# New Techniques for Compact Representation of Boolean Functions

student: **Ján Maťufka** supervisor: Ing. **Ondřej Lengál**, PhD. consultant: Ing. **Vojtěch Havlena**, PhD.

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

## Aims and motivation

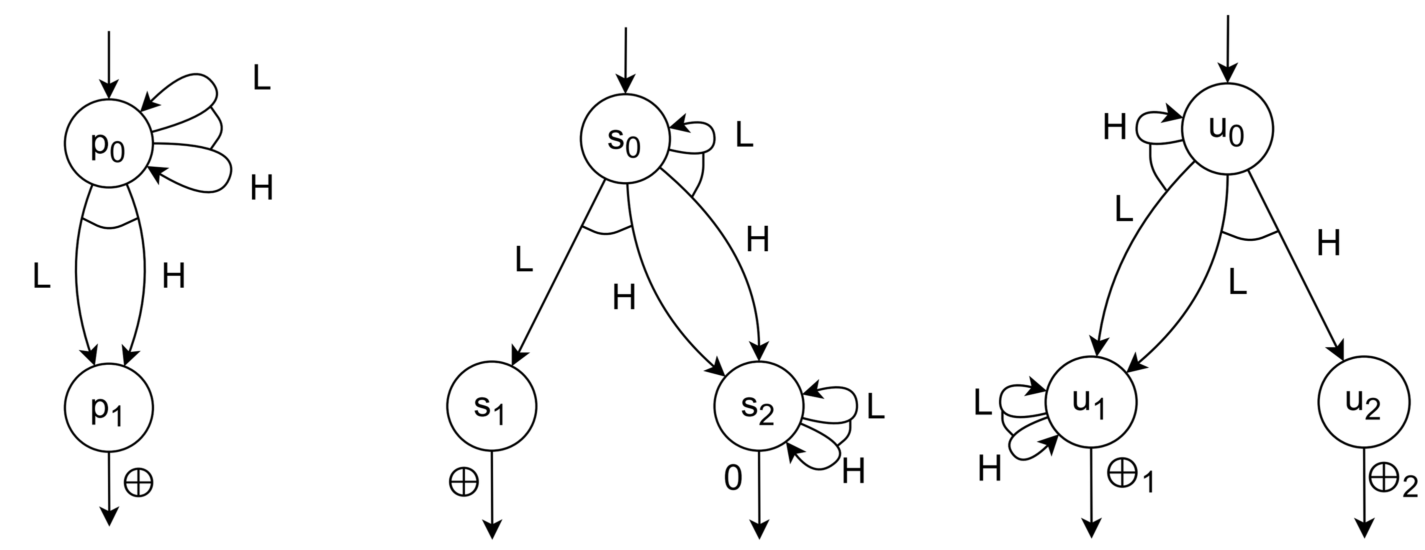
- Provide an extensible and **customizable automaton framework** that can be used to **represent Boolean functions**.
  - Since this model is based on **tree-automata** [5], it will be called Automata-based binary decision diagrams (ABDDs).
- This model should be able to **generalize** (or encompass) all edge-specified binary decision diagrams **reduction rules** from other models:
  - Binary decision diagrams(**BDDs** or ROBDDs – reduced ordered BDDs),
  - Zero-suppressed binary decision diagrams (**ZBDDs**) [6],
  - Tagged BDDs (**TBDDs**) [8],
  - Chain-reduced binary decision diagrams (CBDDs, CZDDs) [4],
  - Binary decision diagrams with edge-specified reductions (**ESRBDDs**) [1].
- Develop algorithms that work with this data structure:
  - Converting the data structure into a **canonical form** – unfolding, normalization, folding.
  - Applying **Boolean operations** over ABDDs (conjunction, disjunction, etc.).
  - Testing **satisfiability**.
- Experimentally evaluate the compactness of ABDDs on a set of benchmarks and compare it to other models.

## Applying reductions – Tree Automata

This project uses special tree automata (in the rest of the poster they will be referred to as **"boxes"**). Boxes should be well-defined (this will help with obtaining a canonical form), which means that they have to have these properties:

### Box properties

- **Non-emptiness** – the language of the box is non-empty.
- **Trimness** – the box does not contain unreachable states (top-down and bottom-up).
- **Root-uniqueness** – box has exactly one root state.
- **Port-consistency** – for every tree from the language of the box there are the same ports used and that their amount is consisent – the number of different ports is called **arity** of the box.
- **Port-uniqueness** – every different port has exactly one state with the corresponding outgoing port-transition.
- **Non-vacuity** (or "non-obsoleteness") – there is no port transition from the root state.
- **Unambiguity** wrt. the variables – if we map a variable to a root and a variable to each port-transition state, then there is at most 1 tree that such that the variable order is followed.




(a) Box $X$. (b) Box $H_0$. (c) Box $L_\oplus$

Figure 1. BDD reduction rule can be represented as box $X$ (can encapsulate don't care chains). ZBDD reduction rule can be represented as box $H_0$. Note that $H_0$ automaton has other 3 alternatives, $H_1$ (has 1 instead of 0 on the output edge) and $L_0$, $L_1$ – they have flipped $LH$ edges compared to $H_0$ and $H_1$. $L_\oplus$ is a more generally applicable reduction rule – as it does not require a leaf node to be viable. This means it can be applied further from leaf nodes (this project also uses a $H_\oplus$ tree automaton as a reduction rule, which just has flipped $LH$ edges).

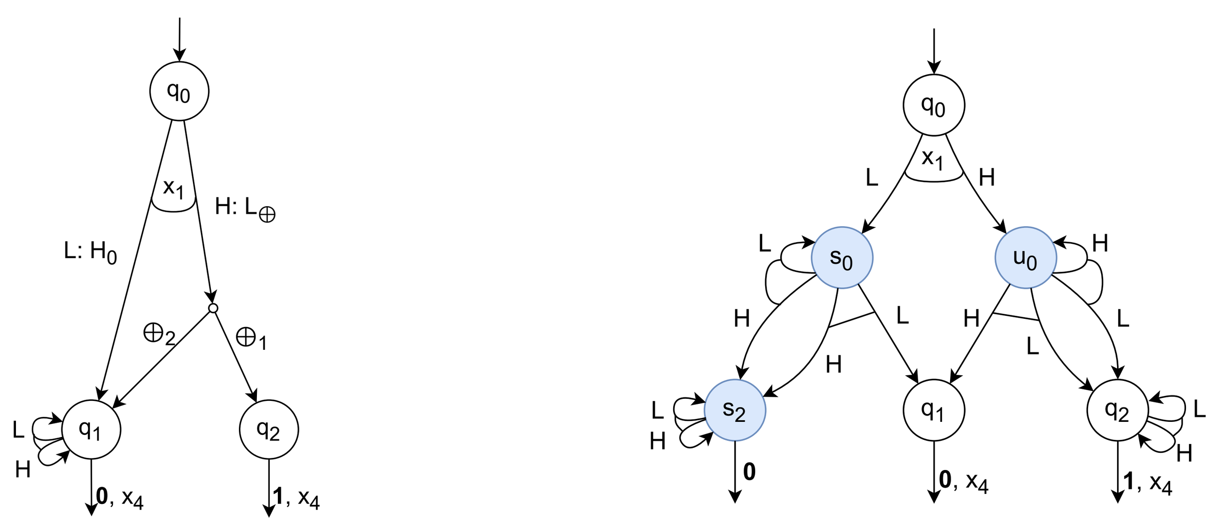
## Canonization and other algorithms

For obtaining a canonical representation of an ABDD, there are three operations that need to be implemented:

1. **Unfolding** – removing boxes from edges and replacing them with the corresponding tree automaton with the correct port-state mapping (see 2).
2. **Normalization** – bottom-up determinization with regards to the variables and saturation of the structure with variables where it is possible to count them.
   - After normalization, no equivalent nodes are present in the binary decision automaton structure.
3. **Folding** – replace repeating patterns with the boxes in a given box order, such that the boxes replace patterns in the maximum possible degree.
   - It utilizes an operation similar to an intersection of two tree automata (an "intersectoid").
   - The difference is that port transitions take prevalence over transitions and have to be put into the result.

Other than that, other important functions over the ABDD structure are:

- **Testing satisfiability** – combination of unfolding and tree-traversal with backtracking using variable assignment.
- **Applying boolean operators** – unfold inputs, unwind cycles (using variables as bounds), then use apply similarly to BDDs, and then normalize and fold into ABDD.




(a) Before unfolding. (b) After unfolding.

Figure 2. Demonstration of a binary decision automaton unfolding. The states in blue were added during the unfolding process and are named according to the state names in boxes from figure 1.

Simulating the decision diagram models was achieved using folding with regards to these box orders:

| | | | | | |
|---|---|---|---|---|---|
| BDDs – | $(X)$ | ZBDDs – | $(H_0)$ | TBDDs – | $(X, H_0)$ |
| ESRBDDs – | $(L_0, H_0, X)$ | CZDDs – | $(H_0, X)$ | CBDDs – | $(X, H_\oplus)$ |
| **ABDDs** – | $(L_0, H_0, L_1, H_1, X, L_\oplus, H_\oplus)$ | | | | |

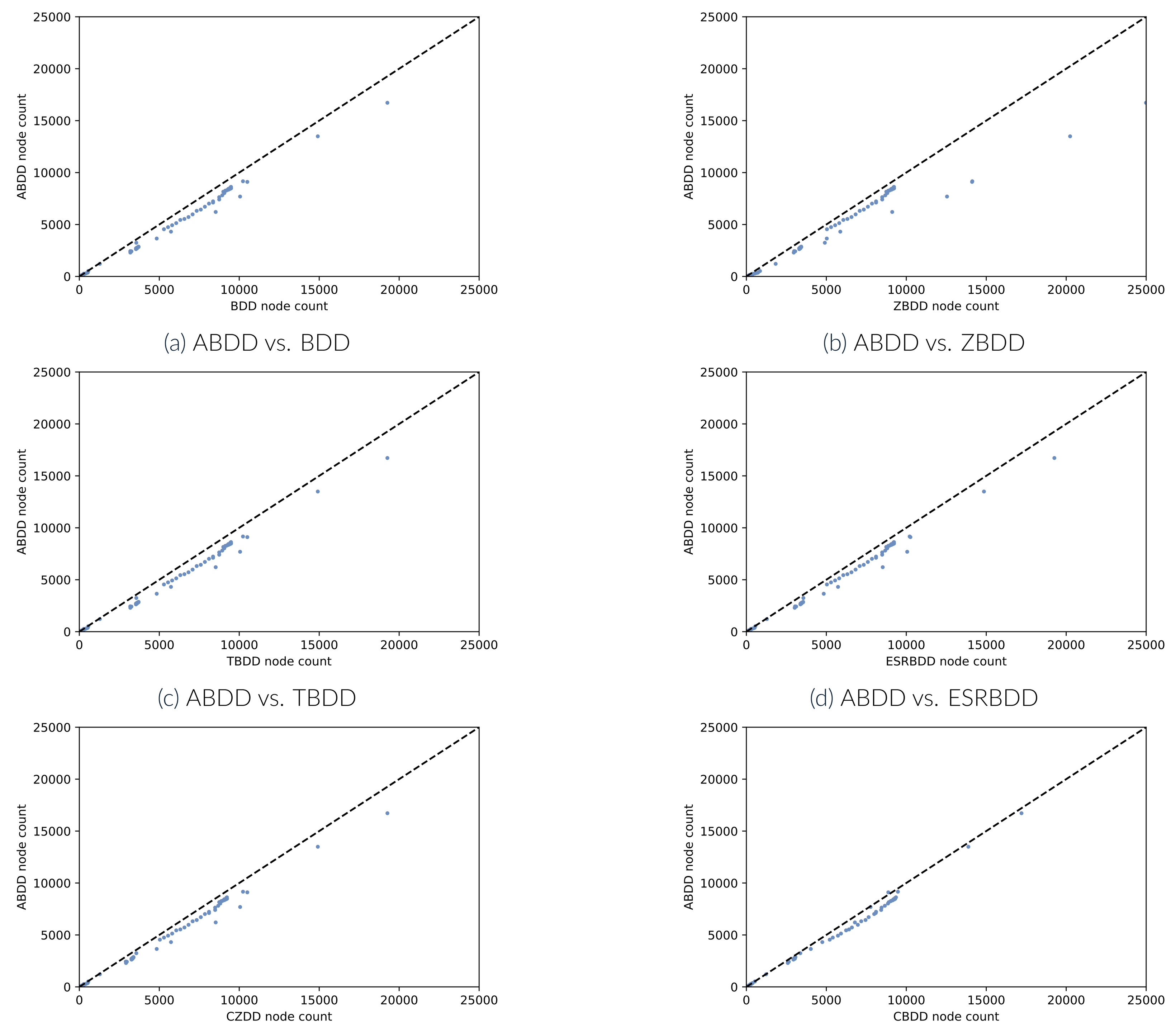
## Results

- Benchmarks were combinational circuits from LGSynth'91 [9].
- They represent 27-channel interrupt controllers, 32-bit SEC, 8-bit ALU, 16-bit SEC/DED.
- Prepared into multiple BDDs using BuDDy library [7].
- BDDs were unfolded, normalized, saturated with variables, and then folded using different box orders to simulate other models.

### Benchmark node counts

| Benchmark | BDD | ZBDD | TBDD | CZDD | CBDD | ESRBDD | ABDD |
|---|---|---|---|---|---|---|---|
| C1355 | 263520 | 255456 | 263520 | 255392 | 258203 | 255360 | **231640** |
| C1908 | 75111 | 72383 | 75095 | 70469 | 61918 | 72587 | **56598** |
| C432 | 2009 | 2821 | 2007 | 2023 | 1578 | 1958 | **1438** |
| C880 | 70645 | 94020 | 70645 | 70660 | 62608 | 70327 | **61414** |
| Total | 411285 | 424680 | 411267 | 398544 | 384307 | 400232 | **351090** |

Table 1. Node counts of tested benchmarks. Each node count marks the sum of multiple output functions from the particular benchmark. ABDDs achieved the lowest node counts of all tested models.




(a) ABDD vs. BDD (b) ABDD vs. ZBDD (c) ABDD vs. TBDD (d) ABDD vs. ESRBDD (e) ABDD vs. CZDD (f) ABDD vs. CBDD

Figure 3. On average, ABDDs achieved 17 % smaller node counts than BDDs (3a), 21 % less nodes than ZBDDs (3b), 17 % smaller node counts than TBDDs (3c), 14 % less nodes than ESRBDDs (3d), 14 % less nodes than CZDDs (3e), and 8.6 % more compact than CBDDs (3f). The smallest node counts on average (apart from ABDDs) were found in CBDDs and the largest in ZBDDs.

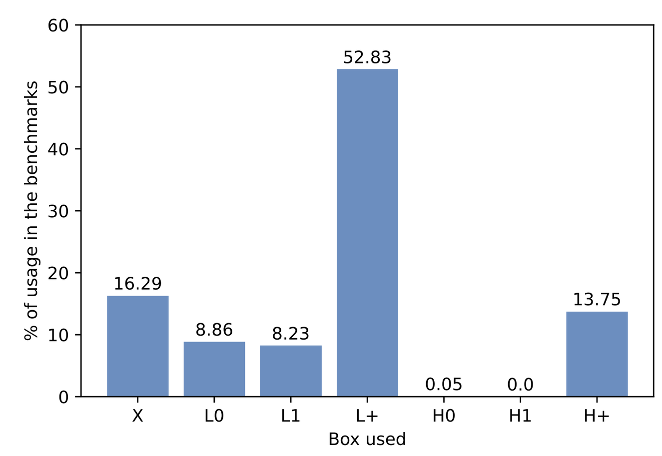



Figure 4. An overview of how often each type of box is used in ABDDs. Box $H_0$ has not been used at all, which was suggested in 3d, while the most often used box was the one introduced in this project – $L_\oplus$. The usage is, however, strongly dependant on the box order in which the folding is applied.

### Conclusion and Future work

- A framework was developed that could **generalize** various binary decision diagram models.
- This was achieved by using the properties of finite tree automata.
- ABDDs also show potential to achieve **better reduction results** than currently used models.
- There are many **improvements and optimizations** possible, as this research is in early stages:
  - better memory-management during bottom-up algorithms (reachability, normalization),
  - faster folding and normalization process,
  - applying boolean operations and testing satisfiability without the need for unfolding,
  - identifiying more patterns that can be reduced with the corresponding boxes that can reduce them,
  - extending it to work with multi-terminal BDDs (MTBDDs).

## References



[1] Junaid Babar, Chuan Jiang, Gianfranco Ciardo, and Andrew S. Miner. Binary decision diagrams with edge-specified reductions. In Tomáš Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II*, volume 11428 of *Lecture Notes in Computer Science*, pages 303–318. Springer, 2019.

[2] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[3] Randal E. Bryant. Binary decision diagrams. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 191–217, Cham, 2018. Springer International Publishing.

[4] Randal E. Bryant. Chain reduction for binary and zero-suppressed decision diagrams. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, volume 10805 of *Lecture Notes in Computer Science*, pages 81–98. Springer, 2018.

[5] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. http://www.grappa.univ-lille3.fr/tata, 2007. Release October, 12th 2007.

[6] Shin ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In Alfred E. Dunlop, editor, *Proceedings of the 30th Design Automation Conference. Dallas, Texas, USA, June 14-18, 1993*, pages 272–277. ACM Press, 1993.

[7] Jørn Lind-Nielsen. BuDDy. A Binary Decision Diagram Library. https://github.com/SSoelvsten/buddy.

[8] T. van Dijk, R. Wille, and R. Meolic. Tagged BDDs: Combining reduction rules from different decision diagram types. In *2017 Formal Methods in Computer Aided Design (FMCAD)*, pages 108–115. IEEE, 2017.

[9] S. Yang. Logic synthesis and optimization benchmarks user guide version 3.0. 1991. https://ddd.fit.cvut.cz/www/prj/Benchmarks/LGSynth91.pdf.