Normal Forms and ...

interpolation in Boolean Rings

Overview

- Interpolation
- Relation between interpolation and normal forms
- ZDD Introduction
- Algorithms
- Practical Results

Interpolation Problem

- O, Z disjoint sets in $\mathbb{Z}_2^n$
- want to compute polynomial f:
  - f(o)=1 for o in O
  - f(z)=0 for z in Z
- define V=O ∪ Z

classical univariate interpolation

- d+1 interpolation points
- solve linear equations system to calculate polynomial over degree d
- terms in $1, x, \ldots, x^d$

multivariate interpolation principles

- n interpolation points
- First step: calculate a set T of possible terms consisting of (at least) n elements
- „combine these terms“

Our approach

- T=standard monomials of I(V)
- do not calculate T in explicitely
- interpolation polynomial automatically a combination of the terms in T

Define ordering on polynomials (not only terms) in BOOLEAN Case

- Extend term ordering to ordering on Boolean polynomials by doing string like comparison with terms as literals
- Example: lex x>y>z
  - x + y + z is bigger than
  - x + z + 1
- **redNF(f,G)=min{g|g in f+<G>}**

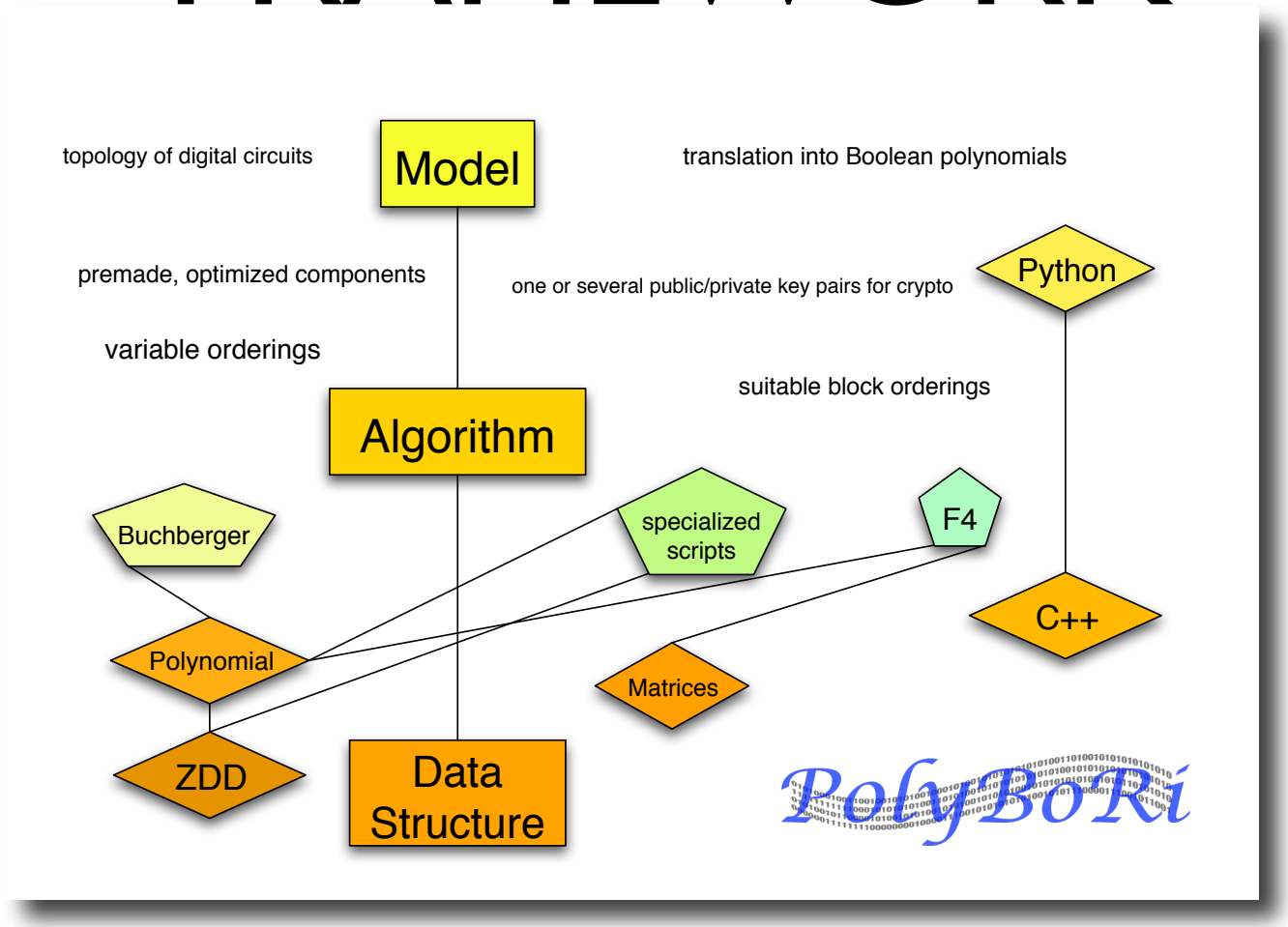
Normal FORM against an IDEAL given by a Boolean Variety I(V)

- g=redNF(f,I(V)) <=>
  - g=**min{g|g in f+I(V)}**
    =min{g|g(x)=f(x) for all x in V}
- So the normal form is a minimal interpolation polynomial

Basic Project 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)

PolyBoRI as FRAMEWORK



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

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

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

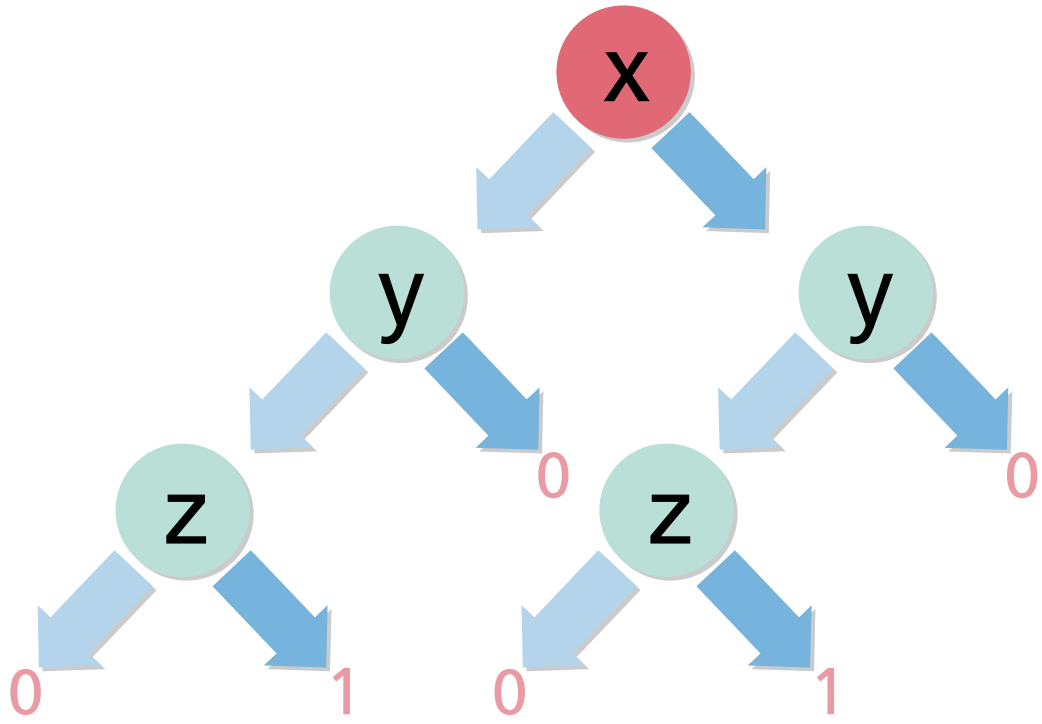
with $S_p = \{\{x_{i_1}, \cdots, x_{i_{n_1}}\}, \cdots, \{x_{i_m}, \cdots, x_{i_{n_m}}\}\}$

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

Zero-suppresed binary decision diagrams

Binary Decision Diagram

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

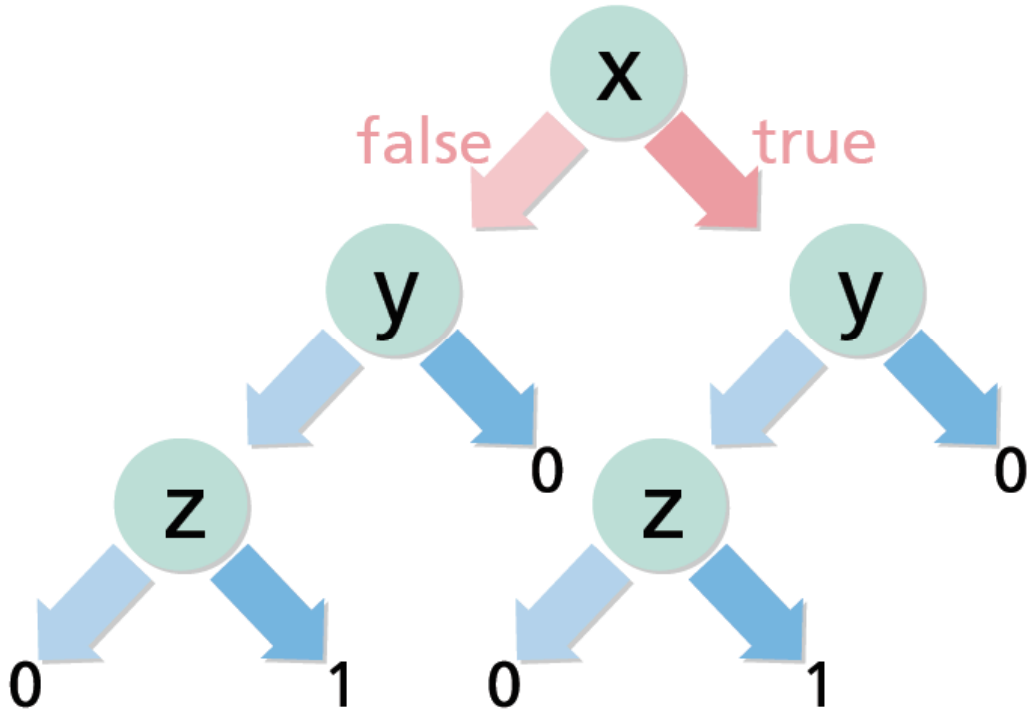


Zero-suppresed binary decision diagrams

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



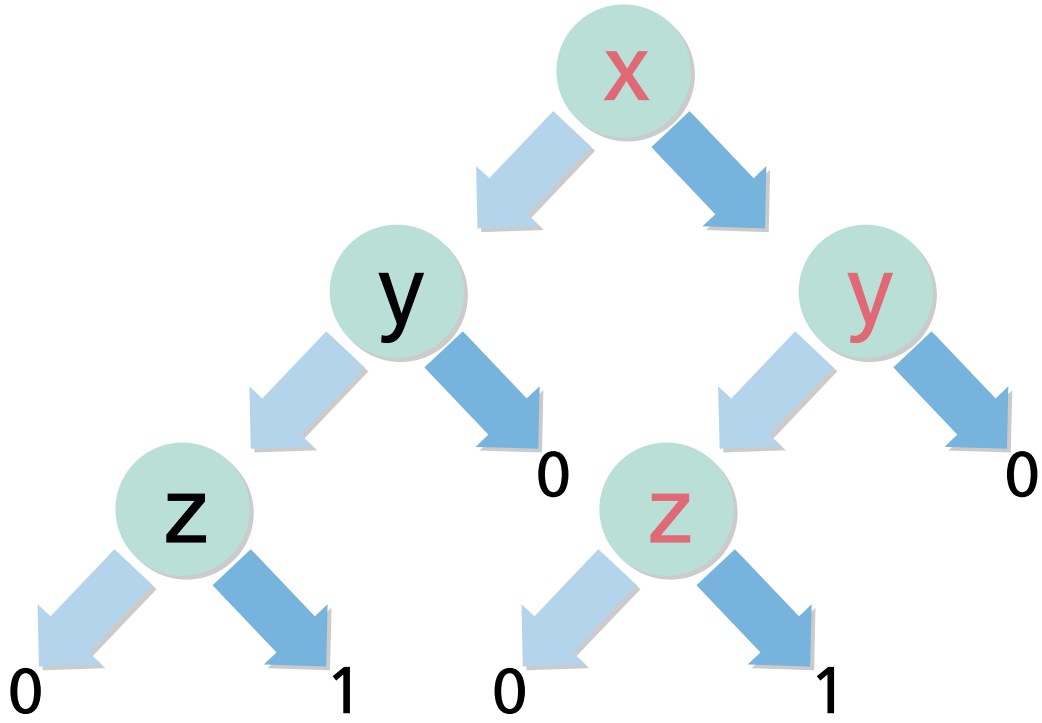
Zero-suppresed binary decision diagrams

Binary Decision Diagram

Rooted, directed, acyclic graph, terminal nodes {0, 1}, decision nodes (≙ Boolean variables)

Two decending edges per node (high/low or then/else)
≙ assign variable to true/false

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



Zero-suppresed binary decision diagrams

Binary Decision Diagram

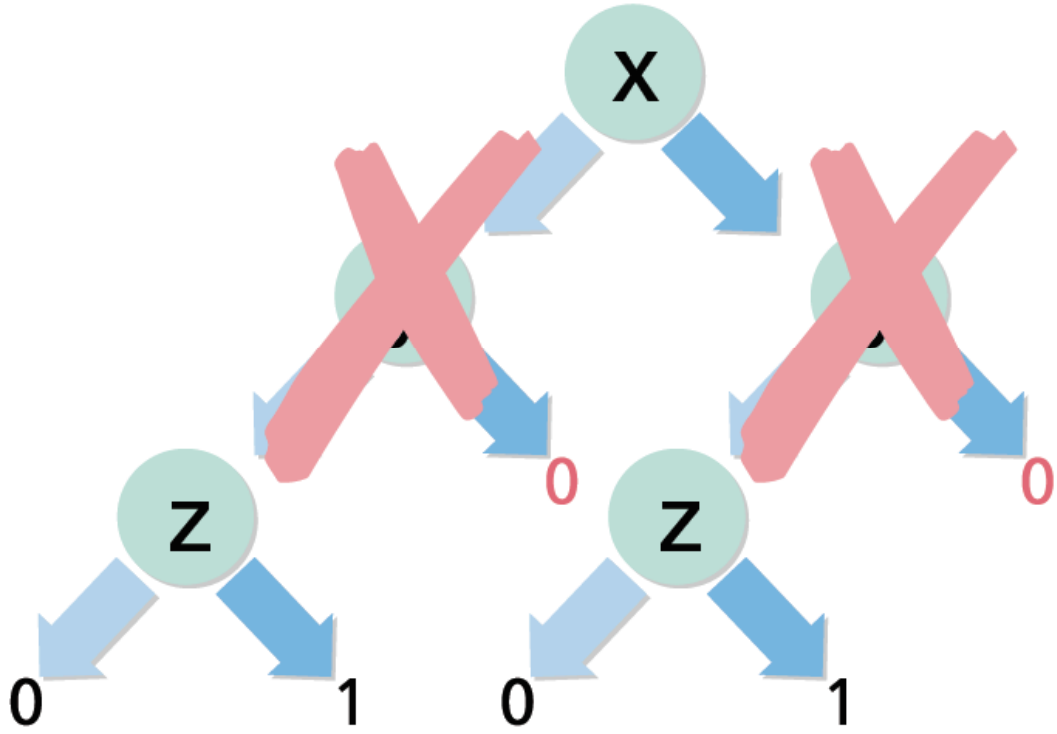
Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

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-suppresed binary decision diagrams

Binary Decision Diagram

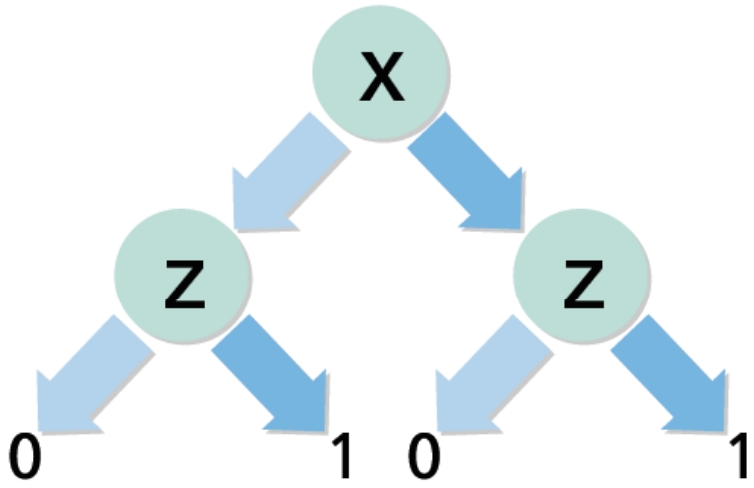
Rooted, directed, acyclic graph, terminal nodes $\{0, 1\}$, decision nodes ($\widehat{=}$ Boolean variables)

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 $\rightarrow 0$



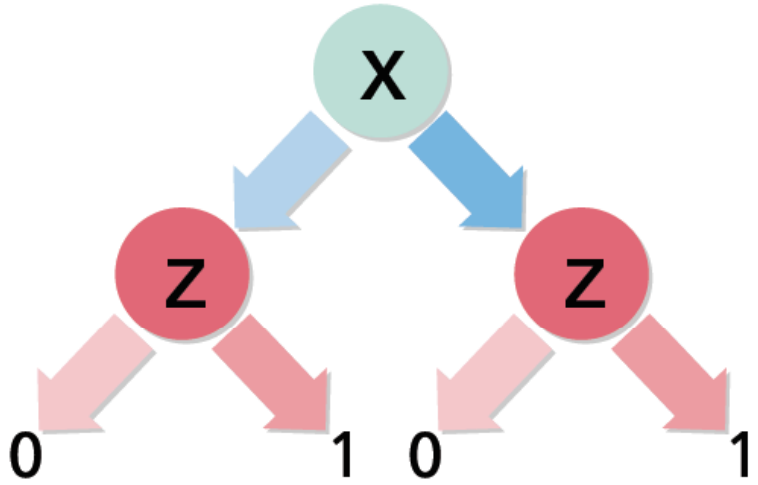
Zero -suppresed binary decision diagrams

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 $\Longleftrightarrow$ then $\rightarrow$ 0

Equal subgraphs merged

Zero -suppresed binary decision diagrams

Binary Decision Diagram

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

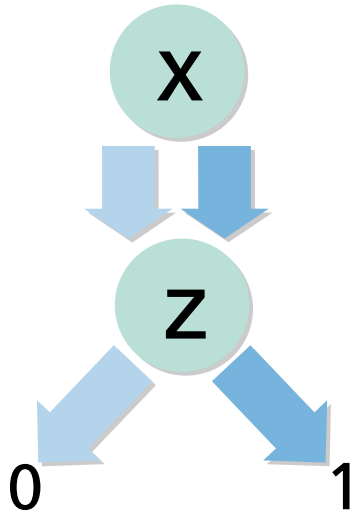
Two decending edges per node (high/low or then/else)
$\widehat{=}$ assign variable to true/false

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

Zero-suppressed BDD (ZDD)

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

Equal subgraphs merged



as Polynomial: x*z+z

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

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)

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

Example: Encoding of vectors in $\mathbb{Z}_2^n$

- Let v=(1,0,0,1,0), w=(0,1,0,0,0)
- Then we can identify
  - v with $\{x_0,x_3\}$ as the 0-th and the third entry are nonzero
  - w with $\{x_1\}$
  - {v,w} with$\{\{x_0,x_3\},\{1\}\}$
- Hence we can identify {v,w} with a ZDD

ZDD operations on sets of vectors

- X={v,w}=ZDD({{x_0,x_3},{x_1}})={(1,0,0,1,0), (0,1,0,0,0)}
- Then we can decompose X into cofactors
  - subset1(X,x_0)={{x_3}} (← {x_0,x_3})
  - subset0(X,x_0)={{x_1}}
- The cofactors can again be interpreted as vectors in a subvector-space of smaller dimension
- subset1(X,x_0)=ZDD({{x_3}})=(0,0,0,1,0)

partial function definitions

- A partial boolean function f is defined by two disjoint sets O, Z
  - f(o)=1 for o in O
  - f(z)=0 for z in Z
- Write f= b_Z^O
- for partial boolean functions f, g
  we define their difference f+g on their common definition space

Algorithm 4 interpolate_simple(b_Z^O)

Require: b_Z^O a partial function definition.
Ensure: interpolate_simple(b_Z^O) $= p$ with $f_p = b_Z^O$ on $Z \cup O$

if $Z = \emptyset$ **then**
  return 1
if $O = \emptyset$ **then**
  return 0
$i = \min(\text{top}(O), \text{top}(Z))$
$Z_1 = \text{subset1}(Z, x_i)$
$Z_0 = \text{subset0}(Z, x_i)$
$O_1 = \text{subset1}(O, x_i)$
$O_0 = \text{subset0}(O, x_i)$
$h_e = \text{interpolate\_simple}(b_{Z_0}^{O_0})$
$h_t = \text{interpolate\_simple}(b_{Z_1}^{O_1}) + h_e$
return $x_i \cdot h_t + h_e$

why this algorithm returns an interpolation

	O_0,Z_0	$x*O_1,x*Z_1$
h_e	affects	affects
x_i*h_t	zero	affects

calculate h_e first -> correct on O_0/Z_0.
adjust on O_1/Z_1 by x_i*h_t

This gives no minimal interpolations polynomial

Optimization principle

- we work with recursion and very easy terminal cases
- it is important to pass the full sets of interpolation candidates to the recursive call for h_t
- this is equivalent to passing as few prescribed points as possible

Observation for lex. ordering

- $f=x_1 * p_f+q_f$ (composition in cofactors of x_1)
- $g=x_1 * p_g+q_g$
- If $p_f>p_g$, then $f>g$
- f is minimal iff
  - p_f is minimal (*)
  - and q_f is minimal under all choices of f satisfying *

Calculate minimal interpolation polynomial

	O_0, Z_0	$x*O_1, x*Z_1$
h_e	affects	affects
x_i*h_t	zero	affects

- $f=x_1*h_t+h_e$
- Calculate h_t or h_e first?
- h_t is more important, so calculate h_t first and adjust value in less important h_e
- How can we make sure, that h_t is the then branch of an interpolation polynomial (without knowing h_e)?

Solution

	O_0, Z_0	$x*O_1, x*Z_1$
h_e	affects	affects
x_i*h_t	zero	affects

- $f=x_1*h_t+h_e$
- $C:=(Z_1 \cup O_1) \cap (Z_0 \cup O_0)$ set of conflicts
- Behavior of h_e on C is known
  $\Rightarrow$ h_t is also uniquely determined on C
- So fixing h_t on C gives only necessary conditions (holds for all interpolation polynomials) -> optimization principle

Algorithm 5 interpolate_smallest_lex(b_Z^O)

Require: b_Z^O a partial function definition.

Ensure: interpolate_smallest_lex(b_Z^O), smallest Boolean polynomial p with respect to lex. with $f_p = b_Z^O$ on $Z \cup O$

if $Z = \emptyset$ **then**

return 1

if $O = \emptyset$ **then**

return 0

$i = \min(\text{top}(O), \text{top}(Z))$

$Z_1 = \text{subset1}(Z, x_i)$

$Z_0 = \text{subset0}(Z, x_i)$

$O_1 = \text{subset1}(O, x_i)$

$O_0 = \text{subset0}(O, x_i)$

$C = (Z_1 \cup O_1) \cap (Z_0 \cup O_0)$ /* C forms the set of conflict points between 0 and 1-subsets */

$f = b_{Z_1}^{O_1}$

$g = b_{Z_0}^{O_0}$

$h_t = \text{interpolate\_smallest\_lex}(f+g)$ /* f+g is only defined on C */

$F = \text{ones}(h_t, ((Z_1 \cup O_1) \backslash C))$

/* non-conflict subset1 terms affected by else branch */

$w = b_{((Z_1 \backslash C) \oplus F) \cup Z_0}^{((O_1 \backslash C) \oplus F) \cup O_0}$

$h_e = \text{interpolate\_smallest\_lex}(w)$

return $x_i \cdot h_t + h_e$

The actual normal form algorithm

Algorithm 6 Reduced lexicographical normal form against a variety: nf_by_interpolate(f, P)

Require: Boolean polynomial f, P set of points in $\mathbb{Z}_2^n$.
Ensure: nf_by_interpolate$(f, P) = \mathrm{NF}(f, \mathrm{I}(P))$
$Z = \mathrm{zeros}(f, P)$
return interpolate_smallest_lex$(b_Z^{P \setminus Z})$

interpolation NF

number of points	time(interpolation)	len(interpolation)	#basis
100	0.01	52	287
500	0.08	253	1943
1000	0.33	485	3393
5000	5.52	2509	10319
10000	18.99	4992	17868
50000	250.95	25012	82929
100000	897.85	50093	162024
500000	2.033.602	249675	636542

Gröbner basis size can be determined without explicit calculation

random data in 100 variables, estimated 100000000000 terms in Gröbner basis of biggest example