttt{L} \in \{\texttt{ADD}, \texttt{SLDD}_+, \texttt{SLDD}_\times, \texttt{AADD}\}$ *and any* $\odot \in \{\max, \min, +, \times\}$, L *does not satisfy* $\odot\mathbf{C}$, $\odot\mathbf{Elim}$, *or* $\mathbf{S}\odot\mathbf{Elim}$.

More difficult is the question of bounded combination. An extension of "apply" algorithm of Bryant (1986), which is polynomial-time on OBDD structures for the AND and OR operators, has been proposed (Sanner and McAllester 2005) to compute the combination (e.g., by $+$, $\times$, max, min) of two AADD formulæ; it can be easily adapted to the other VDD languages considered in this paper. However, the complexity of this "apply" algorithm had not been formally identified. One of the main results of this paper is that bounded combinations are not tractable for AADD, which implies that the extended "apply" is not a polynomial-time algorithm.

Proposition 4.9.

- $\texttt{SLDD}_+$, $\texttt{SLDD}_\times$, *and* AADD *do not satisfy* $\max\mathbf{BC}$, $\min\mathbf{BC}$, $\mathbf{SB}\max\mathbf{Elim}$, *or* $\mathbf{SB}\min\mathbf{Elim}$.
- $\texttt{SLDD}_\times$ *and* AADD *do not satisfy* $+\mathbf{BC}$ *or* $\mathbf{SB}+\mathbf{Elim}$.
- $\texttt{SLDD}_+$ *and* AADD *do not satisfy* $\times\mathbf{BC}$ *or* $\mathbf{SB}\times\mathbf{Elim}$.

The difficulty of $\max\mathbf{BC}$ or $\min\mathbf{BC}$ comes from that of $\mathbf{CUT}_{\succeq\gamma}$and $\mathbf{CUT}_{\preceq\gamma}$. We show the difficulty of $\times\mathbf{BC}$ by considering the two following functions on Boolean variables: $f(\vec{x}) = \sum_{i=0}^{n-1} x_i \cdot 2^i$ (representation of an integer by a bitvector) and $g(\vec{x}) = 2^{n+1} - f(\vec{x})$; each can be represented as an $\texttt{SLDD}_+$ formula (with ordering $x_0 \lhd x_1 \lhd \cdots \lhd x_{n-1}$) with $n+1$ nodes and $2n$ arcs. Then we can show that the $\texttt{SLDD}_+$ representation of $f \times g$ (using the same variable ordering) contains an exponential number of terminal arcs, even if all nodes at the last level have been normalized; this proves that each ordered $\texttt{SLDD}_+$ representation of $f \times g$ is of exponential size. The other proofs are similar, using well-chosen functions f and g.

There are actually only two cases in which bounded combinations are tractable: when the language considered is ADD, and when the combination operator is also the one that aggregates arc values in the language considered (i.e., addition for $\texttt{SLDD}_+$, and product for $\texttt{SLDD}_\times$). In these cases, it is also polynomial to eliminate a single variable with a bounded domain, by definition of variable elimination.

Table 2: Results about transformations (legend in Table 1).

Transformation	ADD	$\texttt{SLDD}_+$	$\texttt{SLDD}_\times$	AADD
$\max\mathbf{BC}$ / $\min\mathbf{BC}$	√	•	•	•
$+\mathbf{BC}$	√	√	•	•
$\times\mathbf{BC}$	√	•	√	•
$\max\mathbf{C}$ / $\min\mathbf{C}$	•	•	•	•
$+\mathbf{C}$ / $\times\mathbf{C}$	•	•	•	•
$\max\mathbf{Elim}$ / $\min\mathbf{Elim}$	•	•	•	•
$+\mathbf{Elim}$ / $\times\mathbf{Elim}$	•	•	•	•
$\mathbf{S}\max\mathbf{Elim}$ / $\mathbf{S}\min\mathbf{Elim}$	•	•	•	•
$\mathbf{S}+\mathbf{Elim}$ / $\mathbf{S}\times\mathbf{Elim}$	•	•	•	•
$\mathbf{SB}\max\mathbf{Elim}$ / $\mathbf{SB}\min\mathbf{Elim}$	√	•	•	•
$\mathbf{SB}+\mathbf{Elim}$	√	√	•	•
$\mathbf{SB}\times\mathbf{Elim}$	√	•	√	•
$\max\mathbf{Marg}$ / $\min\mathbf{Marg}$	√	√	√	√
$+\mathbf{Marg}$	√	√	√	√
$\times\mathbf{Marg}$	√	?	√	?

Proposition 4.10.

- $\texttt{SLDD}_+$ *satisfies* $+\mathbf{BC}$ *and* $\mathbf{SB}+\mathbf{Elim}$.
- $\texttt{SLDD}_\times$ *satisfies* $\times\mathbf{BC}$ *and* $\mathbf{SB}\times\mathbf{Elim}$.
- ADD *satisfies* $\odot\mathbf{BC}$ *and* $\mathbf{SB}\odot\mathbf{Elim}$, *for any operation* $\odot \in \{\times, +, \min, \max\}$.

Finally, we have proved that $\times$-marginalization is tractable for ADD and $\texttt{SLDD}_\times$, and that $\max$-marginalization, $\min$-marginalization and $+$-marginalization are tractable for all languages considered. Whether AADD and $\texttt{SLDD}_+$ satisfy $\times$-marginalization remains an open question.

5 Conclusion

In this paper, we presented a complexity analysis of VDD languages, based on a set of queries and transformations of interest. Requests related to optimization appear as tractable for all VDD languages, as well as additive marginalization and some of the queries related to cutting. Cutting queries prove computationally difficult when a level γ is has to be reached exactly, and this is also the case for the transformations we considered. One of the main results of the paper is that bounded combinations are not tractable on VDD languages, which implies that no "apply" algorithm can run in polynomial time on AADD formulæ in the general case (this is the case even for simple "Boolean-style" operations such as $\min$ and $\max$). When bounded additive (resp. multiplicative) combination is required to be achieved efficiently, the time/space tradeoff offered by $\texttt{SLDD}_+$ (resp. $\texttt{SLDD}_\times$) is valuable. Finally, it turns out that the complexity of the various queries related to optimization is better for the VDD languages than for other languages dedicated to the representation of non-Boolean functions, such as $\texttt{VCSP}_+$ (and similar results are expected for GAI nets or Bayes nets, for which optimization is also hard).

Acknowledgements

This work was partially supported by the project BR4CP ANR-11-BS02-008 of the French National Agency for Research.

References

Amilhastre, J.; Fargier, H.; Niveau, A.; and Pralet, C. 2012. Compiling CSPs: A complexity map of (non-deterministic) multivalued decision diagrams. In *Proceedings of ICTAI'12*, 1–8.

Amilhastre, J.; Fargier, H.; and Marquis, P. 2002. Consistency restoration and explanations in dynamic CSPs: Application to configuration. *Artificial Intelligence* 135(1–2):199–234.

Astesana, J.-M.; Cosserat, L.; and Fargier, H. 2010. Constraint-based vehicle configuration: A case study. In *Proceedings of ICTAI'10*, 68–75.

Bacchus, F., and Grove, A. J. 1995. Graphical models for preference and utility. In *Proceedings of UAI'95*, 3–10.

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proceedings of ICCAD'93*, 188–191.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* 35:677–691.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)* 17:229–264.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Fargier, H.; Marquis, P.; and Schmidt, N. 2013. Semiring labelled decision diagrams, revisited: Canonicity and spatial efficiency issues. In *Proceedings of IJCAI'13*, 884–890.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gogic, G.; Kautz, H.; Papadimitriou, C.; and Selman, B. 1995. The comparative linguistics of knowledge representation. In *Proceedings of IJCAI'95*, 862–869.

Hoey, J.; St-Aubin, R.; Hu, A. J.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of UAI'99*, 279–288.

Lai, Y.-T., and Sastry, S. 1992. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Proceedings of DAC'92*, 608–613.

Lai, Y.-T.; Pedram, M.; and Vrudhula, S. B. K. 1996. Formal verification using edge-valued binary decision diagrams. *IEEE Transactions on Computers* 45(2):247–255.

Pearl, J. 1989. *Probabilistic reasoning in intelligent systems – networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann.

Sanner, S., and McAllester, D. A. 2005. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proceedings of IJCAI'05*, 1384–1390.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of IJCAI'95 (1)*, 631–639.

Srinivasan, A.; Kam, T.; Malik, S.; and Brayton, R. K. 1990. Algorithms for discrete function manipulation. In *Proceedings of ICCAD'90*, 92–95.

Tafertshofer, P., and Pedram, M. 1997. Factored edge-valued binary decision diagrams. *Formal Methods in System Design* 10(2/3).

Wilson, N. 2005. Decision diagrams for the computation of semiring valuations. In *Proceedings of IJCAI'05*, 331–336.