IOWA STATE UNIVERSITY Department of Computer Science

ACI-1642397 SI2-SSE: A next-generation decision diagram library

https://asminer.github.io/meddly/

Andrew Miner (PI)
Gianfranco Ciardo (Co-PI)

MEDDLY, Multi-terminal and Edge-valued Decision Diagram LibrarY, is an open-source decision diagram library written in C++.

Many computer-based scientific and engineering applications store, analyze, and manipulate large data. If this data has enough structure, specialized data structures and algorithms can have dramatically smaller memory and time requirements than explicit approaches. An important case is hardware and software verification, where binary decision diagrams (BDDs) have been successfully employed to study systems with enormous state spaces. Several software libraries have been written to support BDD creation and manipulation. These libraries have also encouraged researchers to use BDDs in other areas, e.g., search, planning, optimization, and satisfiability, in the hope to find and exploit structure often hidden in the data.

However, decision diagram theory has evolved beyond BDDs, with generalizations such as multi-way decisions (MDDs) and multi-valued or numerical outcomes encoded using multiple terminals (MTMDDs) or edge values (EVMDDs). New reduction rules to canonize decision diagrams (often resulting in smaller size) and algorithmic improvements have also been proposed. Unfortunately, libraries have not kept up with these advances.

MEDDLY seeks to fill this gap, by integrating and expanding two existing prototype libraries, a previous version of MEDDLY (designed by PI Miner at Iowa State University) and TEDDY (designed by Co-PI Ciardo at the University of California - Riverside), into a powerful, next-generation decision diagram library that supports a more general theory of decision diagrams.

The new library supports:

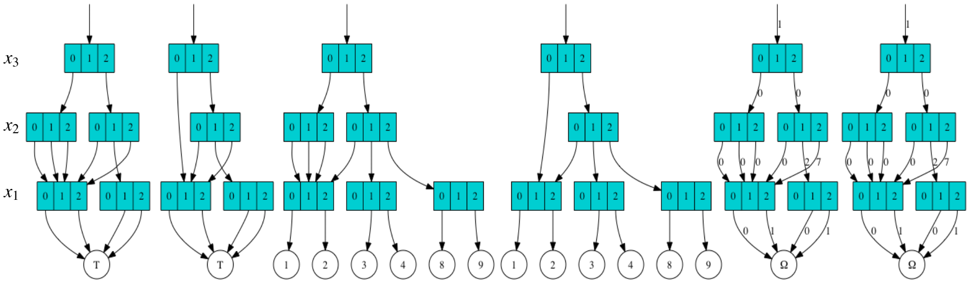
- non-binary variables, including finite discrete domains (possibly with unknown bounds) and even infinite discrete domains (under certain restrictions),
- non-boolean function values, encoding by attaching a value to either the terminal nodes or the edges of the decision diagram,
- a more general definition of canonicity that allows a wide spectrum of reduction rules that can be best at exploiting different structures in the data.

Several project activities will reduce the learning curve for users adopting MEDDLY, including proven methods (e.g., user manuals, tutorials, and examples) and novel methods such as the development of visualization techniques to help debugging and understanding decision diagrams, both statically and dynamically.

Modularity is one of MEDDLY's design goals; its major components interact via the user interface, designed to support future extensions. Users can define their own reduction rules, caching schemes, and decision diagram operations without requiring extensive knowledge of MEDDLY's internals.

Users of MEDDLY include:

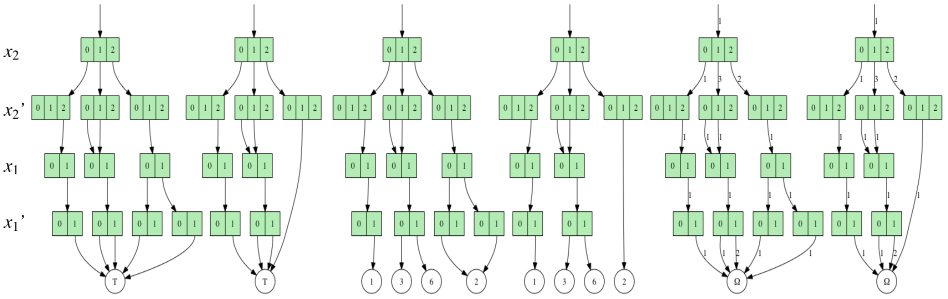
- University of Torino, Italy (*GreatSPN*, state space generation, model checking)
- Guilin University of Electronic Technology, China (constraint solving)
- Formal methods and tools group, University of Twente, the Netherlands
- University of Bergamo, Italy (*CitLAB*, combinatorial interaction testing)
- Research Group Foundations of AI, University of Freiburg, Germany (planning)



Decision diagrams encoding boolean or integer functions. Shown, left to right:

quasi-reduced and fully-reduced MDD encoding the same boolean function (i.e., a set)
$(x_3 = 2, x_2 = 0, x_1 = 2)$ => True
$(x_3 = 2, x_2 = 1, x_1 = 0)$ => False (default)

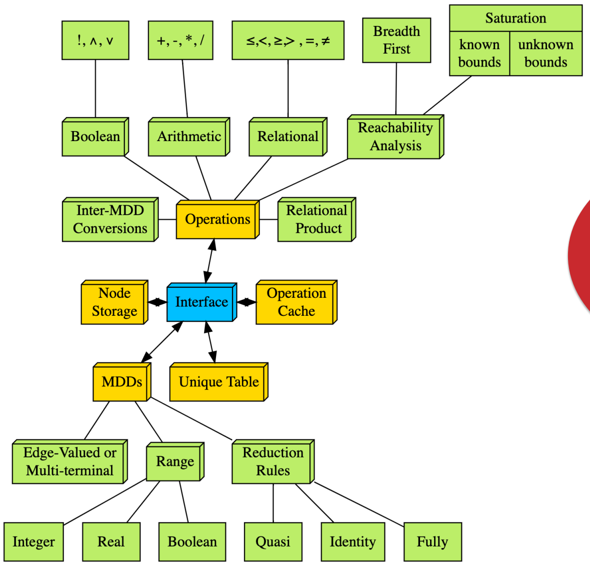
quasi-reduced and fully-reduced MTMDD, quasi-reduced and fully-reduced EV$^+$MDD, all encoding the same integer-valued function
$(x_3 = 2, x_2 = 1, x_1 = 1)$ => 3
$(x_3 = 2, x_2 = 2, x_1 = 1)$ => Infinite (default)



Decision diagrams encoding boolean or integer matrices. Shown, left to right:

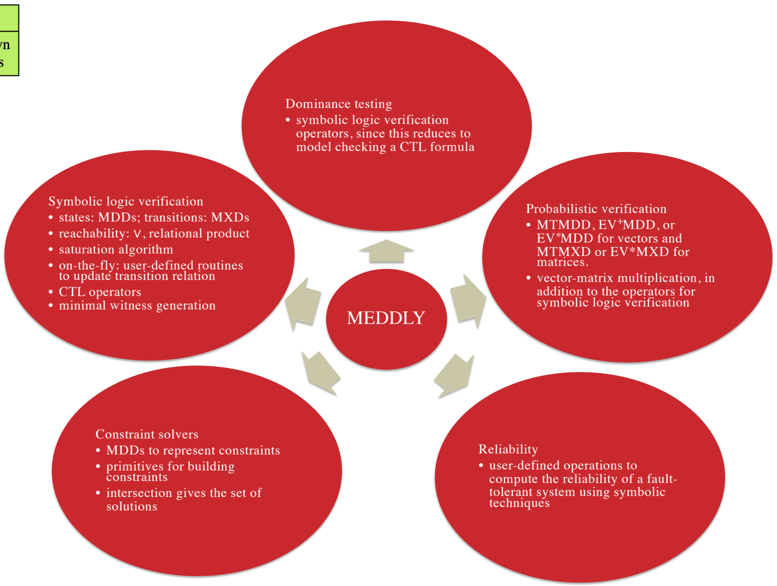
quasi-reduced and fully-identity-reduced MXD encoding the same Boolean matrix (i.e., a relation)
$(x_2 = 0, x_2' = 2, x_1 = 0, x_1' = 0)$ => False (default)
$(x_2 = 1, x_2' = 1, x_1 = 1, x_1' = 0)$ => True

*quasi-reduced and fully-identity-reduced MTMXD, quasi-reduced and fully-identity-reduced EV*MXD, all encoding the same integer-valued function*
$(x_2 = 0, x_2' = 2, x_1 = 1, x_1' = 0)$ => 0 (default)
$(x_2 = 1, x_2' = 1, x_1 = 1, x_1' = 1)$ => 6



!, ∧, ∨
+, -, *, /
≤,<,≥,> ,=,≠
Breadth First
Saturation
known bounds
unknown bounds
Boolean
Arithmetic
Relational
Reachability Analysis
Inter-MDD Conversions
Operations
Relational Product
Node Storage
Interface
Operation Cache
MDDs
Unique Table
Edge-Valued or Multi-terminal
Range
Reduction Rules
Integer
Real
Boolean
Quasi
Identity
Fully

MEDDLY System Diagram: All major components interact through the user interface, maintaining modularity and extensibility.



MEDDLY

Dominance testing
- symbolic logic verification operators, since this reduces to model checking a CTL formula

Symbolic logic verification
- states: MDDs; transitions: MXDs
- reachability: ∨, relational product
- saturation algorithm
- on-the-fly: user-defined routines to update transition relation
- CTL operators
- minimal witness generation

Probabilistic verification
- MTMDD, EV$^+$MDD, or EV*MDD for vectors and MTMXD or EV*MXD for matrices.
- vector-matrix multiplication, in addition to the operators for symbolic logic verification

Constraint solvers
- MDDs to represent constraints
- primitives for building constraints
- intersection gives the set of solutions

Reliability
- user-defined operations to compute the reliability of a fault-tolerant system using symbolic techniques

Major applications for MEDDLY, and main operations supporting them.

Acknowledgements: Junaid Babar and Shruti Biswal