Efficiently representing the integer factorization problem using binary decision diagrams

A reduction of FACT to BDD SAT

David Skidmore

Department of Mathematics and Statistics
Utah State University

27 April 2017

Boolean functions

for this presentation

A *boolean function* is a $\{0, 1\}$-valued function in a finite number of $\{0, 1\}$-valued (boolean) variables.

Example

The unary operation $\neg$ (negation, boolean NOT) is defined by

x	$\neg x$
0	1
1	0

It is common to use $\overline{x}$ to denote $\neg x$.

Example

The binary operation $\wedge$ (conjunction, boolean AND, multiplication) is defined by

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

It is common to use xy to denote $x \wedge y$.

Example

The binary operation $\vee$ (disjunction, boolean OR) is defined by

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Example

The binary operation $\oplus$ (exclusive disjunction, boolean XOR, modulo-2 additon) is defined by

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

The integer factorization problem (FACT)

Given a positive integer $a > 1$, find positive integers $x, y > 1$ such that

$$xy = a.$$

If no such x and y exist then a is *prime*, otherwise a is *composite*, x and y are *factors* of a, and xy is a *factorization* of a.

Representing FACT with boolean functions

Fix a positive integer n. For each nonnegative integer m there is a boolean function $f_m : \{0,1\}^{2n} \to \{0,1\}$ such that $f_m(x_0, x_1, \ldots, x_{n-1}, y_0, y_1, \ldots, y_{n-1})$ (represented by $f_m(\vec{x}, \vec{y})$) gives the the coefficient of 2^m in the binary expansion of the product

$$(x_0 + 2x_1 + \cdots + 2^{n-1}x_{n-1})(y_0 + 2y_1 + \cdots + 2^{n-1}y_{n-1}).$$

Let $a > 1$ be a positive integer with binary expansion $a_0 + 2a_1 + \cdots + 2^{n-1}a_{n-1}$. Every factorization of a corresponds to a solution of

$$F_a(\vec{x}, \vec{y}) = 1$$

where

$$F_a(\vec{x}, \vec{y}) = \prod_{m=0}^{2n-1} [1 \oplus a_m \oplus f_m(\vec{x}, \vec{y})],$$

and if $m \geq n$ then let $a_m = 0$.

$F_a(\vec{x}, \vec{y}) = 1$ is equivalent to the system S_a,

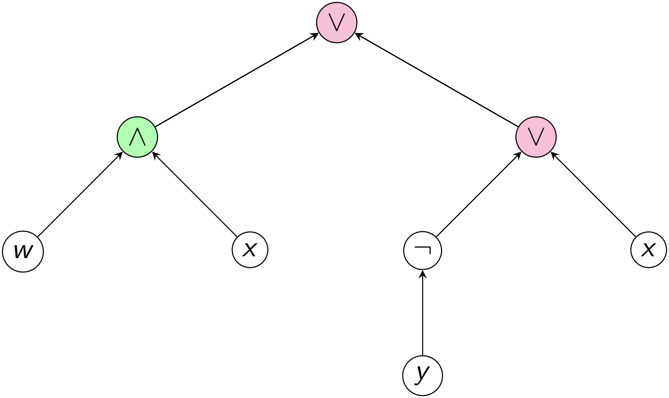
$$\begin{aligned} a_0 \oplus f_0(\vec{x}, \vec{y}) &= 0 \\ a_1 \oplus f_1(\vec{x}, \vec{y}) &= 0 \\ &\vdots \\ a_{n-1} \oplus f_{n-1}(\vec{x}, \vec{y}) &= 0 \\ f_n(\vec{x}, \vec{y}) &= 0 \\ &\vdots \\ f_{2n-1}(\vec{x}, \vec{y}) &= 0 \end{aligned}$$

Boolean formulae

A *boolean formula* is a labeled directed rooted tree representing a mathematical term built from some collection of constant symbols $\{0, 1\}$, and variable and operator symbols corresponding to boolean variables and functions.

Example

Formula: $((w \wedge x) \vee ((\neg y) \vee x))$



Boolean functions from formulae

- A boolean formula with an order on its variables defines a boolean function via substitution.

Boolean functions from formulae

- A boolean formula with an order on its variables defines a boolean function via substitution.
- Two boolean formula with the same variables are equivalent if they represent the same boolean function.

Example

The ternary operator $(\cdot \rightarrow \cdot, \cdot) : \{0,1\}^3 \rightarrow \{0,1\}$ (if-then-else) is defined by the boolean formula in variables $\{x, y, z\}$ with order $x < y < z$,

$$(x \rightarrow y, z) = (\bar{x} \vee y) \wedge (x \vee z)$$

The boolean satisfiability problem (SAT)

Given a boolean formula $\phi(x_1, \ldots, x_n)$, find a solution to

$$\phi(x_1, \ldots, x_n) = 1$$

or prove that no solution exists.

Conjunctive normal form (CNF)

A *literal* is a boolean variable or its negation. A *clause* is a constant or a disjunction of literals. A boolean formula is in *conjunctive normal form* (CNF) if and only if it is a constant or a conjunction of clauses.

Example

In variables $\{x, y, z\}$,

$$(\bar{y} \wedge ((x \vee y) \vee \bar{z})) \wedge (\bar{x} \vee y)$$

is in CNF but

$$((x \wedge \bar{y}) \vee (\bar{z} \wedge \bar{y})) \wedge (\bar{x} \vee y)$$

is not.

Disjunctive normal form (DNF)

A *conjunctive clause* is a constant or a conjunction of literals. A boolean formula is in *disjunctive normal form* (DNF) if and only if it is a constant or a disjunction of conjunctive clauses.

Example

In variables $\{x, y, z\}$,

$$(\bar{y} \vee ((x \wedge y) \wedge \bar{z})) \vee (\bar{x} \wedge y)$$

is in DNF but

$$((x \vee \bar{y}) \wedge (\bar{z} \vee \bar{y})) \vee (\bar{x} \wedge y)$$

is not.

Algebraic normal form (ANF)

A *monomial* is a constant or a conjunction of variables. A boolean formula is in *algebraic normal form* (ANF) if and only if it is a constant or an exclusive disjunction of monomials.

Example

In variables $\{x, y, z\}$,

$$(y \oplus ((xy)z)) \oplus (xy)$$

is in ANF but

$$y(1 \oplus (x(z \oplus 1)))$$

is not.

If-then-else normal form (ITE)

The collection of boolean formulae in *if-then-else* normal form (ITE) in the variables X is the smallest set ITE_X which satisfies,

1. $0, 1 \in ITE_X$.
2. If $x \in X$ and $y, z \in ITE_X$ then $(x \to y, z) \in ITE_X$.

Example

In variables $\{x, y, z\}$,

$$(x \rightarrow (y \rightarrow 0, 1), (z \rightarrow 1, 0))$$

is in ITE but

$$(x \rightarrow \bar{y}, z)$$

is not.

FACT to CNF SAT

- Several bachelor's theses have studied a variety of reductions of FACT to CNF SAT and the performance of different CNF SAT solvers on the resulting reductions [1] [2] [3].
- All such studies proceeded by applying the Tseytin (Tseitin) transformation to different binary multiplier circuits in order to obtain the various CNF SAT instances.
- In all cases, the data indicated an average case exponential time required to factor.

FACT to ANF/DNF SAT

- In 2013 Samuel Lomonaco studied reductions of FACT to ANF and DNF SAT [4]. In his master's thesis S. Bagde further studied and expanded on a FACT to DNF SAT reduction algorithm created by Lomonaco [5].
- In the ANF case, an ad hoc method was used to find a solution to the resulting reduction. The results were poor (exponential time factoring).
- The methods used in the studies to produce the respective DNF SAT instances were found to take exponential time.

Binary decision diagrams (BDD)

- A *binary decision diagram* (BDD) is a labeled rooted directed acyclic graph corresponding to an equivalence class of boolean formulae.

Binary decision diagrams (BDD)

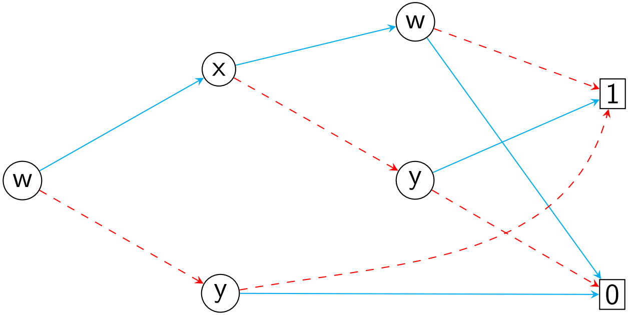
- A *binary decision diagram* (BDD) is a labeled rooted directed acyclic graph corresponding to an equivalence class of boolean formulae.
- Every node in a BDD is labeled by a variable or a constant. Nodes labeled by variables are called *nonterminal* and have outdegree one or two. Nodes labeled by constants are called *terminal* and have outdegree zero.

Binary decision diagrams (BDD)

- A *binary decision diagram* (BDD) is a labeled rooted directed acyclic graph corresponding to an equivalence class of boolean formulae.
- Every node in a BDD is labeled by a variable or a constant. Nodes labeled by variables are called *nonterminal* and have outdegree one or two. Nodes labeled by constants are called *terminal* and have outdegree zero.
- Every edge in a BDD is one of two types, 0 (drawn dashed) or 1 (drawn solid). Two edges leaving the same node must have different types.

Example

Formula: $(w \rightarrow (x \rightarrow (w \rightarrow 0, 1), (y \rightarrow 1, 0)), (y \rightarrow 0, 1))$

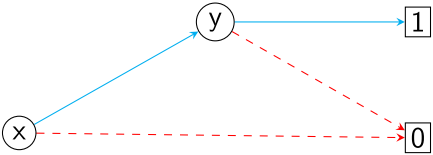


BDD size

The *size* of a BDD is its number of vertices.

Example

The following BDD has size 4:



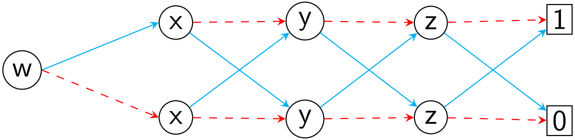
OBDD

A BDD is *ordered* (OBDD) if all paths from its root to a terminal node respect a given linear order on its variable labels.

Example

Function: $f(w, x, y, z) = w \oplus x \oplus y \oplus z$

Order: $w < x < y < z$



In an OBDD,

- Every path from the root to a 1-labeled terminal node corresponds to an assignment for which the corresponding boolean function evaluates to 1.
- Every path from the root to a 0-labeled terminal node corresponds to an assignment for which the corresponding boolean function evaluates to 0.

Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

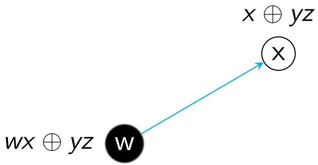
Order: $w < x < y < z$

$wx \oplus yz$ 

Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

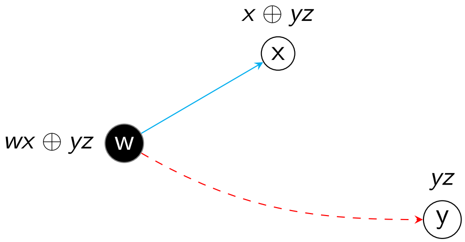
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

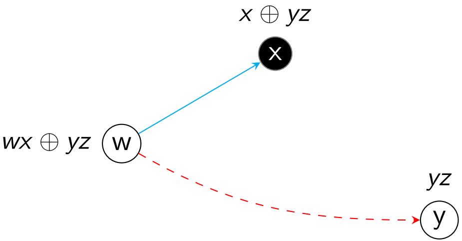
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

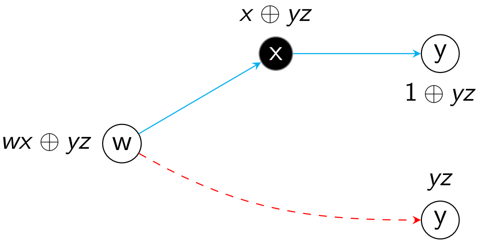
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

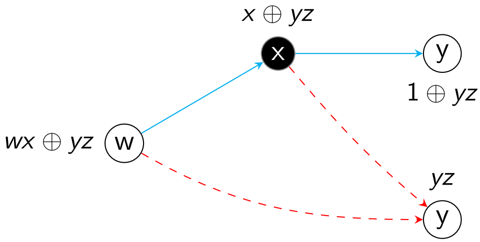
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

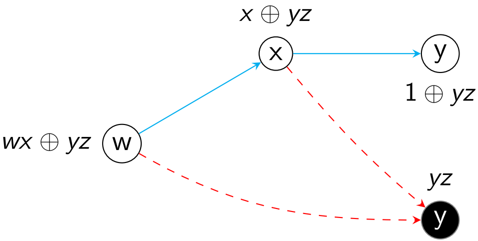
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

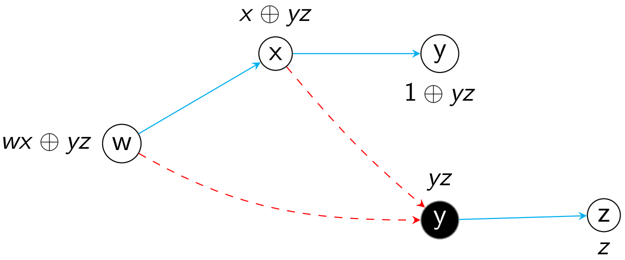
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

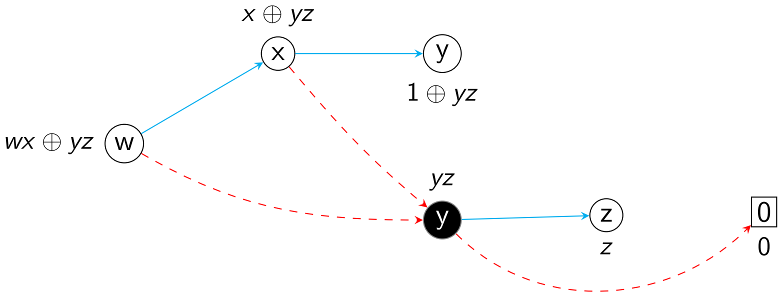
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

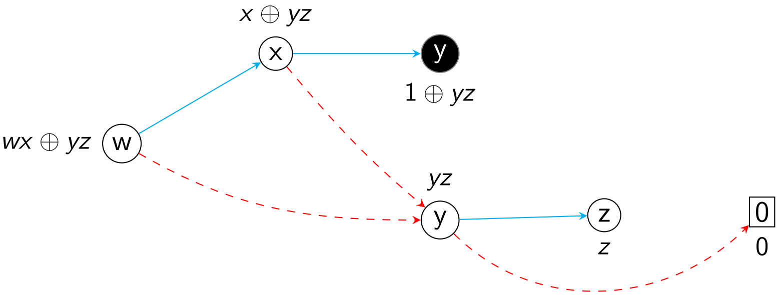
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

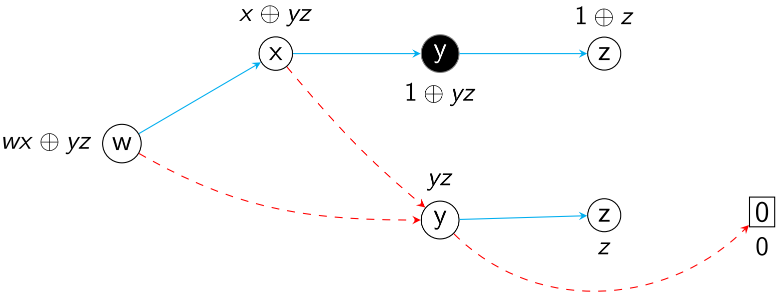
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

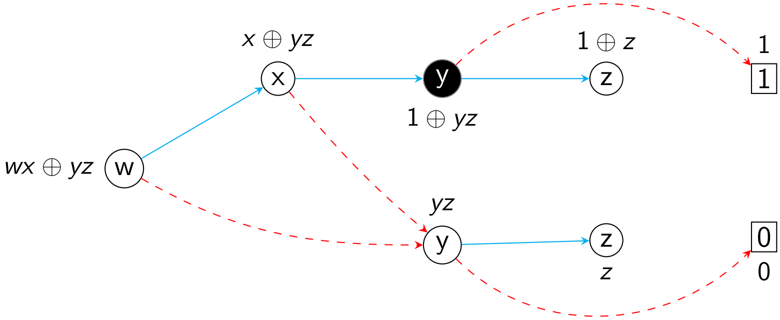
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

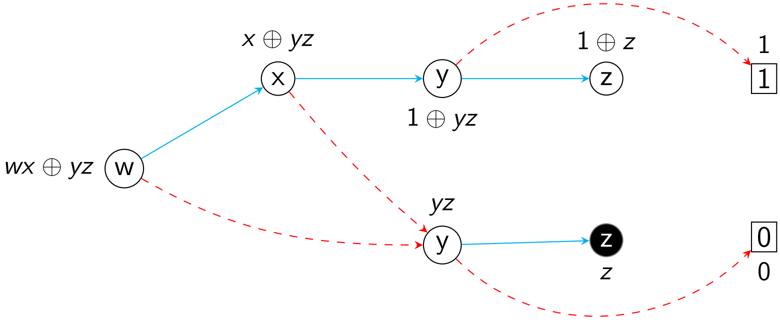
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

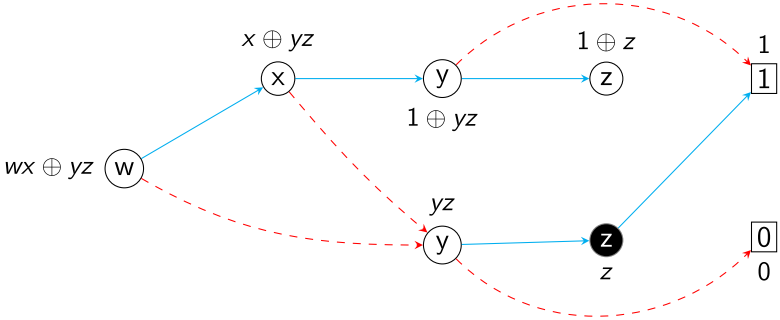
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

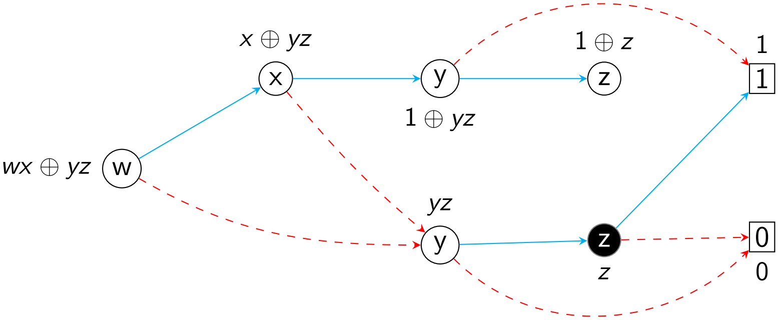
Order: $w < x < y < z$



Example (Construction)

Function: $f(w, x, y, z) = wx \oplus yz$

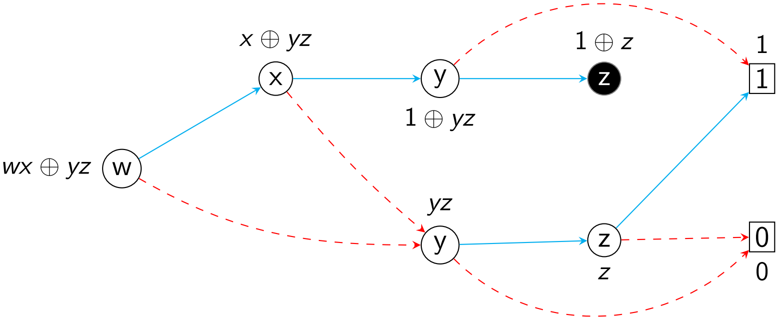
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

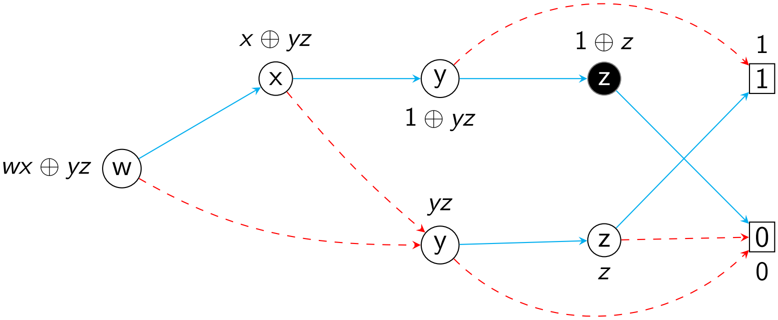
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

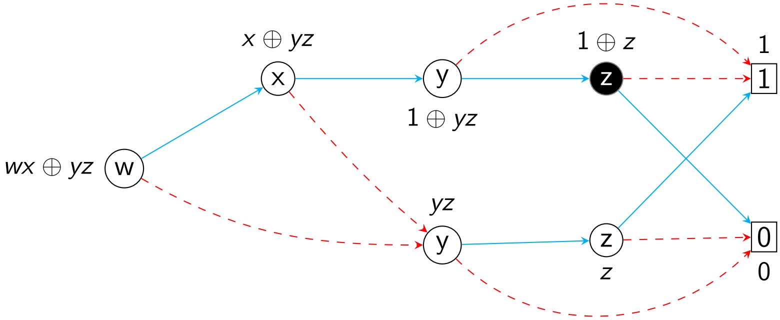
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

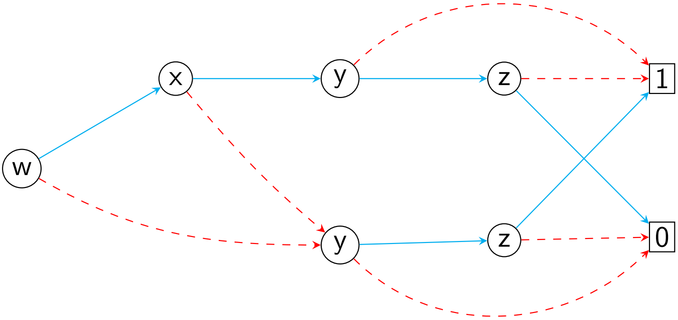
Order: $w < x < y < z$



Example (Construction)

Function: $f(w,x,y,z) = wx \oplus yz$

Order: $w < x < y < z$



FACT to BDD SAT

- Each of the f_m making up F_a can be represented by an OBDD.
- In 2005 Philipp Woelfel showed that regardless of the linear order used, f_{n-1} will have size greater than $2^{\lfloor n/2 \rfloor}/61 - 4$ [6].
- Using F_a as defined previously results in an exponential size representation.
- Conclusion: using F_a as previously defined is infeasible.

How to proceed?

Find an equivalent function (system) to F_a (S_a) and corresponding replacements for each $1 \oplus a_m \oplus f_m$ with smaller OBDD representations.

How to proceed?

Find an equivalent function (system) to F_a (S_a) and corresponding replacements for each $1 \oplus a_m \oplus f_m$ with smaller OBDD representations. ✓

Replacement for each $1 \oplus a_m \oplus f_m$, g_m, has an OBDD size of less than $6(2n)^3$. This results in a replacement for F_a, G_a, with a BDD representation of polynomial size $O(n^4)$.

Problem

The linear order used for each g_m is not the same.

To extract a factorization of a from the BDD, we must find a path from the root to the 1-terminal node such that in the path

1. We never leave a node with a label t along a 0 edge and then later leave another node with the same label t along a 1 edge.
2. We never leave a node with a label t along a 1 edge and then later leave another node with the same label t along a 0 edge.

Open questions and directions for future work

- Compare other normal form representations besides BDDs for G_a to F_a.
- Compare the performance of some CNF SAT solvers on CNF SAT instances obtained from G_a to those obtained from F_a.
- Check the performance of some BDD-based SAT solving algorithms on G_a.

References I

- J. Asketorp, "Attacking rsa moduli with sat solvers," independent thesis basic level, KTH, School of Computer Science and Communication, Stockholm, Sweden, 2014. http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A769846&dswid=9594.
- J. Eriksson and J. Hoglund, "A comparison of reductions from fact to cnf-sat," independent thesis basic level, KTH, School of Computer Science and Communication, Stockholm, Sweden, 2014. http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A769762&dswid=3154.

References II

E. Forsblom and D. Lunden, "Factoring integers with parallel sat solvers," degree project, KTH, School of Computer Science and Communication, Stockholm, Sweden, 2015. http://kth.diva-portal.org/smash/get/diva2:811047/FULLTEXT01.pdf.

S. J. Lomonaco, "Symbolic arithmetic and integer factorization," *ArXiv e-prints*, apr 2013. https://arxiv.org/abs/1304.1944v1.

References III

S. Bagde, *Implementation of the boolean factoring algorithm*. University of Maryland, Baltimore County, 2013. http://contentdm.ad.umbc.edu/cdm/ref/collection/ETD/id/24868.

P. Woelfel, "Bounds on the obdd-size of integer multiplication via universal hashing," *Journal of Computer and System Sciences*, vol. 71, no. 4, pp. 520 – 534, 2005. http://www.sciencedirect.com/science/article/pii/S002200000500067X.