Reduced Ordered Binary Decision Diagrams as compacted tree-structures: Enumeration and Sampling

Antoine Genitrini (Sorbonne Université)
with the collaboration of Julien Clément (CNRS)

The representation of a Boolean function as a binary decision tree has been used for decades. Its main benefit, compared to other representations like a truth table or a Boolean circuit, comes from the underlying *divide-and-conquer* paradigm. Thirty years ago a new data structure emerged, based on the compaction of binary decision trees, and hereafter denoted as Binary Decision Diagrams (or BDDs) [Bry86]. Its takeoff has been so spectacular that many variants of compacted structures have been developed, and called through many acronyms as presented in [Weg00]. One way to represent the different diagrams consists in their embedding as directed acyclic graphs (or DAGs). One reason for the existence of all these variants of diagrams is due to the fact that each DAG correspondence has its own internal agency of the nodes and thus each representation is oriented towards a specific constraint. For example, the case of Reduced Ordered Binary Decision Diagrams (ROBDDs) is such that the variables do appear at most once and in the same order along any path from the source to a sink of the DAG, and furthermore, no two occurrences of the same subgraph do appear in the structure. For such structures and others, like QOBDDs or ZBDDs for example, there is a canonical representation of each Boolean function.

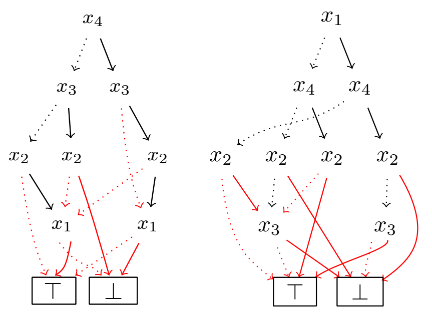

Figure 1: Two Reduced Ordered Binary Decision Diagrams associated to the same Boolean function. Nodes are labeled with Boolean variables; left dotted edges (resp. right solid edges) are `False` links (resp. `True` links).

In his book [Knu11] Knuth proves or recalls combinatorial results, like properties for the profile of a BDD, or the way to combine two structures to represent a more complex function. However, one notes an unseemly fact. There are no results about the distribution of the Boolean functions according to their ROBDD size. In fact in contrast to (e.g.) binary trees where there is a recursive characterization that allows to well specify the trees, we have no local-constraint here for ROBDDs ans thus a similar recurrence is unexpected. Very recently, there is a first study exploring experimentally, numerically, and theoretically the typical and worst-case ROBDD sizes in [NV19]. We aim at obtaining the same kind of combinatorial results but here we design a partition of the decision diagrams that allows us to go much further in terms of size. In particular we obtain an exhaustive enumeration of the diagrams according to their size up to 9 variables. This was unreachable through the exhaustive approach proposed in [NV19] due to the double exponential complexity of the problem: there are 2^{2^k} Boolean functions with k variables. Our C++ implementation fully manages the case of 9 variables (see Fig 2) that corresponds to $2^{512} \approx 10^{154}$ functions. In particular for 9 Boolean variables, our implementation shows one seventh of all ROBDDs are of size 132 (the possible sizes range from 3 to 143). Furthermore, ROBDDs of size between 125 and 143 represents more than 99.8% of all ROBDDs, in accordance with theoretical results from [VB04, GPS04].

Starting from the well-known compaction process (that takes a binary decision tree and outputs its compacted form, the ROBDD), our combinatorial study gives a way of construction for ROBDDs of a given size, but without the compaction step. We further define a total order over the set of ROBDDs and we propose both an unranking and an exhaustive generation algorithm. The first one gives as a by-product a uniform random sampler for ROBDDs of a given number of variables and size. One strength of our approach is that it allows to sample *uniformly* ROBDDs of "small" size, for instance of linear size w.r.t the number k of variables, very efficiently in contrast to a naive rejection algorithm. The usual uniform distribution on Boolean functions [VB04] yields with high probability ROBDDs of near maximal size of order $2^k/k$, although ROBDDs encountered in applications, when tractable, are smaller. As a perspective, once the unranking method is well understood, and in particular the poset underlying the ROBDDs, then we might be able to bias the distribution to sample only in a specific subclass, e.g. ROBDDs corresponding to a particular class of formulas (e.g. read-once formulas).

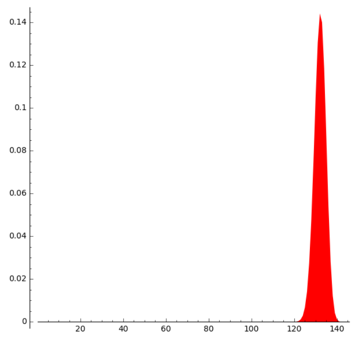

Figure 2: Proportion of BDDs over 9 variables according to their size

Our results have practical applications in several contexts, in particular for testing structures and algorithms. The study given in [DHT03] executes tests for an algorithm whose parameter is a binary decision diagram. It is based on QuickCheck [CH00], the famous software, taking as an entry a random generator and generating test cases for test suites. Using our uniform generator, we aim at obtaining statistical testing, in the sense that the underlying distribution of the samples is uniform, thus allowing to extract statistics thanks to the tests. Another application of our approach allows to derive exhaustive testing for small structures, like the study in [MAD$^+$03], that we also can conduct inside QuickCheck.

During the talk we will first describe our new combinatorial specification allowing to describe ROBDDs and then allowing an efficient enumeration. In a second step we will give the key-points how to obtain an unranking algorithm to build such structures. As a last part we will present a toy example how to use our generator in the context of property-based testing.
As a conclusion we will present other decision tree models where the method for which the enumeration and sampling can be adapted and through which direction we think to improve our approach.

The results are published in the conference paper [CG20].

References

[Bry86] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[CG20] J. Clément and A. Genitrini. Binary Decision Diagrams: from Tree Compaction to Sampling. In *14th Latin American Theoretical Informatics Symposium (LATIN)*, page To appear, 2020.

[CH00] K. Claessen and J. Hughes. Quickcheck: A lightweight tool for random testing of haskell programs. *SIGPLAN Not.*, 35(9):268–279, 2000.

[DHT03] P. Dybjer, Q. Haiyan, and M. Takeyama. Verifying Haskell Programs by Combining Testing and Proving. In *QSIC'03*, pages 272–279, 2003.

[GPS04] Clemens Gröpl, Hans Jürgen Prömel, and Anand Srivastav. Ordered binary decision diagrams and the shannon effect. *Discrete Applied Mathematics*, 142(1):67 – 85, 2004.

[Knu11] D. E. Knuth. *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms.* Addison-Wesley Professional, 2011.

[MAD$^+$03] D. Marinov, A. Andoni, D. Daniliuc, S. Khurshid, and M. Rinard. An evaluation of exhaustive testing for data structures. Technical report, MIT -LCS-TR-921, 2003.

[NV19] J. Newton and D. Verna. A theoretical and numerical analysis of the worst-case size of reduced ordered binary decision diagrams. *ACM TCL*, 20(1):6:1–6:36, 2019.

[VB04] J. Vuillemin and Fr. Béal. On the BDD of a Random Boolean Function. In *ASIAN'04*, pages 483–493, 2004.

[Weg00] I. Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM, 2000.