Parameterized Reductions

Look at some classical reductions:

- Vertex Cover ot Independent Set
- CNF-SAT to 3SAT (weighted)
- Clique to Independent Set

Classical reductions are usually not parameterized reductions.

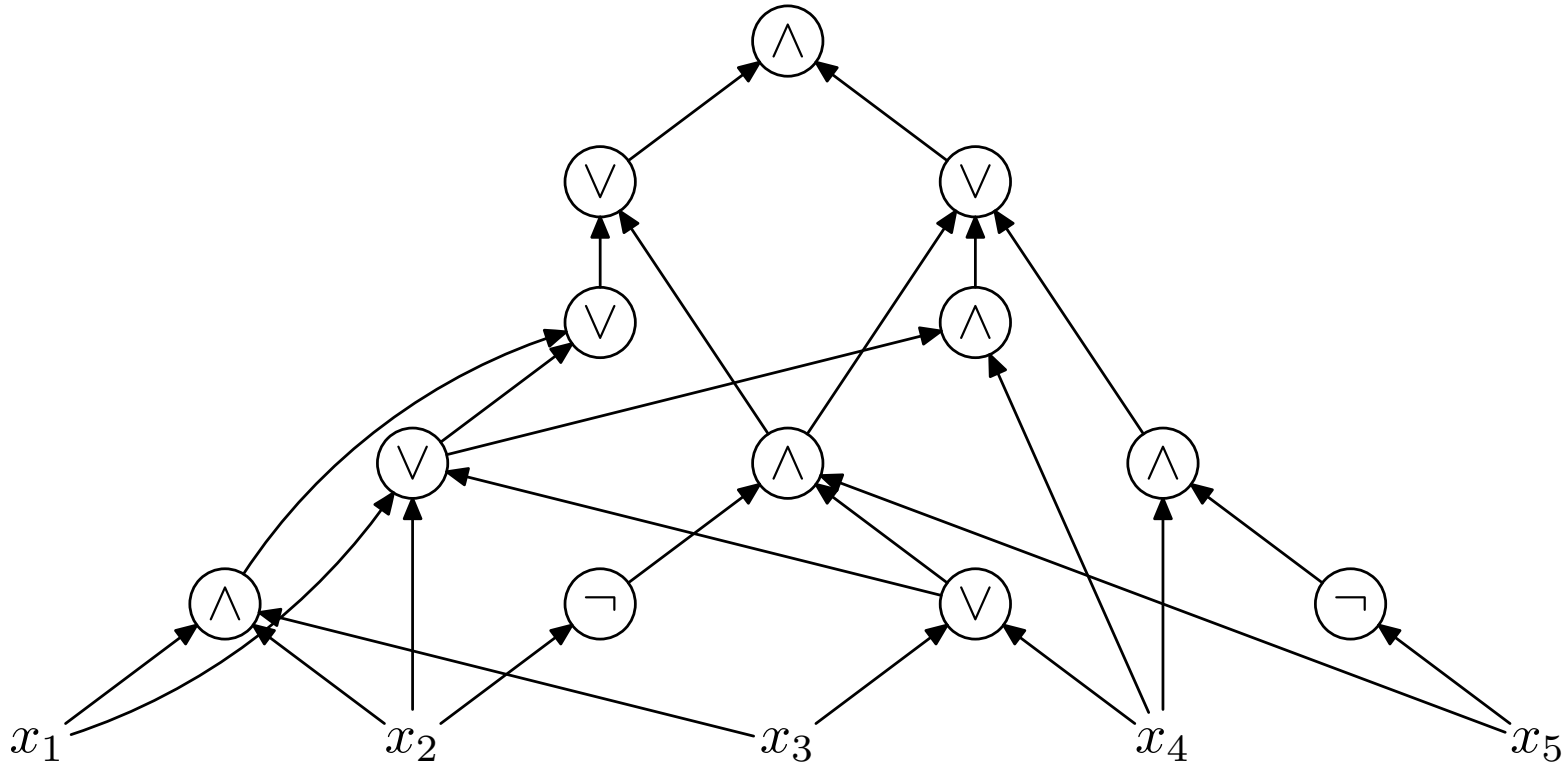
Parameterized Complexity Theory

Definition

A boolean circuit is an acyclic graph such that:

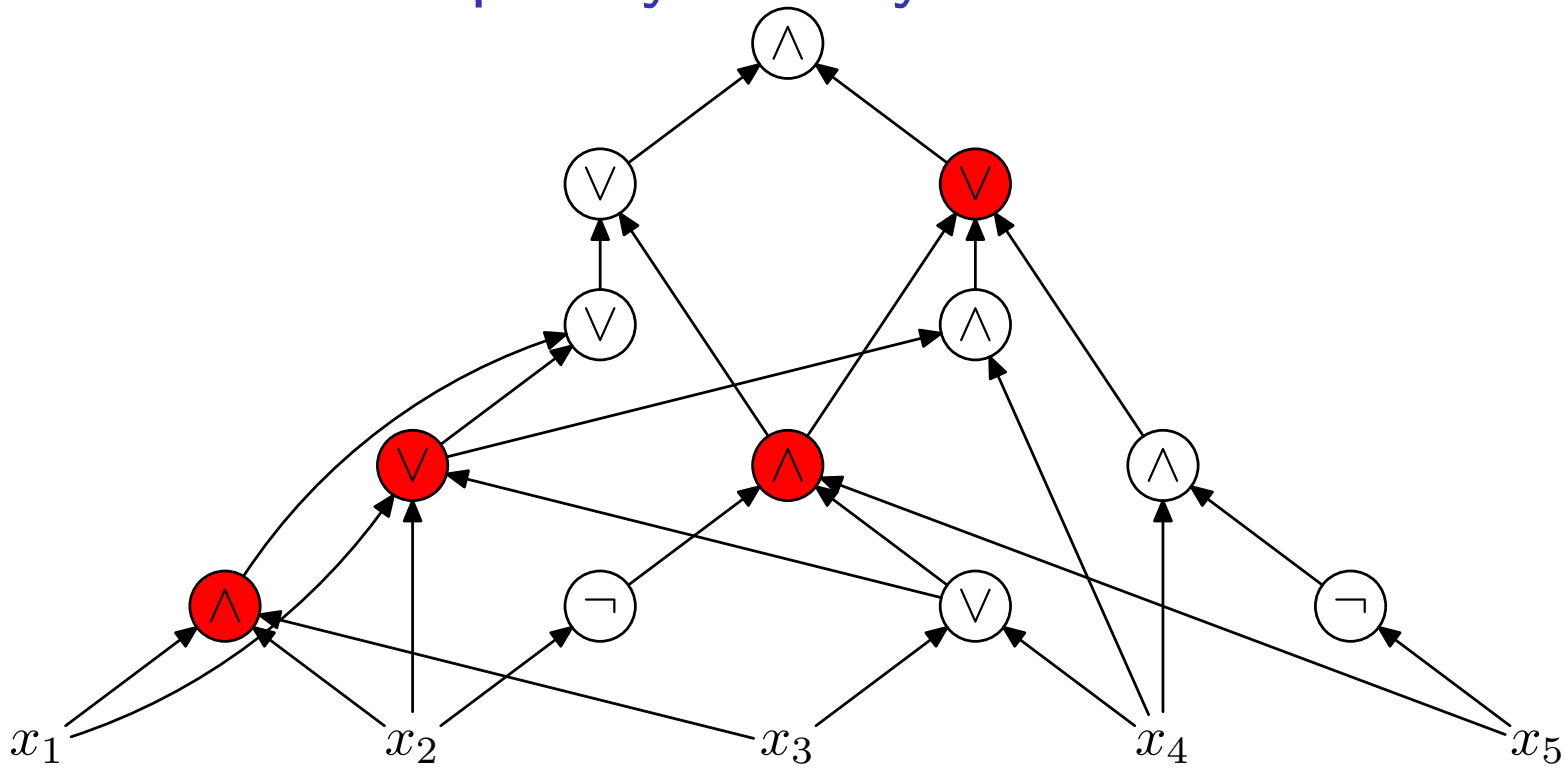
- There is exactly one vertex with outdegree 0, the Output.
- Every vertex with indegree 0 is an Input and labeled by x_i or $\neg x_i$.
- All other vertices are gates and labeled by $\wedge$, $\vee$ or $\neg$ (with indegree 1).

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 problemm 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 $L_{\mathcal{F}(2)}$.

Wir müssen M, w, k auf ein Schaltnetz und eine Zahl $k' = f(k)$ abbilden, so daß es eine erfüllende Belegung mit Gewicht k' genau dann gibt, wenn M das Wort w in höchstens k Schritten akzeptiert.

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 $L_{\mathcal{F}(2)}$.

Wir müssen M, w, k auf ein Schaltnetz und eine Zahl $k' = f(k)$ abbilden, so daß es eine erfüllende Belegung mit Gewicht k' genau dann gibt, wenn M das Wort w in höchstens k Schritten akzeptiert.

Wir können uns die Konfigurationen von M numeriert vorstellen und daher mit den Zahlen $1, 2, \ldots$ identifizieren.

Ein Konfiguration umfaßt dabei

- die Position des Schreib-/Lesekopfes,
- und den Zustand.

Falls i eine Konfiguration und a ein Zeichen ist, dann sei $\delta(i, a) = (j, b)$, wenn M beim Lesen von a in die Konfiguration j wechselt und das a durch ein b überschreibt.

Mit der Variablen $C_{t,i,j,a,b}$ wollen wir modellieren, daß M im t-ten Schritt von der Konfiguration i nach j wechselt, dabei ein a liest und durch ein b überschreibt.

Wir können uns die Konfigurationen von M numeriert vorstellen und daher mit den Zahlen $1, 2, \ldots$ identifizieren.

Ein Konfiguration umfaßt dabei

- die Position des Schreib-/Lesekopfes,
- und den Zustand.

Falls i eine Konfiguration und a ein Zeichen ist, dann sei $\delta(i, a) = (j, b)$, wenn M beim Lesen von a in die Konfiguration j wechselt und das a durch ein b überschreibt.

Mit der Variablen $C_{t,i,j,a,b}$ wollen wir modellieren, daß M im t-ten Schritt von der Konfiguration i nach j wechselt, dabei ein a liest und durch ein b überschreibt.

Wir können uns die Konfigurationen von M numeriert vorstellen und daher mit den Zahlen $1, 2, \ldots$ identifizieren.

Ein Konfiguration umfaßt dabei

- die Position des Schreib-/Lesekopfes,
- und den Zustand.

Falls i eine Konfiguration und a ein Zeichen ist, dann sei $\delta(i, a) = (j, b)$, wenn M beim Lesen von a in die Konfiguration j wechselt und das a durch ein b überschreibt.

Mit der Variablen $C_{t,i,j,a,b}$ wollen wir modellieren, daß M im t-ten Schritt von der Konfiguration i nach j wechselt, dabei ein a liest und durch ein b überschreibt.

Wir können uns die Konfigurationen von M numeriert vorstellen und daher mit den Zahlen $1, 2, \ldots$ identifizieren.

Ein Konfiguration umfaßt dabei

- die Position des Schreib-/Lesekopfes,
- und den Zustand.

Falls i eine Konfiguration und a ein Zeichen ist, dann sei $\delta(i, a) = (j, b)$, wenn M beim Lesen von a in die Konfiguration j wechselt und das a durch ein b überschreibt.

Mit der Variablen $C_{t,i,j,a,b}$ wollen wir modellieren, daß M im t-ten Schritt von der Konfiguration i nach j wechselt, dabei ein a liest und durch ein b überschreibt.

Mit der Variablen $M_{t,p,a,b}$ wollen wir modellieren, daß am Anfang des t-ten Schritts auf der Position p des Arbeitsbandes das Symbol a steht und in diesem Schritt durch ein b überschrieben wird.

Das nächste Ziel besteht nun darin, eine Formel so anzugeben, daß erfüllende Belegungen Berechnungen der Turingmaschine M widerspiegeln.

Das heißt, es gibt genau dann eine erfüllende Belegung mit $C_{t,i,j,a,b} = 1$, wenn es eine Berechnung gibt, in welcher M im t-ten Schritt von der Konfiguration i nach j wechselt, dabei ein a liest und durch ein b überschreibt.

Jeder mögliche Berechnungspfad soll dabei genau einer erfüllenden Belegung entsprechen.

Ausserdem: Es soll nur Berechnungspfade der Länge k und erfüllende Belegungen mit Gewicht $f(k)$ geben!

Mit der Variablen $M_{t,p,a,b}$ wollen wir modellieren, daß am Anfang des t-ten Schritts auf der Position p des Arbeitsbandes das Symbol a steht und in diesem Schritt durch ein b überschrieben wird.

Das nächste Ziel besteht nun darin, eine Formel so anzugeben, daß erfüllende Belegungen Berechnungen der Turingmaschine M widerspiegeln.

Das heißt, es gibt genau dann eine erfüllende Belegung mit $C_{t,i,j,a,b} = 1$, wenn es eine Berechnung gibt, in welcher M im t-ten Schritt von der Konfiguration i nach j wechselt, dabei ein a liest und durch ein b überschreibt.

Jeder mögliche Berechnungspfad soll dabei genau einer erfüllenden Belegung entsprechen.

Ausserdem: Es soll nur Berechnungspfade der Länge k und erfüllende Belegungen mit Gewicht $f(k)$ geben!

Es gibt sehr viele Bedingungen an unsere Variablen, damit die Modellierung stimmt.

Wir wollen diese durch ein UND von ODERs ausdrücken.

Die Klauseln werden jetzt so gewählt, daß eine falsche Modellierung unmöglich wird, weil sie sofort zu einer unerfüllenden Belegung führt.

Es bleiben dann als erfüllende Belegungen genau diejenigen übrig, welche gegen keine Regel verstossen.

Regel 1:

„Es gibt nichts doppelt, da es ein Berechnungspfad ist“

Genauer: Zum Zeitpunkt t ist M in genau einem Zustand, wechselt zu genau einem anderen, liest dabei genau ein Zeichen und überschreibt dieses durch genau ein (anderes) Zeichen.

Wie läßt sich diese Tatsache durch Klauseln darstellen?

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

beziehungsweise

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

für alle $t, i, j, a, b, i', j', a', b'$ mit $(i, j, a, b) \neq (i', j', a', b')$ und

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

für alle t, p, a, b, a', b' mit $(a, b) \neq (a', b')$.

Regel 1:

„Es gibt nichts doppelt, da es ein Berechnungspfad ist“

Genauer: Zum Zeitpunkt t ist M in genau einem Zustand, wechselt zu genau einem anderen, liest dabei genau ein Zeichen und überschreibt dieses durch genau ein (anderes) Zeichen.

Wie läßt sich diese Tatsache durch Klauseln darstellen?

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

beziehungsweise

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

für alle $t, i, j, a, b, i', j', a', b'$ mit $(i, j, a, b) \neq (i', j', a', b')$ und

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

für alle t, p, a, b, a', b' mit $(a, b) \neq (a', b')$.

Regel 1:

„Es gibt nichts doppelt, da es ein Berechnungspfad ist“

Genauer: Zum Zeitpunkt t ist M in genau einem Zustand, wechselt zu genau einem anderen, liest dabei genau ein Zeichen und überschreibt dieses durch genau ein (anderes) Zeichen.

Wie läßt sich diese Tatsache durch Klauseln darstellen?

Frage:

Warum nur eine Möglichkeit? Es geht doch um nichtdeterministische Turingmaschinen?

beziehungsw

für alle $t, i, j, a, b, i', j', a', b'$ mit $(i, j, a, b) \neq (i', j', a', b')$ und

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

für alle t, p, a, b, a', b' mit $(a, b) \neq (a', b')$.

Regel 2:

„M und C muß zusammenpassen.“

Genauer: Wenn ein Zeichen gelesen wird, dann muß es dort auch stehen. Wird ein Zeichen irgendwohin geschrieben, muß es danach dort auch stehen.

Wie läßt sich diese Tatsache durch Klauseln darstellen?

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

Für alle t, i, j, a, b, wobei $p(i)$ die Kopfposition der Konfiguration i bezeichnet.

Regel 2:

„M und C muß zusammenpassen.“

Genauer: Wenn ein Zeichen gelesen wird, dann muß es dort auch stehen. Wird ein Zeichen irgendwohin geschrieben, muß es danach dort auch stehen.

Wie läßt sich diese Tatsache durch Klauseln darstellen?

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

Für alle t, i, j, a, b, wobei $p(i)$ die Kopfposition der Konfiguration i bezeichnet.

Regel 2:

„M und C muß zusammenpassen.“

Genauer: Wenn ein Zeichen gelesen wird, dann muß es dort auch stehen. Wird es danach dort auch ste

Wie läßt sich

Frage:

Benötigen wir hier eine Art Rückrichtung?

„Falls ein Zeichen irgendwo steht, dann muß es auch gelesen werden.“

Für alle t, i, j, a, b, wobei $p(i)$ die Kopfposition der Konfiguration i bezeichnet.

Regel 3:

„Aufeinanderfolgende Schritte müssen zusammenpassen.“

Genauer: Beende ich einen Schritt in einer bestimmten Konfiguration, dann muß der nächste Schritt dort auch wieder beginnen. Der Inhalt einer Zelle nach dem t-ten und vor dem $t+1$-ten Schritt ist identisch.

Wie läßt sich diese Tatsache durch Klauseln darstellen?

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

Für alle $t, i, j, i', j' a, b, c, d$ mit $i' \neq j$ und

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

für alle t, p, a, b, c, d mit $b \neq c$.

Regel 3:

„Aufeinanderfolgende Schritte müssen zusammenpassen.“

Genauer: Beende ich einen Schritt in einer bestimmten Konfiguration, dann muß der nächste Schritt dort auch wieder beginnen. Der Inhalt einer Zelle nach dem t-ten und vor dem $t+1$-ten Schritt ist identisch.

Wie läßt sich diese Tatsache durch Klauseln darstellen?

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

Für alle $t, i, j, i', j' a, b, c, d$ mit $i' \neq j$ und

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

für alle t, p, a, b, c, d mit $b \neq c$.

Regel 4:

„Der Anfang und das Ende müssen stimmen. Der Berechnungspfad muß akzeptierend sein.“

$\rightarrow$ Übungsaufgabe

Da alle Regeln gleichzeitig erfüllt werden müssen, können wir sie durch ein riesiges UND verbinde.

Das ergibt eine $\mathcal{F}(2)$-Formel, wie gewünscht.

Es gibt einen akzeptierenden Berechnungspfad der Länge k genau dann, wenn es eine erfüllende Belegung des Gewichts k' gibt.

Regel 4:

„Der Anfang und das Ende müssen stimmen. Der Berechnungspfad muß akzeptierend sein.“

$\rightarrow$ Übungsaufgabe

Da alle Regeln gleichzeitig erfüllt werden müssen, können wir sie durch ein riesiges UND verbinde.

Das ergibt eine $\mathcal{F}(2)$-Formel, wie gewünscht.

Es gibt einen akzeptierenden Berechnungspfad der Länge k genau dann, wenn es eine erfüllende Belegung des Gewichts k' gibt.

Regel 4:

„Der Anfang und das Ende müssen stimmen. Der Berechnungspfad muß akzeptierend sein.“

→ Übungsaufgabe

Da alle Regeln gleichzeitig erfüllt werden müssen, können wir sie durch

Frage:

Wie groß ist k'?

Das er wünscht.

Es gibt einen akzeptierenden Berechnungspfad der Länge k genau dann, wenn es eine erfüllende Belegung des Gewichts k' gibt.

Zur Erinnerung.

Wir haben gerade folgendes bewiesen:

Theorem

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