Supplement: Decision Diagram Representations of Selector Functions*

Randal E. Bryant
Carnegie Mellon University

December 20, 2019

Notice

The material in this document is supplementary material to publication [3]. This document is being made available to the public, subject to copyright provisions. You are free to copy and distribute it, but you must give attribution for any use of this material.

This work was supported, in part, by NSF STARSS grant 1525527.

Overview

For Boolean variables $x_1, \ldots, x_n$, a *selector function* is defined as a Boolean function of the form:

$$f(x_1, \ldots, x_n) \quad = \quad \bigwedge_{i=1}^{n} \widetilde{x}_i \tag{1}$$

where for each value of i, $\widetilde{x}_i$ is a *literal*, equal to either x_i or $\neg x_i$. We encode these literals with a set of *phase* values p_i, where $p_i = 1$ when $\widetilde{x}_i = x_i$, and $p_i = 0$ when $\widetilde{x}_i = \neg x_i$.

In this document, we consider the complexity of encoding such functions with different types of decision diagrams (DDs):

- Reduced Ordered Binary Decision Diagrams (BDDs) [4]
- Zero-suppressed Binary Decision Diagrams (ZDDs) [5, 6]
- Chain-reduced Binary Binary Decision Diagrams (CBDDs) [2, 3]
- Edge-Specified-Reduction Binary Decision Diagrams (ESRBDDs) [1]

None of the DD types requires more than $n + 2$ nodes, including the two leaf nodes. Here we consider the average-case sizes, assuming the phase values $p_1, \ldots, p_n$ are independent and uniformly distributed. That is, each phase p_i is either 0 or 1 with probability $1/2$, and the value of p_i is independent from that of any other phase p_j. We also derive the worst-case sizes.

DD	Average	Worst Case
BDD	1	1
CBDD	$3/4$	1
ZDD	$1/2$	1
ESRBDD	$1/3$	$1/2$

Table 1: **Size coefficients α_{avg} for representing selector functions with different DD types**

We derive results showing that each DD type has an average and worst-case size of the form $\alpha\, n + 2$, where the coefficients α_{avg} (average) and α_{wc} (worst case) are shown in Table 1. The first three average-case results are presented in [3]. The results for ESRBDDs are new to this document.

As described in [3], these average sizes are significant when using a DD to represent a set of strings, such as when representing a dictionary or a set of configurations of a chess board. For strings over some alphabet A having $|A| = r$, we can use a *binary* encoding to represent each symbol using $k = \lceil \log_2 r \rceil$ Boolean variables, and we can then represent a string of length m as a selector function with with $m \cdot k$ Boolean variables. The function representing a set of strings is a disjunction of selector functions, and hence there can be sharing among nodes in the DD representation. Still, the different coefficients of Table 1 can indicate that the four different DD types are more or less compact when representing sets of this form.

Benchmark	BDD	CBDD		ZDD		ESRBDD	
	Nodes	Nodes	Ratio	Nodes	Ratio	Nodes	Ratio
Word compact	1,120,437	971,387	0.87	657,969	0.59	484,765	0.43
Word full	1,285,285	1,153,438	0.90	851,555	0.66	520,576	0.41
Password compact	5,704,777	4,542,925	0.80	2,960,478	0.52	2,399,272	0.42
Password full	5,648,670	4,960,446	0.88	3,532,847	0.63	2,410,589	0.43

Table 2: **Benchmark results for dictionary representations**. Data taken from protect[1]. The size ratios are relative to BDDs.

Table 2 displays data on the sizes of the DDs generated when representing four dictionaries using binary codings of the characters. The data are taken from [1]. The table also shows the ratios of sizes relative to those of BDDs. We see that these results slightly exceed predictions based on the average case results of Table 1. Several factors could explain why the measured numbers do not match those of Table 1:

- The dictionaries consist of sets of words, not just individual words, and so the DDs encode disjunctions of selector functions.
- There can be a sharing of nodes among the different words, and this sharing may be greater with the more homogeneously structured BDDs than for other DD types.

- The probabilities of the phase values are not independent and are not uniformly distributed.

Further investigation would be required to fully quantify these phenomena. For here, it suffices that they indicate a general trend of BDDs, CBDDs, ZDDs, and ESRBDDs being progressively more compact.

BDDs

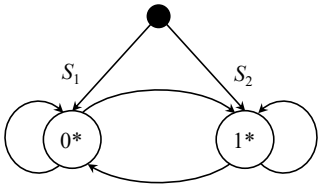

Figure 1: **State machine for BDD node generation.** The forward phase string is processed starting at the initial state (solid circle). Each state marked with an asterisk indicates that a node must be generated.

The size of the BDD representation of a selector function is straightforward: there must be a nonleaf node for each level i, as well as two leaf nodes.

This simple case allows us to introduce a *state machine* representation for generating the DD nodes of a selector function, as illustrated in Figure 1. Define the *forward phase string* p as $p = p_1\, p_2\, \cdots\, p_n$. Generating the BDD starts at the initial state (solid circle) in Figure 1. It then processes the string, transitioning from the current state to the (unique) state labeled by p_i on each step i. Those states that are labeled by an asterisk ('*') denote ones that require a new DD node.

We can also view the state machine of Fig. 1 as a Markov chain, where each arc denotes a transition probability of $1/2$. Transitions among the states S_1 and S_2 can then be represented by the transition matrix:

$$T_{\mathrm{BDD}} \;=\; \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \tag{2}$$

The set of steady-state probabilites for the two states is then the unique value $\vec{s}$ such that: 1) $s_i \geq 0$ for all i, 2) $\sum_{i=1,n} s_i = 1$, and 3) $T\vec{s} = \vec{s}$. In this case, one can easily see that $s_1 = s_2 = 1/2$, and since both states require generating a BDD node, the average number of nodes generated per level will be $\alpha_{\mathrm{avg}} = 1/2 + 1/2 = 1$.

The worst-case result of $\alpha_{\mathrm{wc}} = 1$ is straightforward—every BDD representation of a selector function must have $n + 2$ nodes.

CBDDs

When chain suppression is applied to BDDs, a consecutive set of phase values $p_i, p_{i+1}, \ldots, p_{i+k}$ equal to 0 can be encoded as a single node have top level $t = i$ and bottom level $b = i + k$. This property is

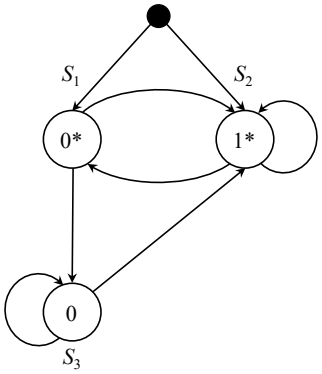

Figure 2: **State machine for CBDD node generation.** The additional state S_3 denotes a position where chain reduction applies.

represented by the state machine of Figure 2. The additional state S_3 encodes the processing of the substring $p_{i+1} \cdots p_{i+k}$—no nodes are generated.

The transition matrix for the corresponding Markov chain is:

$$T_{\text{CBDD}} = \begin{bmatrix} 1/2 & 1/2 & 1/2 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1/2 \end{bmatrix} \tag{3}$$

The steady-state probabilities for this matrix have $s_2 = 1/2$ and $s_1 = s_3 = 1/4$. The average number of nodes generated per level will therefore be $\alpha_{\text{avg}} = 1/2 + 1/4 = 3/4$.

The worst-case result of $\alpha_{\text{wc}} = 1$ can be seen when $p_i = 1$ for all i. Only state S_2 will be visited during the generation process.

ZDDs

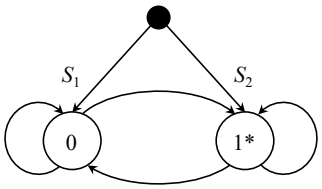

Figure 3: **State machine for ZDD node generation.** Only state S_2 requires generating a node.

The zero suppression rule of ZDDs avoids the need to have a node at level i when $p_i = 0$. We can therefore represent the node generation process with the state machine shown in Figure 3. It is identical to the one for BDDs, except that transitions to state S_1 do not require generating a node. The steady-state probabilities will be $s_1 = s_2 = 1/2$, and the average number of non-leaf nodes generated per level will be $\alpha_{\text{avg}} = 1/2$.

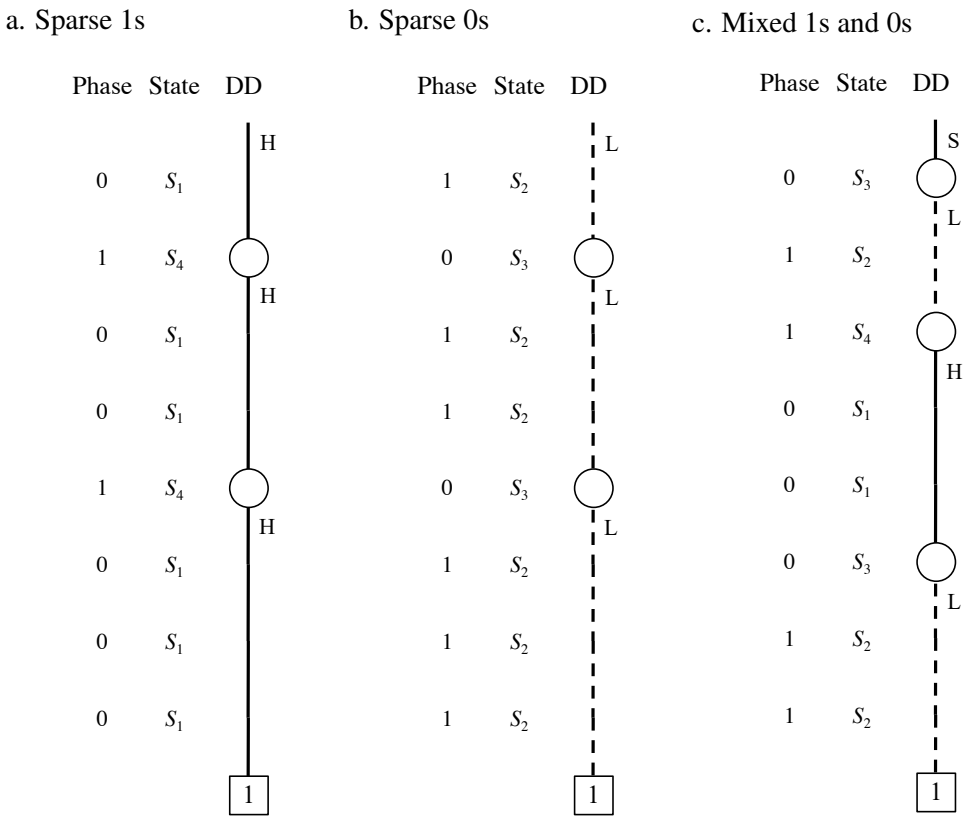

Figure 4: **ESRBDD selector function representation examples**. Each edge is labeled as high zero-suppressed (H), low zero-suppressed (L), or short (S)

The worst-case result of $\alpha_{\text{wc}} = 1$ can be seen when $p_i = 1$ for all i. Only state S_2 will be visited during the generation process.

ESRBDDs

ESRBDDs have edges with four different labels:

S The edge is between nodes at adjacent levels, or it is the initial edge to a node at level 1.

X The skipped levels should be treated as don't cares.

H The skipped levels should be *high-zero* suppressed. That is, the function will yield 0 if any of the skipped variables are assigned value 1.

L The skipped levels should be *low-zero suppressed*. That is, the function will yield 0 if any of the skipped variables are assigned value 0.

When representing a selector function, there are no edges with label X. Restricting the labels to S and H would yield a ZDD representation. Label L introduces a case where a level with phase value $p_i = 1$ need not have a node at that level.

Figure 4 illustrates the ESRBDD for three example selector functions. When drawing DDs, we do not show the leaf with value 0 nor any edges to it. The *lo* edge from a node is shown as a dashed line, and the *hi* edge is shown as a solid line.

Figure 4a illustrates the case where the phases with value 1 are sparse, i.e., no two adjacent levels have phase 1. The ESRBDD structure matches that of a ZDD, with nodes only at the levels with phase value 0. The edges are labeled H, including the edge to the root, to indicate that the skipped levels are high-zero suppressed. Conversely, when the phases with value 0 are sparse (4b), the representation can take advantage of low-zero suppression to avoid any node at a level with phase 1. When the function has a mix of consecutive 1s and consecutive 0s (4c), a different pattern emerges. Here, a node occurs at the lowest level of each run to indicate the phase change.

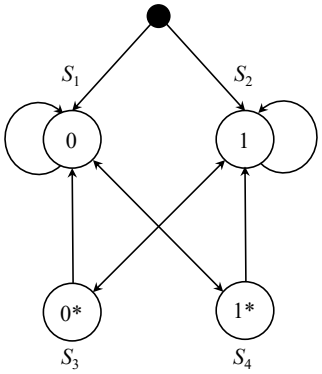

Figure 5: **State machine for ESRBDD node generation.** Nodes are generated when processing the reverse of the phase string. The two zero-suppression rules yield symmetrical cases where no new DD node is required.

These patterns are captured by the state machine of Figure 5. Since the reduction rules are applied from the leaf nodes upward, the state-machine representation of this process must consider the *reversed phase string* $p^R = p_n\, p_{n-1} \cdots p_1$. The three examples of Figure 4 also indicate the states that would occur when processing the phase values from the bottom up. For a string with sparse 1s (4a), only states S_1 and S_4 are encountered. Similarly, generating the DD when processing a string with sparse 0s (4b) encounters only states S_2 and S_3. With a mix of 1s and 0s (4c), transitioning from a series of consecutive 0s to 1 (resp., 1s to 0) involves a transition from S_1 to S_4 (resp., S_2 to S_3), generating a new node.

The transition matrix for the corresponding Markov chain is:

$$T_{\mathrm{ESRBDD}} \;=\; \begin{bmatrix} 1/2 & 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \end{bmatrix} \tag{4}$$

The steady-state probabilities for this matrix have $s_1 = s_2 = 1/3$ and $s_3 = s_4 = 1/6$. The average number of nodes generated will per level will therefore be $\alpha_{\text{avg}} = 1/6 + 1/6 = 1/3$

Unlike the other DD types, the worst-case analysis for ESRBDDs is nontrivial. We can see that a string consisting of $n/2$ alternations of 0 and 1 will yield a ESRBDD with $n/2 + 2$ nodes, either transitioning between states S_1 and S_4 or between states S_2 and S_3, depending on p_n. Indeed, this pattern is the worst case, since there are no self loops for the two states that require generating a node, and there are no transitions between these states. We therefore conclude the $\alpha_{\text{wc}} = 1/2$ for ESRBDDs.

References

[1] J. Babar, C Jiang, G. Ciardo, and A. Miner. Binary decision diagrams with edge-specified reductions. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 11428 of *Lecture Notes in Computer Science*, pages 303–318, 2019.

[2] R. E. Bryant. Chain reduction for binary and zero-suppressed decision diagrams. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 10805 of *Lecture Notes in Computer Science*, pages 81–98, 2018.

[3] R. E. Bryant. Chain reduction for binary and zero-suppressed decision diagrams. *Journal of Automated Reasoning*, 2020.

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

[5] Shin-ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pages 272–277, June 1993.

[6] Shin-ichi Minato. *Binary Decision Diagrams and Applications for VLSI CAD*. Kluwer Academic Publishers, 1995.