PolyBoRi - A Gröbner Basis Framework for Boolean Polynomials

Michael Brickenstein

Mathematisches Forschungsinstitut Oberwolfach

Alexander Dreyer*

Fraunhofer Institut für Techno- und Wirtschaftsmathematik (ITWM)

Oberseminar im Wintersemester 2007/2008
Kryptographie und Computeralgebra, TU Darmstadt, 31. 1. 2008

*Presentation

Fraunhofer Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Basic Data

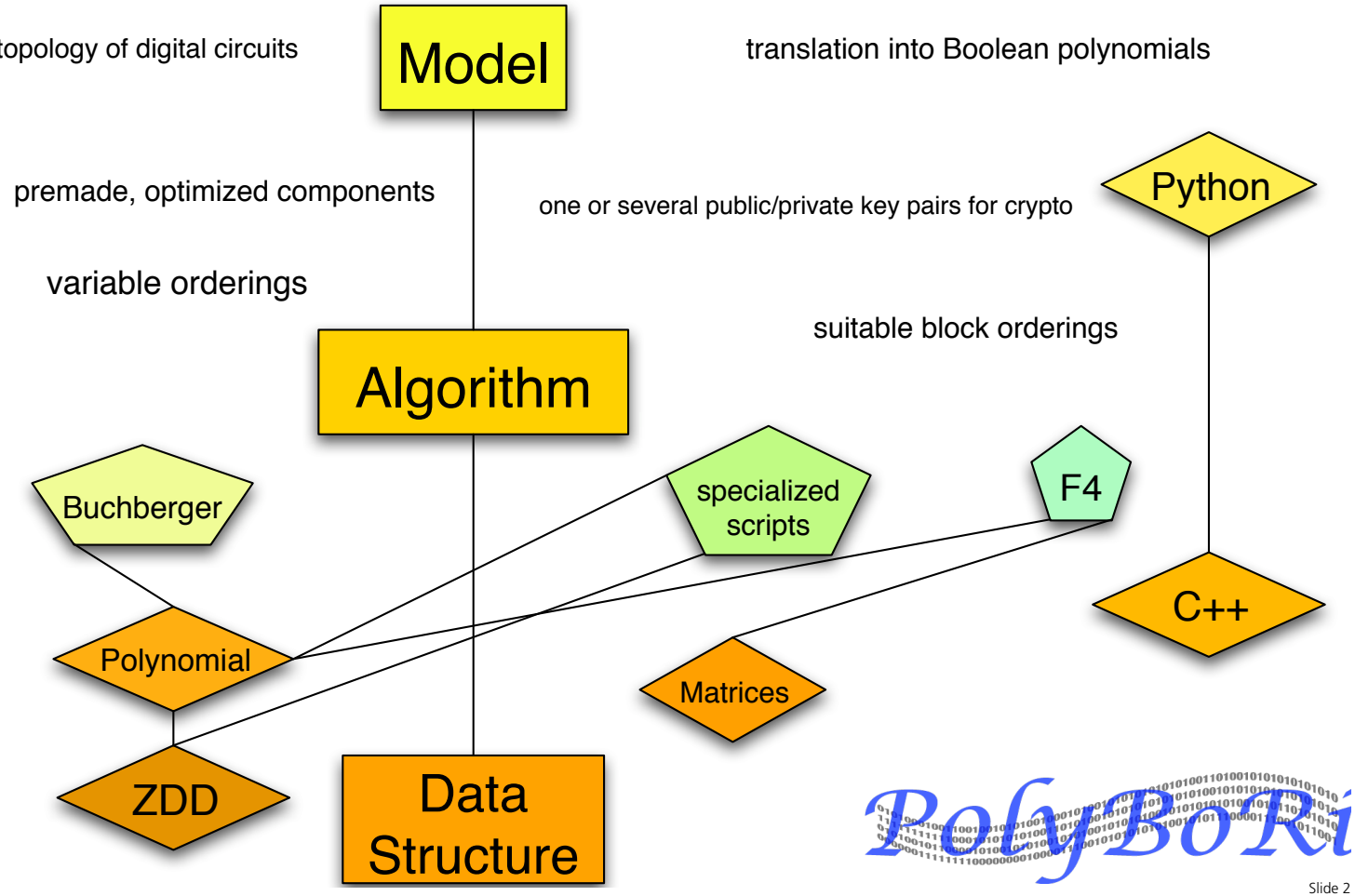
PolyBoRi	***Poly***nomials over ***Bo***olean ***Ri***ngs
Website	`http://polybori.sourceforge.net`
DFG Project	"Development, implementation and application of mathematical-algebraic algorithms for formal verification of digital systems with arithmetic blocks"
Fraunhofer ITWM	Alexander Dreyer (Department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

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Overview

Optimization on many levels



Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Translation from $\mathrm{GF}(2^r)$ to $\mathrm{GF}(2)$

Identification of fields

$\mathrm{GF}(2^r) \leftrightarrow \mathrm{GF}(2)[a]/\langle m \rangle =: F$,
irreducible $m \in \mathrm{GF}(2)[a]$ with $\deg m = r$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Translation from $\mathrm{GF}(2^r)$ to $\mathrm{GF}(2)$

Identification of fields

$$\mathrm{GF}(2^r) \leftrightarrow \mathrm{GF}(2)[a]/\langle m \rangle =: F,$$
irreducible $m \in \mathrm{GF}(2)[a]$ with $\deg m = r$

Identification of rings

$$\mathrm{GF}(2^r)[x_1, \ldots x_n] \leftrightarrow \mathrm{GF}(2)[a][x_1, \ldots, x_n]/\langle m \rangle =: P$$

Monomorphism of F-algebras

$$\phi : P \to P_2, \quad x_i \mapsto \sum_{j=0}^{r-1} y_{i,j} \cdot a^j$$

$$P_2 = F[y_{1,0}, \ldots, y_{1,r-1}, y_{2,0}, \ldots, y_{2,r-1}, \ldots, y_{n,0}, \ldots, y_{n,r-1}]$$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Translation from $\mathrm{GF}(2^r)$ to $\mathrm{GF}(2)$

Identification of fields

$\mathrm{GF}(2^r) \leftrightarrow \mathrm{GF}(2)[a]/\langle m \rangle =: F$,
irreducible $m \in \mathrm{GF}(2)[a]$ with $\deg m = r$

Identification of rings

$$\mathrm{GF}(2^r)[x_1, \ldots x_n] \leftrightarrow \mathrm{GF}(2)[a][x_1, \ldots, x_n]/\langle m \rangle =: P$$

Monomorphism of F-algebras

$$\phi : P \to P_2, \quad x_i \mapsto \sum_{j=0}^{r-1} y_{i,j} \cdot a^j$$

$$P_2 = F[y_{1,0}, \ldots, y_{1,r-1}, y_{2,0}, \ldots, y_{2,r-1}, \ldots, y_{n,0}, \ldots, y_{n,r-1}]$$

Canonical components f_i

Let $f \in P \Rightarrow \exists f_i \in \mathrm{GF}(2)[y_{1,0}, \ldots, y_{1,r-1}, y_{2,0}, \ldots, y_{n,r-1}] =: Q$
s. th. $\phi(f) = \sum_{i=0}^{r-1} a^i \cdot f_i$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Translation from $\mathrm{GF}(2^r)$ to $\mathrm{GF}(2)$

Canonical components $\phi(f) = \sum_{i=0}^{r-1} a^i \cdot f_i$

Map to canonical components $\tau : P \to \mathsf{PowerSet}(Q), \quad f \mapsto \{f_0, \ldots, f_{r-1}\}$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Translation from $\mathrm{GF}(2^r)$ to $\mathrm{GF}(2)$

Canonical components $\phi(f) = \sum_{i=0}^{r-1} a^i \cdot f_i$

Map to canonical components $\tau : P \to \mathrm{PowerSet}(Q), \quad f \mapsto \{f_0, \ldots, f_{r-1}\}$

Correspondence $G = \{g_1, \ldots, g_m\} \subset P \leftrightarrow H = \cup_{g \in G} \tau(f) \subset Q$

Isomorphism of $\mathrm{GF}(2)$-vector spaces $\lambda : F^n \to \mathrm{GF}(2)^{r \cdot n}$, $a^j \cdot e_i \mapsto E_{(i-1)\cdot r + j + 1}$ $\quad (F = \mathrm{GF}(2)[a]/\langle m \rangle \cong \mathrm{GF}(2^r))$

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Translation from $\mathrm{GF}(2^r)$ to $\mathrm{GF}(2)$

Canonical components $\phi(f) = \sum_{i=0}^{r-1} a^i \cdot f_i$

Map to canonical components $\tau : P \to \mathsf{PowerSet}(Q), \quad f \mapsto \{f_0, \ldots, f_{r-1}\}$

Correspondence $G = \{g_1, \ldots, g_m\} \subset P \;\leftrightarrow\; H = \cup_{g \in G} \tau(f) \subset Q$

Isomorphism of $\mathrm{GF}(2)$-vector spaces

$\lambda\colon\ F^n \to \mathrm{GF}(2)^{r \cdot n}$ $\qquad (F = \mathrm{GF}(2)[a]/\langle m \rangle \mathrel{\widehat{=}} \mathrm{GF}(2^r))$

$a^j \cdot e_i \mapsto E_{(i-1)\cdot r + j + 1}$

Theorem $\lambda(\mathsf{V}(G \cup \mathrm{FP}_P)) = \mathsf{V}(H \cup \mathrm{FP}_Q)$

Field polynomials

$\mathrm{FP}_P = \{x_1^{2^r} - x_1, \ldots, x_n^{2^r} - x_n\},$
$\mathrm{FP}_Q = \{y_{i,j}^2 - y_{i,j} \mid 1 \le i \le n, 0 \le j < r\}$

Describe vanishing points over $\mathrm{GF}(2^r)$ and $\mathrm{GF}(2)$, resp.

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Translation from $\mathrm{GF}(2^r)$ to $\mathrm{GF}(2)$

Disadvantages

- More equations (r times as much)

Advantages

- Simpler base field $\mathrm{GF}(2)$
- Lower degree per variable (bound to 1, if FP_Q is omitted)
- Boolean polynomial approach applicable (PolyBoRi)

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

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

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

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

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

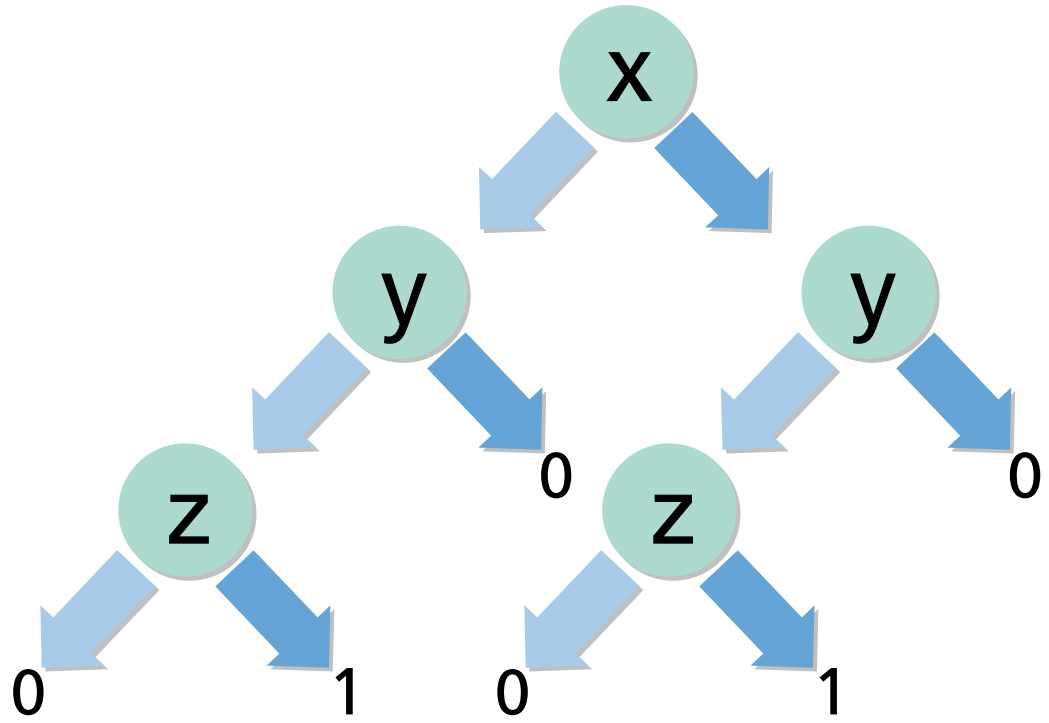
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Zero-suppressed Binary Decision Diagrams

Binary Decision Diagram

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



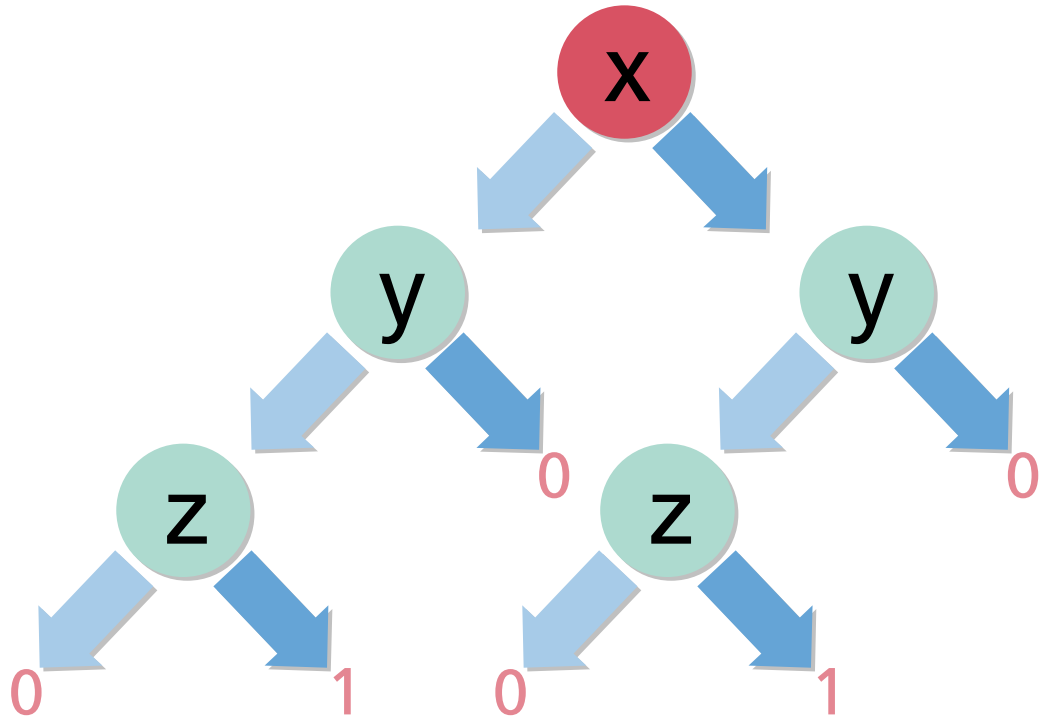
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

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

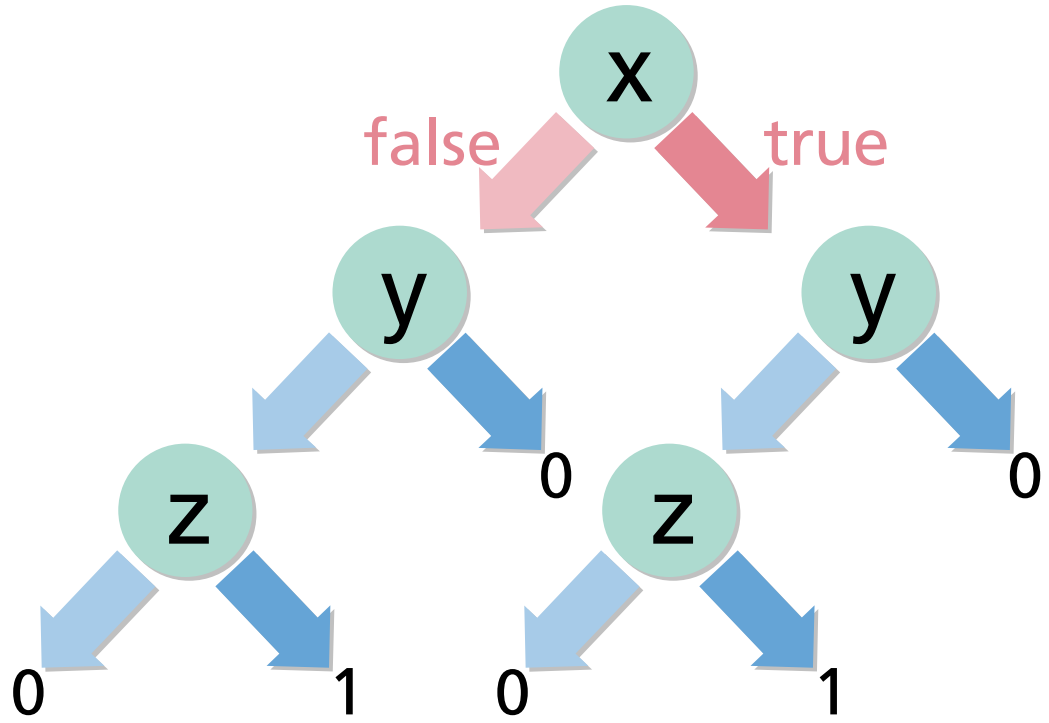
Mathematisches Forschungsinstitut Oberwolfach

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

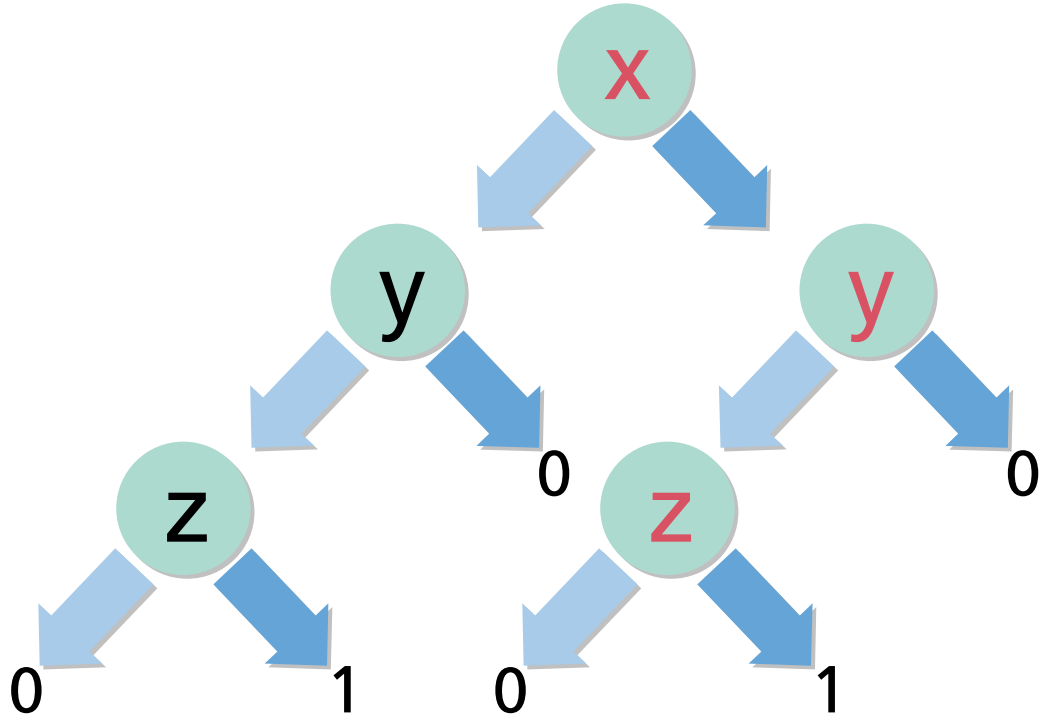
Zero-suppressed Binary Decision Diagrams

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



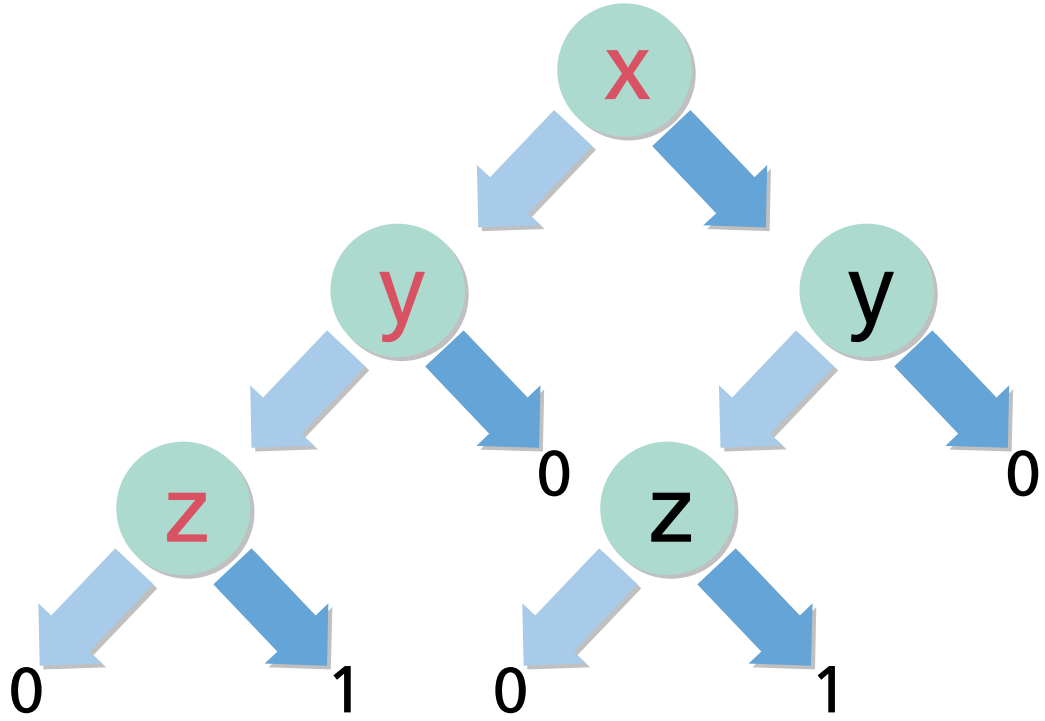
Zero-suppressed Binary Decision Diagrams

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

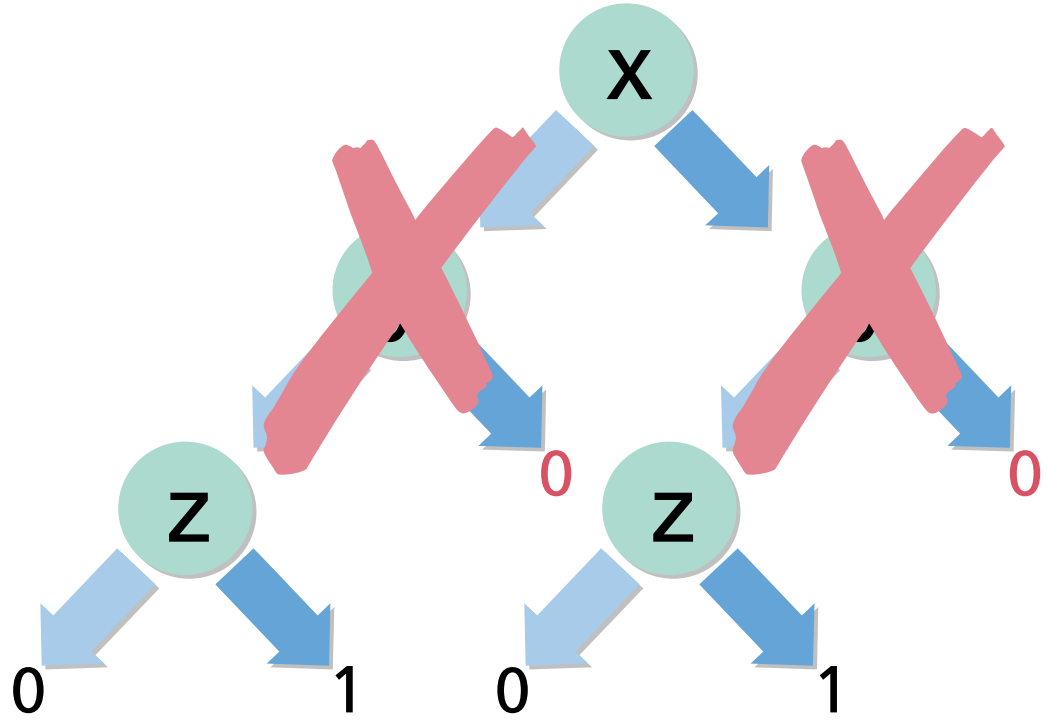
Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

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

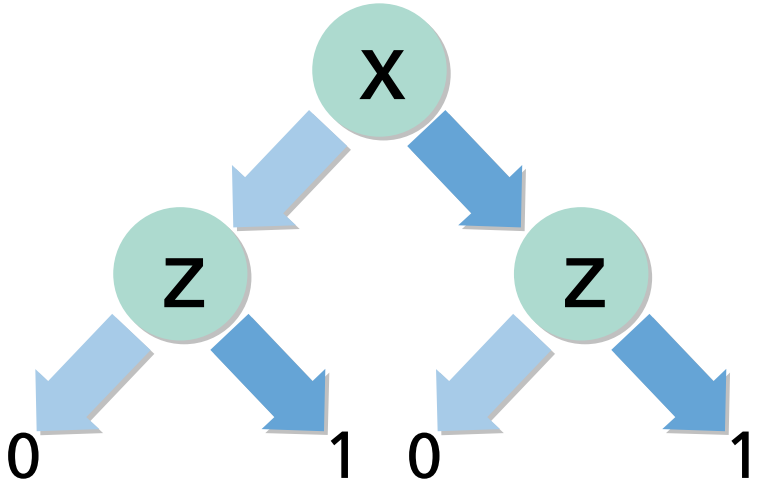
Zero-suppressed Binary Decision Diagrams

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

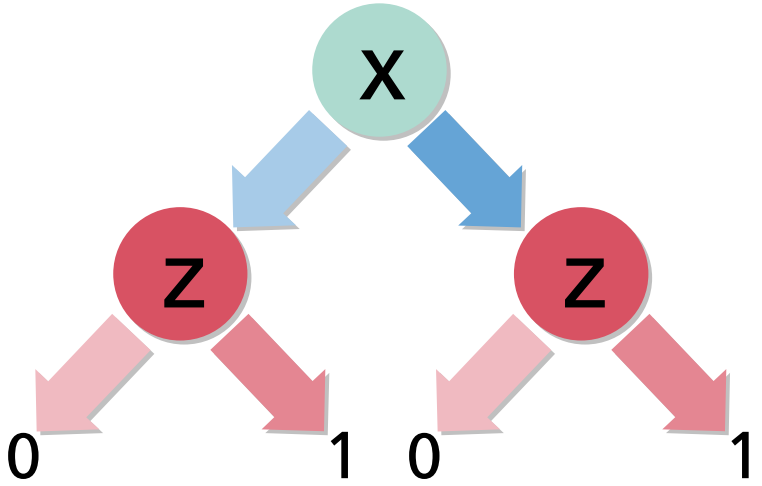
Zero-suppressed Binary Decision Diagrams

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

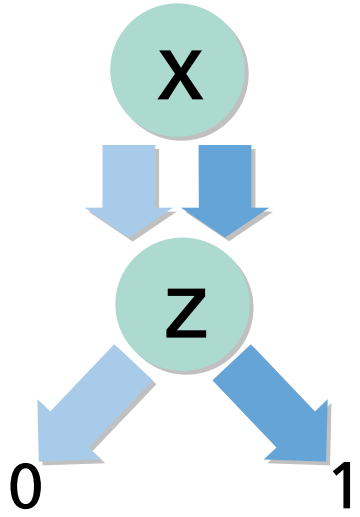
Zero-suppressed Binary Decision Diagrams

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



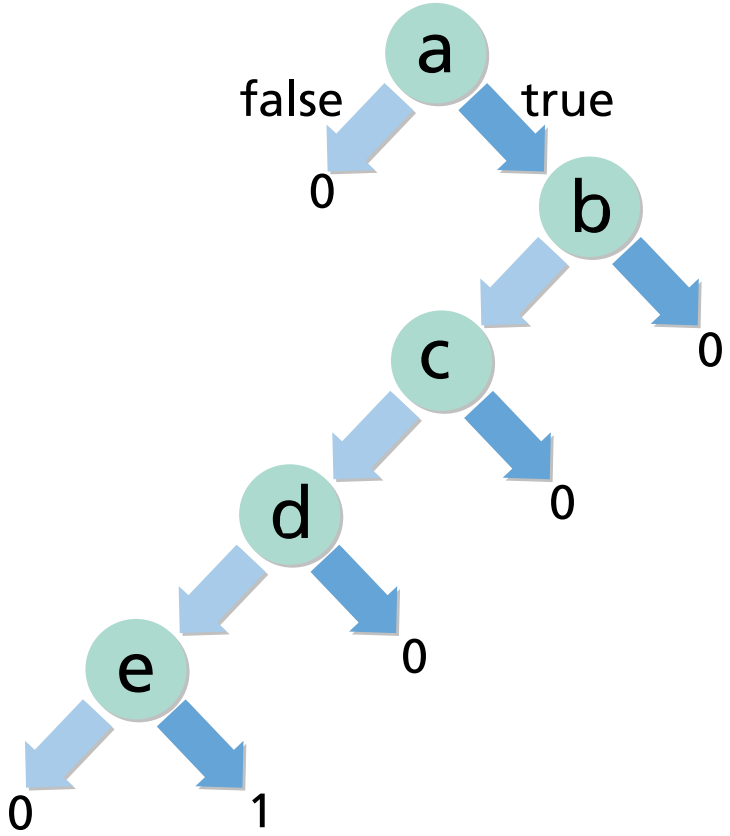
Zero-suppressed BDD (ZDD)

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

Equal subgraphs merged

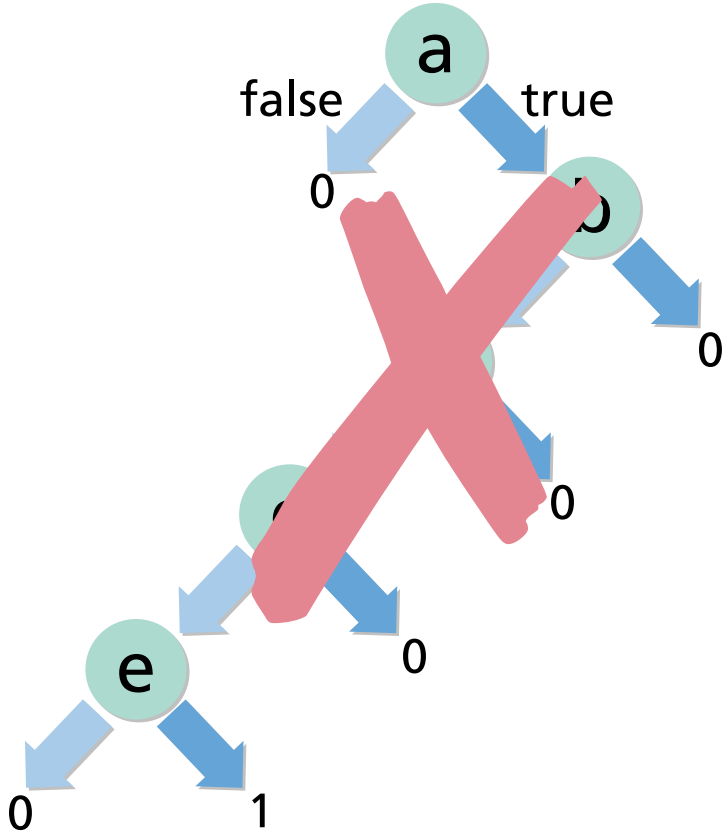
ZDDs for Sparse Sets

Example: $\{\{a, e\}\}$



ZDDs for Sparse Sets

Example: $\{\{a, e\}\}$

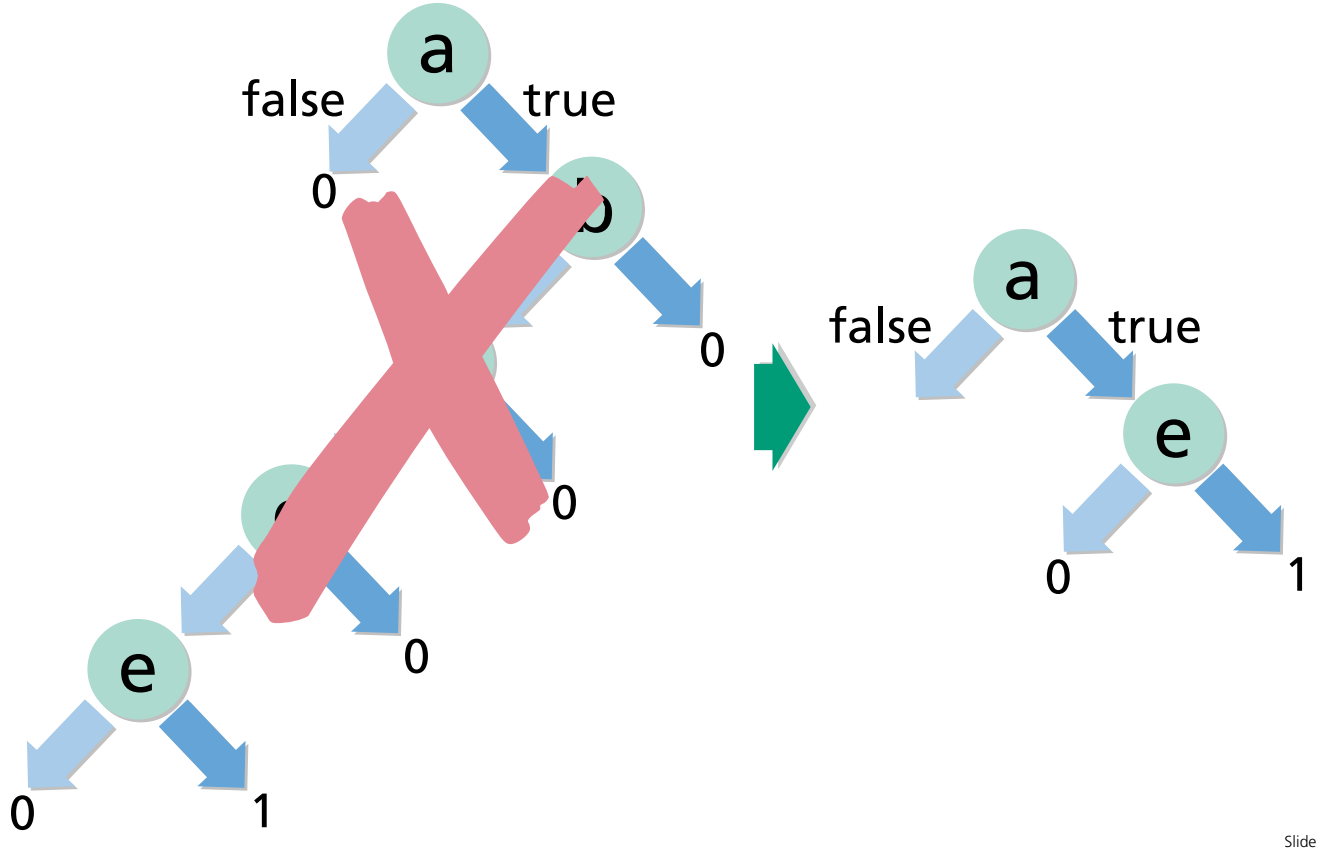


Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

ZDDs for Sparse Sets

Example: $\{\{a, e\}\}$



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 and structured subsets of power sets

Advantages of ZDDs

Idea

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

Reasons

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

Advantages of ZDDs

Idea

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

Reasons

- Compact data structure
- Suitable for sparse and structured subsets of power sets
- Polynomial structure recognizable
  - Paths $\widehat{=}$ polynomial terms
  - Natural path sequence $\widehat{=}$ lexicographical term ordering
- Unique diagram nodes (less memory, reference counting)
- Caching of operations

Boolean Polynomials as ZDDs

Example $x\,y + x + z \mathrel{\widehat{=}} \{\{x, y\}, \{x\}, \{z\}\}$

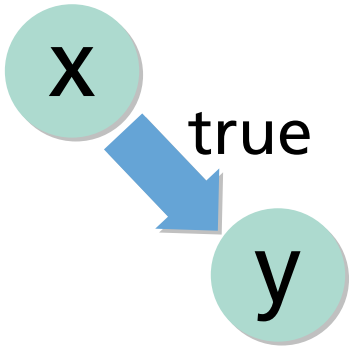
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Boolean Polynomials as ZDDs

Example

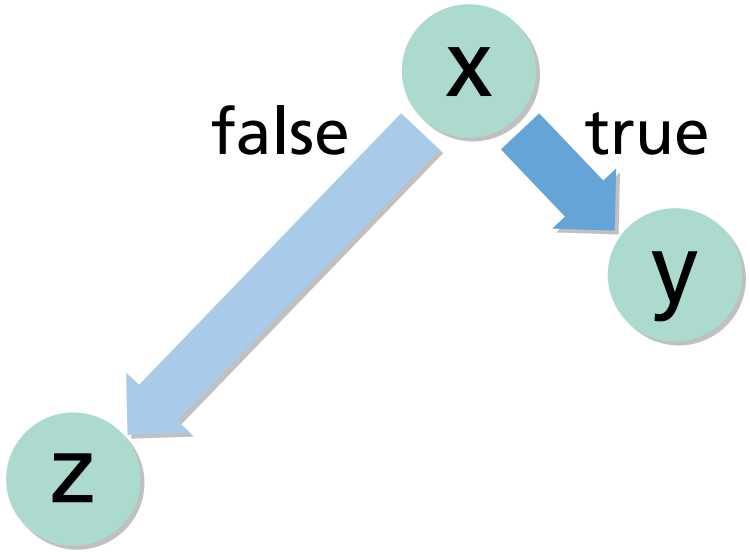
$$x\,y + x + z \mathrel{\widehat{=}} \{\{x, y\}, \{x\}, \{z\}\}$$



Boolean Polynomials as ZDDs

Example

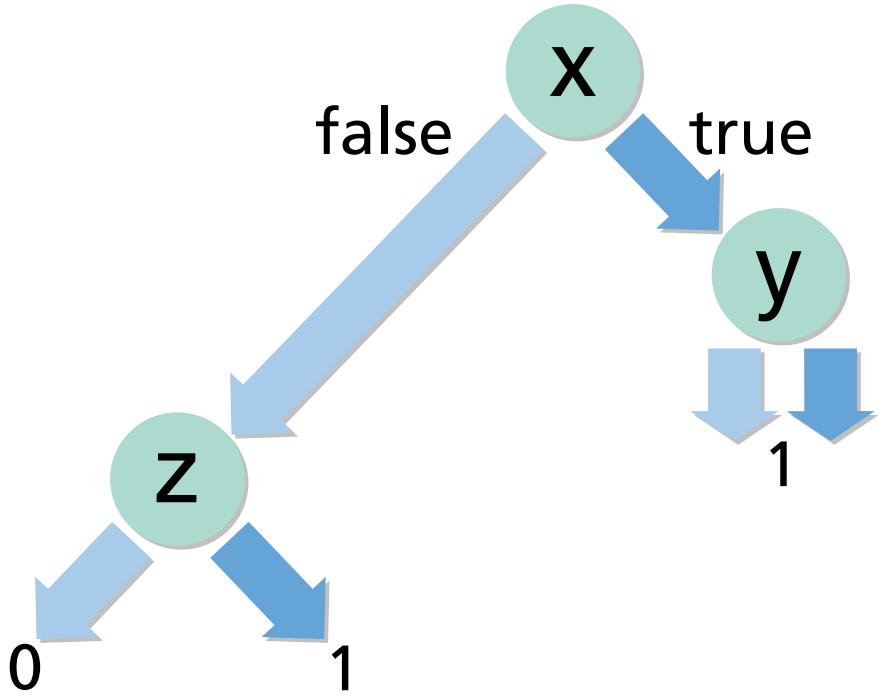
$$x\,y + x + z \mathrel{\widehat{=}} \{\{x, y\}, \{x\}, \{z\}\}$$



Boolean Polynomials as ZDDs

Example

$$x\,y + x + z \mathrel{\widehat{=}} \{\{x, y\}, \{x\}, \{z\}\}$$



Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

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

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

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)

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

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
Interfaces `www.sagemath.org` `singular.mathematik.uni-kl.de`	SAGE (python code compatible; community-made) Singular (Prototype)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Data Structures

Polynomial Ring
(tiny excerpt)

Generates base components

Ordering (runtime changeable)

Active ring (shared pointer) based on decision diagram manager

```
class BoolePolyRing {
 public:
  // ...
  BoolePolyRing(size_type nvars, ordercode_type order = lp);

  dd_type variable(idx_type i) const;  // i-th variable
  dd_type zero() const;          // Constant polynomials
  dd_type one() const;

  size_type nVariables() const;

  void changeOrdering(ordercode_type);


 protected:
  manager_ptr pMgr;
  // ...
};
```

Data Structures

Example
(tiny excerpt)

Object-oriented operations

High-level abstractions

Likewise for monomials,
variables, and exponent vector

```
class BoolePolynomial {
 public:
  // ...
  BoolePolynomial();                // Default constructor
  BoolePolynomial(bool_type);       // Construct from 0 or 1

  self& operator+=(const self&); // Arithmetical operations
  self& operator*=(const self&);

  bool operator==(const self&) const; // Logical operations
  bool operator!=(const self&) const;

  monom_type lead() const;          // Get leading term
  size_type deg() const;            // Degree of the polynomial
  monom_type usedVariables() const;
  // ... various term access functions ...
 private:
  dd_type m_dd;                     // Actual decision diagram
};
```

Term-access Functions

Forward iterator over terms

```
const_iterator begin() const;
const_iterator end() const;
```

in natural – lexicographical – term ordering

Term-access Functions

Forward iterator over terms

```
const_iterator begin() const;
const_iterator end() const;
```

in natural – lexicographical – term ordering

Runtime-selected ordering

```
ordered_iterator orderedBegin() const;
ordered_iterator orderedEnd() const;
```

Term-access Functions

Forward iterator over terms

```
const_iterator begin() const;
const_iterator end() const;
```

in natural – lexicographical – term ordering

Runtime-selected ordering

```
ordered_iterator orderedBegin() const;
ordered_iterator orderedEnd() const;
```

Compile-time selected ordering

$\langle order \rangle$`_iterator genericBegin(`$\langle order \rangle$`_tag) const;`
$\langle order \rangle$`_iterator genericEnd(`$\langle order \rangle$`_tag) const;`

currently for lexicographic, degree-lexicographic, and degree-reverse-lexicographic (ascending variable order) orderings

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Term-access Functions

Forward iterator over terms

```
const_iterator begin() const;
const_iterator end() const;
```

in natural – lexicographical – term ordering

Runtime-selected ordering

```
ordered_iterator orderedBegin() const;
ordered_iterator orderedEnd() const;
```

Compile-time selected ordering

$\langle order \rangle$`_iterator genericBegin(`$\langle order \rangle$`_tag) const;`
$\langle order \rangle$`_iterator genericEnd(`$\langle order \rangle$`_tag) const;`

currently for lexicographic, degree-lexicographic, and degree-reverse-lexicographic (ascending variable order) orderings

Iterator-like structure for navigation over ZDD nodes

```
navigator navigation() const;
```

`incrementThen()`, `incrementElse()` replacing `operator++()`

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

Python Interface (Data Format)

Script level

Easy input and handling of complex polynomial systems

```
declare_ring([
  AlternatingBlock(["c","s"],56,start_index=1,reverse=True),
  AlternatingBlock(["a","b"],8,reverse=True),
],globals())

ideal = [
  a(0)*b(1)*a(1)*b(0)+c(1),
  a(0)*b(1)+a(1)*b(0)+s(1),
  a(1)*b(1)*a(2)*b(0)+c(3)
]

ideal = function(ideal);
```

Several variants for Gröber base computation

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Comparison of Storage Types for Boolean Polynomials

Tune strategies and heuristics to strengths of underlying data structure!

	ZDD	linked list	buckets[1]	factory[2]
leading term	++|0 |– –[3]	++	+	+ (close ZDD)
p1 + p2	++	+	++	0
tail(p)[4]	–	++	+	0
redNF(p,m)	++	0	–	– –
canonicalize	++	+	– –	+
memory use	++[5]	+	–	– –
iteration	0[6]	++	– –	– (see ZDD)
p1 * p2	++[7]	+	– –	+
factorize	++	–	– –	++

[1]) partly reduced container [2]) recursive expression
[3]) lex | special cases | general case [4]) often during reduction with monomial [5]) every substructure stored once
[6]) possible, but ordering-dependent [7]) automatically reduced

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

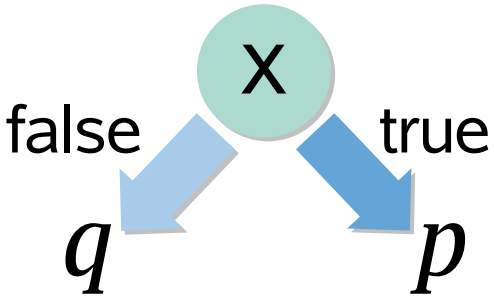
ZDD normalform	Unique diagram root nodes $a = b \iff \texttt{rootnode}(a)$ is $\texttt{rootnode}(b)$
Reference counting	Lower memory usage (no deep copies)
Caching of operations	$a \diamond b$ never evaluated twice (also for sub-diagrams)
Advantage	Speed-up recursive procedures

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Caching and Recursion

Example: $\deg(f)$



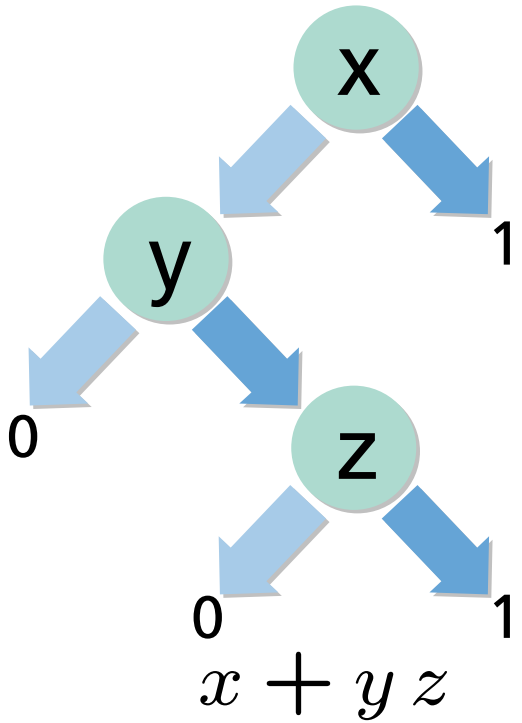
$$f = q + x \cdot p$$

Input: f Boolean polynomial **Output:** $d = \deg(f)$

if $f \in \{0, 1\}$ **then**
  set $d := 0$
else if isCached(deg, f) **then**
  set $d :=$ cache(deg, f)
else
  set $d :=$ max(deg(thenBranch(f)) + 1,
    deg(elseBranch(f)))
  insert cache(deg, f) $:= d$
end if

Caching and Recursion

Example: $\deg(f)$

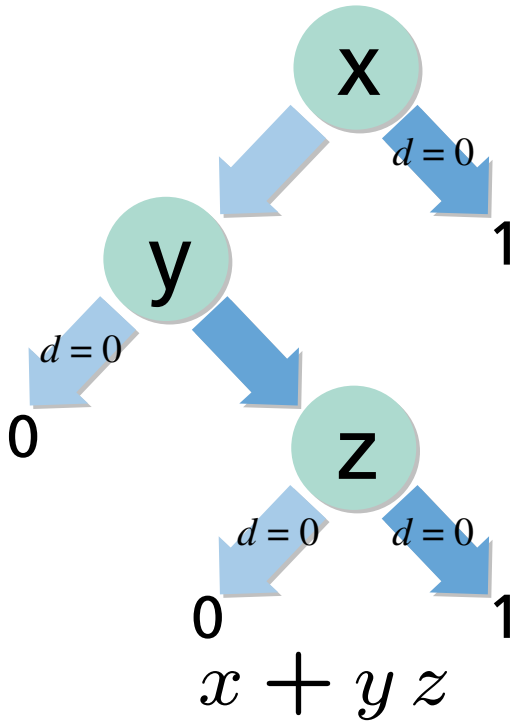


Input: f Boolean polynomial **Output:** $d = \deg(f)$

if $f \in \{0, 1\}$ **then**
  set $d := 0$
else if isCached(deg, f) **then**
  set $d :=$ cache(deg, f)
else
  set $d :=$ max(deg(thenBranch(f)) + 1,
    deg(elseBranch(f)))
  insert cache(deg, f) $:= d$
end if

Caching and Recursion

Example: $\deg(f)$

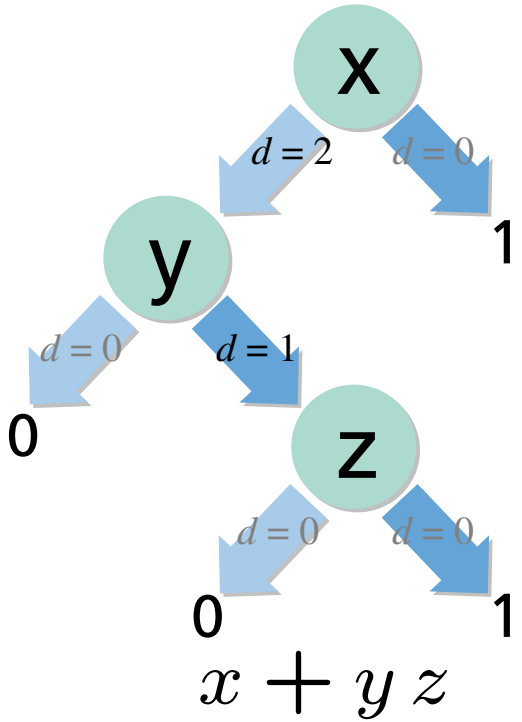


Input: f Boolean polynomial **Output:** $d = \deg(f)$

if $f \in \{0, 1\}$ **then**
  set $d := 0$
else if isCached(deg, f) **then**
  set $d :=$ cache(deg, f)
else
  set $d := \max(\deg(\text{thenBranch}(f)) + 1,$
    $\deg(\text{elseBranch}(f)))$
  insert cache(deg, f) $:= d$
end if

Caching and Recursion

Example: $\deg(f)$

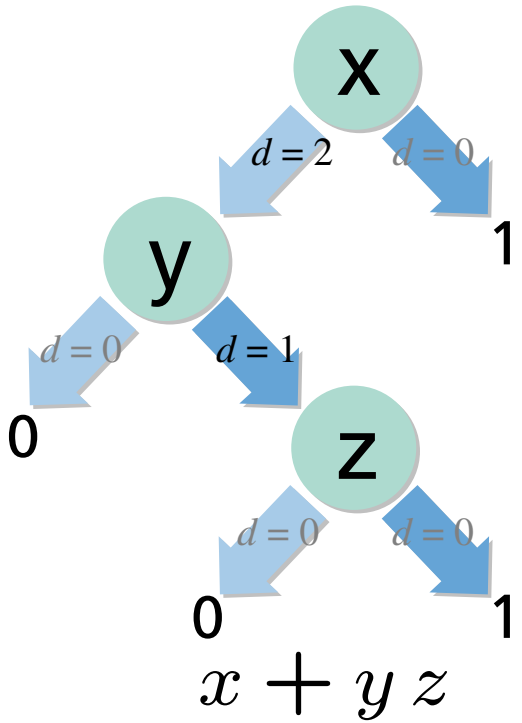


Input: f Boolean polynomial **Output:** $d = \deg(f)$

if $f \in \{0, 1\}$ **then**
  set $d := 0$
else if isCached(deg, f) **then**
  set $d :=$ cache(deg, f)
else
  set $d := \max(\deg(\text{thenBranch}(f)) + 1,$
    $\deg(\text{elseBranch}(f)))$
  insert cache(deg, f) $:= d$
end if

Caching and Recursion

Example: $\deg(f)$



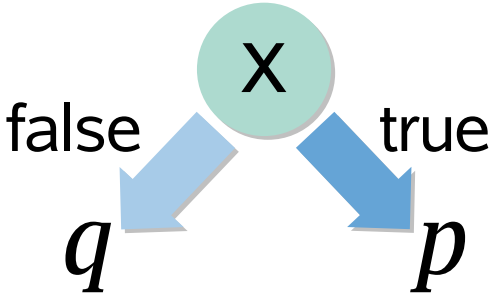
Input: f Boolean polynomial **Output:** $d = \deg(f)$

if $f \in \{0, 1\}$ **then**
  set $d := 0$
else if isCached(deg, f) **then**
  set $d :=$ cache(deg, f)
else
  set $d :=$ max(deg(thenBranch(f)) + 1,
    deg(elseBranch(f)))
  insert cache(deg, f) $:= d$
end if

- Reuse for polynomials with likewise structure.
- Fast access to intermediate results by other functions.

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)

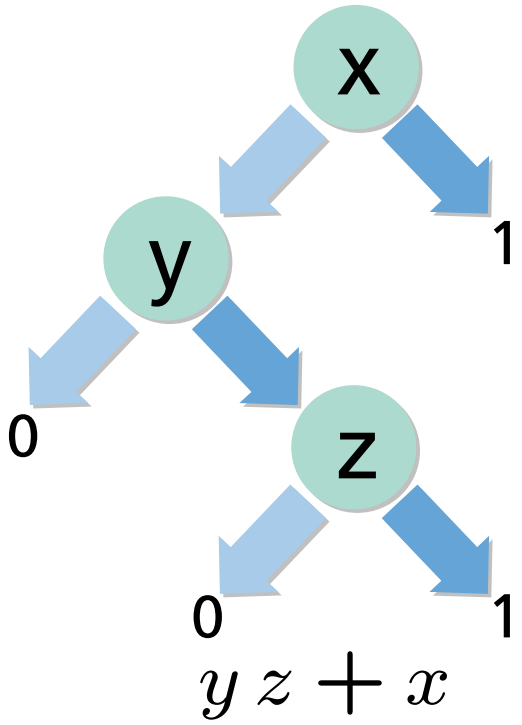
if $f \in \{0, 1\}$ **then**
  set $t := 1$
else if isCached(lead, f) **then**
  set $t :=$ cache(lead, f)
else
  set $x :=$ root variable of f
  if $\deg(f) = \deg(\mathrm{thenBranch}(f)) + 1$ **then**
    set $t := x \cdot \mathrm{lead}(\mathrm{thenBranch}(f))$
  else
    set $t := \mathrm{lead}(\mathrm{elseBranch}(f))$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

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Caching and Recursion

Example: lead(f)



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

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

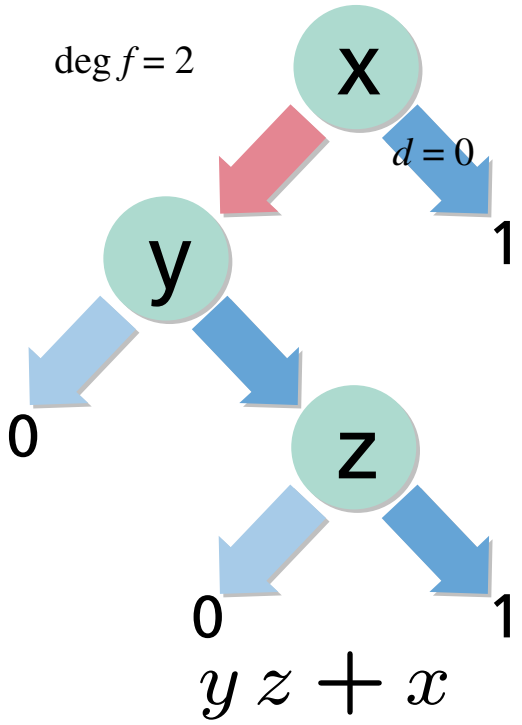
if $f \in \{0, 1\}$ **then**
  set $t := 1$
else if isCached(lead, f) **then**
  set $t :=$ cache(lead, f)
else
  set $x :=$ root variable of f
  if $\deg(f) = \deg(\mathrm{thenBranch}(f)) + 1$ **then**
    set $t := x \cdot \mathrm{lead}(\mathrm{thenBranch}(f))$
  else
    set $t := \mathrm{lead}(\mathrm{elseBranch}(f))$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

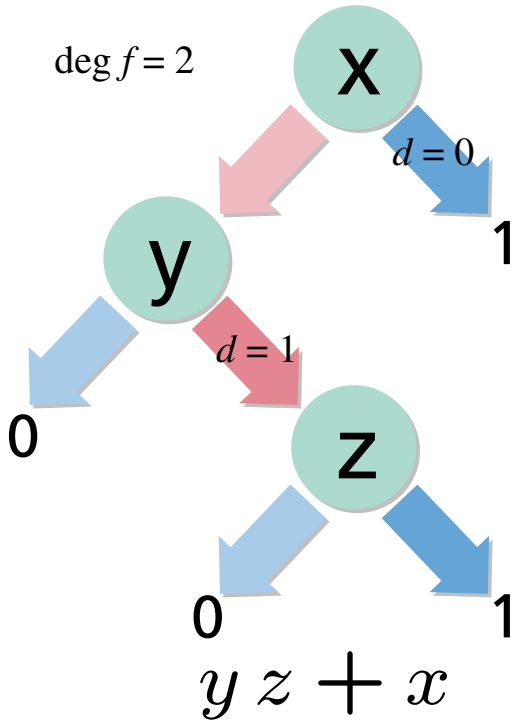
if $f \in \{0, 1\}$ **then**
  set $t := 1$
else if isCached(lead, f) **then**
  set $t :=$ cache(lead, f)
else
  set $x :=$ root variable of f
  if $\deg(f) = \deg(\mathrm{thenBranch}(f)) + 1$ **then**
    set $t := x \cdot \mathrm{lead}(\mathrm{thenBranch}(f))$
  else
    set $t := \mathrm{lead}(\mathrm{elseBranch}(f))$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

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 $\longrightarrow 1$
  $\mathsf{Or}(x, y) \longrightarrow x \cdot y$, $\mathsf{Not}(x) \longrightarrow 1 + x$

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

Gröbner Basis

- Let $G = \{g_1, \ldots, g_m\}$ be (multivariate) polynomials
- Define an ordering $>$ on the monomials of $K[x_1, \ldots, x_m]$
- For a polynomial p define lead(p) the biggest monomial occurring in p

Gröbner Basis

- Let $G = \{g_1, \ldots, g_m\}$ be (multivariate) polynomials
- Define an ordering $>$ on the monomials of $K[x_1, \ldots, x_m]$
- For a polynomial p define lead(p) the biggest monomial occurring in p

G is a Gröbner basis $\iff$ for every $0 \neq p$ in the ideal generated by G there exists $g \in G$: lead(g) divides lead(p)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Gröbner Basis

- Let $G = \{g_1, \ldots, g_m\}$ be (multivariate) polynomials
- Define an ordering $>$ on the monomials of $K[x_1, \ldots, x_m]$
- For a polynomial p define lead(p) the biggest monomial occurring in p

G is a Gröbner basis $\iff$ for every $0 \neq p$ in the ideal generated by G there exists $g \in G$: lead(g) divides lead(p)

Examples

- $\langle G \rangle$ is the whole ring, then $G \ni 1$
- $\{x_1 - c_1, \ldots, x_n - c_n\}$ for a (radical) ideal with unique solution $c_1, \ldots, c_n$

Linear Lead-Rewriting System

System of Boolean polynomials
- $G = \{g_1, \ldots, g_m\}$ in variables $x_1, \ldots, x_n$

Suppose, the following holds:
- For each i: lead(g_i) is a variable x_{gi} ($\rightarrow x_{gi}$ not in $g_i -$ lead(g_i))
- For $i \neq j$: lead$(g_i) \neq$ lead(g_j)

$\Rightarrow$ Then G is a Gröbner basis and
$R := G \cup \{x_i^2 + x_i | i = 1, \ldots, n\}$ is a Gröbner basis

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Linear Lead-Rewriting System

System of Boolean polynomials

- $G = \{g_1, \ldots, g_m\}$ in variables $x_1, \ldots, x_n$

Suppose, the following holds:

- For each i: lead(g_i) is a variable x_{gi} ($\rightarrow x_{gi}$ not in $g_i -$ lead(g_i))
- For $i \neq j$: lead$(g_i) \neq$ lead(g_j)

$\Rightarrow$ Then G is a Gröbner basis and
$R := G \cup \{x_i^2 + x_i | i = 1, \ldots, n\}$ is a Gröbner basis

For each Boolean polynomial p there exists exactly one Boolean polynomial q in $\{x_1, .., x_n\} \backslash \{$lead$(g_1), ...,$ lead$(g_m)\}$: $p - q \in \langle R \rangle$

There exists an algorithm to calculate $q =:$ NF(p, R)

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Linear Lead-Rewriting System

System of Boolean polynomials
- $G = \{g_1, \ldots, g_m\}$ in variables $x_1, \ldots, x_n$

Suppose, the following holds:
- For each i: lead(g_i) is a variable x_{gi} ($\rightarrow x_{gi}$ not in $g_i -$ lead(g_i))
- For $i \neq j$: lead$(g_i) \neq$ lead(g_j)

$\Rightarrow$ Then G is a Gröbner basis and
$R := G \cup \{x_i^2 + x_i | i = 1, \ldots, n\}$ is a Gröbner basis

For each Boolean polynomial p there exists exactly one Boolean polynomial q in $\{x_1, .., x_n\} \backslash \{\text{lead}(g_1), ..., \text{lead}(g_m)\}$: $p - q \in \langle R \rangle$

There exists an algorithm to calculate $q =: \text{NF}(p, R)$

Example: $G = \{x - 1, y - 1\}$, $p = x \cdot y \cdot z \rightarrow \text{NF}(p, R) = z$

Special Situation in Formal Verification

- A digital circuit is described by a linear lead rewriting system
- Each output signal o_j is described by a Boolean functions $f(i_1, ...i_s)$ with respect to its input signals $i_1, \ldots, i_s$

Special Situation in Formal Verification

- A digital circuit is described by a linear lead rewriting system
- Each output signal o_j is described by a Boolean functions $f(i_1, ...i_s)$ with respect to its input signals $i_1, \ldots, i_s$
- Take $e_j := o_j - f(i_1, \ldots, i_s)$ as equation
- $R := \{e_j | o_j \text{ output signal}\} \cup \{x_i^2 + x_i | i = 1, \ldots, n\}$

Special Situation in Formal Verification

- A digital circuit is described by a linear lead rewriting system
- Each output signal o_j is described by a Boolean functions $f(i_1, ...i_s)$ with respect to its input signals $i_1, \ldots, i_s$
- Take $e_j := o_j - f(i_1, \ldots, i_s)$ as equation
- $R := \{e_j | o_j \text{ output signal}\} \cup \{x_i^2 + x_i | i = 1, \ldots, n\}$
- For the system described by this equations, we would like to check a property p, which is given as Boolean expression
- Check via normal form computation: $\mathsf{NF}(p, R) = 0$?

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

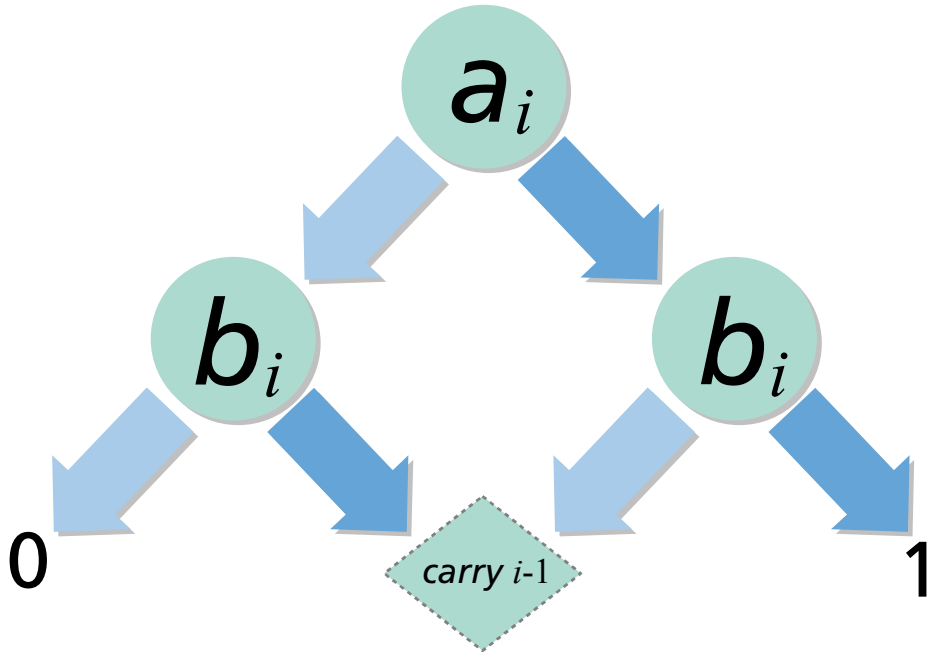
$c(i) + carry_i$
$s(i) + a(i) + b(i) + carry_i$

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

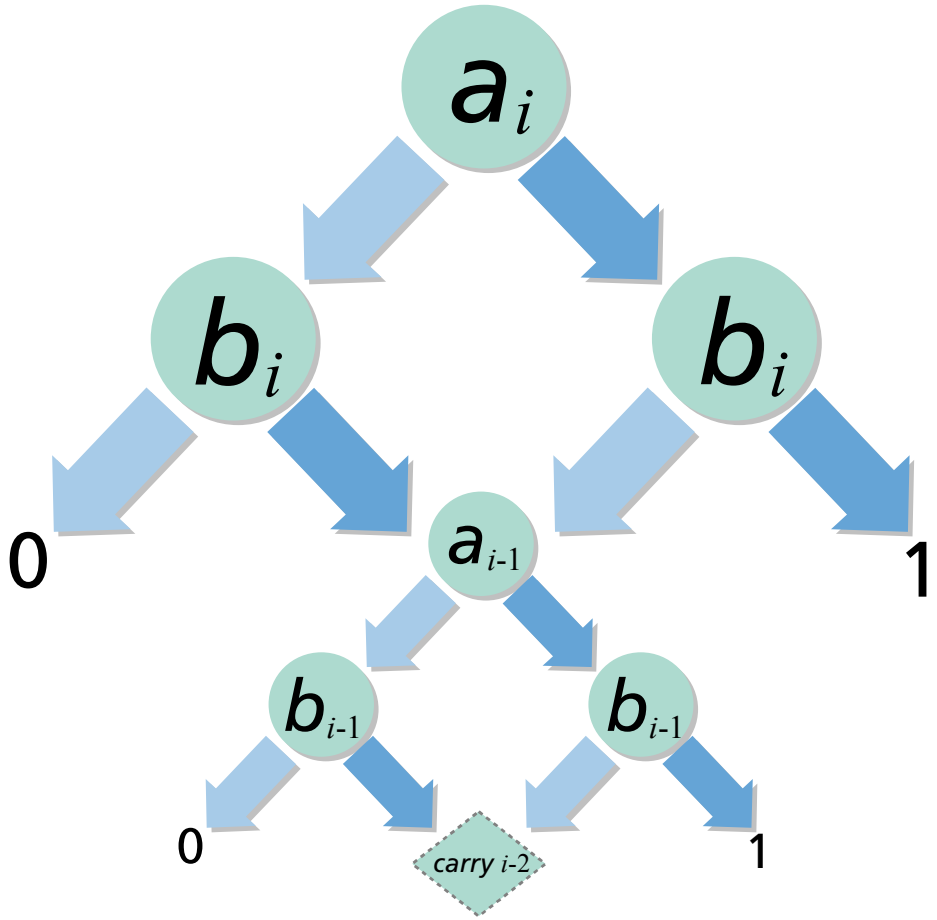
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

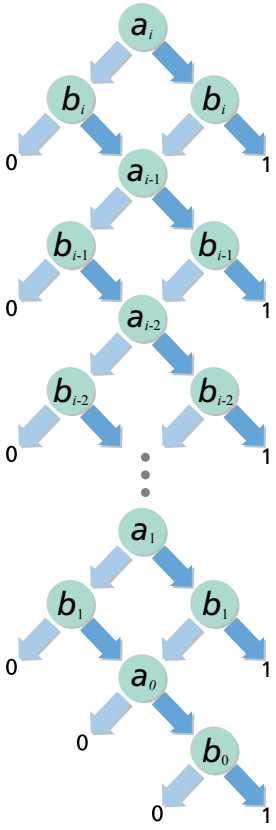
Carry-Bit Graph



Carry-Bit Graph



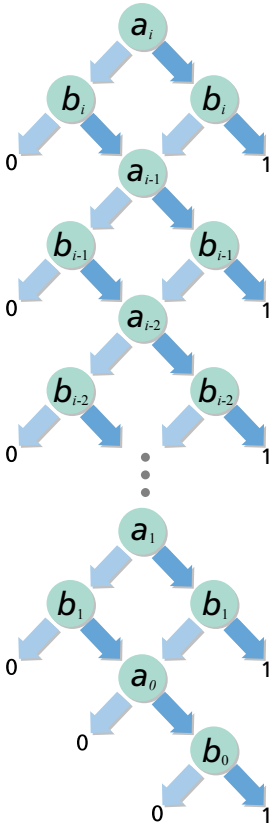
Carry-Bit Graph



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

Benchmarks: Pigeon Hole Problems

	Vars./Eqs.		PolyBoRi		MiniSat	
hole6	42	133	0.13 s	51.64 MB	0.01 s	1.96 MB
hole7	56	204	0.46 s	51.89 MB	0.06 s	1.96 MB
hole8	72	297	1.88 s	56.59 MB	0.30 s	2.08 MB
hole9	90	415	8.01 s	84.04 MB	2.31 s	2.35 MB
hole10	110	561	44.40 s	97.68 MB	25.20 s	3.24 MB
hole11	132	738	643.14 s	130.83 MB	782.65 s	7.19 MB
hole12	156	949	10264.92 s	338.66 MB	22920.20 s	17.13 MB

Remark

Good performance mainly due to fast Boolean multiplication

Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik

Mathematisches Forschungsinstitut Oberwolfach

Benchmarks: Lex. Normalform Computations without Gröbner basis

Interpolation vs. Gröbner approaches

# points	smallest lex. time/1 s	smallest lex. length	interpolate simple time/1 s	interpolate simple length	# basis
100	0.01	42	0.00	12771	287
500	0.06	249	$0.01 \approx 2.9 \cdot 10^{10}$		1943
1000	0.29	508	$0.01 \approx 8.1 \cdot 10^{13}$		3393
5000	5.53	2552	$1.47 \approx 4.5 \cdot 10^{23}$		10319
10000	19.78	5020	$37.18 \approx 1.6 \cdot 10^{26}$		17868
50000	250.95	25012			82929
100000	897.85	50093			162024
200000	3488.61	99868			296697
500000	20336.02	249675			636542

Remark

Normalform w. r. t. Boolean functions, given by points in $\mathbb{Z}_2^n$

Gröbner basis sizes calculated without explicit basis computation

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

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)