Boolean Circuit Depth (II)

Yijia Chen
Fudan University

A Quick Recap

Definition

The depth $d(C)$ of a circuit C is the length of the longest path from the output node to an input node. The size $L(F)$ of a formula F is the number of its input nodes.

For a function f, the depth complexity $d(f)$ is the minimum depth of a circuit computing f and the size complexity $L(f)$ is the minimum size of a formula computing f.

The measure $d_m(C)$, $L_m(F)$, $d_m(f)$, and $L_m(f)$ are defined similarly for monotone circuits, formulas, and functions respectively.

Definition

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ let

$$X = f^{-1}(1) \quad \text{and} \quad Y = f^{-1}(0).$$

We define

$$R_f = \left\{(x, y, i) \;\middle|\; x \in X,\ y \in Y, \text{ and } i \in \{1, \dots, n\} \text{ with } x_i \neq y_i\right\}.$$

For monotone f we also define

$$M_f = \left\{(x, y, i) \;\middle|\; x \in X,\ y \in Y, \text{ and } i \in \{1, \dots, n\} \text{ with } x_i = 1 \text{ and } y_i = 0\right\}.$$

Theorem

$$d(f) = D(R_f) \quad and \quad L(f) = C^P(R_f).$$

Theorem

$$d_m(f) = D(CM_f) \quad and \quad L_m(f) = C^P(M_f).$$

We prove circuit lower bounds by reductions to lower bounds for communication complexity.

Matching

Given a graph G on n vertices,

$$\textsc{match}(G) = \begin{cases} 1, & \text{if there is a matching of size } \geq n/3 \text{ in } G, \\ 0, & \text{otherwise.} \end{cases}$$

Theorem

$$d_m(\textsc{match}) = \Omega(n).$$

Given a directed graph G on n nodes,

$$\text{STCON}(G) = \begin{cases} 1, & \text{if there is a path in } G \text{ from vertex 1 to vertex } n \\ 0, & \text{otherwise.} \end{cases}$$

Theorem

$$d_m(\text{STCON}) = \Omega(\log^2 n).$$

Set Cover

Let $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ whose deterministic communication complexity $D(g)$ is significantly larger than its nondeterministic communication complexity $N(g)$.

Let $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ whose deterministic communication complexity $D(g)$ is significantly larger than its nondeterministic communication complexity $N(g)$.

Let $R_1, \ldots, R_t$ be a cover (possibly with intersections) of the matrix M_g corresponding to g with monochromatic rectangles.

Let $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ whose deterministic communication complexity $D(g)$ is significantly larger than its nondeterministic communication complexity $N(g)$.

Let $R_1, \ldots, R_t$ be a cover (possibly with intersections) of the matrix M_g corresponding to g with monochromatic rectangles. Thus

$$N(g) \leq t.$$

Let $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ whose deterministic communication complexity $D(g)$ is significantly larger than its nondeterministic communication complexity $N(g)$.

Let $R_1, \ldots, R_t$ be a cover (possibly with intersections) of the matrix M_g corresponding to g with monochromatic rectangles. Thus

$$N(g) \leq t.$$

We define

$$M = \{(x, y, i) \mid x, y \in \{0,1\}^n \text{ and } (x, y) \in R_i\}.$$

M is a total relation,

Let $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ whose deterministic communication complexity $D(g)$ is significantly larger than its nondeterministic communication complexity $N(g)$.

Let $R_1, \ldots, R_t$ be a cover (possibly with intersections) of the matrix M_g corresponding to g with monochromatic rectangles. Thus

$$N(g) \leq t.$$

We define

$$M = \{(x, y, i) \mid x, y \in \{0,1\}^n \text{ and } (x, y) \in R_i\}.$$

M is a total relation, and

$$D(g) \leq D(M).$$

We construct a function $f : \{0,1\}^t \to \{0,1\}$ such that $D(M_f) \geq D(M)$.

We construct a function $f : \{0,1\}^t \to \{0,1\}$ such that $D(M_f) \geq D(M)$.

$$f(z_1, \dots, z_t) = \begin{cases} 1, & \text{if there exists a row } x \text{ of } M_g \text{ such that} \\ & \qquad \text{for all } i \text{ we have } (x \in R_i \implies z_i = 1) \\ 0, & \text{otherwise.} \end{cases}$$

We construct a function $f : \{0,1\}^t \to \{0,1\}$ such that $D(M_f) \geq D(M)$.

$$f(z_1, \ldots, z_t) = \begin{cases} 1, & \text{if there exists a row } x \text{ of } M_g \text{ such that} \\ & \quad\quad \text{for all } i \text{ we have } (x \in R_i \implies z_i = 1) \\ 0, & \text{otherwise.} \end{cases}$$

f is monotone.

Reduction from M to M_f

Reduction from M to M_f

1. Alice, given $x \in \{0,1\}^n$, constructs $x' \in \{0,1\}^t$ by assigning $x'_i = 1$ if the the row x belongs to R_i and 0 otherwise.

Reduction from M to M_f

1. Alice, given $x \in \{0,1\}^n$, constructs $x' \in \{0,1\}^t$ by assigning $x'_i = 1$ if the the row x belongs to R_i and 0 otherwise. So $f(x') = 1$.

Reduction from M to M_f

1. Alice, given $x \in \{0,1\}^n$, constructs $x' \in \{0,1\}^t$ by assigning $x'_i = 1$ if the the row x belongs to R_i and 0 otherwise. So $f(x') = 1$.
2. Bob, given $y \in \{0,1\}^n$, constructs $y' \in \{0,1\}^t$ by assigning $y'_i = 0$ if the column y belongs to R_i and 1 otherwise.

Reduction from M to M_f

1. Alice, given $x \in \{0,1\}^n$, constructs $x' \in \{0,1\}^t$ by assigning $x'_i = 1$ if the the row x belongs to R_i and 0 otherwise. So $f(x') = 1$.

2. Bob, given $y \in \{0,1\}^n$, constructs $y' \in \{0,1\}^t$ by assigning $y'_i = 0$ if the column y belongs to R_i and 1 otherwise. So $f(y') = 0$.

Reduction from M to M_f

1. Alice, given $x \in \{0,1\}^n$, constructs $x' \in \{0,1\}^t$ by assigning $x'_i = 1$ if the the row x belongs to R_i and 0 otherwise. So $f(x') = 1$.
2. Bob, given $y \in \{0,1\}^n$, constructs $y' \in \{0,1\}^t$ by assigning $y'_i = 0$ if the column y belongs to R_i and 1 otherwise. So $f(y') = 0$.
3. Alice and Bob use the protocol for the relation M_f on (x', y') to get an index i with $x'_i = 1$ and $y'_i = 0$. Thus, both x and y intersect R_i, i.e., $(x, y, i) \in M$.

Reduction from M to M_f

1. Alice, given $x \in \{0,1\}^n$, constructs $x' \in \{0,1\}^t$ by assigning $x'_i = 1$ if the the row x belongs to R_i and 0 otherwise. So $f(x') = 1$.
2. Bob, given $y \in \{0,1\}^n$, constructs $y' \in \{0,1\}^t$ by assigning $y'_i = 0$ if the column y belongs to R_i and 1 otherwise. So $f(y') = 0$.
3. Alice and Bob use the protocol for the relation M_f on (x', y') to get an index i with $x'_i = 1$ and $y'_i = 0$. Thus, both x and y intersect R_i, i.e., $(x, y, i) \in M$.

Assume $D(g) = N^2(g)$, then the function f has $t = 2^{N(g)}$ variables and

$$d_m(f) = D(M_f) \geq D(M) \geq D(g) = \log^2 t.$$

Similarly $L(f) = \Omega(t^{\log t})$.

We can write

$$f(z_1, \ldots, z_t) \equiv \exists x \in \{0,1\}^n : \\ \big[(x \in R_1) \implies (z_1 = 1)\big] \wedge \cdots \wedge \big[(x \in R_t) \implies (z_t = 1)\big].$$

We can write

$$f(z_1, \ldots, z_t) \equiv \exists x \in \{0,1\}^n : \\ \big[(x \in R_1) \Longrightarrow (z_1 = 1)\big] \wedge \cdots \wedge \big[(x \in R_t) \Longrightarrow (z_t = 1)\big].$$

If deciding "$x \in R_i$" can be done in time polynomial in t, then f is a function in NP,

We can write

$$f(z_1, \dots, z_t) \equiv \exists x \in \{0,1\}^n : \\ \big[(x \in R_1) \implies (z_1 = 1)\big] \wedge \cdots \wedge \big[(x \in R_t) \implies (z_t = 1)\big].$$

If deciding "$x \in R_i$" can be done in time polynomial in t, then f is a function in NP, and can be rewritten to a 3-CNF formula

$$f(z_1, \dots, z_t) \equiv \exists x_1 \cdots x_p \big(\varphi_1 \wedge \cdots \wedge \varphi_s\big),$$

We can write

$$f(z_1, \ldots, z_t) \equiv \exists x \in \{0,1\}^n : \\ \big[(x \in R_1) \Longrightarrow (z_1 = 1)\big] \wedge \cdots \wedge \big[(x \in R_t) \Longrightarrow (z_t = 1)\big].$$

If deciding "$x \in R_i$" can be done in time polynomial in t, then f is a function in NP, and can be rewritten to a 3-CNF formula

$$f(z_1, \ldots, z_t) \equiv \exists x_1 \cdots x_p \big(\varphi_1 \wedge \cdots \wedge \varphi_s\big),$$

where

1. $x_{n+1}, \ldots, x_p$ are auxiliary variables,

We can write

$$f(z_1, \ldots, z_t) \equiv \exists x \in \{0,1\}^n : \quad \big[(x \in R_1) \Longrightarrow (z_1 = 1)\big] \wedge \cdots \wedge \big[(x \in R_t) \Longrightarrow (z_t = 1)\big].$$

If deciding "$x \in R_i$" can be done in time polynomial in t, then f is a function in NP, and can be rewritten to a 3-CNF formula

$$f(z_1, \ldots, z_t) \equiv \exists x_1 \cdots x_p \big(\varphi_1 \wedge \cdots \wedge \varphi_s\big),$$

where

1. $x_{n+1}, \ldots, x_p$ are auxiliary variables,
2. each φ_i is a disjunction of 3 literals on the variables $x_1, \ldots, x_p$,

We can write

$$f(z_1,\ldots,z_t) \equiv \exists x \in \{0,1\}^n : \\ \big[(x \in R_1) \implies (z_1 = 1)\big] \wedge \cdots \wedge \big[(x \in R_t) \implies (z_t = 1)\big].$$

If deciding "$x \in R_i$" can be done in time polynomial in t, then f is a function in NP, and can be rewritten to a 3-CNF formula

$$f(z_1,\ldots,z_t) \equiv \exists x_1 \cdots x_p \big(\varphi_1 \wedge \cdots \wedge \varphi_s\big),$$

where

1. $x_{n+1},\ldots,x_p$ are auxiliary variables,
2. each φ_i is a disjunction of 3 literals on the variables $x_1,\ldots,x_p$,
3. and both p and s are polynomially bounded in t.

The set-cover problem

The set-cover problem

SET-COVER	
Input:	A collection of m sets over a universe of ℓ elements and a number d.
Problem:	Is there a subcollection of d sets that covers the whole universe?

Reduction to the set-cover problem

Reduction to the set-cover problem

Recall

$$f(z_1, \ldots, z_t) \equiv \exists x_1 \cdots x_p (\varphi_1 \wedge \cdots \wedge \varphi_s).$$

Reduction to the set-cover problem

Recall

$$f(z_1, \ldots, z_t) \equiv \exists x_1 \cdots x_p \big(\varphi_1 \wedge \cdots \wedge \varphi_s\big).$$

1. The universe is of size $s + p$, one element for each φ_i, and one element for each $x_i \vee \bar{x}_i$.

Reduction to the set-cover problem

Recall

$$f(z_1, \dots, z_t) \equiv \exists x_1 \cdots x_p \big(\varphi_1 \wedge \cdots \wedge \varphi_s\big).$$

1. The universe is of size $s + p$, one element for each φ_i, and one element for each $x_i \vee \bar{x}_i$.

2. For every x_i there are two sets $A_{x_i=1}$ and $A_{x_i=0}$. $A_{x_i=1}$ contains all terms in which x_i appears, and $A_{x_i=0}$ contains all terms in which $\bar{x}_i$ appears.

Reduction to the set-cover problem

Recall

$$f(z_1, \ldots, z_t) \equiv \exists x_1 \cdots x_p \big(\varphi_1 \wedge \cdots \wedge \varphi_s\big).$$

1. The universe is of size $s + p$, one element for each φ_i, and one element for each $x_i \vee \bar{x}_i$.
2. For every x_i there are two sets $A_{x_i=1}$ and $A_{x_i=0}$. $A_{x_i=1}$ contains all terms in which x_i appears, and $A_{x_i=0}$ contains all terms in which $\bar{x}_i$ appears.
3. Finally, set $d = p$.

The correctness

The correctness

If f is 1, then there exists an assignment for $x_1, \ldots, x_p$ that satisfies all the terms. Then the corresponding p sets from a cover.

The correctness

If f is 1, then there exists an assignment for $x_1, \ldots, x_p$ that satisfies all the terms. Then the corresponding p sets from a cover.

If there is cover, then for every i at least one of $A_{x_1=1}$ and $A_{x_i=0}$ is in the cover in order to cover the term $x_i \vee \bar{x}_i$.

The correctness

If f is 1, then there exists an assignment for $x_1, \ldots, x_p$ that satisfies all the terms. Then the corresponding p sets from a cover.

If there is cover, then for every i at least one of $A_{x_1=1}$ and $A_{x_i=0}$ is in the cover in order to cover the term $x_i \vee \bar{x}_i$. Since the cover is of size p, exactly one of $A_{x_1=1}$ and $A_{x_i=0}$ is in the cover.

The correctness

If f is 1, then there exists an assignment for $x_1, \ldots, x_p$ that satisfies all the terms. Then the corresponding p sets from a cover.

If there is cover, then for every i at least one of $A_{x_1=1}$ and $A_{x_i=0}$ is in the cover in order to cover the term $x_i \vee \bar{x}_i$. Since the cover is of size p, exactly one of $A_{x_1=1}$ and $A_{x_i=0}$ is in the cover.

Then the cover induces a satisfying assignment, since the universe contains all the terms.

Reduction to the set-cover problem (3)

Reduction to the set-cover problem (3)

The reduction can be performed in a small depth $O(\log t)$.

Reduction to the set-cover problem (3)

The reduction can be performed in a small depth $O(\log t)$. Hence

$$d_m(\textsc{set-cover}) \geq d(f) - O(\log t) = \Omega(\log^2 t).$$

Monotone Constant-Depth Circuits

Circuits of unbounded fan-in

Circuits of unbounded fan-in

Now $\wedge$- and $\vee$-gates can have unbounded number of inputs.

Circuits of unbounded fan-in

Now $\wedge$- and $\vee$-gates can have unbounded number of inputs. Among others, constant-depth circuits become meaningful.

Circuits of unbounded fan-in

Now $\wedge$- and $\vee$-gates can have unbounded number of inputs. Among others, constant-depth circuits become meaningful.

We can define similarly $d(f)$ and $L(f)$.

Circuits of unbounded fan-in

Now $\wedge$- and $\vee$-gates can have unbounded number of inputs. Among others, constant-depth circuits become meaningful.

We can define similarly $d(f)$ and $L(f)$.

It is still the case that $L(F)$, the size of a formula F, translate to the protocol partition number $C^P(f)$.

Circuits of unbounded fan-in

Now $\wedge$- and $\vee$-gates can have unbounded number of inputs. Among others, constant-depth circuits become meaningful.

We can define similarly $d(f)$ and $L(f)$.

It is still the case that $L(F)$, the size of a formula F, translate to the protocol partition number $C^P(f)$.

However, the depth $d(f)$ is equal to the round complexity of the protocol, the number of alternations between the communication from Alice to Bob and the communication from Bob to Alice.

Depth k vs. depth $k-1$ for monotone circuits

Depth k vs. depth $k-1$ for monotone circuits

We construct a formula $f : \{0,1\}^n \to \{0,1\}$ with $n = m^k$ as follows.

Depth k vs. depth $k-1$ for monotone circuits

We construct a formula $f : \{0,1\}^n \to \{0,1\}$ with $n = m^k$ as follows.

1. f consists of a complete m-ary tree of depth k.

Depth k vs. depth $k-1$ for monotone circuits

We construct a formula $f : \{0,1\}^n \to \{0,1\}$ with $n = m^k$ as follows.

1. f consists of a complete m-ary tree of depth k.
2. Each of its m^k leaves is labelled by a unique variable in $\{x_1, \ldots, x_n\}$.

Depth k vs. depth $k-1$ for monotone circuits

We construct a formula $f : \{0,1\}^n \to \{0,1\}$ with $n = m^k$ as follows.

1. f consists of a complete m-ary tree of depth k.
2. Each of its m^k leaves is labelled by a unique variable in $\{x_1, \ldots, x_n\}$.
3. The gates in the odd levels (including the root) are labelled by $\wedge$, and those in the even levels are labelled by $\vee$.

Depth k vs. depth $k-1$ for monotone circuits

We construct a formula $f : \{0,1\}^n \to \{0,1\}$ with $n = m^k$ as follows.

1. f consists of a complete m-ary tree of depth k.
2. Each of its m^k leaves is labelled by a unique variable in $\{x_1, \ldots, x_n\}$.
3. The gates in the odd levels (including the root) are labelled by $\wedge$, and those in the even levels are labelled by $\vee$.

We show that any depth $k-1$ formula computing f has size exponential in m.

The tree problem T_k

The tree problem T_k

Consider the complete m-ary tree of depth k.

The tree problem T_k

Consider the complete m-ary tree of depth k. A labelling of the tree assigns to each leaf a bit, and to each internal node a number in $\{1, \ldots, m\}$.

The tree problem T_k

Consider the complete m-ary tree of depth k. A labelling of the tree assigns to each leaf a bit, and to each internal node a number in $\{1, \ldots, m\}$.

The labels of the internal nodes define a (unique) path from the root to a leaf, where the label of each internal node is viewed as a pointer to one of its children.

The tree problem T_k

Consider the complete m-ary tree of depth k. A labelling of the tree assigns to each leaf a bit, and to each internal node a number in $\{1, \ldots, m\}$.

The labels of the internal nodes define a (unique) path from the root to a leaf, where the label of each internal node is viewed as a pointer to one of its children.

An input to the tree problem is a labelling of the tree, where

The tree problem T_k

Consider the complete m-ary tree of depth k. A labelling of the tree assigns to each leaf a bit, and to each internal node a number in $\{1, \ldots, m\}$.

The labels of the internal nodes define a (unique) path from the root to a leaf, where the label of each internal node is viewed as a pointer to one of its children.

An input to the tree problem is a labelling of the tree, where

1. Bob gets as his input the labels of all nodes in the odd levels,

The tree problem T_k

Consider the complete m-ary tree of depth k. A labelling of the tree assigns to each leaf a bit, and to each internal node a number in $\{1, \ldots, m\}$.

The labels of the internal nodes define a (unique) path from the root to a leaf, where the label of each internal node is viewed as a pointer to one of its children.

An input to the tree problem is a labelling of the tree, where

1. Bob gets as his input the labels of all nodes in the odd levels,
2. and Alice gets her input the labels of all nodes in even level.

The tree problem T_k

Consider the complete m-ary tree of depth k. A labelling of the tree assigns to each leaf a bit, and to each internal node a number in $\{1, \ldots, m\}$.

The labels of the internal nodes define a (unique) path from the root to a leaf, where the label of each internal node is viewed as a pointer to one of its children.

An input to the tree problem is a labelling of the tree, where

1. Bob gets as his input the labels of all nodes in the odd levels,
2. and Alice gets her input the labels of all nodes in even level.

The goal is to compute the label of the leaf reached by the path induced by the labelling.

The tree problem T_k

Consider the complete m-ary tree of depth k. A labelling of the tree assigns to each leaf a bit, and to each internal node a number in $\{1, \ldots, m\}$.

The labels of the internal nodes define a (unique) path from the root to a leaf, where the label of each internal node is viewed as a pointer to one of its children.

An input to the tree problem is a labelling of the tree, where

1. Bob gets as his input the labels of all nodes in the odd levels,
2. and Alice gets her input the labels of all nodes in even level.

The goal is to compute the label of the leaf reached by the path induced by the labelling.

It is known that the $(k-1)$-round communication complexity $D^{k-1}(T_k)$ of T_k is

$$D^{k-1}(T_k) = \Omega\big(m/\text{polylog}(m)\big).$$

Reduction from T_k to M_f (1)

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:
   - S_1 contains only the root of the tree.

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:
   - S_1 contains only the root of the tree.
   - If i is even, then

$$S_{i+1} = \left\{ \text{the child of } v \text{ defined by the labelling given to Alice} \;\middle|\; v \in S_i \right\}$$

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:
   - S_1 contains only the root of the tree.
   - If i is even, then
   $$S_{i+1} = \left\{ \text{the child of } v \text{ defined by the labelling given to Alice} \,\middle|\, v \in S_i \right\}$$
   - If i is odd, then
   $$S_{i+1} = \left\{ \text{all the children of } v \,\middle|\, v \in S_i \right\}$$

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:
   - S_1 contains only the root of the tree.
   - If i is even, then
   $$S_{i+1} = \left\{ \text{the child of } v \text{ defined by the labelling given to Alice} \;\middle|\; v \in S_i \right\}$$
   - If i is odd, then
   $$S_{i+1} = \left\{ \text{all the children of } v \;\middle|\; v \in S_i \right\}$$
2. Bob computes a sequence of sets $Q_1, \ldots, Q_k$ inductively:

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:
   - S_1 contains only the root of the tree.
   - If i is even, then

     $$S_{i+1} = \left\{ \text{the child of } v \text{ defined by the labelling given to Alice} \;\middle|\; v \in S_i \right\}$$

   - If i is odd, then

     $$S_{i+1} = \left\{ \text{all the children of } v \;\middle|\; v \in S_i \right\}$$

2. Bob computes a sequence of sets $Q_1, \ldots, Q_k$ inductively:
   - Q_1 contains only the root of the tree.

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:
   - S_1 contains only the root of the tree.
   - If i is even, then
     $$S_{i+1} = \{\text{the child of } v \text{ defined by the labelling given to Alice} \mid v \in S_i\}$$
   - If i is odd, then
     $$S_{i+1} = \{\text{all the children of } v \mid v \in S_i\}$$
2. Bob computes a sequence of sets $Q_1, \ldots, Q_k$ inductively:
   - Q_1 contains only the root of the tree.
   - If i is even, then
     $$Q_{i+1} = \{\text{all the children of } v \mid v \in Q_i\}$$

Reduction from T_k to M_f (1)

1. Alice computes a sequence of sets $S_1, \ldots, S_k$ inductively:
   - S_1 contains only the root of the tree.
   - If i is even, then
   $$S_{i+1} = \left\{\text{the child of } v \text{ defined by the labelling given to Alice} \;\middle|\; v \in S_i\right\}$$
   - If i is odd, then
   $$S_{i+1} = \left\{\text{all the children of } v \;\middle|\; v \in S_i\right\}$$
2. Bob computes a sequence of sets $Q_1, \ldots, Q_k$ inductively:
   - Q_1 contains only the root of the tree.
   - If i is even, then
   $$Q_{i+1} = \left\{\text{all the children of } v \;\middle|\; v \in Q_i\right\}$$
   - If i is odd, then
   $$Q_{i+1} = \left\{\text{the child of } v \text{ defined by the labelling given to Bob} \;\middle|\; v \in Q_i\right\}$$

Reduction from T_k to M_f (2)

Reduction from T_k to M_f (2)

3. Alice computes a string x of length n by putting 1 in all coordinates j for $j \in S_k$ and 0 elsewhere.

Reduction from T_k to M_f (2)

3. Alice computes a string x of length n by putting 1 in all coordinates j for $j \in S_k$ and 0 elsewhere.

4. Bob computes a string y of length n by putting 0 in all coordinates j for $j \in Q_k$ and 1 elsewhere.

Reduction from T_k to M_f (2)

3. Alice computes a string x of length n by putting 1 in all coordinates j for $j \in S_k$ and 0 elsewhere.
4. Bob computes a string y of length n by putting 0 in all coordinates j for $j \in Q_k$ and 1 elsewhere.
5. Finally, Alice and Bob use the protocol for M_f on (x, y) and output the result.

The correctness (1)

The correctness (1)

We first show

$$f(x) = 1 \quad \text{and} \quad f(y) = 0$$

The correctness (1)

We first show

$$f(x) = 1 \quad \text{and} \quad f(y) = 0$$

$f(x) = 1$ By induction on i from $k-1$ to 1, if each node in S_{i+1} computes the value 1, then so do all the nodes in S_i.

The correctness (1)

We first show

$$f(x) = 1 \quad \text{and} \quad f(y) = 0$$

$f(x) = 1$ By induction on i from $k-1$ to 1, if each node in S_{i+1} computes the value 1, then so do all the nodes in S_i.

$f(y) = 0$ By induction on i from $k-1$ to 1 if each node in Q_{i+1} computes the value 0, then so do all the nodes in Q_i.

The correctness (2)

The correctness (2)

Finally, we prove that there is exactly one j with $x_j = 1$ and $y_j = 0$ by showing that for every $i \in \{1, \ldots, k\}$ the set $S_i \cap Q_i$ includes a single node v_i, which is the node in level i that the path from the root reaches.

The correctness (2)

Finally, we prove that there is exactly one j with $x_j = 1$ and $y_j = 0$ by showing that for every $i \in \{1, \ldots, k\}$ the set $S_i \cap Q_i$ includes a single node v_i, which is the node in level i that the path from the root reaches.

- It is trivially true for $i = 1$, i.e., $S_1 = Q_1 = \{\mathrm{root}\}$.

The correctness (2)

Finally, we prove that there is exactly one j with $x_j = 1$ and $y_j = 0$ by showing that for every $i \in \{1, \ldots, k\}$ the set $S_i \cap Q_i$ includes a single node v_i, which is the node in level i that the path from the root reaches.

- It is trivially true for $i = 1$, i.e., $S_1 = Q_1 = \{\mathrm{root}\}$.
- If i is odd, then we put all the children of S_i to S_{i+1}, and only those defined by the labelling to Q_{i+1}.

The correctness (2)

Finally, we prove that there is exactly one j with $x_j = 1$ and $y_j = 0$ by showing that for every $i \in \{1, \dots, k\}$ the set $S_i \cap Q_i$ includes a single node v_i, which is the node in level i that the path from the root reaches.

- It is trivially true for $i = 1$, i.e., $S_1 = Q_1 = \{\mathrm{root}\}$.
- If i is odd, then we put all the children of S_i to S_{i+1}, and only those defined by the labelling to Q_{i+1}. Since $v_i \in S_i \cap Q_i$, then the next node v_{i+1} on the path is in $S_{i+1} \cap Q_{i+1}$.

The correctness (2)

Finally, we prove that there is exactly one j with $x_j = 1$ and $y_j = 0$ by showing that for every $i \in \{1, \ldots, k\}$ the set $S_i \cap Q_i$ includes a single node v_i, which is the node in level i that the path from the root reaches.

- It is trivially true for $i = 1$, i.e., $S_1 = Q_1 = \{\mathrm{root}\}$.
- If i is odd, then we put all the children of S_i to S_{i+1}, and only those defined by the labelling to Q_{i+1}. Since $v_i \in S_i \cap Q_i$, then the next node v_{i+1} on the path is in $S_{i+1} \cap Q_{i+1}$. Conversely, if $v \in S_{i+1} \cap Q_{i+1}$, then its father is in $S_i \cap Q_i = \{v_i\}$. Thus, $v = v_{i+1}$.

The correctness (2)

Finally, we prove that there is exactly one j with $x_j = 1$ and $y_j = 0$ by showing that for every $i \in \{1, \ldots, k\}$ the set $S_i \cap Q_i$ includes a single node v_i, which is the node in level i that the path from the root reaches.

- It is trivially true for $i = 1$, i.e., $S_1 = Q_1 = \{\mathrm{root}\}$.
- If i is odd, then we put all the children of S_i to S_{i+1}, and only those defined by the labelling to Q_{i+1}. Since $v_i \in S_i \cap Q_i$, then the next node v_{i+1} on the path is in $S_{i+1} \cap Q_{i+1}$. Conversely, if $v \in S_{i+1} \cap Q_{i+1}$, then its father is in $S_i \cap Q_i = \{v_i\}$. Thus, $v = v_{i+1}$.
- The case for even i is symmetric.

The lower bound

We conclude for any constant k, the size of any depth $k-1$ formula for f is

$$C^{P,k-1}(M_f) = \Omega\left(2^{D^{k-1}(M_f)/(k-1)}\right) = \Omega\left(2^{D^{k-1}(T_f)/(k-1)}\right) = \Omega\left(2^{m/\text{polylog}(m)}\right).$$

Small Circuits

Q-Circuits

A Q-circuit is a directed acyclic graph whose gates are taken from a fixed family of gates Q.

Q-Circuits

A Q-circuit is a directed acyclic graph whose gates are taken from a fixed family of gates Q.

The cost of a circuit is its size, i.e., the number of gates.

Q-Circuits

A Q-circuit is a directed acyclic graph whose gates are taken from a fixed family of gates Q.

The cost of a circuit is its size, i.e., the number of gates.

Definition

The Q-circuits complexity of a function f, denoted by $S_Q(f)$, is the minimum cost of a Q-circuit computing f.

Worst-case partition

Worst-case partition

Definition

Let $f : \{0,1\}^m \to \{0,1\}$ be a function. Let S and T be a partition of the variables $x_1, \ldots, x_m$ into two disjoint sets. The (deterministic) communication complexity of f between S and T, denoted $D^{S:T}(f)$, is the complexity of computing f where Alice sees all bits in S, and Bob sees all bits in T.

Worst-case partition

Definition

Let $f : \{0,1\}^m \to \{0,1\}$ be a function. Let S and T be a partition of the variables $x_1, \ldots, x_m$ into two disjoint sets. The (deterministic) communication complexity of f between S and T, denoted $D^{S:T}(f)$, is the complexity of computing f where Alice sees all bits in S, and Bob sees all bits in T.

The worse-case communication complexity of f, denoted by $D^{\mathrm{worst}}(f)$, is the maximum of $D^{S;T}(f)$ over all such partitions.

Lemma

Denote $c_Q = \max_{q \in Q} D^{\mathrm{worst}}(q)$.

Lemma

Denote $c_Q = \max_{q \in Q} D^{\mathrm{worst}}(q)$. *Then, for all* f *we have*

$$S_Q(f) \geq \frac{D^{\mathrm{worst}}(f)}{c_Q}.$$

Proof

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a “bottom-up” order of the gates.

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a "bottom-up" order of the gates.
2. For every gate q there are some inputs for q that are the results of previous gates (whose values are known to both player) and some input variables.

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a “bottom-up” order of the gates.
2. For every gate q there are some inputs for q that are the results of previous gates (whose values are known to both player) and some input variables.

   Alice and Bob compute the value of q using the best protocol for computing q with respect to any partition that has

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a "bottom-up" order of the gates.
2. For every gate q there are some inputs for q that are the results of previous gates (whose values are known to both player) and some input variables.

   Alice and Bob compute the value of q using the best protocol for computing q with respect to any partition that has

   - all the inputs to the gate that are Alice's variables in one set,

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a “bottom-up” order of the gates.
2. For every gate q there are some inputs for q that are the results of previous gates (whose values are known to both player) and some input variables.

   Alice and Bob compute the value of q using the best protocol for computing q with respect to any partition that has

   - all the inputs to the gate that are Alice’s variables in one set,
   - all the inputs to the gate that are Bob’s variables in the other set,

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a “bottom-up” order of the gates.
2. For every gate q there are some inputs for q that are the results of previous gates (whose values are known to both player) and some input variables.

   Alice and Bob compute the value of q using the best protocol for computing q with respect to any partition that has

   - all the inputs to the gate that are Alice’s variables in one set,
   - all the inputs to the gate that are Bob’s variables in the other set,
   - and all the other inputs (that both players know) are partitioned in an arbitrary way.

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a "bottom-up" order of the gates.
2. For every gate q there are some inputs for q that are the results of previous gates (whose values are known to both player) and some input variables.

   Alice and Bob compute the value of q using the best protocol for computing q with respect to any partition that has

   - all the inputs to the gate that are Alice's variables in one set,
   - all the inputs to the gate that are Bob's variables in the other set,
   - and all the other inputs (that both players know) are partitioned in an arbitrary way.

   Thus, to simulate each gate, c_Q bits of communication are sufficient.

Proof

Fix an arbitrary partition of the input bits into two disjoint sets.

1. Alice and Bob agree on a "bottom-up" order of the gates.
2. For every gate q there are some inputs for q that are the results of previous gates (whose values are known to both player) and some input variables.

   Alice and Bob compute the value of q using the best protocol for computing q with respect to any partition that has

   - all the inputs to the gate that are Alice's variables in one set,
   - all the inputs to the gate that are Bob's variables in the other set,
   - and all the other inputs (that both players know) are partitioned in an arbitrary way.

   Thus, to simulate each gate, c_Q bits of communication are sufficient.

Because the circuit is of size $S_Q(f)$, the whole simulation uses at most

$$c_Q \cdot S_Q(f)$$

bits. □

Threshold gates

Threshold gates

A threshold gate is determined by:

Threshold gates

A threshold gate is determined by:

1. t edges $z_1, \ldots, z_t$ entering the gate,

Threshold gates

A threshold gate is determined by:

1. t edges $z_1, \ldots, z_t$ entering the gate,
2. each edge is associated with an integer weight w_i,

Threshold gates

A threshold gate is determined by:

1. t edges $z_1, \ldots, z_t$ entering the gate,
2. each edge is associated with an integer weight w_i,
3. an integer θ.

Threshold gates

A threshold gate is determined by:

1. t edges $z_1, \ldots, z_t$ entering the gate,
2. each edge is associated with an integer weight w_i,
3. an integer θ.

Then the gate computes whether

$$\sum_{i=1}^{t} w_i \cdot z_i > \theta.$$

GT by threshold gates

GT by threshold gates

The "greater than" function

$$\textsc{gt}(x, y) = \begin{cases} 1, & \text{if } x > y \\ 0, & \text{otherwise.} \end{cases}$$

$\textsc{gt}$ by threshold gates

The "greater than" function

$$\textsc{gt}(x, y) = \begin{cases} 1, & \text{if } x > y \\ 0, & \text{otherwise.} \end{cases}$$

Assume $x = x_n \cdots x_1$ and $y = y_n \cdots y_1$. Then

$$x > y \iff \sum_{i=1}^{n} 2^{i-1} x_i + \sum_{i=1}^{n} -2^{i-1} y_i = x - y > \theta = 0.$$

The total weight of a threshold gate

The total weight of a threshold gate

For a threshold gate specified by $(w_1, \ldots, w_t, \theta)$, its total weight is

$$\sum_{i=1}^{t} |w_i|.$$

The total weight of a threshold gate

For a threshold gate specified by $(w_1, \ldots, w_t, \theta)$, its total weight is

$$\sum_{i=1}^{t} |w_i|.$$

For the previous threshold gate $(1, 2, \ldots, 2^{n-1}, -1, -2, \ldots, -2^{n-1}, 0)$, its total weight is

$$W = 2 \cdot \sum_{i=1}^{n} 2^{i-1} = 2^{n+1} - 2.$$

The total weight of a threshold gate

For a threshold gate specified by $(w_1, \ldots, w_t, \theta)$, its total weight is

$$\sum_{i=1}^{t} |w_i|.$$

For the previous threshold gate $(1, 2, \ldots, 2^{n-1}, -1, -2, \ldots, -2^{n-1}, 0)$, its total weight is

$$W = 2 \cdot \sum_{i=1}^{n} 2^{i-1} = 2^{n+1} - 2.$$

We will show that an exponential weight, $W \geq 2^n$ is necessary for computing GT with a single gate.

$c_Q \leq \log W + 1$

$c_Q \leq \log W + 1$

Let $S : T$ be an arbitrary partition of the input bits.

$c_Q \leq \log W + 1$

Let $S : T$ be an arbitrary partition of the input bits.

1. Alice computes $\sum_{z_i \in S} w_{z_i} z_i$ and send the result to Bob.

$c_Q \leq \log W + 1$

Let $S : T$ be an arbitrary partition of the input bits.

1. Alice computes $\sum_{z_i \in S} w_{z_i} z_i$ and send the result to Bob.
2. Bob computes $\sum_{z_i \in T} w_{z_i} z_i$, adds the result to the number received from Alice, and compares the sum with θ.

Now we apply

$$S_Q(\mathrm{GT}) \geq D^{\mathrm{worst}}(\mathrm{GT})/c_Q$$

and $D^{\mathrm{worst}}(\mathrm{GT}) = D(\mathrm{GT}) = n + 1$.

Now we apply

$$S_Q(\textsc{gt}) \geq D^{\text{worst}}(\textsc{gt})/c_Q$$

and $D^{\text{worst}}(\textsc{gt}) = D(\textsc{gt}) = n + 1$. Thus the size of any Q-circuit computing $\textsc{gt}$ is at least

$$\frac{n+1}{\log W + 1}.$$

Depth 2 Threshold Circuits

Lemma

Assume that a function $f : \{0,1\}^m \to \{0,1\}$ *can be computed by a depth 2 threshold circuit, whether the total weight of each gate is bounded by* W*. Then*

$$R^{\mathrm{pub,worst}}_{1/2+1/(4W)}(f) \leq \log W + 1.$$

Proof (1)

Proof (1)

The first step is to covert a given circuit for f to a circuit with the following properties.

Proof (1)

The first step is to covert a given circuit for f to a circuit with the following properties.

1. The top gate is a threshold gate whose threshold $\theta' = 0$.

Proof (1)

The first step is to covert a given circuit for f to a circuit with the following properties.

1. The top gate is a threshold gate whose threshold $\theta' = 0$. We feed the gate with the constant 1 with weight $-\theta$.

Proof (1)

The first step is to covert a given circuit for f to a circuit with the following properties.

1. The top gate is a threshold gate whose threshold $\theta' = 0$. We feed the gate with the constant 1 with weight $-\theta$.

2. The weighted sum computed by the gate is always nonzero.

Proof (1)

The first step is to covert a given circuit for f to a circuit with the following properties.

1. The top gate is a threshold gate whose threshold $\theta' = 0$. We feed the gate with the constant 1 with weight $-\theta$.

2. The weighted sum computed by the gate is always nonzero. We multiply each weight by 2,

Proof (1)

The first step is to covert a given circuit for f to a circuit with the following properties.

1. The top gate is a threshold gate whose threshold $\theta' = 0$. We feed the gate with the constant 1 with weight $-\theta$.

2. The weighted sum computed by the gate is always nonzero. We multiply each weight by 2, and decrease the weight of the constant 1 by 1, i.e., its weight is $-2\theta + 1$.

Proof (1)

The first step is to covert a given circuit for f to a circuit with the following properties.

1. The top gate is a threshold gate whose threshold $\theta' = 0$. We feed the gate with the constant 1 with weight $-\theta$.

2. The weighted sum computed by the gate is always nonzero. We multiply each weight by 2, and decrease the weight of the constant 1 by 1, i.e., its weight is $-2\theta + 1$.

The function does not change, and the new total weight $W' \leq 4W$.

Proof (2)

Proof (2)

Let $f_1, \ldots, f_t$ be the functions that are the inputs to the top gate and $w_1, \ldots, w_t$. These functions are either constants or input variables or threshold gates, all satisfy

$$D^{\mathrm{worst}}(f_i) \leq \log W + 1.$$

Proof (3)

Proof (3)

Fix an arbitrary partition of the inputs, and in the public coin model.

Proof (3)

Fix an arbitrary partition of the inputs, and in the public coin model.

1. Alice and Bob choose at random an index $1 \leq i \leq t$ with

$$\Pr\left[i \text{ is chosen}\right] = \frac{|w_i|}{W'}.$$

Proof (3)

Fix an arbitrary partition of the inputs, and in the public coin model.

1. Alice and Bob choose at random an index $1 \leq i \leq t$ with

$$\Pr\left[i \text{ is chosen}\right] = \frac{|w_i|}{W'}.$$

2. They run the deterministic protocol for f_i with the fixed partition to get an output b.

Proof (3)

Fix an arbitrary partition of the inputs, and in the public coin model.

1. Alice and Bob choose at random an index $1 \leq i \leq t$ with

$$\Pr\left[i \text{ is chosen}\right] = \frac{|w_i|}{W'}.$$

2. They run the deterministic protocol for f_i with the fixed partition to get an output b.

3. 3.1 If $b = 0$, then the output is chosen uniformly at random from 0 and 1.

Proof (3)

Fix an arbitrary partition of the inputs, and in the public coin model.

1. Alice and Bob choose at random an index $1 \leq i \leq t$ with

$$\Pr\left[i \text{ is chosen}\right] = \frac{|w_i|}{W'}.$$

2. They run the deterministic protocol for f_i with the fixed partition to get an output b.
3. 3.1 If $b = 0$, then the output is chosen uniformly at random from 0 and 1.
   3.2 If $b = 1$, then the output is 1 if $w_i > 0$ and 0 if $w_i < 0$.

Proof (4)

Proof (4)

Consider an input x with $f(x) = 1$.

Proof (4)

Consider an input x with $f(x) = 1$. Let

$$\alpha = \Pr_{1 \leq i \leq t}\left[f_i(x) = 0\right].$$

Proof (4)

Consider an input x with $f(x) = 1$. Let

$$\alpha = \Pr_{1 \leq i \leq t} \left[f_i(x) = 0 \right].$$

The contribution of these indices to the probability that the output is 1 is $\alpha/2$.

Proof (4)

Consider an input x with $f(x) = 1$. Let

$$\alpha = \Pr_{1 \leq i \leq t}\left[f_i(x) = 0\right].$$

The contribution of these indices to the probability that the output is 1 is $\alpha/2$.

Moreover

$$\Pr_i\left[f_i(x) = 1\right] = 1 - \alpha.$$

Proof (4)

Consider an input x with $f(x) = 1$. Let

$$\alpha = \Pr_{1 \leq i \leq t} \left[f_i(x) = 0 \right].$$

The contribution of these indices to the probability that the output is 1 is $\alpha/2$.

Moreover

$$\Pr_i \left[f_i(x) = 1 \right] = 1 - \alpha.$$

By $f(x) = \text{``}\sum_i w_i \cdot f_i(x) > 0?''$ we have

$$\sum_{i : f_i(x) = 1} w_i > 0,$$

Proof (4)

Consider an input x with $f(x) = 1$. Let

$$\alpha = \Pr_{1 \leq i \leq t}\left[f_i(x) = 0\right].$$

The contribution of these indices to the probability that the output is 1 is $\alpha/2$.

Moreover

$$\Pr_i\left[f_i(x) = 1\right] = 1 - \alpha.$$

By $f(x) =$ "$\sum_i w_i \cdot f_i(x) > 0?$" we have

$$\sum_{i : f_i(x) = 1} w_i > 0,$$

and note the weights are integers,

Proof (4)

Consider an input x with $f(x) = 1$. Let

$$\alpha = \Pr_{1 \le i \le t}\left[f_i(x) = 0\right].$$

The contribution of these indices to the probability that the output is 1 is $\alpha/2$.

Moreover

$$\Pr_i\left[f_i(x) = 1\right] = 1 - \alpha.$$

By $f(x) = \text{``}\sum_i w_i \cdot f_i(x) > 0?''$ we have

$$\sum_{i:f_i(x)=1} w_i > 0,$$

and note the weights are integers, the contribution of these indices to the probability that the output is 1 is at least $(1-\alpha)/2 + 1/W'$.

Proof (4)

Consider an input x with $f(x) = 1$. Let

$$\alpha = \Pr_{1 \leq i \leq t}\left[f_i(x) = 0\right].$$

The contribution of these indices to the probability that the output is 1 is $\alpha/2$.

Moreover

$$\Pr_i\left[f_i(x) = 1\right] = 1 - \alpha.$$

By $f(x) =$ "$\sum_i w_i \cdot f_i(x) > 0?$" we have

$$\sum_{i:f_i(x)=1} w_i > 0,$$

and note the weights are integers, the contribution of these indices to the probability that the output is 1 is at least $(1-\alpha)/2 + 1/W'$.

Therefore the total success probability is at least

$$\frac{\alpha}{2} + \frac{1-\alpha}{2} + \frac{1}{W'} = \frac{1}{2} + \frac{1}{W'}.$$

Proof (5)

Proof (5)

The case for an x with $f(x) = 0$ is symmetric.

Proof (5)

The case for an x with $f(x) = 0$ is symmetric. Here, we crucially use the fact that

$$\sum_i w_i \cdot f_i(x) \neq 0.$$

Proof (5)

The case for an x with $f(x) = 0$ is symmetric. Here, we crucially use the fact that

$$\sum_i w_i \cdot f_i(x) \neq 0.$$

In both case, we get the correct answer with probability

$$\frac{1}{2} + \frac{1}{W'} \geq \frac{1}{2} + \frac{1}{4W}.$$

Proof (5)

The case for an x with $f(x) = 0$ is symmetric. Here, we crucially use the fact that

$$\sum_i w_i \cdot f_i(x) \neq 0.$$

In both case, we get the correct answer with probability

$$\frac{1}{2} + \frac{1}{W'} \geq \frac{1}{2} + \frac{1}{4W}.$$

And

$$D^{\text{worst}}(f_i) \leq \log W + 1. \qquad \square$$

IP by depth 2 threshold circuits

IP by depth 2 threshold circuits

By Exercise 3.30

$$R^{\mathrm{pub,worst}}_{1/2+1/W}(\mathrm{IP}) \geq m - O(\log W).$$

IP by depth 2 threshold circuits

By Exercise 3.30

$$R^{\mathrm{pub,worst}}_{1/2+1/W}(\mathrm{IP}) \geq m - O(\log W).$$

By the previous lemma any depth 2 threshold circuit for IP must satisfy

$$R^{\mathrm{pub,worst}}_{1/2+1/W}(\mathrm{IP}) \leq \log(W/4) + 1.$$

IP by depth 2 threshold circuits

By Exercise 3.30

$$R^{\mathrm{pub,worst}}_{1/2+1/W}(\mathrm{IP}) \geq m - O(\log W).$$

By the previous lemma any depth 2 threshold circuit for IP must satisfy

$$R^{\mathrm{pub,worst}}_{1/2+1/W}(\mathrm{IP}) \leq \log(W/4) + 1.$$

Thus

$$W = 2^{\Omega(m)}.$$

IP by depth 2 threshold circuits

By Exercise 3.30

$$R^{\text{pub,worst}}_{1/2+1/W}(\text{IP}) \geq m - O(\log W).$$

By the previous lemma any depth 2 threshold circuit for IP must satisfy

$$R^{\text{pub,worst}}_{1/2+1/W}(\text{IP}) \leq \log(W/4) + 1.$$

Thus

$$W = 2^{\Omega(m)}.$$

If the circuit has s gates with each w_i and θ is at most w, then

$$s = 2^{\Omega(m)}/w.$$

$\textsc{IP}$ by depth 2 threshold circuits

By Exercise 3.30

$$R^{\text{pub,worst}}_{1/2+1/W}(\textsc{IP}) \geq m - O(\log W).$$

By the previous lemma any depth 2 threshold circuit for $\textsc{IP}$ must satisfy

$$R^{\text{pub,worst}}_{1/2+1/W}(\textsc{IP}) \leq \log(W/4) + 1.$$

Thus

$$W = 2^{\Omega(m)}.$$

If the circuit has s gates with each w_i and θ is at most w, then

$$s = 2^{\Omega(m)}/w.$$

That is, provided the weights are small, the size of the circuit has to be exponential.

Thank You!