PolyBoRi - A Framework for Gröbner Basis Computations with Boolean Polynomials

Michael Brickenstein

Mathematisches Forschungsinstitut Oberwolfach

Alexander Dreyer*

Fraunhofer Institute for Industrial Mathematics (ITWM)

MEGA 2007 – Effective Methods in Algebraic Geometry
Strobl, Austria, June 26th, 2007

*Presentation

Fraunhofer Institut Techno- und Wirtschaftsmathematik

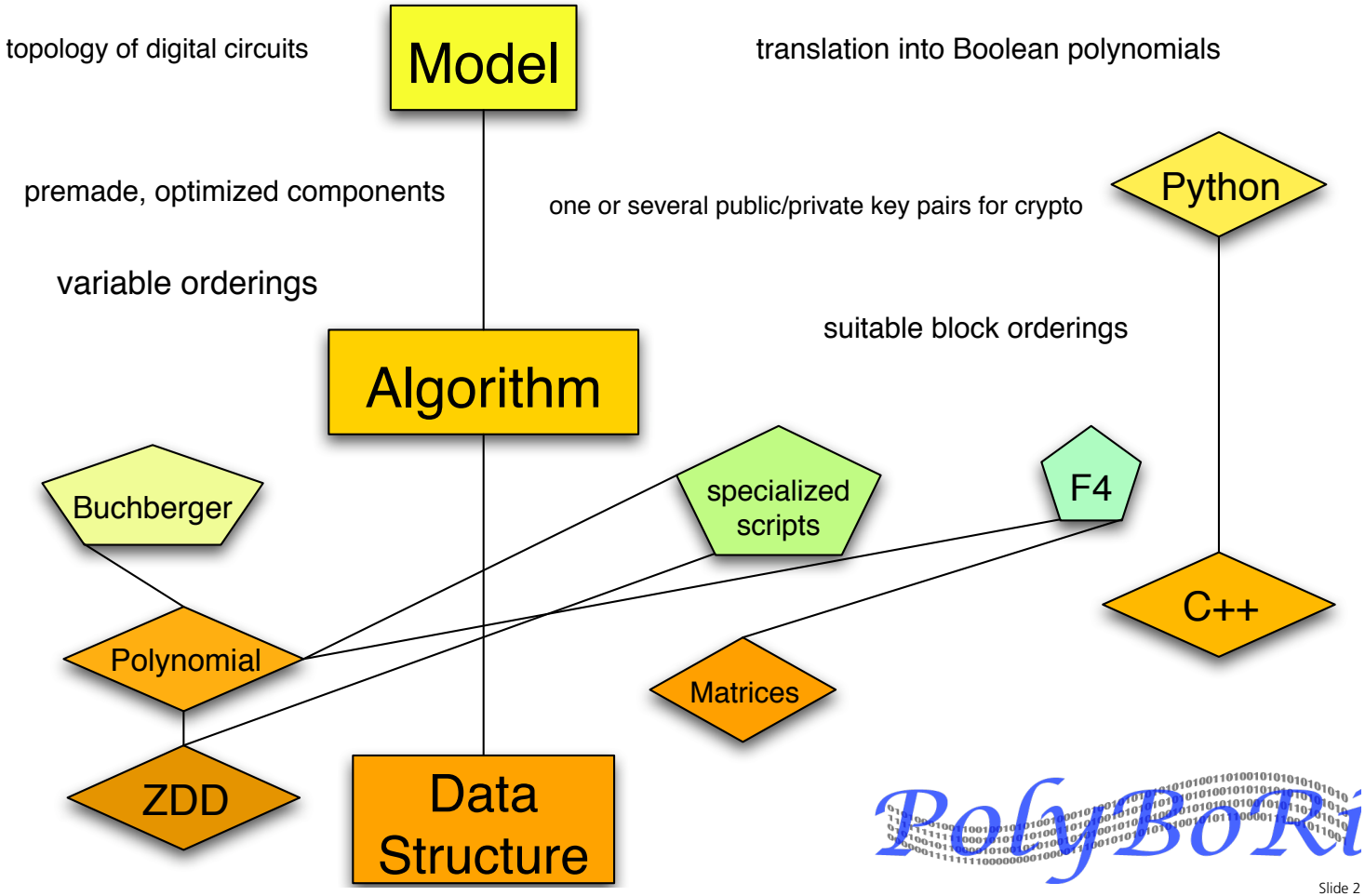
Mathematisches Forschungsinstitut Oberwolfach

Basic Data

PolyBoRi	***Poly***nomials over ***Bo***olean ***Ri***ngs
DFG Project	"Development, implementation and application of mathematical-algebraic algorithms for formal verification of digital systems with arithmetic blocks"
Fraunhofer ITWM	Alexander Dreyer (Department Adaptive Systems)
University of Kaiserslautern	Algebra, Geometry and Computer Algebra Group (Prof. Greuel, Department of Mathematics)
Doctoral thesis	Michael Brickenstein (Mathematisches Forschungsinstitut Oberwolfach)

Overview

Optimization on many levels



Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Digital systems verification and Boolean Polynomials

Interprete Boolean expressions as polynomials over $\mathbb{Z}_2$

logical operations $\longrightarrow$ arithmetical operations and $\{\mathsf{true}, \mathsf{false}\} \longrightarrow \{0, 1\}$

$$p \in \mathbb{Z}_2[x_1, \cdots, x_n]/\langle x_1^2 - x_1, \cdots, x_n^2 - x_n \rangle$$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Digital systems verification and Boolean Polynomials

Interprete Boolean expressions as polynomials over $\mathbb{Z}_2$

logical operations $\rightarrow$ arithmetical operations and $\{\mathsf{true}, \mathsf{false}\} \rightarrow \{0, 1\}$

$$p \in \mathbb{Z}_2[x_1, \cdots, x_n]/\langle x_1^2 - x_1, \cdots, x_n^2 - x_n \rangle$$

Polynomials as sets

$$p = a_1 \cdot x_1^{\nu_{11}} \cdot \ldots \cdot x_n^{\nu_{1n}} + \ldots + a_{2^n} \cdot x_1^{\nu_{2^n 1}} \cdot \ldots \cdot x_n^{\nu_{2^n n}}$$

$$= \sum_{s \in S_p} \left(\prod_{x_\nu \in s} x_\nu \right),$$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Digital systems verification and Boolean Polynomials

Interprete Boolean expressions as polynomials over $\mathbb{Z}_2$

logical operations $\rightarrow$ arithmetical operations and $\{\mathsf{true}, \mathsf{false}\} \rightarrow \{0, 1\}$

$$p \in \mathbb{Z}_2[x_1, \cdots, x_n]/\langle x_1^2 - x_1, \cdots, x_n^2 - x_n \rangle$$

Polynomials as sets

$$p = a_1 \cdot x_1^{\nu_{11}} \cdot \ldots \cdot x_n^{\nu_{1n}} + \ldots + a_{2^n} \cdot x_1^{\nu_{2^n 1}} \cdot \ldots \cdot x_n^{\nu_{2^n n}}$$

$$= \sum_{s \in S_p} \left(\prod_{x_\nu \in s} x_\nu \right),$$

$$\text{with } S_p = \{\{x_{i_1}, \cdots, x_{i_{n_1}}\}, \cdots, \{x_{i_m}, \cdots, x_{i_{nm}}\}\}$$

$$\subseteq \mathsf{PowerSet}(x_1, \cdots, x_n)$$

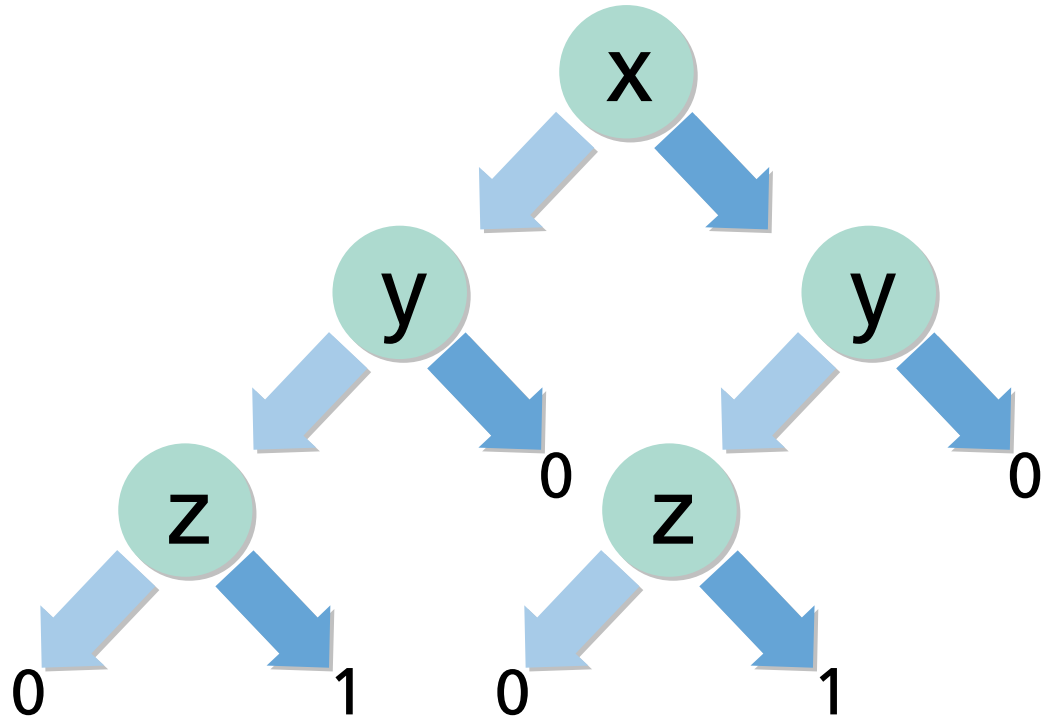
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)



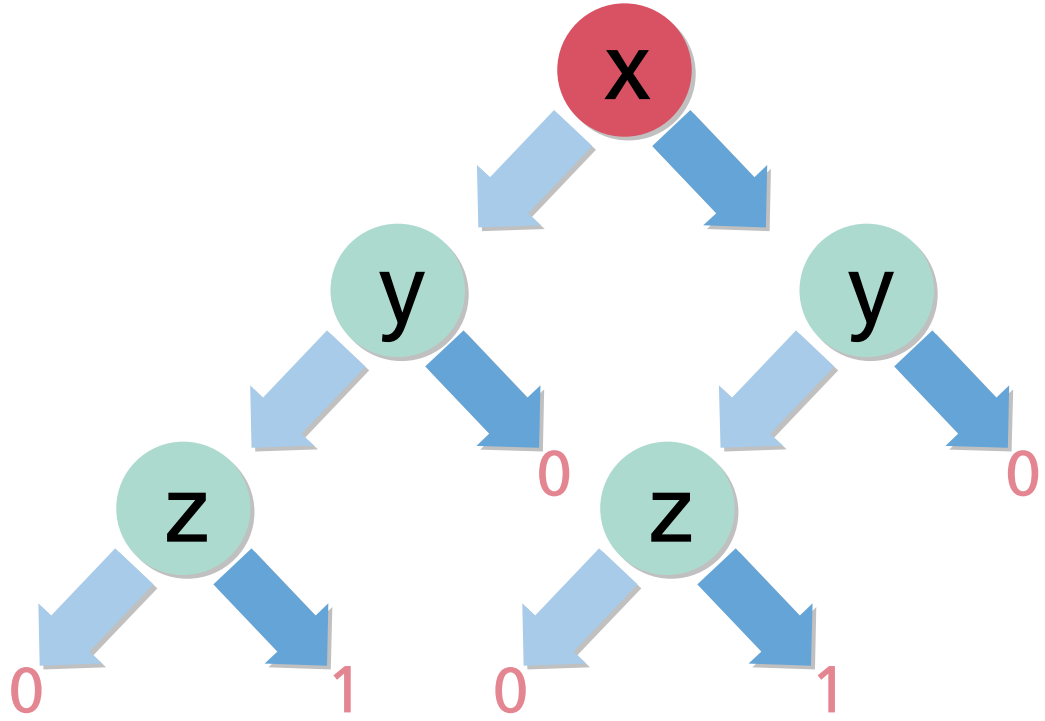
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)



Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

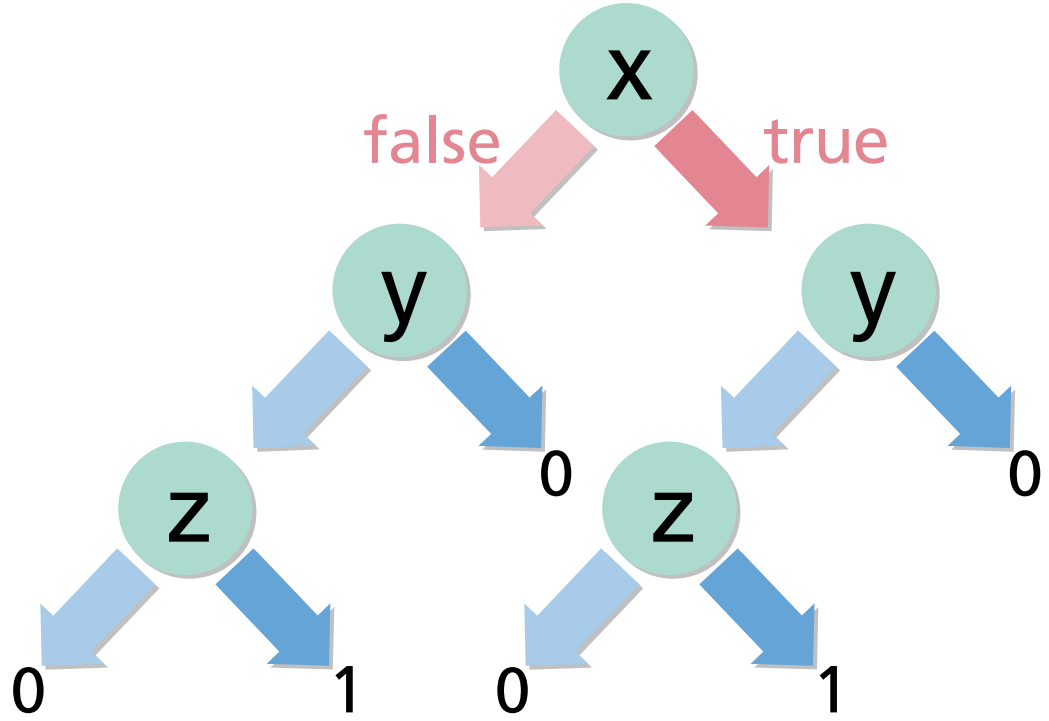
Mathematisches Forschungsinstitut Oberwolfach

Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false



Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

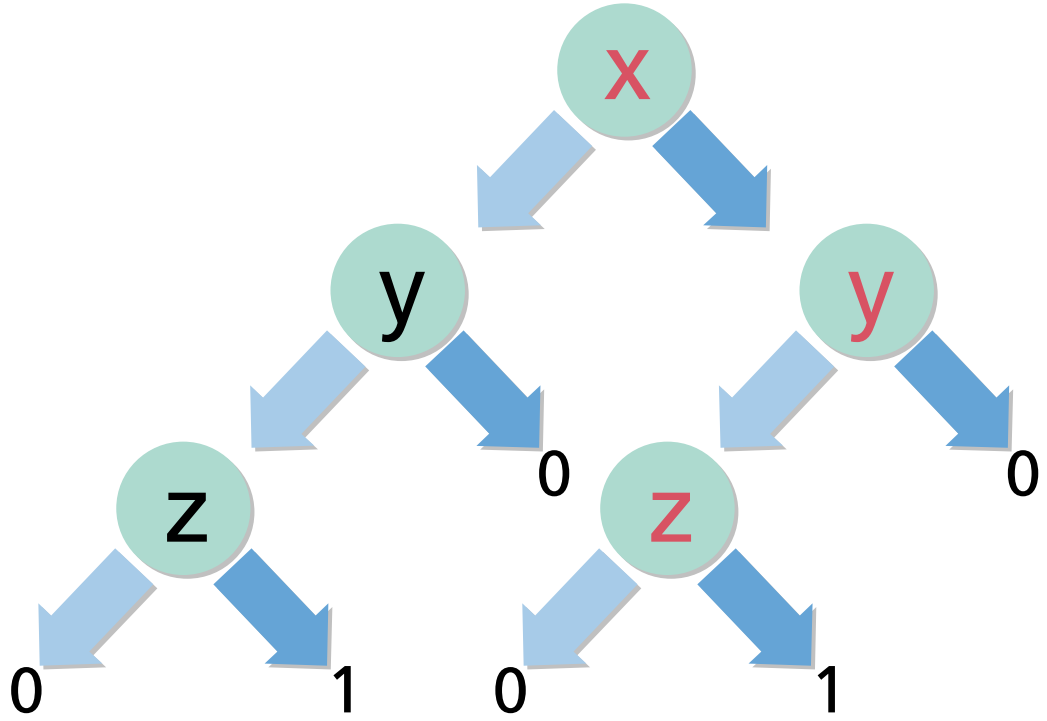
Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false

»Ordered« if the variable order is constant over all paths



Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

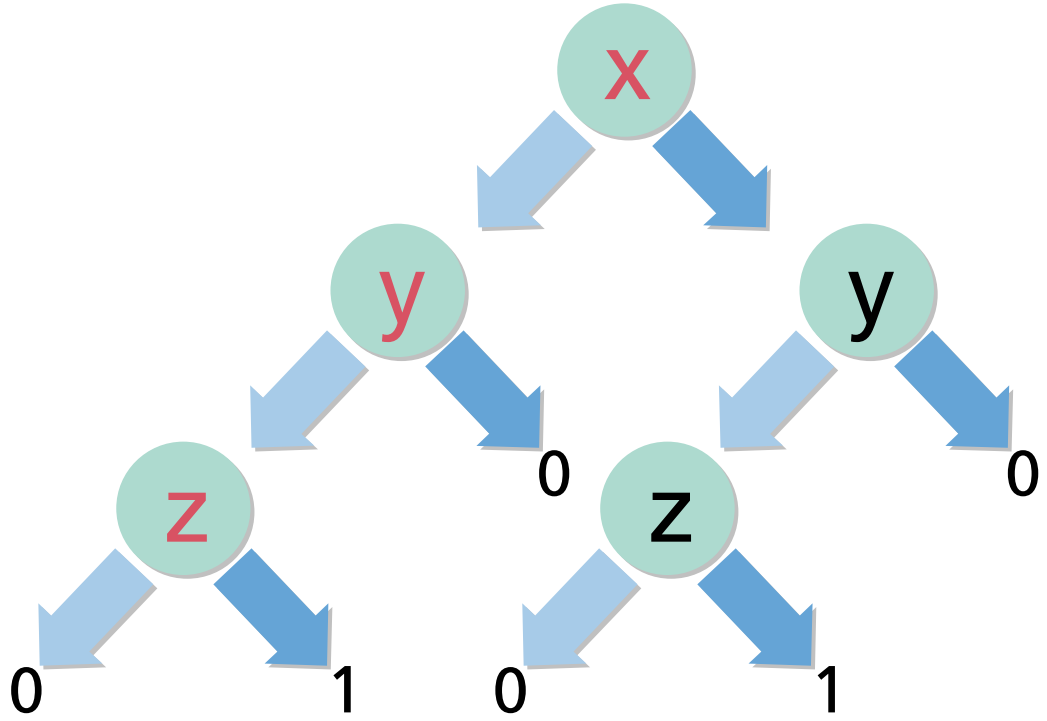
Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false

»Ordered« if the variable order is constant over all paths



Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

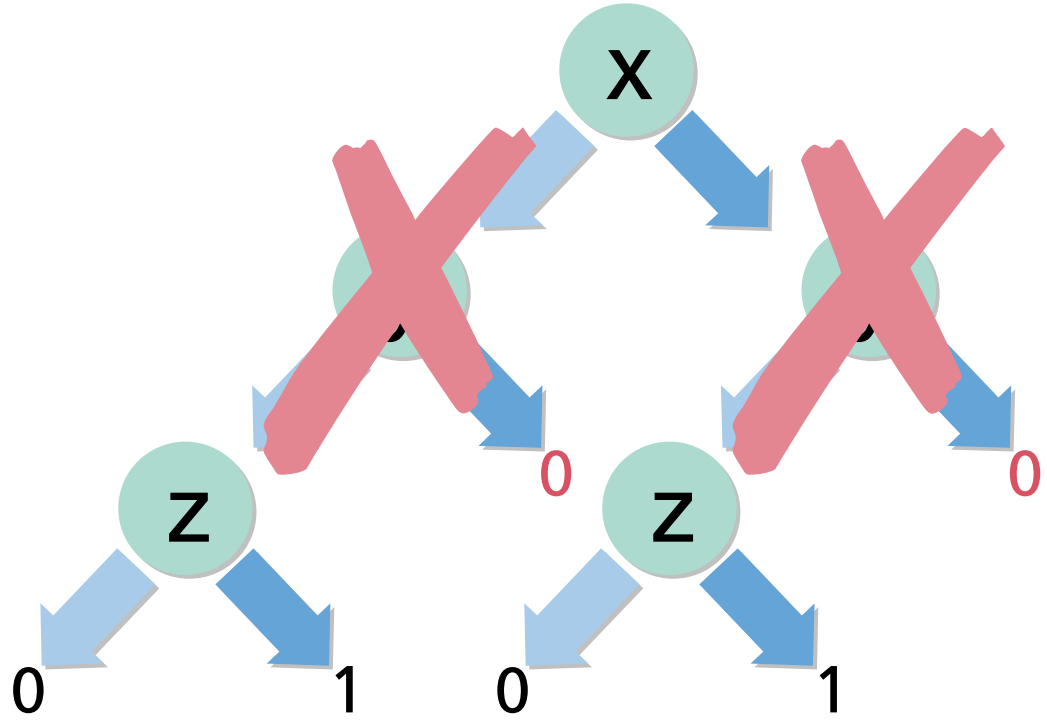
Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false

»Ordered« if the variable order is constant over all paths

Zero-suppressed BDD (ZDD)

Node eliminated $\iff$ then $\rightarrow 0$



Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

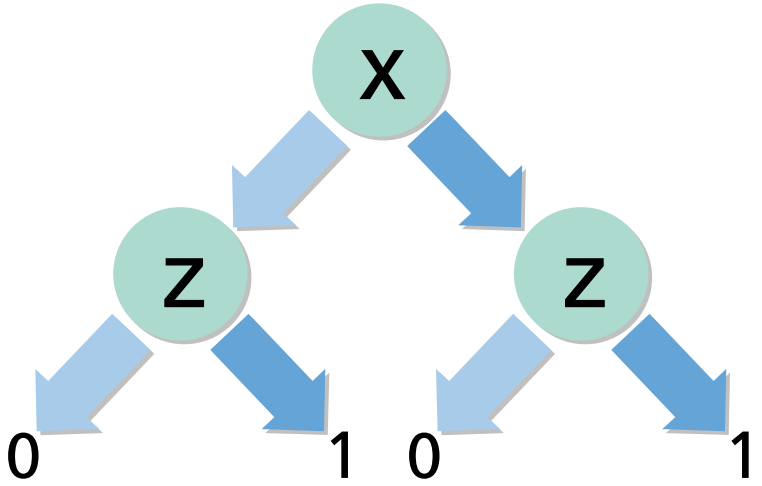
Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false

»Ordered« if the variable order is constant over all paths



Zero-suppressed BDD (ZDD)

Node eliminated $\iff$ then $\rightarrow$ 0

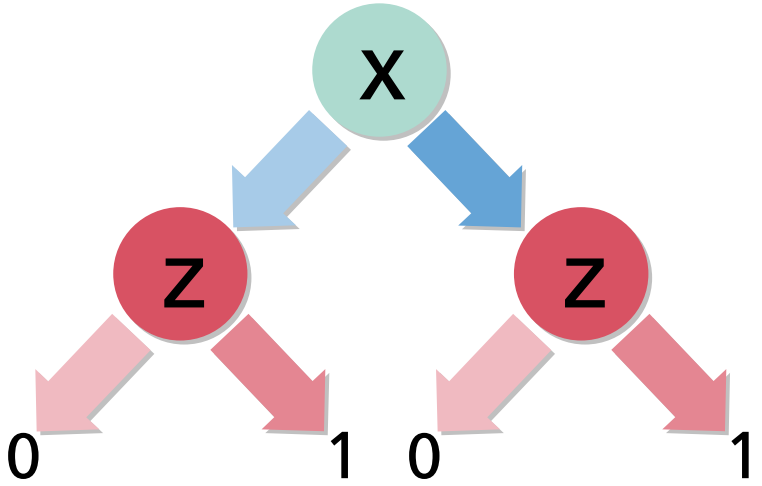
Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false

»Ordered« if the variable order is constant over all paths



Zero-suppressed BDD (ZDD)

Node eliminated $\iff$ then $\rightarrow 0$

Equal subgraphs merged

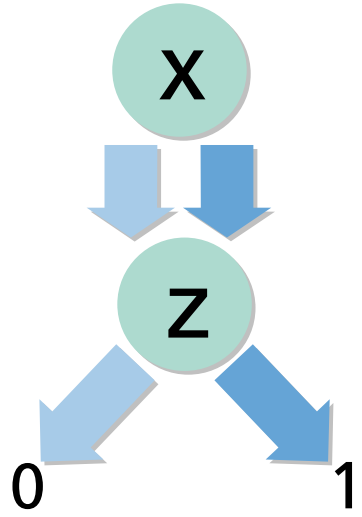
Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false

»Ordered« if the variable order is constant over all paths



Zero-suppressed BDD (ZDD)

Node eliminated $\iff$ then $\rightarrow 0$

Equal subgraphs merged

Advantages of ZDDs

Idea

ZDDs store term structure (**not** the Boolean function behind)

Advantages of ZDDs

Idea — ZDDs store term structure (**not** the Boolean function behind)

Reasons

– Compact data structure

Advantages of ZDDs

Idea

ZDDs store term structure (**not** the Boolean function behind)

Reasons

- Compact data structure
- Suitable for sparse sets of subsets

Advantages of ZDDs

Idea

ZDDs store term structure (**not** the Boolean function behind)

Reasons

- Compact data structure
- Suitable for sparse sets of subsets
- Polynomial structure recognizable
  - Valid paths $\widehat{=}$ polynomial terms
  - Natural path sequence $\widehat{=}$ lexicographical term ordering

Polynomial Arithmetic

Boolean polynomial operations Correspond to set operations

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Polynomial Arithmetic

Boolean polynomial operations — Correspond to set operations

Example

$$(x + x\,y) + (x\,y + z) = x + 2\,x\,y + z \equiv x + z \iff$$

$$(S_1 \cup S_2) \backslash (S_1 \cap S_2) = \{\{x\}, \{z\}\}$$
(with $S_1 = \{\{x\}, \{x, y\}\}, S_2 = \{\{x, y\}, \{z\}\}$)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Polynomial Arithmetic

Boolean polynomial operations — Correspond to set operations

Example

$$(x + x\,y) + (x\,y + z) = x + 2\,x\,y + z \equiv x + z \iff$$

$$(S_1 \cup S_2) \backslash (S_1 \cap S_2) = \{\{x\}, \{z\}\}$$
(with $S_1 = \{\{x\}, \{x, y\}\}, S_2 = \{\{x, y\}, \{z\}\}$)

$$x \cdot (y + z) = x\,y + y\,z \iff$$

$$\{\{x\} \cup \{y\}, \{x\} \cup \{z\}\} = \{\{x, y\}, \{x, z\}\}$$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Polynomial Arithmetic

Boolean polynomial operations — Correspond to set operations

Example

$$(x + x\,y) + (x\,y + z) = x + 2\,x\,y + z \equiv x + z \iff$$

$$(S_1 \cup S_2) \backslash (S_1 \cap S_2) = \{\{x\}, \{z\}\}$$
(with $S_1 = \{\{x\}, \{x, y\}\}, S_2 = \{\{x, y\}, \{z\}\}$)

$$x \cdot (y + z) = x\,y + y\,z \iff$$

$$\{\{x\} \cup \{y\}, \{x\} \cup \{z\}\} = \{\{x, y\}, \{x, z\}\}$$

Likewise (but more complicated) — Factors/multiples of monomials, degree of a polynomial...

ZDD Implementation — Free C/C++ Library Cudd: Fabio Somenzi (University of Colorado)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Caching and Recursion

ZDD normalform

Unique diagram root nodes

$a = b \iff$ rootnode(a) is rootnode(b)

Caching and Recursion

ZDD normalform	Unique diagram root nodes $a = b \iff \texttt{rootnode}(a)$ is $\texttt{rootnode}(b)$
Reference counting	Lower memory usage (no deep copies)
Caching of operations	$a \diamond b$ never evaluated twice (also for sub-diagrams)

Caching and Recursion

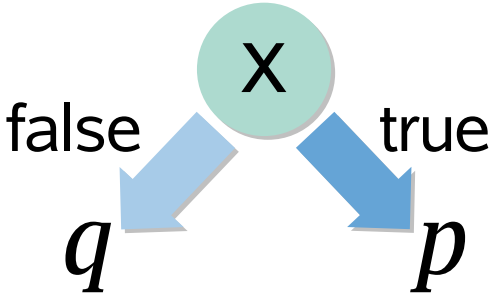
ZDD normalform	Unique diagram root nodes $a = b \iff \texttt{rootnode}(a)$ is $\texttt{rootnode}(b)$
Reference counting	Lower memory usage (no deep copies)
Caching of operations	$a \diamond b$ never evaluated twice (also for sub-diagrams)
Advantage	Speed-up recursive procedures

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Caching and Recursion

Example: lead(f)



$$f = q + x \cdot p$$

First (lexicographical) term t in f with $\deg t = \deg f$.

Input: f Boolean polynomial **Output:** $t = \mathrm{lead}(f)$ (deg-lex)

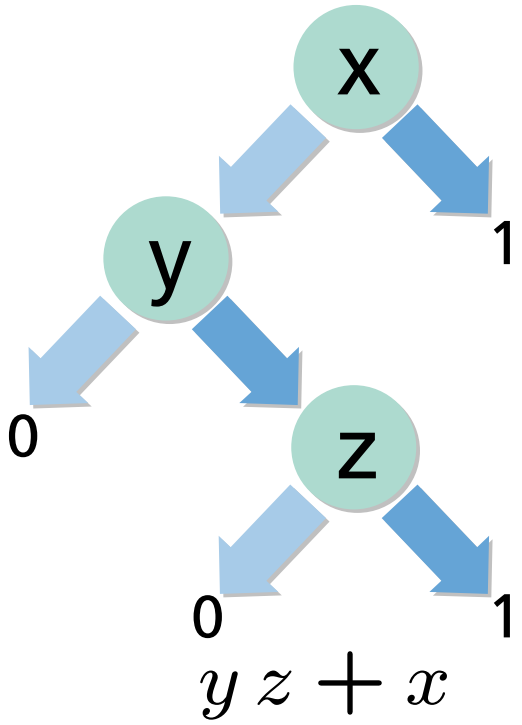
if $f \in \{0, 1\}$ **then**
  set $t := 1$
else if isCached(lead, f) **then**
  set $t :=$ cache(lead, f)
else
  set $x :=$ root variable of f
  if $\deg(f) = \deg(\mathrm{thenBranch}(f)) + 1$ **then**
    set $t := x \cdot \mathrm{lead}(\mathrm{thenBranch}(f))$
  else
    set $t := \mathrm{lead}(\mathrm{elseBranch}(f))$
  end if
  insert cache(lead, f) $:= t$
end if

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Caching and Recursion

Example: lead(f)



$y\,z + x$

First (lexicographical) term t in f with $\deg t = \deg f$.

Input: f Boolean polynomial **Output:** $t = \mathrm{lead}(f)$ (deg-lex)

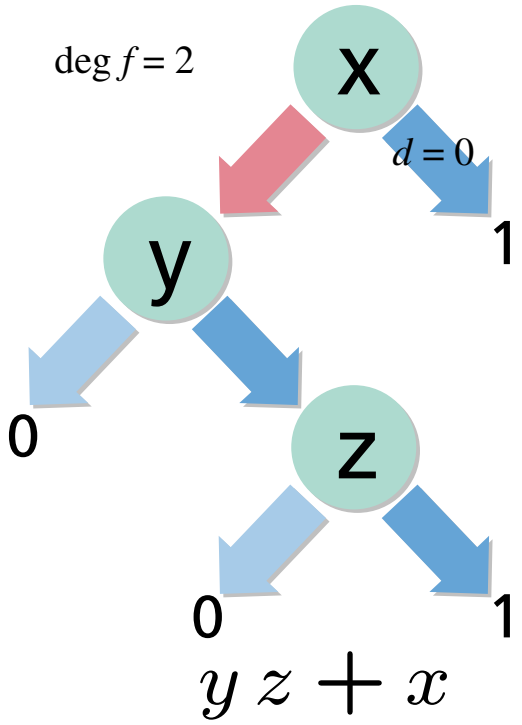
if $f \in \{0, 1\}$ **then**
  set $t := 1$
else if isCached(lead, f) **then**
  set $t :=$ cache(lead, f)
else
  set $x :=$ root variable of f
  if $\deg(f) = \deg(\mathrm{thenBranch}(f)) + 1$ **then**
    set $t := x \cdot \mathrm{lead}(\mathrm{thenBranch}(f))$
  else
    set $t := \mathrm{lead}(\mathrm{elseBranch}(f))$
  end if
  insert cache(lead, f) $:= t$
end if

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Caching and Recursion

Example: lead(f)



First (lexicographical) term t in f with $\deg t = \deg f$.

Input: f Boolean polynomial **Output:** $t = \mathrm{lead}(f)$ (deg-lex)

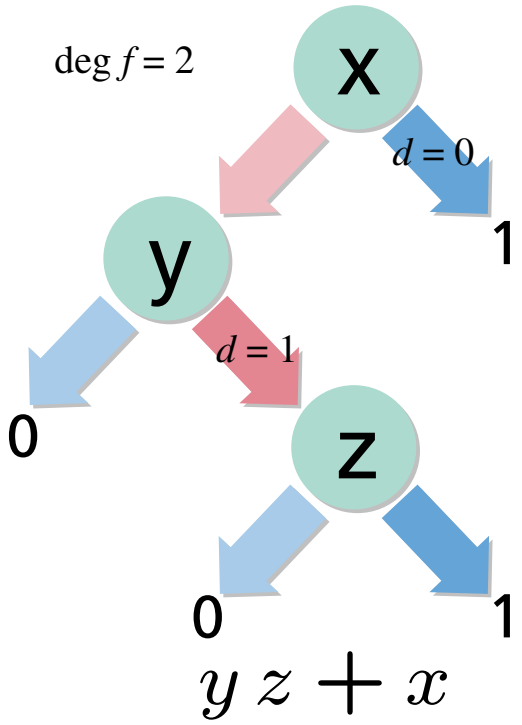
if $f \in \{0, 1\}$ **then**
  set $t := 1$
else if isCached(lead, f) **then**
  set $t :=$ cache(lead, f)
else
  set $x :=$ root variable of f
  if $\deg(f) = \deg(\mathrm{thenBranch}(f)) + 1$ **then**
    set $t := x \cdot \mathrm{lead}(\mathrm{thenBranch}(f))$
  else
    set $t := \mathrm{lead}(\mathrm{elseBranch}(f))$
  end if
  insert cache(lead, f) $:= t$
end if

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Caching and Recursion

Example: lead(f)



First (lexicographical) term t in f with $\deg t = \deg f$.

Input: f Boolean polynomial **Output:** $t = \mathrm{lead}(f)$ (deg-lex)

if $f \in \{0, 1\}$ **then**
  set $t := 1$
else if isCached(lead, f) **then**
  set $t :=$ cache(lead, f)
else
  set $x :=$ root variable of f
  if $\deg(f) = \deg(\mathrm{thenBranch}(f)) + 1$ **then**
    set $t := x \cdot \mathrm{lead}(\mathrm{thenBranch}(f))$
  else
    set $t := \mathrm{lead}(\mathrm{elseBranch}(f))$
  end if
  insert cache(lead, f) $:= t$
end if

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

From ZDDs to PolyBoRi

C++-Library

High-level data types for Boolean polynomials, monomials, exponent vectors, and underlying rings
Implements polynomial operations and basic functionality

From ZDDs to PolyBoRi

C++-Library	High-level data types for Boolean polynomials, monomials, exponent vectors, and underlying rings Implements polynomial operations and basic functionality
ZDDs	Internal handling of polynomial structure (utilizing cache and uniqueness)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

From ZDDs to PolyBoRi

C++-Library	High-level data types for Boolean polynomials, monomials, exponent vectors, and underlying rings Implements polynomial operations and basic functionality
ZDDs	Internal handling of polynomial structure (utilizing cache and uniqueness)
Ordering-dependend functions	Leading term computation, monomial comparisons, . . . added

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

From ZDDs to PolyBoRi

C++-Library	High-level data types for Boolean polynomials, monomials, exponent vectors, and underlying rings Implements polynomial operations and basic functionality
ZDDs	Internal handling of polynomial structure (utilizing cache and uniqueness)
Ordering-dependend functions	Leading term computation, monomial comparisons, . . . added
Non-trivial Monomial Orderings (run- & compile-time selectable)	Lexicographic, degree-lexicographic, and degree-reverse-lexicographic (ascending variable order) orderings Orderings consisting of blocks of degree-orderings

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

From ZDDs to PolyBoRi

C++-Library	High-level data types for Boolean polynomials, monomials, exponent vectors, and underlying rings Implements polynomial operations and basic functionality
ZDDs	Internal handling of polynomial structure (utilizing cache and uniqueness)
Ordering-dependend functions	Leading term computation, monomial comparisons, . . . added
Non-trivial Monomial Orderings (run- & compile-time selectable)	Lexicographic, degree-lexicographic, and degree-reverse-lexicographic (ascending variable order) orderings Orderings consisting of blocks of degree-orderings
Singular Interface	Prototype

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

PolyBoRi's Python Interface

Python interface allows for

- Parsing of complex polynomial systems (Inner-domain specific language)
- Interactive use (via ipython)

PolyBoRi's Python Interface

Python interface allows for
- Parsing of complex polynomial systems (Inner-domain specific language)
- Interactive use (via ipython)

Extensive testsuite
- Mainly satisfiability examples; some from cryptography

Rapid Prototyping
- Many algorithms existed in python first

PolyBoRi's Python Interface

Python interface allows for
- Parsing of complex polynomial systems (Inner-domain specific language)
- Interactive use (via ipython)

Extensive testsuite
- Mainly satisfiability examples; some from cryptography

Rapid Prototyping
- Many algorithms existed in python first
- Sophisticated and easy extendable strategies for Gröbner base computation

Formal Verification

Boolean Polynomials and Logical Expressions

- Propositional logic $\longrightarrow$ Boolean polynomial
- Each formula from propositional logic can be stated using Or, Not, True, False
- Choose a mapping: True $\longrightarrow 0$, False $\longrightarrow 1$
  $\mathrm{Or}(x, y) \longrightarrow x \cdot y$, $\mathrm{Not}(x) \longrightarrow 1 + x$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Formal Verification

Boolean Polynomials and Logical Expressions

- Propositional logic $\longrightarrow$ Boolean polynomial
- Each formula from propositional logic can be stated using Or, Not, True, False
- Choose a mapping: True $\longrightarrow 0$, False $\longrightarrow 1$
  $\mathrm{Or}(x, y) \longrightarrow x \cdot y$, $\mathrm{Not}(x) \longrightarrow 1 + x$

Property Checking

- Digital system given as set R of Boolean polynomials
- Property p (Boolean expression) to be verified w. r. t. R
- Check via normal form computation: $\mathrm{NF}(p, R) = 0$?

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Representation of Adder Blocks with Carry Bit

Variables

$s(0), \ldots, s(n-1)$ (sums), $c(0), \ldots, c(n-1)$ (carry bits),
$a(0), \ldots, a(n-1), b(0), \ldots, b(n-1)$ (inputs)

Topological Variable Order

$s(n-1), c(n-1), a(n-1), b(n-1),$
$s(n-2), c(n-2), a(n-2), b(n-2), \ldots$

Outputs before inputs, a, b reverse alternating (fast ZDD handling)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Representation of Adder Blocks with Carry Bit

Variables

$s(0), \ldots, s(n-1)$ (sums), $c(0), \ldots, c(n-1)$ (carry bits),
$a(0), \ldots, a(n-1), b(0), \ldots, b(n-1)$ (inputs)

Topological Variable Order

$s(n-1), c(n-1), a(n-1), b(n-1),$
$s(n-2), c(n-2), a(n-2), b(n-2), \ldots$

Outputs before inputs, a, b reverse alternating (fast ZDD handling)

Equations

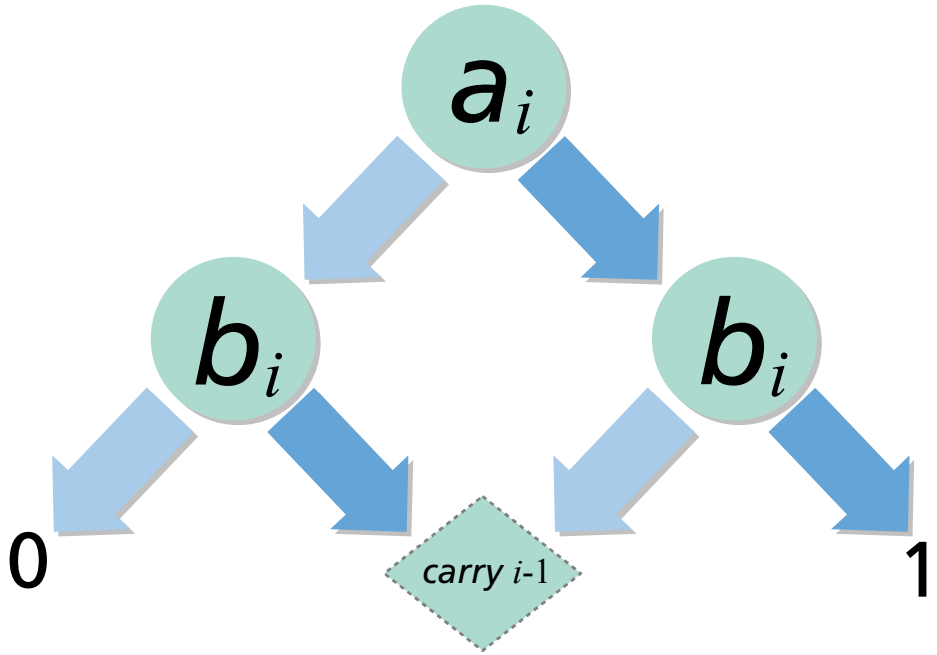
$c(i) + carry_i$
$s(i) + a(i) + b(i) + carry_i$

with $carry_{-1} = 0$
and $carry_i = a(i) \cdot (b(i) + carry_{i-1}) + b(i) \cdot carry_{i-1}$

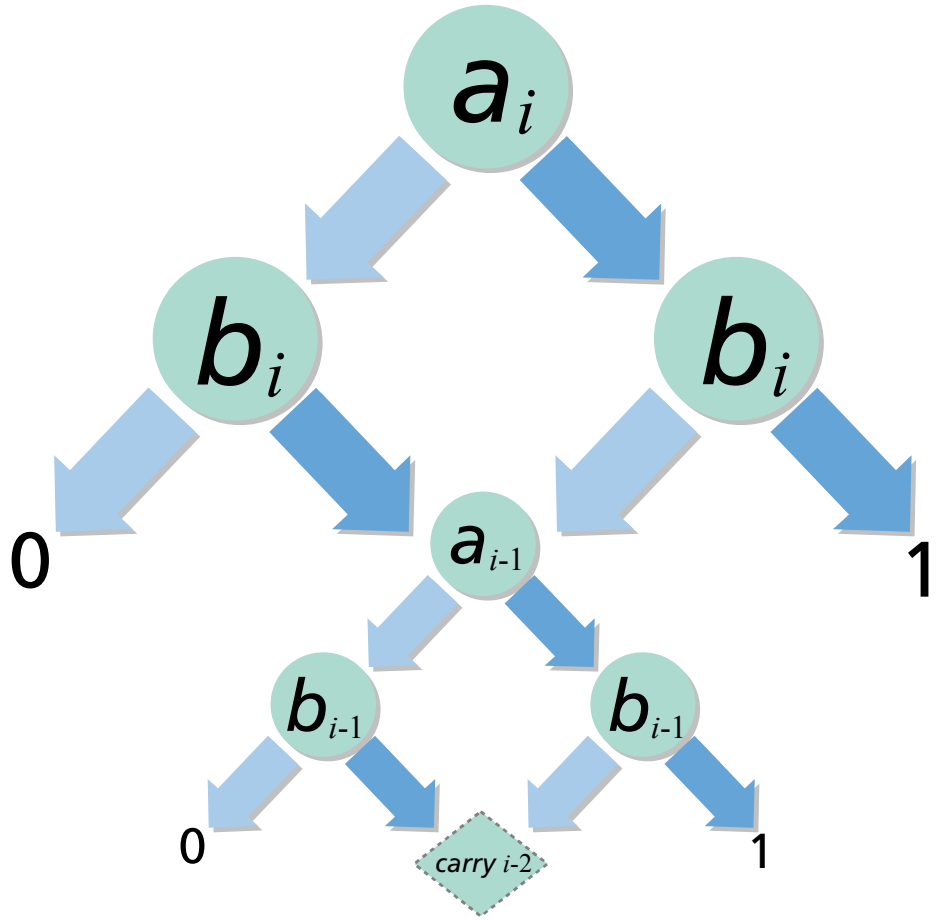
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

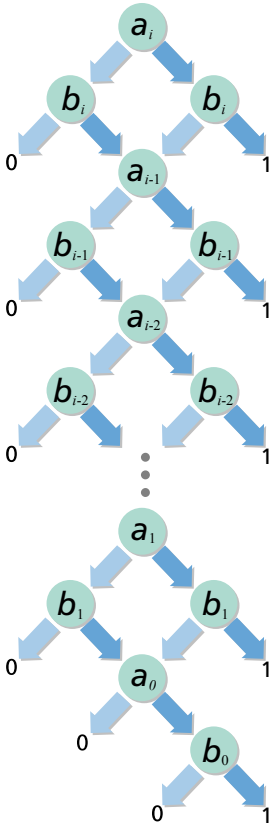
Carry-Bit Graph



Carry-Bit Graph



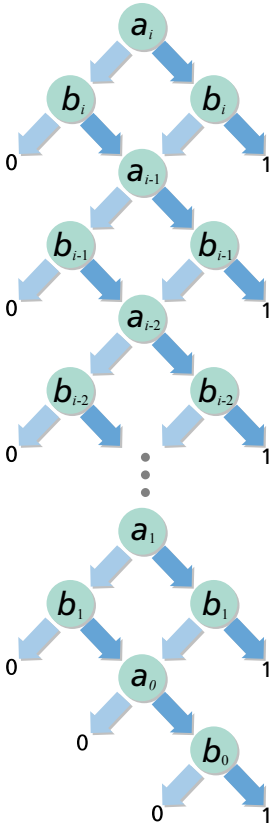
Carry-Bit Graph



i-th bit	terms	nodes
1	1	2
2	3	5
3	7	8
4	15	11
5	31	14
6	63	17
7	127	20
8	255	23
9	511	26
10	1023	29
11	2047	32
n	$2^n - 1$	$3\,n - 1$

- Sophisticated ordering of variables
- Linear growth of ZDDs (plaited structure)
- Fast generation and arithmetic operations

Carry-Bit Graph



i-th bit	terms	nodes
1	1	2
2	3	5
3	7	8
4	15	11
5	31	14
6	63	17
7	127	20
8	255	23
9	511	26
10	1023	29
11	2047	32
n	$2^n - 1$	$3\,n - 1$

- Sophisticated ordering of variables
- Linear growth of ZDDs (plaited structure)
- Fast generation and arithmetic operations

Demo: $n = 4096$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Application: Abstract 4096-Bit Adder and Logic

```
adder_bits=4096
adder_block=AdderBlock(sums="s",carries="c",
                       input1="a",input2="b",
                       adder_bits=adder_bits,start_index=1)

declare_ring([Block("x",100),adder_block,Block("y",100)])
ideal=[
    y(0)+y(1)+1,
    a(adder_bits)+1,b(adder_bits)+y(0)+y(1),
    x(0)+(c(adder_bits))*x(1)
]
adder_block.implement(ideal)

claims=[
  c(adder_bits)+1,
  if_then([x(1)],[x(0)]),
  if_then([a(1),b(1)],[c(1),s(1)]),
  if_then([a(1)+1],[c(1)+b(1),s(1)+b(1)+1]),
  s(adder_bits)+c(adder_bits-1),
  x(33)
]
```

Counter Example for Failed Claims

- A claim c fails $\iff$ $q := \mathsf{NF}(c, R) \neq 0$
- A Boolean polynomial is 0 $\iff$ it is 0 as a function

Algorithm

Boolean polynomial q reduced w. r. t. R, R red. Gröbner basis

$\mathsf{Variables}(q) \subseteq V := \{x_1, .., x_n\} \backslash \{\text{Leading Terms}(R)\}$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Counter Example for Failed Claims

- A claim c fails $\iff$ $q := \mathsf{NF}(c, R) \neq 0$
- A Boolean polynomial is 0 $\iff$ it is 0 as a function

Algorithm

Boolean polynomial q reduced w. r. t. R, R red. Gröbner basis

$\mathsf{Variables}(q) \subseteq V := \{x_1, .., x_n\} \backslash \{\text{Leading Terms}(R)\}$

Step 1 Find $v_1, \ldots, v_n \in \{0, 1\}$, where $q(v_1, ..., v_n) = 1$

Now we have to make it compatible with relations in R:

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Counter Example for Failed Claims

- A claim c fails $\iff$ $q := \mathsf{NF}(c, R) \neq 0$
- A Boolean polynomial is 0 $\iff$ it is 0 as a function

Algorithm

Boolean polynomial q reduced w. r. t. R, R red. Gröbner basis

$\mathsf{Variables}(q) \subseteq V := \{x_1, .., x_n\} \backslash \{\text{Leading Terms}(R)\}$

Step 1 Find $v_1, \ldots, v_n \in \{0, 1\}$, where $q(v_1, ..., v_n) = 1$

Now we have to make it compatible with relations in R:

Step 2 For every $x_i \in \mathsf{lead}(R)$, we have $p \in R$, with $x_i = \mathsf{lead}(x)$ and $\mathsf{tail}(p) \in K[V]$

Redefine $v_i := \mathsf{tail}(p)(v_1, \ldots, v_n)$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Counter Example for Failed Claims

- A claim c fails $\iff$ $q := \mathsf{NF}(c, R) \neq 0$
- A Boolean polynomial is 0 $\iff$ it is 0 as a function

Algorithm

Boolean polynomial q reduced w. r. t. R, R red. Gröbner basis

$\mathsf{Variables}(q) \subseteq V := \{x_1, .., x_n\} \backslash \{\text{Leading Terms}(R)\}$

Step 1 Find $v_1, \ldots, v_n \in \{0, 1\}$, where $q(v_1, ..., v_n) = 1$

Now we have to make it compatible with relations in R:

Step 2 For every $x_i \in \mathsf{lead}(R)$, we have $p \in R$, with $x_i = \mathsf{lead}(x)$ and $\mathsf{tail}(p) \in K[V]$

Redefine $v_i := \mathsf{tail}(p)(v_1, \ldots, v_n)$

Theorem

$v_1, \ldots, v_n$ provides a valid counter example

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Algorithm: Find a One of a Boolean Polynomial

Input: Boolean polynomial $p \neq 0$

Out: $\{x_i \to v_i\}_i : p(v_1, \ldots, v_k) = 1$

if $p = 1$ **then**
  return $\emptyset$ // empty tuple
end if
Set $r :=$ rootnode(p)
Set $x :=$ Variable corresponding to r
if elseBranch(r) is 0 **then**
  Set $v = 1$
  Set $q =$ thenBranch(r) // plug in 1
else
  Set $v = 0$
  Set $q =$ elseBranch(r) // plug in 0
end if
return $\{x \to v\} \cup$ find_one(q)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Summary

- Boolean polynomials as subsets of the power set of the ring variables, which can be represented efficiently as ZDDs

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Summary

- Boolean polynomials as subsets of the power set of the ring variables, which can be represented efficiently as ZDDs
- High-level C++ library: internal ZDD structure in polynomials encapsulated, as well as transparent access to ZDD structure

Summary

- Boolean polynomials as subsets of the power set of the ring variables, which can be represented efficiently as ZDDs
- High-level C++ library: internal ZDD structure in polynomials encapsulated, as well as transparent access to ZDD structure
- Easy usable Python interface for rapid prototyping of sophisticated heuristics for Gröbner base computations

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Summary

- Boolean polynomials as subsets of the power set of the ring variables, which can be represented efficiently as ZDDs
- High-level C++ library: internal ZDD structure in polynomials encapsulated, as well as transparent access to ZDD structure
- Easy usable Python interface for rapid prototyping of sophisticated heuristics for Gröbner base computations
- Gröbner property checker for formal verification of given properties. Generation of counter examples, if claims fail

Summary

- Boolean polynomials as subsets of the power set of the ring variables, which can be represented efficiently as ZDDs
- High-level C++ library: internal ZDD structure in polynomials encapsulated, as well as transparent access to ZDD structure
- Easy usable Python interface for rapid prototyping of sophisticated heuristics for Gröbner base computations
- Gröbner property checker for formal verification of given properties. Generation of counter examples, if claims fail
- Optimized setup of arithmetic components (adder)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach