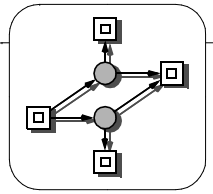
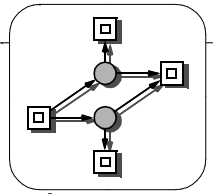


COMPRESSED STATE SPACE REPRESENTATIONS - BINARY DECISION DIAGRAMS



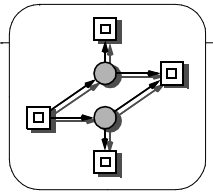
QUALITATIVE ANALYIS METHODS, OVERVIEW

- NET REDUCTION
- STRUCTURAL PROPERTIES
- LINEAR PROGRAMMING
  - place / transition invariants
  - state equation
  - trap equation

} static analysis

- REACHABILITY ANALYSIS
  - (complete) reachability graph
  - compressed state spaces
    - BDDs, NDDs, ..., XDDs
    - Kronecker products
  - reduced state spaces
    - coverability graph
    - symmetry
    - stubborn sets
  - branching process

} dynamic analysis *(model checking)*



BINARY DECISION DIAGRAM (BDD)

- general data structure to implement efficiently Boolean functions

  $f : B^n \rightarrow B$

  $f(x_1, x_2, ..., x_n) \rightarrow \{T, F\}, x_i \in \{T, F\}$

- worst-case effort

  -> *exponential*

- BUT, for many real-live examples

  -> *much better*

- source of origin

  -> *hardware design and verification*

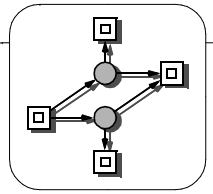
- basis for many successful analysis tools, e.g.

  -> *SVM, NuSVM*
  (communicating state machines)

  . . .

  -> *BDD-CTL, BDD-LTL* ***-> MARCIE***
  (1-bounded Petri nets)

  -> *Rabbit*
  (Cottbus timed automata)



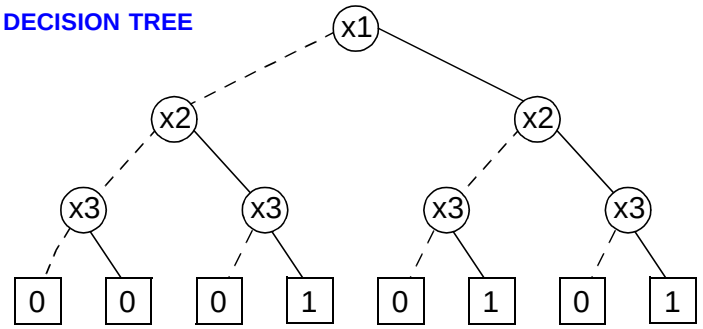
BDD, EXAMPLE 1 (1)

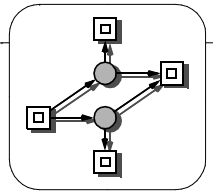
$$f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge x_3$$

DECISION TABLE

x1	x2	x3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

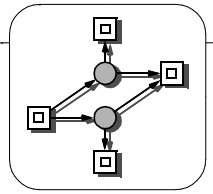
DECISION TREE





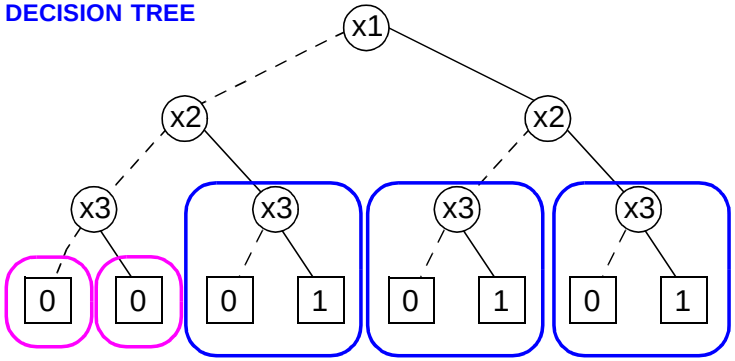
DECISION TREE, PROPERTIES

- two node types
  - -> *non-terminal nodes - Boolean variables*
  - -> *terminal nodes - function values* $\{T, F\}$
- each non-terminal node has two outgoing arcs
  - -> *lo : value of variable is false* - - - - -
  - -> *hi : value of variable is true* ———
- variables are totally ordered
  - -> *the variables appear in the same order along each path (root, terminal node)*
- terminal nodes - from left to right - correspond to table function values - from top to bottom -
- obvious procedure to determine the function value for a given setting of the Boolean variables



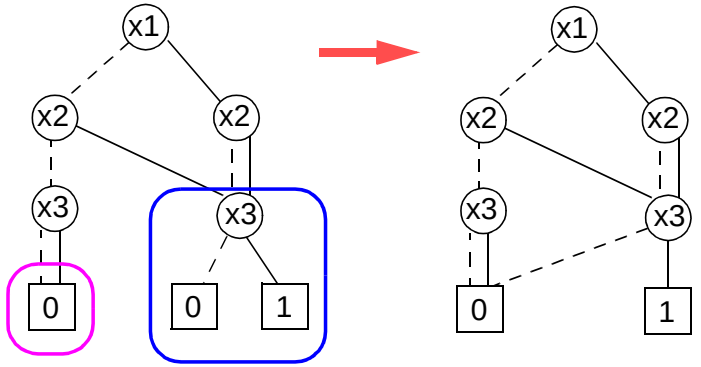
BDD, EXAMPLE 1 (2)

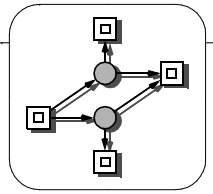
DECISION TREE



REDUCTION RULE 1

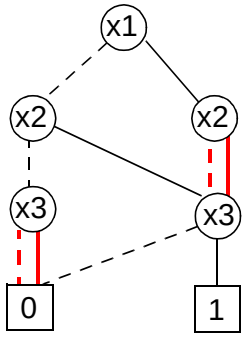
DAG - DIRECTED ACYCLIC ROOT GRAPH





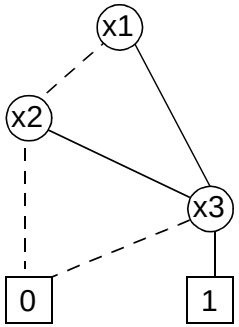
BDD, EXAMPLE 1 (3)

OBDD



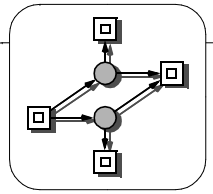
REDUCTION RULE 2

ROBDD



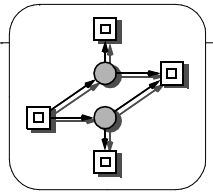
5 nodes, 6 arcs
instead of
15 nodes, 14 arcs

$$f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge x_3$$



REDUCED ORDERED BINARY DECISION DIAGRAM

- the success story of ROBDDs relies on the properties
  - -> *ordered*
  - -> *reduced*
  - -> *in the following, BDD stands shortly for ROBDD*
- no further reduction possible
  - -> *no isomorphic subtrees (rule 1)*
    (equally named nodes with similar subtrees)
  - -> *no node with identical successors (rule 2)*
- **directed acyclic root graph** (DARG)
  - -> *direction always from top to bottom*
  - -> *no cycles, but rejoining (opposite to trees)*
  - -> *all nodes reachable starting at root node*
- **unique result**
  - -> *for a given total order of all variables*
  - -> *insensitive to the order of reductions applied*
- **canonical representation** for Boolean functions
  - -> *some decision problems turn into easy ones*



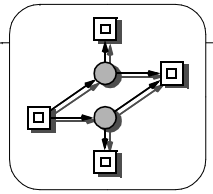
BDD-BASED SOLUTION OF SOME DECISION PROBLEMS

- **equivalence** of two given functions f and g
  (the same function values for all variable settings)
  -> *BDD(f) is isomorphic to BDD(g) in node and arc inscriptions*

- a given function *f* is a **tautology**
  (the function value is true for all variable settings)
  -> *BDD(f)* ⟶ 1

- a given function *f* is **satisfiable**
  (there is at least one variable setting yielding true)
  -> *BDD(f) contains the node* 1

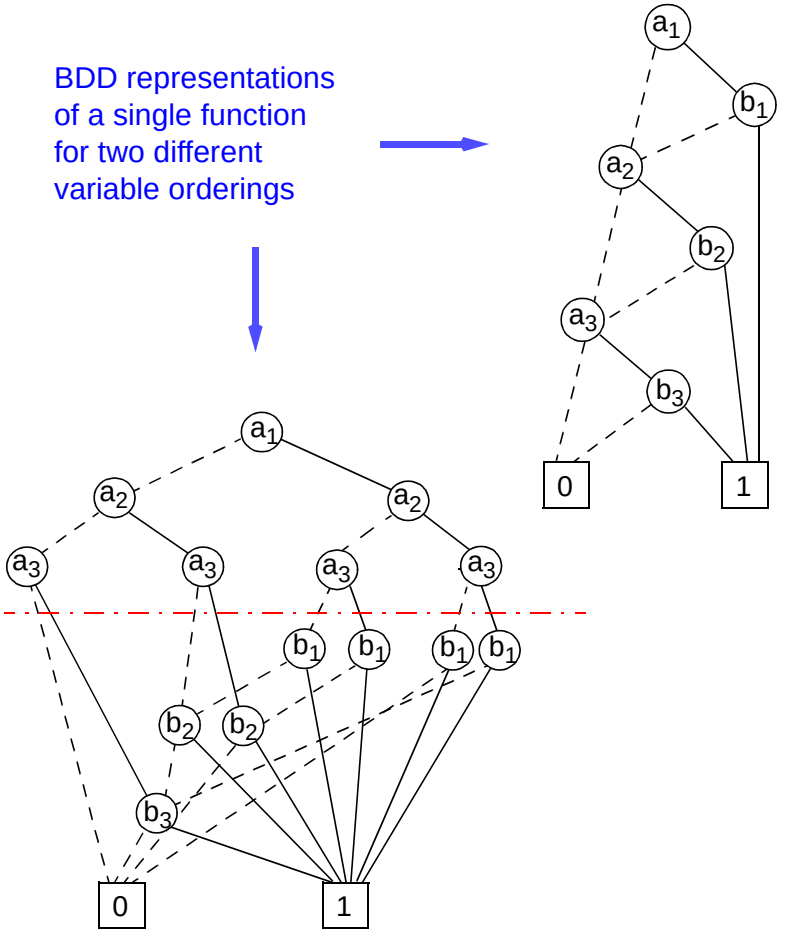
- the function *f* is independent of a given variable *x*
  -> *the BDD(f) does not contain a node x*

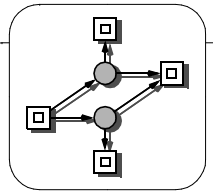


BDD, EXAMPLE 2

$$f(a_1, b_1, a_2, b_2, a_3, b_3) = a_1 \wedge b_1 \vee a_2 \wedge b_2 \vee a_3 \wedge b_3$$

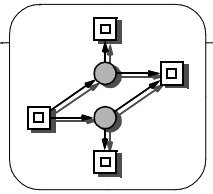
BDD representations of a single function for two different variable orderings





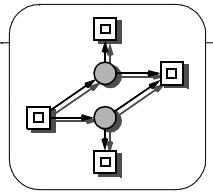
OBSERVATIONS BY EXAMPLE 2

- the reduction effect may depend on the chosen order of all variables
- worst-case: there is no reduction
  -> *space demand grows exponentially with number of variables*
- successful tools apply successful heuristics to determine a suitable (not necessarily optimal) order of variables
- the choosen order of variables does not influence the correctness of the result
- despite of the exponential growth in the worst case, BDDs are often successful for real-live examples



BDD CONSTRUCTION

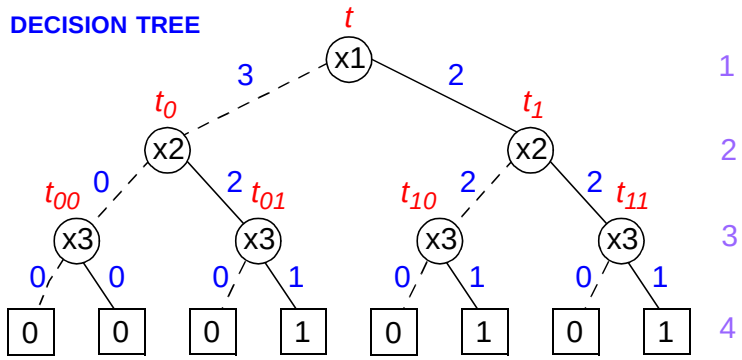
- **objective:**
  no a posteriori reduction necessary
- **approach 1:**
  top-down on-the-fly construction
- **approach 2:**
  syntax-oriented bottom-up construction
  by step-wise composition of two BDDs
  -> *Boolean operations to combine two BDDs*
- there exist very efficient algorithms for both approaches



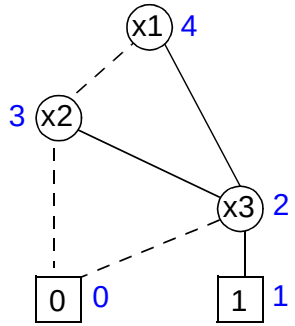
TOP-DOWN APPROACH, BDD EXAMPLE 1

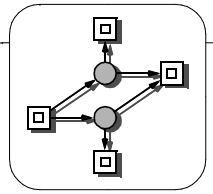
$$f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge x_3$$

DECISION TREE



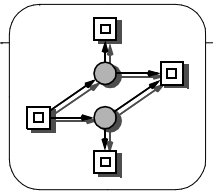
ROBDD





BDD & PETRI NET (1)

- 1-bounded Petri nets
  - -> *a place may be interpreted as Boolean variable*
- different ways to represent a marking *m*
  - -> *set of marked places (subset of P)* -> *{p1, p4}*
  - -> *bit vector b[1..n] , n = card(P), with* -> *(1,0,0,1)*
    bi = 0, if $pi \notin m$
    bi = 1, if $pi \in m$
  - -> *Boolean function* -> $p1 \wedge \overline{p2} \wedge \overline{p3} \wedge p4$
    conjunction of all variables,
    whereby non-marked places are negated
    -->> characteristic function χ_m
- set of markings M (-> state space)
  - -> *set of bit vectors*
  - -> *characteristic function* χ_M *of* M
    -> *n-ary Boolean function,*
    yielding the function value 'true'
    for all markings belonging to M
    -> *BDD representation on hand*
    (often called ***symbolic representation****)*

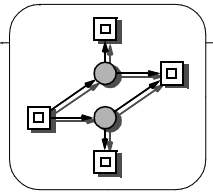


BDD & PETRI NET (2)

- all operations on sets of markings
  can be realized as
  logical operations on their characteristic functions
  - -> $\chi_{M1 \cup M2} \equiv \chi_{M1} \vee \chi_{M2}$
  - -> $\chi_{M1 \cap M2} \equiv \chi_{M1} \wedge \chi_{M2}$
  - -> $\chi_{\overline{M}} \equiv \overline{\chi_{M}}$

- that's all we need to implement
  BDD-based Petri net analysis algorithms
  - -> ***symbolic Petri net analysis***
    with symbolic representation of marking sets

- remark:
  larger sets may result into smaller BDDs
  than smaller sets



REFERENCES

[Akers 78]
Akers, S. B.:
Binary Decision Diagrams;
IEEE Transactions on Computers C-27(1978), 509-516.

[Andersen 99]
Andersen, H. R.:
An Introduction to Binary Decision Diagrams;
Lecture Notes for Efficient Algorithms and Programsm, Fall 1999, IT Univ. Copenhagen.

[Bryant 86]
Bryant, R. E.:
Graph-based Algorithms for Boolean Function Manipulation;
IEEE Transactions on Computers C-35(1986)6, 677–691.

[Bryant 92]
BRYANT, R. E.:
Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams;
ACM Computing Survey 24(1992)3, 293-318.

[Lee 59]
Lee, C. Y.:
Representation of Switching Circuits by Binary Decision Programs;
Bell System Technical Journal 38(1959), 985–999.

[Noack 99]
Noack, A.:
A ZBBD Package for Efficient Model Checking of Petri Nets (in German);
BTU Cottbus, Dep. of CS, Major Individual Project, 1999.

[Spranger 2001]
Spranger, J.:
Symbolic LTL Verification of Petri Nets (in German);
PhD thesis, BTU Cottbus, Dep. of CS, December 2001.

[Wimmel 97]
WIMMEL, G.:
A BDD-based Model Checker for the PEP Tool;
Univ. of Newcastle, Dep. of CS, Major Individual Project, May 1997.

[Tovchigrechko 2008]
Tovchigrechko, A:
Efficient symbolic analysis of bounded Petri nets using Interval Decision Diagrams;
PhD thesis, BTU Cottbus, Dep. of CS, October 2008.

[Marcie 2013]
M Heiner, C Rohr and M Schwarick:
MARCIE - Model checking And Reachability analysis done effiCIEntly;
Proc. PETRI NETS 2013, Milano, Springer, LNCS 7927, 389–399, June 2013.

[Marcie 2016]
M Heiner, C Rohr, M Schwarick and A Tovchigrechko:
MARCIE's secrets of efficient model checking;
Transactions on Petri Nets and Other Models of Concurrency XI, LNCS 9930, 2016