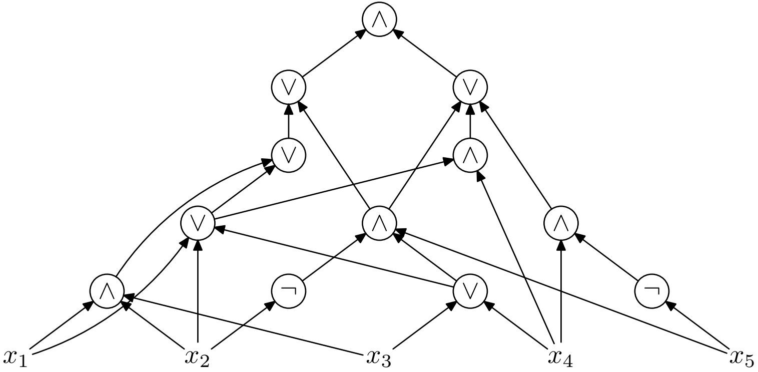
Parameterized Complexity Theory

Definition

A boolean circuit is an acyclic graph such that:

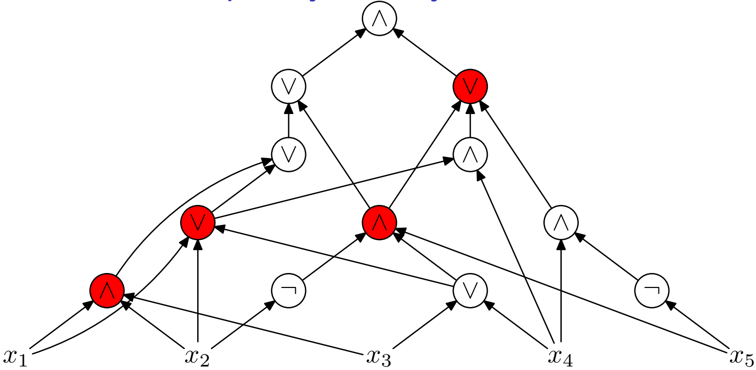
- There is exactly one vertex with outdegree 0, the Output.
- Every vertex with indegree 0 is an Input and labeled by x_i or $\neg x_i$.
- All other vertices are gates and labeled by $\wedge$, $\vee$ or $\neg$ (with indegree 1).

Parameterized Complexity Theory



To find out whether a boolean circuit has a satisfying assignment is *NP*-complete.

Parameterized Complexity Theory



Big gates have indegree > 2.

The height is the length of the longest path.

The weft is the maximal number of big gates on some path.

Parameterized Complexity Theory

Definition

Let $\mathcal{F}(t, h)$ be the set of all boolean circuits with height h and weft t.

Definition

The weighted satisfiability problem $L_{\mathcal{F}(t,h)}$:

Input: (G, k), where $G \in \mathcal{F}(t, h)$
Parameter: k
Question: Has G a satisfiying assingment of weight k

The weight of an assignment is the number of 1s.

Parameterized Complexity Theory

Definition

A parameterized problem is in the complexity class $W[t]$ if it can be reduced to $L_{\mathcal{F}(t,h)}$ for some h by a parameterized reduction.

Example

Independent Set is in $W[1]$.

Dominating Set is in $W[2]$.

Question: Why?

Parameterized Complexity Theory

Definition

A problem L is $W[t]$-hard, if every problem in $W[t]$ can be reduced to L by a parameterized reduction.

Definition

A problem is $W[t]$-complete, if it belongs to $W[t]$ and is $W[t]$-hard.

Parameterized Complexity Theory

Theorem

Let A be a $W[t]$-complete problem.
Assume that A can be reduced to B by a parameterized reduction and $B \in W[t]$.
Then B is also $W[t]$-complete.

Proof

We already assumed that $B \in W[t]$.
Since every problem in $W[t]$ can be reduced to A, the $W[t]$-hardness follows from the transitivity of parameterized reducibility.

Short Turing Machine Acceptance

We will see later that the following problem is $W[1]$-complete:

Definition

Short Turing Machine Acceptance:

Input: A non-deterministic Turing maching M, a word w, a number k.
Parameter: k
Question: Does M have an accepting path of length at most k on input w?

The class $W[1, s]$ and $W[1, 2]$

We consider the weighted satisfiability problem for very simple circuits.

Definition

Let $s > 1$ be a number. By $\mathcal{F}(s)$ we denote the family of all circuit whose output is an AND-gate that is connected to OR-gates with indegrees at most s. The OR-gates are directly connected to inputs (literals, i.e., variables or negated variables).

We define $W[1, s]$ as the class of all problems in $W[1]$ that can be reduced to $L_{\mathcal{F}(s)}$ by a parameterized reduction.

We will prove the following theorem:

Theorem

Short Turing Machine Acceptance $\in W[1,2]$

For this end we need a reduction from Short Turing Machine Acceptance to $L_{\mathcal{F}(2)}$.

We have to map M, w, k to a circuit and a number $k' = f(k)$ in such a way that there is a satisfying assignment of weight k' iff M accepts w in at most k steps.

We will prove the following theorem:

Theorem

Short Turing Machine Acceptance $\in W[1,2]$

For this end we need a reduction from Short Turing Machine Acceptance to $L_{\mathcal{F}(2)}$.

We have to map M, w, k to a circuit and a number $k' = f(k)$ in such a way that there is a satisfying assignment of weight k' iff M accepts w in at most k steps.

We can enumerate all configurations of M and therefore identify them with the numbers $1, 2, \ldots$

A configuration consists of

- the position of the read/write-head,
- the state.

If i is a configuration and a is a symbol, then let $\delta(i, a) = (j, b)$ hold if M changes from configuration i into j when reading the symbol a overwriting it with b.

We model with the variable $C_{t,i,j,a,b}$ that M changes from configuration i to j in the tth step while reading an a and overwriting it with a b.

We can enumerate all configurations of M and therefore identify them with the numbers $1, 2, \ldots$

A configuration consists of

- the position of the read/write-head,
- the state.

If i is a configuration and a is a symbol, then let $\delta(i, a) = (j, b)$ hold if M changes from configuration i into j when reading the symbol a overwriting it with b.

We model with the variable $C_{t,i,j,a,b}$ that M changes from configuration i to j in the tth step while reading an a and overwriting it with a b.

We can enumerate all configurations of M and therefore identify them with the numbers $1, 2, \ldots$

A configuration consists of

- the position of the read/write-head,
- the state.

If i is a configuration and a is a symbol, then let $\delta(i, a) = (j, b)$ hold if M changes from configuration i into j when reading the symbol a overwriting it with b.

We model with the variable $C_{t,i,j,a,b}$ that M changes from configuration i to j in the tth step while reading an a and overwriting it with a b.

We can enumerate all configurations of M and therefore identify them with the numbers $1, 2, \ldots$

A configuration consists of

- the position of the read/write-head,
- the state.

If i is a configuration and a is a symbol, then let $\delta(i, a) = (j, b)$ hold if M changes from configuration i into j when reading the symbol a overwriting it with b.

We model with the variable $C_{t,i,j,a,b}$ that M changes from configuration i to j in the tth step while reading an a and overwriting it with a b.

The variable $M_{t,p,a,b}$ models that at the beginning of the tth step the symbol a can be found at the pth position of the tape and that it is overwritten with a b during this step.

Our next goal is to construct a formula whose satisfying assignments model a computation of the Turing maching M.

In particular there will be a satisfying assignment with $C_{t,i,j,a,b} = 1$ iff there is a computation where M goes from configuration i to j in its tth step reading an a and overwriting it with a b.

Every possible computation path should correspond to exactly one (weighted) satisfying assignment.

Moreover: There should exist only accepting computation of length k and satisfying assignments with weight $f(k)$.

The variable $M_{t,p,a,b}$ models that at the beginning of the tth step the symbol a can be found at the pth position of the tape and that it is overwritten with a b during this step.

Our next goal is to construct a formula whose satisfying assignments model a computation of the Turing maching M.

In particular there will be a satisfying assignment with $C_{t,i,j,a,b} = 1$ iff there is a computation where M goes from configuration i to j in its tth step reading an a and overwriting it with a b.

Every possible computation path should correspond to exactly one (weighted) satisfying assignment.

Moreover: There should exist only accepting computation of length k and satisfying assignments with weight $f(k)$.

We need a lot of constraints in order to make this model work.

We will express each constraint by an AND of ORs.

We define clauses in such a way that a wrong modelling automatically leads to a non-satisfying assignment.

In that way, satisfying assignments are those that do not overstep any rule.

Rule 1:

"Nothing exists twice because we have a computation path"

More precisely: In step t the machine M is in exactly one state, changes to exactly one other, reads exactly one symbol, and overwrites it by exactly one symbol.

How can we enforce Rule 1 by clauses?

$$C_{t,i,j,a,b} \rightarrow \overline{C_{t,i',j',a',b'}}$$

or, equivalently,

$$\overline{C_{t,i,j,a,b}} \vee \overline{C_{t,i',j',a',b'}}$$

for all $t, i, j, a, b, i', j', a', b'$ with $(i, j, a, b) \neq (i', j', a', b')$ and

$$M_{t,p,a,b} \rightarrow \overline{M_{t,p,a',b'}}$$

for all t, p, a, b, a', b' with $(a, b) \neq (a', b')$.

Rule 1:

"Nothing exists twice because we have a computation path"

More precisely: In step t the machine M is in exactly one state, changes to exactly one other, reads exactly one symbol, and overwrites it by exactly one symbol.

How can we enforce Rule 1 by clauses?

$$C_{t,i,j,a,b} \to \overline{C_{t,i',j',a',b'}}$$

or, equivalently,

$$\overline{C_{t,i,j,a,b}} \vee \overline{C_{t,i',j',a',b'}}$$

for all $t, i, j, a, b, i', j', a', b'$ with $(i, j, a, b) \neq (i', j', a', b')$ and

$$M_{t,p,a,b} \to \overline{M_{t,p,a',b'}}$$

for all t, p, a, b, a', b' with $(a, b) \neq (a', b')$.

Rule 1:

"Nothing exists twice because we have a computation path"

More precisely: In step t the machine M is in exactly one state, changes to exactly one other, reads exactly one symbol, and overwrites it by exactly one symbol.

How can we enforce Rule 1 by clauses?

Question:
Why only one possibility? We have nondeterministic TMs after all?

or, equivalently,

$$\overline{C_{t,i,j,a,b}} \vee \overline{C_{t,i',j',a',b'}}$$

for all $t, i, j, a, b, i', j', a', b'$ with $(i, j, a, b) \neq (i', j', a', b')$ and

$$M_{t,p,a,b} \rightarrow \overline{M_{t,p,a',b'}}$$

for all t, p, a, b, a', b' with $(a, b) \neq (a', b')$.

Rule 2:

"M and C must fit together."

More precisely: If we read a symbol, it has to be there beforehand. If we write a symbol, it must be there afterwards.

How can we enforce Rule 2 by clauses?

$$C_{t,i,j,a,b} \rightarrow M_{t,p(i),a,b}$$

for all t, i, j, a, b, where $p(i)$ is the position of the read/write head in configuration i.

Rule 2:

"M and C must fit together."

More precisely: If we read a symbol, it has to be there beforehand. If we write a symbol, it must be there afterwards.

How can we enforce Rule 2 by clauses?

$$C_{t,i,j,a,b} \to M_{t,p(i),a,b}$$

for all t, i, j, a, b, where $p(i)$ is the position of the read/write head in configuration i.

Rule 2:

"M and C must fit together."

More precise

If we write a

How can we

Question:

Do we need the other direction, too?

"If there is a symbol on the tape, then this symbol is read."

beforehand.

$$C_{t,i,j,a,b} \to M_{t,p(i),a,b}$$

for all t, i, j, a, b, where $p(i)$ is the position of the read/write head in configuration i.

Rule 3:

"Subsequent steps have to fit together."

More precisely: If one step ends with a configuration, the next step has to start with the same one. The tape content cannot change from step t to step $t+1$ at most places.

How can we enforce Rule 3 by clauses?

$$C_{t,i,j,a,b} \to \overline{C_{t+1,i',j',c,d}}$$

for all $t, i, j, i', j' a, b, c, d$ with $i' \neq j$ and

$$M_{t,p,a,b} \to \overline{M_{t+1,p,c,d}}$$

for all t, p, a, b, c, d with $b \neq c$.

Rule 3:

"Subsequent steps have to fit together."

More precisely: If one step ends with a configuration, the next step has to start with the same one. The tape content cannot change from step t to step $t+1$ at most places.

How can we enforce Rule 3 by clauses?

$$C_{t,i,j,a,b} \to \overline{C_{t+1,i',j',c,d}}$$

for all $t, i, j, i', j' a, b, c, d$ with $i' \neq j$ and

$$M_{t,p,a,b} \to \overline{M_{t+1,p,c,d}}$$

for all t, p, a, b, c, d with $b \neq c$.

Rule 4:

“The beginning and the end have to be correct. The computation path must be accepting.”

$\rightarrow$ Exercise

Because all rules have to hold simultaneously we can combine them with a big AND.

This yields an $\mathcal{F}(2)$-formula as desired.

There is an accepting path of length k iff there is a satisfying assignment with weight k'.

Rule 4:

"The beginning and the end have to be correct. The computation path must be accepting."

$\rightarrow$ Exercise

Because all rules have to hold simultaneously we can combine them with a big AND.

This yields an $\mathcal{F}(2)$-formula as desired.

There is an accepting path of length k iff there is a satisfying assignment with weight k'.

Rule 4:

"The beginning and the end have to be correct. The computation path must be accepting."

→ Exercise

Because all rules have to hold simultaneously we can combine them w

Question:

How big is k'?

This yields an $\mathcal{F}(2)$-formula as desired.

There is an accepting path of length k iff there is a satisfying assignment with weight k'.

Remember:

We just proved the following.

Theorem

Short Turing Machine Acceptance $\in W[1,2]$

The class antimonotone-$W[1, s]$

We consider the weighted satisfiability problem for very simple structured circuits.

Definition

Let $s > 1$ be a number. By antimonotone-$\mathcal{F}(s)$ we denote the family of all circuits whose output is an AND-gate connected to OR-gates with indegree at most s. The OR-gates are connected to negative literals only (negated variables).

Antimonotone-$W[1, s]$ is the class of all parameterized problems in $W[1]$ that can be reduced to $L_{\text{Antimonoton-}\mathcal{F}(s)}$ by a parameterized reduction.

Theorem

$L_{Antimonoton\text{-}\mathcal{F}(s)}$ can be reduced to Short Turing Machine Acceptance be a parameterized reduction.

Corollary

antimonotone-$W[1, s] \subseteq$ antimonotone-$W[1, 2]$

Proof

Construct a turing machine that works as follows:

1. Guess k variables onto the tape.
2. Visit all subsets of size s of them.
3. Verify for each subset that it does not cover a clause.

Our real goal is:

$W[1] \subseteq$ antimonotone-$W[1, 2]$

Theorem

$L_{Antimonoton\text{-}\mathcal{F}(s)}$ can be reduced to Short Turing Machine Acceptance be a parameterized reduction.

Corollary

antimonotone-$W[1,s] \subseteq$ antimonotone-$W[1,2]$

Proof

Construct a turing machine that works as follows:

1. Guess k variables onto the tape.
2. Visit all subsets of size s of them.
3. Verify for each subset that it does not cover a clause.

Our real goal is:

$W[1] \subseteq$ antimonotone-$W[1,2]$

Theorem

$L_{Antimonoton\text{-}\mathcal{F}(s)}$ can be reduced to Short Turing Machine Acceptance be a parameterized reduction.

Corollary

antimonotone-$W[1, s] \subseteq$ *antimonotone-*$W[1, 2]$

Proof

Construct a turing machine that works as follows:

1. Guess k variables onto the tape.
2. Visit all subsets of size s of them.
3. Verify for each subset that it does not cover a clause.

Our real goal is:

$W[1] \subseteq \text{antimonotone-}W[1, 2]$

Theorem

$L_{Antimonoton\text{-}\mathcal{F}(s)}$ can be reduced to Short Turing Machine Acceptance be a parameterized reduction.

Corollary

antimonoton $1, 2]$

Question:

What is the running time?

Proof

Construct a turing machine that works as follows:

1. Guess k variables onto the tape.
2. Visit all subsets of size s of them.
3. Verify for each subset that it does not cover a clause.

Our real goal is:

$W[1] \subseteq$ antimonotone-$W[1, 2]$

Theorem

$L_{Antimonoton\text{-}\mathcal{F}(s)}$ can be reduced to Short Turing Machine Acceptance be a parameterized reduction.

Corollary

antimonotone-$W[1,s] \subseteq$ antimonotone-$W[1,2]$

Proof

Construct a turing machine that works as follows:

1. Guess k variables onto the tape.
2. Visit all subsets of size s of them.
3. Verify for each subset that it does not cover a clause.

Our real goal is:

$W[1] \subseteq$ antimonotone-$W[1,2]$

The class $W[1,1,s]$

Now we consider the weighted satisfiability problem for different, but still very simple circuits.

Definition

Let $s > 1$ be a number. By $\mathcal{F}(1,1,s)$ we denote the family of all circuits whose output is an OR-gate connected to AND-gates that are connected to OR-gates with indegree at most s.

By $W[1,1,s]$ we denote the class of problems in $W[1]$ that can be reduced to $L_{\mathcal{F}(1,1,s)}$ by a parameterized reduction.

Simplification of Weft-1-Circuits

Theorem

Consider a circuit of weft 1 and height h.
Then we can construct an equivalent circuit $\mathcal{F}(1, 1, s)$ in polynomial time, where s depends only on h.

Proof

- DNF und CNF
- de Morgan
- Combination of gates of same type
- Distributive law

Simplification of Weft-1-Circuits

Theorem

Consider a circuit of weft 1 and height h.
Then we can construct an equivalent circuit $\mathcal{F}(1, 1, s)$ in polynomial time, where s depends only on h.

Proof

- DNF und CNF
- de Morgan
- Combination of gates of same type
- Distributive law

$L_{\mathcal{F}(1,1,s)}$

Goal: Reduce $L_{\mathcal{F}(1,1,s)}$ to STMA.

Intermediate step:

Reduce $L_{\mathcal{F}(1,1,s)}$ to one TM M_i for all subcircuits below the output gate.

One TM guesses whicht M_i will be used for the simulation.

Still not known:

How to reduce $L_{\mathcal{F}(1,s)}$ to STMA.

$L_{\mathcal{F}(1,1,s)}$

Goal: Reduce $L_{\mathcal{F}(1,1,s)}$ to STMA.

Intermediate step:

Reduce $L_{\mathcal{F}(1,1,s)}$ to one TM M_i for all subcircuits below the output gate.

One TM guesses whicht M_i will be used for the simulation.

Still not known:

How to reduce $L_{\mathcal{F}(1,s)}$ to STMA.

Reduction of $L_{\mathcal{F}(1,s)}$ to STMA:

Construct a TM that guesses an assignment on the tape and then computes two numbers:

- $A =$ the number of clauses that are satisfied by negated variables.
- $M =$ the number of clauses that are satisfied only by positive literals.

Assignment is satisfying iff $A + M =$ number of clauses.

Reduction of $L_{\mathcal{F}(1,s)}$ to STMA:

Construct a TM that guesses an assignment on the tape and then computes two numbers:

- $A =$ the number of clauses that are satisfied by negated variables.
- $M =$ the number of clauses that are satisfied only by positive literals.

Assignment is satisfying iff $A + M =$ number of clauses.

Reduction of $L_{\mathcal{F}(1,s)}$ to STMA:

Construct a TM that guesses an assignment on the tape and then computes two numbers:

- $A =$ the number of clauses that are satisfied by negated variables.
- $M =$ the number of clauses that are satisfied only by positive literals.

Assignment is satisfying iff $A + M =$ number of clauses.

Reduction of $L_{\mathcal{F}(1,s)}$ to STMA:

Construct a TM that guesses an assignment on the tape and then computes two numbers:

- $A =$ the number of clauses that are satisfied by negated variables.
- $M =$ the number of clauses that are satisfied only by positive literals.

Assignment is satisfying iff $A + M =$ number of clauses.