Small Normalized Boolean Circuits for Semi-disjoint Bilinear Forms Require Logarithmic Conjunction-depth

Andrzej Lingas, Lund university

CCC 2018
A set $F$ of quadratic polynomials over a semi-ring, defined on the set of variables $X \cup Y$ is a semi-disjoint bilinear form if the following properties hold.

1. For each polynomial $P$ in $F$ and each variable $z \in X \cup Y$, there is at most one monomial (in the Boolean case, called a prime implicant) of $P$ containing $z$.

2. Each monomial of a polynomial in $F$ consists of exactly one variable in $X$ and one variable in $Y$.

3. The sets of monomials of polynomials in $F$ are pairwise disjoint.
Boolean Vector Convolution

The $n$-dimensional Boolean vector convolution is an important example of semi-disjoint bilinear forms, where $|X| = |Y| = n$ and $|F| = 2n - 1$. It is related to integer multiplication and string matching.

For two $n$-dimensional Boolean vectors $a = (a_0, ..., a_{n-1})$ and $b = (b_0, ..., b_{n-1})$ over the Boolean semi-ring ($\{0, 1\}, \lor, \land$), their convolution over the semi-ring is a Boolean vector $c = (c_0, ..., c_{2n-2})$, where $c_i = \bigvee_{l=\max\{i-n+1, 0\}}^{\min\{i,n-1\}} a_l \land b_{i-l}$ for $i = 0, ..., 2n - 2$. 
The $n \times n$ matrix product is another very important example of semi-disjoint bilinear forms, where $|X| = |Y| = |F| = n^2$.

For a $n \times n$ Boolean matrix $A$ and a $n \times n$ Boolean matrix $B$ over the semi-ring $\langle \{0, 1\}, \lor, \land \rangle$, their matrix product over the semi-ring is a $n \times n$ Boolean matrix $C$ such that $C[i, j] = \bigvee_{m=1}^{n} A[i, m] \land B[m, j]$ for $1 \leq i \leq n$ and $1 \leq j \leq n$. 
A (Boolean) circuit is a finite directed acyclic graph with the following properties:

1. The indegree of each vertex (termed gate) is either 0, 1 or 2.
2. The source vertices (i.e., vertices with indegree 0 called input gates) are labeled by elements in some set of literals, i.e., variables and their negations, and the Boolean constants 0, 1.
3. The vertices of indegree 2 are labeled by elements of the set \{and, or\} and termed and-gates and or-gates, respectively.
4. The vertices of indegree 1 are labeled by negation and termed negation-gates.
A Boolean circuit is *normalized* if it does not use negation-gates. A Boolean circuit is *monotone* if it is normalized and it does not use negated variables.

The *size* of a Boolean circuit is the total number of not input gates. The *depth* of the circuit is the maximum length of a directed path in the circuit. A normalized circuit is of *and-depth* $d$ if the number of and-gates on any directed path in the circuit does no exceed $d$.

A form composed of $k$ functions is computed by a Boolean circuit if the circuit contains $k$ distinguished gates computing the $k$ functions.
Known bounds for convolution and matrix product

• Any monotone circuit for $n$-dimensional Boolean convolution uses $\Omega(n^2 / \log^6 n)$ disjunctions (Grinchuk and Sergeev 2011) and $n^{4/3}$ conjunctions (Blum 1980). On the other hand, one can construct a normalized circuit for the convolution of size $\tilde{O}(n)$ by a reduction to fast integer multiplication (Fisher and Paterson 1974).

• Any monotone circuit for $n \times n$ Boolean matrix product uses $n^2(n - 1)$ disjunctions (Paterson 1975, Mehlhorn-Galil 1976) and $n^3$ conjunctions (Paterson 1975, Pratt 1975, Mehlhorn-Galil 1976). On the other hand, one can construct a normalized circuit for the matrix product of size $\tilde{O}(n^\omega)$, where $\omega$ stands for the exponent of fast matrix multiplication known to not exceed 2.373 (Vassilevska Williams 2012, Le Gall 2014).
A set $T(g)$ of terms associated to a circuit gate $g$

If $g$ is labelled by a variable or a negated variable or a constant $z$
then $T(g) \leftarrow \{z\}$.

If $g$ is an OR gate then $T(g) \leftarrow T(g_1) \cup T(g_2)$, where $g_1$ and $g_2$ are
direct predecessors of $g$.

If $g$ is an AND gate then $T(g) \leftarrow \{t_1t_2|t_1 \in T(g_1) \& t_2 \in T(g_2)\}$,
where $g_1$ and $g_2$ are direct predecessors of $g$. 
An implicant of a set $F$ of Boolean functions is a conjunction of some variables and/or some negated variables of $F$ and/or Boolean constants (monom) such that there is a function belonging to $F$ which is true whenever the conjunction is true. If the conjunction includes the Boolean $0$ or a variable $x$ and its negation $\bar{x}$ then it is a trivial implicant of (any) $F$.

A non-trivial implicant of $F$ that is minimal with respect to included literals is a prime implicant of $F$.

$F = \{x_0y_0, x_0y_1 \lor x_1y_0, x_0y_2 \lor x_1y_1 \lor x_2y_0, x_1y_2 \lor x_2y_1, x_2y_2\}$

The set of prime implicants of $F$ consists of all monoms $xy$, where $x \in \{x_0, x_1, x_2\}$ and $y \in \{y_0, y_1, y_2\}$

For example, $x_1\bar{x}_2y_0, x_0y_1y_2$ are (not prime) implicants of $F$
Single term representation of implicants

The monom represented by a term $t$ is obtained by replacing concatenations in $t$ with conjunctions, respectively.

We shall say that an implicant (in particular, a prime implicant) of a function $f_g$ computed at the gate $g$ is \textit{represented by a single term} in $T(g)$ if there is a term $t \in T(g)$ such that the monom represented by $t$ is equivalent to the implicant.

In monotone circuits, each prime implicant of a function computed at a gate $h$ has to be represented by a single term in $T(h)$. This is not the case in normalized circuits generally. E.g., $xy$ could be represented by $\{xyz, xy\bar{z}\}$. 
The first key lemma

**Lemma 1**  Let $C$ be a normalized Boolean circuit computing a form $F$. For each prime implicant of the function $f_o \in F$ computed at the output gate $o$ of $C$, there is a term in $T(o)$ representing the (whole) prime implicant or a conjunction of the prime implicant with solely negated variables.

**Proof:** idea. Consider a prime implicant of $f_o$. Assign the Boolean 1 to the variables in the prime implicant and the Boolean 0 to all remaining variables in $F$.  

\[\blacksquare\]
Corollary 2  Let $C$ be a normalized Boolean circuit computing a form $F$ with $p$ prime implicants. Suppose that each prime implicant of $F$ is composed of $q$ (not negated) variables and each output term of $C$ contains at most $k$ distinct literals. Let $0 < \beta < 1$. There is a subset of the set of variables of $F$ such that after setting them to the Boolean $0$ there are at least $p\beta^q(1 - \beta)^{k-q}$ prime implicants of $F$ represented by single output terms of the circuit $C'$ resulting from $C$. Note that the circuit $C'$ computes a form $F'$ whose set of prime implicants is a subset of that of $F$. 
Lemma 3 Let $C$ be a normalized Boolean circuit computing a semi-disjoint bilinear form $F$ on the variables $x_0, \ldots, x_{n-1}$ and $y_0, \ldots, y_{n-1}$. Suppose that for each output gate $o$ in $C$, each term in $T(o)$ contains at most $k$ different literals.

Let $h$ be a gate connected by directed paths with some output gates in $C$ such that the function computed at $h$ has prime implicants $z_{q_1}, \ldots, z_{q_{l(h)}}$ which are single (not negated) variables represented by single terms in $T(h)$, and possibly some other prime implicants.

The inequality $l(h) \leq k$ holds or $h$ can be replaced by the Boolean constant 1.
The second key lemma - proof

Case 1: For each output gate \( o \) reachable by a directed path from the gate \( h \), for each \( z \in \{ z_{q_1}, \ldots, z_{q_l(h)} \} \), and each term \( t_1 z t_2 \in T(o) \), the term \( t_1 t_2 \) represents an implicant of the function computed at 0. Then, \( h \) can be replaced by the constant 1 gate.

Case 2: For an output gate \( o \) reachable by a directed path from the gate \( h \), for a \( z \in \{ z_{q_1}, \ldots, z_{q_l(h)} \} \), and a term \( t_1 z t_2 \in T(o) \), the term \( t_1 t_2 \) does not represent an implicant of the function computed at 0. Then the term \( t_1 t_2 \) has to contain for each \( z \in \{ z_{q_1}, \ldots, z_{q_l(h)} \} \) the variable \( z' \) completing \( z \) to a prime implicant \( z z' \) of the function or \( \bar{z} \) so the term \( t_1 z t_2 \) becomes a trivial implicant, totally \( l(h) \) different variables.
Lemma 4 Let $C$ be a normalized Boolean circuit of $d$-bounded conjunction-depth computing a form $F$. Each term, in particular, each output term of $C$ includes at most $2^d$ literals.

Proof: An and-gate can at most double the number of literals in single terms while an or-gate does not increase it. Hence, by induction on the maximum number $d$ of and-gates on a path from an input gate to a gate $g$ in $C$, any term in $T(g)$ includes at most $2^d$ literals.
Theorem 1  Let $C$ be a normalized Boolean circuit of conjunction-depth at most $d$ computing a semi-disjoint bilinear form $F$ with $p$ prime implicants. The circuit $C$ has at least 
\[ \frac{p}{2^d} (1 - \frac{1}{2^d})^{2^d-2} \] and-gates.

Proof sketch

1. Apply Lemma 2 with $\beta = \frac{1}{2^d}$ and $q = 2$ to the circuit $C$, where $k \leq 2^d$ by Corollary 4. The resulting circuit $C'$ computes a form $F'$ with at least 
\[ \frac{p}{2^d} (1 - \frac{1}{2^d})^{2^d-2} \] prime implicants inherited from $F$ and represented by single output terms of $C'$.

2. Prune $C'$ by iteratively eliminating all and-gates that can be replaced by the constant 1 without affecting the form computed by the circuit.
3. For each prime implicant $xy$ of $F'$ represented by a simple term there exists a gate $g$ of $S$ such that $xy$ is an implicant of the function computed at $g$ represented by single terms in $T(g)$ and none of the direct predecessor of $g$ in $C'$ has the aforementioned property. By Lemma 3, each direct predecessor of $g$ cannot have more single variable implicants represented by simple terms than the upper bound $2^d$ on the number of literals in a term. Hence, $g$ can be assigned this way to at most $2^{2d}$ prime implicants of $F'$ represented by simple terms.
Corollaries for vector convolution and matrix product

Corollary 5  Any normalized circuit of $\epsilon \log n$-bounded and-depth that computes the $n$-dimensional Boolean vector convolution has $\Omega(n^{2-4\epsilon})$ and-gates.

Corollary 6  Any normalized circuit of $\epsilon \log n$-bounded and-depth that computes the $n \times n$ Boolean matrix product has $\Omega(n^{3-4\epsilon})$ and-gates.
An upper bound trade-off for vector convolution

**Proposition 1** There is a positive constant $c \leq 1$ such that for any $\epsilon \in (0, \frac{1}{c})$, the $n$-dimensional Boolean vector convolution can be computed by a normalized Boolean circuit of $\epsilon \log n$-bounded conjunction-depth and $O(n^{2-ce} \log^2 n \log \log n)$ size.

**Proof:** By the known fast algorithms for Boolean convolution, for some positive constant $c \leq 1$, an $n^{ce}$-dimensional Boolean vector convolution can be computed by a normalized Boolean circuit of $\epsilon \log n$-bounded depth and $O(n^{ce} \log^2 n \log \log n)$ size. On the other hand, since $ce < 1$, the $n$-dimensional Boolean vector convolution can be easily reduced to $n^{2-2ce}$ $n^{ce}$-dimensional Boolean vector convolutions using just disjunctions. □
An upper bound trade-off for matrix product

**Proposition 2** There is a positive constant \( c \leq 1 \) such that for any \( \epsilon \in (0, \frac{1}{c}) \), the \( n \times n \) Boolean matrix product can be computed by a normalized Boolean circuit of \( \epsilon \log n \)-bounded conjunction-depth and \( O(n^{3-(3-\omega)c\epsilon}) \) size.

**Proof:** By the fast algorithms for matrix multiplication, there is a positive constant \( c \leq 1 \) such that an \( n^{ce} \times n^{ce} \) Boolean matrix product can be computed by a normalized Boolean circuit of \( \epsilon \log n \)-bounded depth and \( O(n^{\omega ce}) \) size. On the other hand, since \( ce < 1 \), the \( n \times n \) Boolean matrix product can be easily reduced to \( n^{3-3ce} \times n^{ce} \times n^{ce} \) Boolean matrix products using just disjunctions. □
Lemma 7 Let the normalized Boolean circuits $C, C'$ and the forms $F, F'$ computed by them be defined as in Lemma 2. Let $F''$ be a form having the following properties: for each $f'' \in F''$ different from a constant there is a distinct $f \in F$ such that the prime implicants of $f''$ are implicants of $f$ and all prime implicants of $f$ represented by single output terms in $C'$ are also prime implicants of $f''$, Suppose that any monotone Boolean circuit computing such form $F''$ has at least $u$ and-gates and at least $w$ or-gates. Then the circuits $C, C'$ have also at least $u$ and-gates and at least $w$ or-gates.

Proof: sketch.

Substitute for each negated variable the Boolean 0. \hfill \Box
A stronger lower-bound trade-off for matrix product

**Lemma 8** Suppose that for each function $h$ in a form $H$ there is a distinct function $f$ in the $n \times n$ Boolean matrix product such that each prime implicant of $h$ is an implicant of $f$. Let $p$ be the total number of prime implicants of $H$ that are also prime implicants of the $n \times n$ Boolean matrix product. Any monotone Boolean circuit computing $H$ has at least $p$ and-gates.

**Theorem 2** Let $C$ be a normalized Boolean circuit computing the $n \times n$ Boolean matrix product. Suppose that each output term of $C$ contains at most $k$ distinct negated variables. The circuit has at least $\frac{n^3}{k^2}(1 - \frac{1}{k})^k$ and-gates. In particular, if $C$ is is of negation-dependent conjunction-depth $d$ then it has at least $\frac{n^3}{2^d}(1 - \frac{1}{2^d})^{2^d}$ and-gates. Finally, if $d = \epsilon \log n$ then $C$ has $\Omega(n^{3-2\epsilon})$ and-gates.