Mechanisation of SETH $\Longrightarrow$ OVH

or proving a conditional lower bound in P using the extraction-framework

Sergey Bozhko

Advisors: Prof. Gert Smolka and Fabian Kunze

20.03.2020

UNIVERSITÄT DES SAARLANDES

Conditional lower bounds

- Ideally, for any problem A we would like to
  - Find: an algorithm with running time $\mathcal{O}(t(n))$
  - Prove: any algorithm solving A has runtime $\Omega(t(n))$

- Real world is quire different
- We have **no** techniques for proving **unconditional** lower bounds
  - We can prove that SAT $\in \Omega(n^2)$

- However, we know how to prove **conditional** lower bounds
  - Let $\mathrm{P} \neq \mathrm{NP}$, then Clique has no poly-time algorithm
  - Many others: 3SUM, APSP, ETH, SETH, Clique $\notin$ FPT,…

Conditional lower bounds

- Ideally, for any problem A we would like to
  - Find: an algorithm with running time $\mathcal{O}(t(n))$
  - Prove: any algorithm solving A has runtime $\Omega(t(n))$

- Real world is quire different
- We have **no** techniques for proving **unconditional** lower bounds
  - We can prove that $\mathsf{SAT} \in \Omega(n^2)$

- However, we know how to prove **conditional** lower bounds
  - Let $\mathsf{P} \neq \mathsf{NP}$, then Clique has no poly-time algorithm
  - Many others: $\mathsf{3SUM}$, APSP, ETH, SETH, $\mathsf{Clique} \notin \mathsf{FPT}, \ldots$

Strong Exponential Time Hypothesis (SETH)

- Algorithms for solving kSAT
  - Trivial algorithm: kSAT $\in \mathcal{O}(2^n \cdot \text{poly}(size))$
  - Paruti: kSAT $\in \mathcal{O}(2^{(1-\mu_k/(k-1))n} \cdot \text{poly}(size))$
  - Impagliazzo: kSAT $\in \mathcal{O}(2^{(1-d/k)n} \cdot \text{poly}(size))$

- With big k the runtime tends towards $\mathcal{O}(2^n \cdot \text{poly}(size))$
  - E.g., $\exists k$, kSAT $\notin \mathcal{O}(2^{0.9999 \cdot n} \cdot \text{poly}(size))$

- Strong Exponential Time Hypothesis (SETH)
  - There is no "fast" algorithm for kSAT
  - Formally: $\forall \varepsilon \in \mathbb{R} : \varepsilon > 0, \exists k,$ kSAT $\notin \mathcal{O}(2^{(1-\varepsilon)\cdot n} \cdot \text{poly}(size))$

Orthogonal Vectors Problem (OV)

- Problem: Given two sets of vectors $A, B \in \mathbb{N}^{d \times n}$, we are to decide whether there are $a \in A$ and $b \in B$ such that $\langle a, b \rangle = 0$

- Given $a, b \in \mathbb{N}^d$, define the dot product of vectors as follows: $\langle a, b \rangle = \sum_{i=0}^{d-1} a_i \cdot b_i$

0	2	1	3	0	1
3	0	0	1	1	0
7	0	1	3	4	2
0	7	1	0	2	2
2	0	0	3	7	0
5	1	5	4	3	1
3	0	8	6	0	3

7	0	2	0	2	0
0	4	1	4	5	2
5	3	0	7	3	2
3	0	6	5	0	0
7	1	6	0	4	1
0	0	1	4	1	1
2	3	3	0	4	3

Orthogonal Vectors Problem (OV)

- Naive algorithm: OV $\in \mathcal{O}(n^2 \cdot d)$

- Question: can we solve OV faster?
  - Is OV $\in \mathcal{O}(n^{1.9} \cdot d)$?
  - Is OV $\in \mathcal{O}(n^{1.999} \cdot d^2)$?
  - Or maybe OV $\in \mathcal{O}(n^{1.9999} \cdot d^{100})$?

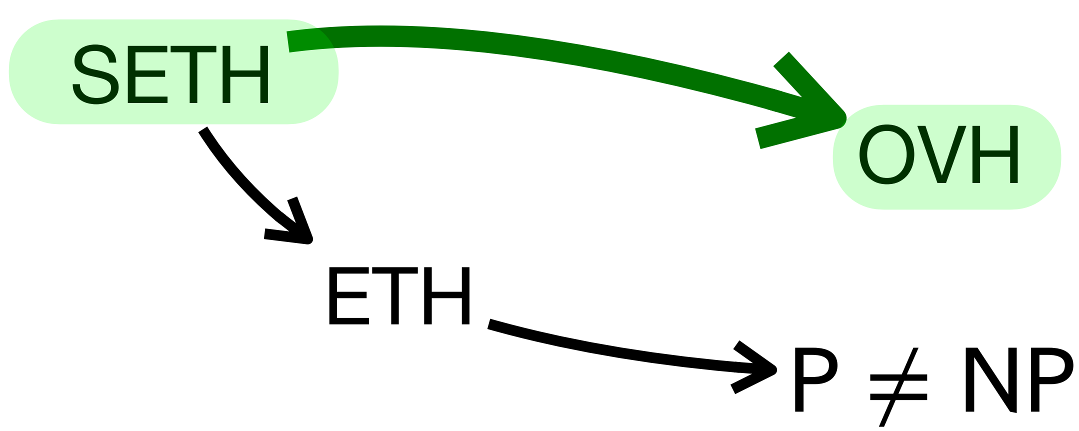
0	2	1	3	0	1
3	0	0	1	1	0
7	0	1	3	4	2
0	7	1	0	2	2
2	0	0	3	7	0
5	1	5	4	3	1
3	0	8	6	0	3

7	0	2	0	2	0
0	4	1	4	5	2
5	3	0	7	3	2
3	0	6	5	0	0
7	1	6	0	4	1
0	0	1	4	1	1
2	3	3	0	4	3

Orthogonal Vectors Hypothesis (OVH)

- Orthogonal Vectors Hypothesis (OVH):
  - There is **no** “fast” algorithm for OV
  - That is, $\forall \varepsilon > 0 \wedge \text{OV} \notin \mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$

- Today’s theorem: if SETH holds, then OVH holds



Proof via Reduction

$$\text{SETH} \implies \text{OVH}$$

- Proof idea:

  Assuming a "fast" algorithm ov for OV,

  construct a "fast" algorithm ksat for kSAT
- Map an instance of kSAT to an instance of OV
  - Define a reduction $f : \mathsf{kSAT} \mapsto \mathsf{OV}$
  - Prove its correctness $\psi \in \mathsf{kSAT} \iff f(\psi) \in \mathsf{OV}$
  - Prove runtime bound for f
- Combine reduction and the algorithm for OV:

  $\mathsf{ksat} := \mathsf{ov} \circ f$

Remarks

$$\text{SETH} \implies \text{OVH}$$

- Assumption about NP, implication about P
- The reduction is in exp-time, which blows-up an instance exponentially
- Requires tricky work with complexity analysis
- The theorem is quite recent [1]

[1]: Ryan Williams. “A new algorithm for optimal 2-constraint satisfaction and its implications”. 2005

Contributions

- Define reduction and prove its correctness
- Prove runtime bound on the reduction
- Combine previous steps to obtain* $\mathsf{SETH} \implies \mathsf{OVH}$

*(up to some problems with the handling of $\mathbb{R}$)

Contributions

- Define reduction and prove its correctness
  - Pen&Paper proof is well-known
  - RIL: The proof is mechanised
- Prove runtime bound on the reduction
- Combine previous steps to obtain $\mathsf{SETH} \implies \mathsf{OVH}$

Reduction and its Correctness

- Given: a formula
$$\psi = (l_{1,1} \vee \ldots \vee l_{1,k}) \wedge \ldots \wedge (l_{m,1} \vee \ldots \vee l_{m,k})$$
  - N variables, M clauses, each clause has at most K literals

- Want: construct two sets A and B such that
$$\exists \alpha \models \psi \iff \exists a \in A, b \in B, \langle a, b \rangle = 0$$

Reduction and its Correctness

$$(l_{1,1} \vee \ldots \vee l_{1,k})$$
$$(l_{2,1} \vee \ldots \vee l_{2,k})$$
$$(l_{3,1} \vee \ldots \vee l_{3,k})$$
$$\ldots$$
$$(l_{m,1} \vee \ldots \vee l_{m,k})$$

Reduction and its Correctness

$$
\begin{gathered}
(l_{1,1} \vee \ldots \vee l_{1,k}) \\
(l_{2,1} \vee \ldots \vee l_{2,k}) \\
(l_{3,1} \vee \ldots \vee l_{3,k}) \\
\ldots \\
(l_{m,1} \vee \ldots \vee l_{m,k})
\end{gathered}
$$

$$\alpha = \{\boxed{x_1 \mapsto \bot, x_2 \mapsto \bot, \ldots, x_{n/2} \mapsto \bot}, x_{n/2+1} \mapsto ?, \ldots, x_n \mapsto ?\}$$

Partial assignment

$x \mapsto$ Some $\bot$ | Some $\top$ | None

Reduction and its Correctness

$$(l_{1,1} \lor \ldots \lor l_{1,k}) \xrightarrow{\alpha} \top$$

$$(l_{2,1} \lor \ldots \lor l_{2,k}) \xrightarrow{\alpha} \bot$$

$$(l_{3,1} \lor \ldots \lor l_{3,k}) \xrightarrow{\alpha} ?$$

$$\ldots \qquad \ldots$$

$$(l_{m,1} \lor \ldots \lor l_{m,k}) \xrightarrow{\alpha} \bot$$

$$\alpha = \{\boxed{x_1 \mapsto \bot, x_2 \mapsto \bot, \ldots, x_{n/2} \mapsto \bot}, x_{n/2+1} \mapsto ?, \ldots, x_n \mapsto ?\}$$

Partial assignment

$x \mapsto$ Some $\bot$ | Some $\top$ | None

Reduction and its Correctness

$$
\begin{array}{cccccc}
(l_{1,1} \lor \ldots \lor l_{1,k}) & \xrightarrow{\alpha} & \top & \longrightarrow & 0 \\
(l_{2,1} \lor \ldots \lor l_{2,k}) & \xrightarrow{\alpha} & \bot & \longrightarrow & 1 \\
(l_{3,1} \lor \ldots \lor l_{3,k}) & \xrightarrow{\alpha} & ? & \longrightarrow & 1 \\
\ldots & \ldots & & \ldots & \ldots \\
(l_{m,1} \lor \ldots \lor l_{m,k}) & \xrightarrow{\alpha} & \bot & \longrightarrow & 1
\end{array}
$$

$$\alpha = \{x_1 \mapsto \bot, x_2 \mapsto \bot, \ldots, x_{n/2} \mapsto \bot, x_{n/2+1} \mapsto ?, \ldots, x_n \mapsto ?\}$$

Partial assignment

$x \mapsto$ Some $\bot$ | Some $\top$ | None

Reduction and its Correctness

Goes to A

$$\begin{array}{ccccc}
(l_{1,1} \vee \ldots \vee l_{1,k}) & \xrightarrow{\quad\alpha\quad} & \top & \xrightarrow{\qquad} & 0 \\
(l_{2,1} \vee \ldots \vee l_{2,k}) & \xrightarrow{\quad\alpha\quad} & \bot & \xrightarrow{\qquad} & 1 \\
(l_{3,1} \vee \ldots \vee l_{3,k}) & \xrightarrow{\quad\alpha\quad} & ? & \xrightarrow{\qquad} & 1 \\
\ldots & \ldots & & \ldots & \ldots \\
(l_{m,1} \vee \ldots \vee l_{m,k}) & \xrightarrow{\quad\alpha\quad} & \bot & \xrightarrow{\qquad} & 1
\end{array}$$

$$\alpha = \{x_1 \mapsto \bot, x_2 \mapsto \bot, \ldots, x_{n/2} \mapsto \bot, x_{n/2+1} \mapsto ?, \ldots, x_n \mapsto ?\}$$

Partial assignment

$x \mapsto$ Some $\bot$ | Some $\top$ | None

Reduction and its Correctness

Goes to A

$$
\begin{array}{cccc}
(l_{1,1} \lor \ldots \lor l_{1,k}) & \xrightarrow{\alpha} \top & \longrightarrow & 0 \\
(l_{2,1} \lor \ldots \lor l_{2,k}) & \xrightarrow{\alpha} \bot & \longrightarrow & 1 \\
(l_{3,1} \lor \ldots \lor l_{3,k}) & \xrightarrow{\alpha} \top & \longrightarrow & 0 \\
\ldots & \ldots & \ldots & \ldots \\
(l_{m,1} \lor \ldots \lor l_{m,k}) & \xrightarrow{\alpha} ? & \longrightarrow & 1
\end{array}
$$

$$\alpha = \{\boxed{x_1 \mapsto \top, x_2 \mapsto \bot, \ldots, x_{n/2} \mapsto \bot}, x_{n/2+1} \mapsto ?, \ldots, x_n \mapsto ?\}$$

Reduction and its Correctness

Goes to A

$$\begin{array}{ccccc}
(l_{1,1} \lor \ldots \lor l_{1,k}) & \xrightarrow{\alpha} & \top & \longrightarrow & 0 \\
(l_{2,1} \lor \ldots \lor l_{2,k}) & \xrightarrow{\alpha} & \bot & \longrightarrow & 1 \\
(l_{3,1} \lor \ldots \lor l_{3,k}) & \xrightarrow{\alpha} & ? & \longrightarrow & 1 \\
\ldots & \ldots & & \ldots & \ldots \\
(l_{m,1} \lor \ldots \lor l_{m,k}) & \xrightarrow{\alpha} & \bot & \longrightarrow & 1
\end{array}$$

$$\alpha = \{x_1 \mapsto \bot, x_2 \mapsto \top, \ldots, x_{n/2} \mapsto \bot, x_{n/2+1} \mapsto ?, \ldots, x_n \mapsto ?\}$$

Reduction and its Correctness

Goes to A

$$\begin{array}{ccccc}
(l_{1,1} \vee \dots \vee l_{1,k}) & \xrightarrow{\alpha} & \top & \longrightarrow & 0 \\
(l_{2,1} \vee \dots \vee l_{2,k}) & \xrightarrow{\alpha} & ? & \longrightarrow & 1 \\
(l_{3,1} \vee \dots \vee l_{3,k}) & \xrightarrow{\alpha} & \bot & \longrightarrow & 1 \\
\dots & \dots & & \dots & \dots \\
(l_{m,1} \vee \dots \vee l_{m,k}) & \xrightarrow{\alpha} & \bot & \longrightarrow & 1
\end{array}$$

A contains $2^{N/2}$ vectors

$$\alpha = \{x_1 \mapsto \top, x_2 \mapsto \top, \dots, x_{n/2} \mapsto \top, x_{n/2+1} \mapsto ?, \dots, x_n \mapsto ?\}$$

Repeat this $2^{N/2}$ times

Reduction and its Correctness

Goes to B

$$\begin{array}{ccccc}
(l_{1,1} \vee \ldots \vee l_{1,k}) & \xrightarrow{\alpha} & \top & \longrightarrow & 0 \\
(l_{2,1} \vee \ldots \vee l_{2,k}) & \xrightarrow{\alpha} & \perp & \longrightarrow & 1 \\
(l_{3,1} \vee \ldots \vee l_{3,k}) & \xrightarrow{\alpha} & ? & \longrightarrow & 1 \\
\ldots & \ldots & & \ldots & \ldots \\
(l_{m,1} \vee \ldots \vee l_{m,k}) & \xrightarrow{\alpha} & \perp & \longrightarrow & 1
\end{array}$$

$$\alpha = \{x_1 \mapsto ?, \ldots, x_{n/2} \mapsto ?, x_{n/2+1} \mapsto \perp, x_{n/2+2} \mapsto \perp, \ldots, x_n \mapsto \perp\}$$

Reduction and its Correctness

$$
\begin{array}{cccc}
(l_{1,1} \vee \ldots \vee l_{1,k}) & \xrightarrow{\alpha} \top \longrightarrow & 0 \\
(l_{2,1} \vee \ldots \vee l_{2,k}) & \xrightarrow{\alpha} \perp \longrightarrow & 1 \\
(l_{3,1} \vee \ldots \vee l_{3,k}) & \xrightarrow{\alpha} \ ? \ \longrightarrow & 1 \\
\ldots & \ldots \qquad \ldots & \ldots \\
(l_{m,1} \vee \ldots \vee l_{m,k}) & \xrightarrow{\alpha} \perp \longrightarrow & 1
\end{array}
$$

Goes to B

B contains $2^{N/2}$ vectors

$$\alpha = \{x_1 \mapsto ?, \ldots, x_{n/2} \mapsto ?, x_{n/2+1} \mapsto \perp, x_{n/2+2} \mapsto \perp, \ldots, x_n \mapsto \perp\}$$

Repeat this $2^{N/2}$ times

Reduction and its Correctness

$f : \psi \mapsto (A, B)$ such that $\exists \alpha \vDash \psi \iff \exists a \in A, b \in B, \langle a, b \rangle = 0$

For ψ with (N, M, K), f generates an instance $(n := 2^{N/2}, d := M)$

Set A

M

0	1	1	0	0	1	1	0	1	1	1
1	0	0	1	1	0	0	1	0	0	0
1	0	1	1	1	0	1	0	1	0	0
0	0	1	0	0	1	0	1	0	1	1
1	0	0	0	1	0	0	1	0	1	1

$2^{N/2}$

Set B

M

0	1	1	0	0	1	1	0	1	1	1
1	0	0	1	1	1	0	1	0	0	0
1	0	1	1	1	0	1	1	1	1	0
0	0	0	0	0	1	0	1	0	1	1
1	0	0	0	1	0	0	1	0	1	1

$2^{N/2}$

Contributions

- Define reduction and prove its correctness ✓
- Prove runtime bound on the reduction
  - Pen&Paper proof is well-known
  - RIL: O-Notation is introduced
  - RIL: The proof is mechanised
- Combine previous steps to obtain $\mathsf{SETH} \implies \mathsf{OVH}$

Complexity of the Reduction

- The reduction is defined ✓
- Want to prove: $f \in \mathcal{O}(2^{N/2}) \cdot \text{poly(size)}$
- That is, we need the runtime-bound in terms of N, M, K:

```
time_reduction (ψ : cnf) :=
  k0 * 2^(N/2) * N^k1 * M^k2 * K^k3
```

Complexity of the Reduction

- The reduction is defined ✓
- Want to prove: $f \in \mathcal{O}(2^{N/2}) \cdot \text{poly}(\text{size})$
- That is, we need the runtime-bound in terms of N, M, K:

```
time_reduction (ψ : cnf) :=
  k0 * 2^(N/2) * N^k1 * M^k2 * K^k3
```

- One cannot just run "`extract; solvrec`" from the extraction framework, since it gives bounds of the form:

```
time_reduction (ψ : cnf) := time_sizeN ψ +
time_sizeM ψ + time_reductionA ψ +
time_reductionB ψ + 19.
```

Complexity of the Reduction

- The reduction is defined ✓

- Want to prove: $f \in \mathcal{O}(2^{N/2}) \cdot \mathrm{poly}(\mathrm{size})$

- That is, we need the runtime-bound in terms of N, M, K:

```
time_reduction (ψ : cnf) :=
  k0 ∗ 2^(N/2) ∗ N^k1 ∗ M^k2 ∗ K^k3
```

- One cannot just run “`extract; solvrec`" from the extraction framework, since it gives bounds of the form:

```
time_reduction (ψ : cnf) := time_sizeN ψ +
time_sizeM ψ + time_reductionA ψ +
time_reductionB ψ + 19.
```

Complexity of the Reduction

To give a concrete example

Complexity of the Reduction

To give a concrete example

- Runtime for split_A function

Complexity of the Reduction

To give a concrete example

- Runtime for split_A function
- Extraction framework:

```
time_sizeN ψ + time_div2 (sizeN ψ) + time_cnf_varset ψ +
time_firstn (div2 (sizeN ψ)) (cnf_varset ψ) + 6
```

Complexity of the Reduction

To give a concrete example

- Runtime for split_A function
- Extraction framework:

```
time_sizeN ψ + time_div2 (sizeN ψ) + time_cnf_varset ψ +
time_firstn (div2 (sizeN ψ)) (cnf_varset ψ) + 6
```

- I can upper-bound each function call:

```
194 * (1 + sizeN ψ) * (1 + sizeK ψ)^2 * (1 + sizeM ψ)^2 +
(8 + 7*sizeN ψ) + 174 * (1 + sizeK ψ)^2 * (1 + sizeM ψ)^2
+ (8 + 17 * sizeN ψ) + 6
```

Complexity of the Reduction

To give a concrete example

- Runtime for $split_A$ function
- Extraction framework:

```
time_sizeN ψ + time_div2 (sizeN ψ) + time_cnf_varset ψ +
time_firstn (div2 (sizeN ψ)) (cnf_varset ψ) + 6
```

- I can upper-bound each function call:

```
194 ∗ (1 + sizeN ψ) ∗ (1 + sizeK ψ)^2 ∗ (1 + sizeM ψ)^2 +
(8 + 7∗sizeN ψ) + 174 ∗ (1 + sizeK ψ)^2 ∗ (1 + sizeM ψ)^2
+ (8 + 17 ∗ sizeN ψ) + 6
```

- In the end, it can be bounded by this monomial:

```
414 ∗ (1 + sizeN ψ) ∗ (1 + sizeK ψ)^2 ∗ (1 + sizeM ψ)^2
```

O-Notation

- Simple for 1-dimensional case:
  $f \in \mathcal{O}(g) := \exists a, N : \forall n > N : g(n) \leq a \cdot f(n)$

O-Notation

- Simple for 1-dimensional case:
  $f \in \mathcal{O}(g) := \exists a, N : \forall n > N : g(n) \leq a \cdot f(n)$

- Tricky for k-dimensional case; e.g., a naive generalisation does not satisfy all the natural properties
  $f \in \mathcal{O}(g) := \exists a, N : \forall n_1 > N, \ldots, n_k > N : g(n_1, \ldots, n_k) \leq a \cdot f(n_1, \ldots, n_k)$

There is a paper discussing this issue [1]

[1]: R. R. Howell, "On Asymptotic Notation with Multiple Variables", 2008

O-Notation

- Simple for 1-dimensional case:
  $f \in \mathcal{O}(g) := \exists a, N : \forall n > N : g(n) \leq a \cdot f(n)$

- Tricky for k-dimensional case; e.g., a naive generalisation does not satisfy all the natural properties
  $f \in \mathcal{O}(g) := \exists a, N : \forall n_1 > N, \ldots, n_k > N : g(n_1, \ldots, n_k) \leq a \cdot f(n_1, \ldots, n_k)$

- Not clear for a function $f : X \to Y$,
  where type X might have no order ($<$)

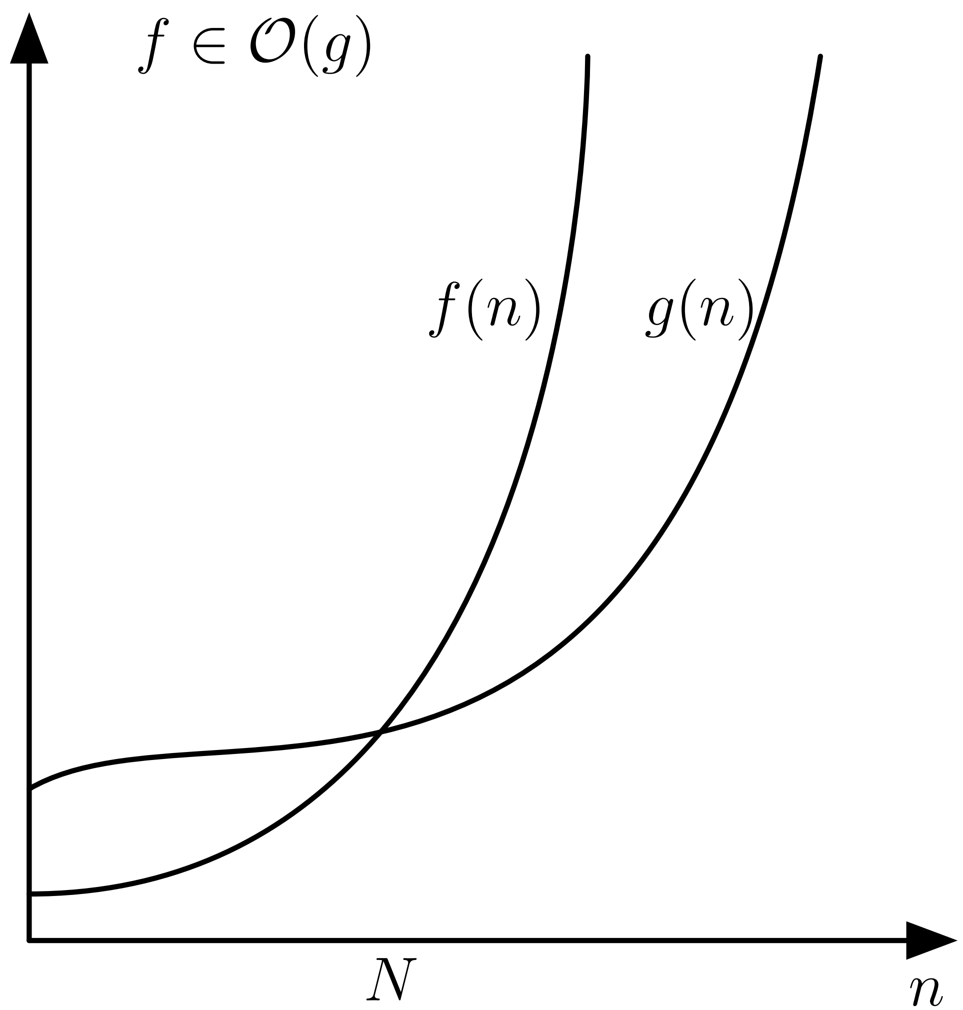
There is a paper discussing this issue [1]

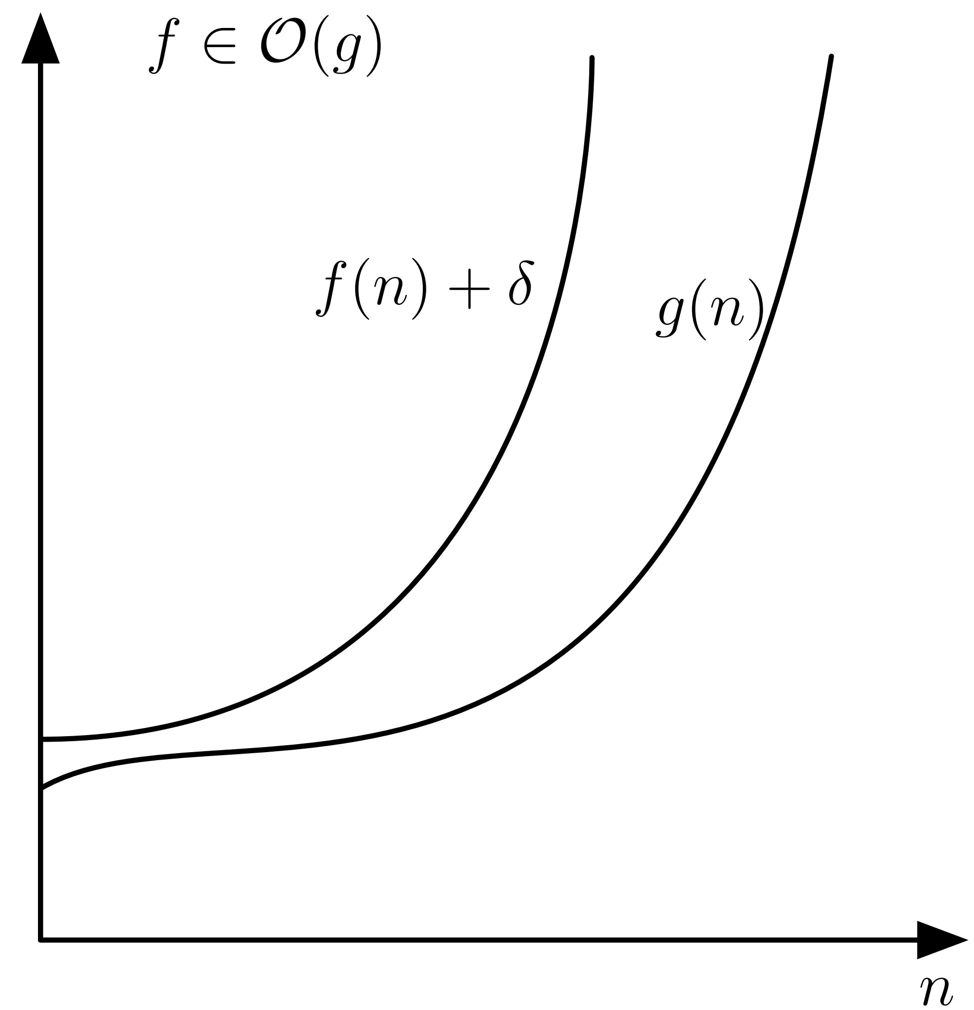
There is a mechanisation [2] in terms of filters, but it hard to use

[1]: R. R. Howell, "On Asymptotic Notation with Multiple Variables", 2008

[2]: A. Guéneau et. Al., "A Fistful of Dollars: Formalizing Asymptotic Complexity Claims via Deductive Program Verification", 2018

O-Notation





$$\exists a, N : \forall n > N : g(n) \leq a \cdot f(n)$$

$$\exists a, \delta : \forall n : g(n) \leq a \cdot f(n) + \delta$$

n should be of an ordered type

n can be of any type X (for $f, g : X \to \mathbb{N}$)

O-Notation

For an arbitrary type X

And some complexity parameters, e.g. $n, k : X \to \mathbb{N}$

Size
Number of vertices
Number of items

- $a \leq b \implies n^a \in \mathcal{O}(n^b)$
- $a < b \implies n^a \in o(n^b)$
- $f \in \mathscr{C} \land g \in \mathscr{C} \implies f + g \in \mathcal{O}(\mathscr{C})$
- $\text{poly}(n) := \mathcal{O}(n^{\mathcal{O}(1)})$
- $2^n \notin \text{poly}(n)$
- $(\log n)^k \in \text{Any}(k) \times \text{poly}(n)$
- $n^k \notin \text{Any}(k) \times \text{poly}(n)$

- Equivalent to the asymptotic O-Notation for "reasonable" functions $\mathbb{N} \to \mathbb{N}$

O-Notation and Composition

- Main benefit: lemmas about composition
- The notion seems very promising

```
Variable f : X -> Y.
Variable F : ParamCompl X.
Hypothesis H1 : L_computable_inParamTime f F.

Variable g : Y -> Z.
Variable G : ParamCompl Y.
Hypothesis H2 : L_computable_inParamTime g G.

Lemma computable_inParamTime_composition:
   L_computable_inParamTime (g ∘ f) (O (F + (G ∘ f))).
```

Complexity of the Reduction

- Unfortunately, I decided to switch to O-Notation too late
- A lot of work has been done in terms of concrete functions. Like this one:

```
414 ∗ (1 + sizeN ψ) ∗ (1 + sizeK ψ) ^ 2 ∗ (1 + sizeM ψ) ^ 2
```

- Eventually, I proved the following lemma

```
Lemma time_reduction :
  L_computable_inParamTime reduction
     (⟦fun ψ => 2^(sizeN ψ / 2)⟧ ⊗ poly size).
```

Contributions

- Define reduction and prove its correctness ✓
- Prove runtime bound on the reduction ✓
- Combine previous steps to obtain $\mathsf{SETH} \implies \mathsf{OVH}$
  - Pen&Paper proof is well-known
  - RIL: The proof is mechanised
  - RIL: There are problems with the handling of $\mathbb{R}$

Statement of the Theorem

$$\text{SETH} := \forall \varepsilon > 0, \exists k, \text{kSAT} \notin \mathcal{O}(2^{(1-\varepsilon)\cdot n} \cdot \text{poly}(size))$$

```
Definition SETH :=
  forall (ε : Q),
    ε > 0 ->
    exists k, L_undecidable_inParamTime
              (SAT k) (O(⟦fun ψ => 2^((1-ε)*sizeN ψ)⟧) ⊗ poly size).
```

Statement of the Theorem

$$\text{SETH} := \forall \varepsilon > 0, \exists k, \text{kSAT} \notin \mathcal{O}(2^{(1-\varepsilon)\cdot n} \cdot \text{poly}(size))$$

```
Definition SETH :=
  forall (ε : Q),
    ε > 0 ->
    exists k, L_undecidable_inParamTime
           (SAT k) (O(⟦fun ψ => 2^((1-ε)*sizeN ψ)⟧) ⊗ poly size).
```

$$\text{OVH} := \forall \varepsilon > 0 : \text{OV} \notin \mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$$

```
Definition OVH :=
  forall (ε : Q),
    ε > 0 ->
    L_undecidable_inParamTime
      OV ((⟦_n⟧ ↑ ⟦fun _ => 2 - ε⟧) ⊗ poly _d).
```

Statement of the Theorem

SETH := $\forall \varepsilon > 0, \exists k,$ kSAT $\notin \mathcal{O}(2^{(1-\varepsilon)\cdot n} \cdot \text{poly}(size))$

```
Definition SETH :=
  forall (ε : Q),
    ε > 0 ->
    exists k, L_undecidable_inParamTime
          (SAT k) (O(⟦fun ψ => 2^((1-ε)*sizeN ψ)⟧) ⊗ poly size).
```

OVH := $\forall \varepsilon > 0 :$ OV $\notin \mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$

```
Definition OVH :=
  forall (ε : Q),
    ε > 0 ->
    L_undecidable_inParamTime
      OV ((⟦_n⟧ ↑ ⟦fun _ => 2 - ε⟧) ⊗ poly _d).
```

Main theorem:

```
Theorem SETH_implies_OVH: SETH -> OVH.
```

Statement of the Theorem

SETH := $\forall \varepsilon > 0, \exists k,$ kSAT $\notin \mathcal{O}(2^{(1-\varepsilon)\cdot n} \cdot \text{poly}(size))$

```
Definition SETH :=
  forall (ε : Q),
    ε > 0 ->
    exists k, L_undecidable_inParamTime
           (SAT k) (O(⟦fun ψ => 2^((1-ε)*sizeN ψ)⟧) ⊗ poly size).
```

OVH := $\forall \varepsilon > 0$: OV $\notin \mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$

```
Definition OVH :=
  forall (ε : Q),
    ε > 0 ->
    L_undecidable_inParamTime
      OV ((⟦_n⟧ ↑ ⟦fun _ => 2 - ε⟧) ⊗ poly _d).
```

Fake rational number

Q := nat

Coq has no support for $\mathbb{Q}/\mathbb{R}$ powers

Main theorem:

```
Theorem SETH_implies_OVH: SETH -> OVH.
```

Proof of the Theorem

- Let ov solves OV in $\mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$ for some $\varepsilon > 0$
- We can show that $\mathsf{ksat} := \mathsf{ov} \circ f$ solves kSAT in time $\mathcal{O}(2^{(1-\varepsilon/2)N}) \cdot \text{poly}(\text{size})$ for any k

Proof of the Theorem

- Let ov solves OV in $\mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$ for some $\varepsilon > 0$
- We can show that ksat $:=$ ov $\circ f$ solves kSAT in time $\mathcal{O}(2^{(1-\varepsilon/2)N}) \cdot \text{poly(size)}$ for any k

- Also know that $f \in \mathcal{O}(2^{N/2}) \cdot \text{poly(size)}$
- For ψ with (N, M, K), f generates an instance $(n := 2^{N/2}, d := M)$

Proof of the Theorem

- Let ov solves OV in $\mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$ for some $\varepsilon > 0$
- We can show that ksat := ov $\circ f$ solves kSAT in time $\mathcal{O}(2^{(1-\varepsilon/2)N}) \cdot \text{poly(size)}$ for any k

- Also know that $f \in \mathcal{O}(2^{N/2}) \cdot \text{poly(size)}$
- For ψ with (N, M, K), f generates an instance $(n := 2^{N/2}, d := M)$

- The runtime of ksat:
  $\mathcal{O}(2^{N/2}) \cdot \text{poly(size)} + \mathcal{O}((n \circ f)^{2-\varepsilon} \cdot \text{poly}(d \circ f))$

Proof of the Theorem

- Let ov solves OV in $\mathcal{O}(n^{2-\varepsilon} \cdot \text{poly}(d))$ for some $\varepsilon > 0$
- We can show that ksat := ov $\circ f$ solves kSAT in time $\mathcal{O}(2^{(1-\varepsilon/2)N}) \cdot \text{poly(size)}$ for any k

- Also know that $f \in \mathcal{O}(2^{N/2}) \cdot \text{poly(size)}$
- For ψ with (N, M, K), f generates an instance $(n := 2^{N/2}, d := M)$

- The runtime of ksat:

$$\mathcal{O}(2^{N/2}) \cdot \text{poly(size)} + \mathcal{O}((n \circ f)^{2-\varepsilon} \cdot \text{poly}(d \circ f)) \subseteq$$
$$\mathcal{O}(2^{N/2}) \cdot \text{poly(size)} + \mathcal{O}(2^{N/2})^{2-\varepsilon} \cdot \text{poly}(M) \subseteq$$
$$\mathcal{O}(2^{N/2}) \cdot \text{poly(size)} + \mathcal{O}(2^{N/2})^{2-\varepsilon} \cdot \text{poly(size)} \subseteq$$
$$(\mathcal{O}(2^{N/2}) + \mathcal{O}(2^{N/2})^{2-\varepsilon}) \cdot \text{poly(size)} \subseteq$$
$$(\mathcal{O}(2^{N/2}) + \mathcal{O}(2^{(1-\varepsilon/2)N}) \cdot \text{poly(size)} \subseteq$$
$$\mathcal{O}(2^{(1-\varepsilon/2)N}) \cdot \text{poly(size)}$$

Contributions

- Defined reduction and proved its correctness
- Proved runtime bound on the reduction
- Introduced the O-Notation
- Mechanised SETH and OVH
- Proved $\mathsf{SETH} \implies \mathsf{OVH}$ (up to the handling of $\mathbb{R}$)

	spec	**proof**
Auxiliary Things	681	1346
Operations on Partial Assignments	267	894
Reduction	139	615
ONotation	207	790
SETH implies OVH	53	256
	1347	3901

Total LOC: 5248

Future work:

- Further investigate the O-Notation
- Fix the problem with rational numbers
- SETH depends on the computational model

  (so, $\mathsf{SETH}_{\mathsf{L}}$ doesn't imply $\mathsf{SETH}_{\mathsf{RAM}}$)

Backup Slides

Definition for O-Notation

```
Inductive ParamCompl (X : Type) : Type :=
| id__pc (f : X -> nat) : ParamCompl X
| o__pc : ParamCompl X -> ParamCompl X
| O__pc : ParamCompl X -> ParamCompl X
| add__pc : ParamCompl X -> ParamCompl X -> ParamCompl X
| mul__pc : ParamCompl X -> ParamCompl X -> ParamCompl X
| exp__pc : ParamCompl X -> ParamCompl X -> ParamCompl X
| Any__pc : ParamCompl X -> ParamCompl X.
```

```
Reserved Notation "f ∈p F" (at level 65).
Fixpoint InParamCompl {X : Type} (f : X -> nat) (F : ParamCompl X) : Prop :=
  match F with
  | ⟦F⟧ => forall x, f x <= F x
  | O(F) => exists g, g ∈p F /\ exists a b,       forall x, f x <= a * g x + b
  | o(F) => exists g, g ∈p F /\ forall a b, exists δ, forall x, a * f x + b <= g x + δ
  | F1⊕F2 => exists g1 g2, g1 ∈p F1 /\ g2 ∈p F2 /\ forall x, f x <= g1 x + g2 x
  | F1⊗F2 => exists g1 g2, g1 ∈p F1 /\ g2 ∈p F2 /\ forall x, f x <= g1 x * g2 x
  | F1↑F2 => exists g1 g2, (forall x, 0 < g1 x) /\ g1 ∈p F1 /\ g2 ∈p F2 /\ forall x, f x <= g1 x ^ g2 x
  | Any(F) => exists G, monotonic G /\ exists g, g ∈p F /\ forall x, f x <= G (g x)
  end
where "f ∈p F" := (InParamCompl f F).
Notation "f ∉p F" := (~ InParamCompl f F) (at level 65).
```

Exponential Time Hypothesis (ETH)

- Algorithms for solving 3SAT
  - Trivial algorithm: $3SAT \in \mathcal{O}(2^n \cdot \text{poly}(size))$
  - Monien and Speckenmeyer: $3SAT \in \mathcal{O}(2^{0.7n} \cdot \text{poly}(size))$
  - Schöning, randomized: $3SAT \in \mathcal{O}(2^{0.42n} \cdot \text{poly}(size))$
  - Moser, Scheder: $3SAT \in \mathcal{O}(2^{0.42n} \cdot \text{poly}(size))$
  - …
  - PPSZ algorithm, randomized: $3SAT \in \mathcal{O}(2^{0.387n} \cdot \text{poly}(size))$

- Exponential Time Hypothesis (ETH)
  - There is no “fast” algorithm for 3SAT
  - Formally: $\exists \varepsilon > 0, 3SAT \notin \mathcal{O}(2^{\varepsilon \cdot n} \cdot \text{poly}(size))$

$\mathcal{O}(2^{\sqrt{n}} \cdot \text{poly}(size))$

Reduction and its Correctness

Reduction and its Correctness

$$\exists \alpha \models \psi \implies \exists a \in A, b \in B, \langle a, b \rangle = 0$$

Let $\alpha = \{x_1 \mapsto b_1, \ldots, x_N \mapsto b_N\}$ and $\alpha \models \psi$

$$\begin{pmatrix} x_1 \mapsto ? \\ \ldots \\ x_{N/2} \mapsto ? \\ x_{N/2+1} \mapsto b_{N/2+1} \\ \ldots \\ x_N \mapsto b_N \end{pmatrix} \qquad \begin{pmatrix} x_1 \mapsto b_1 \\ \ldots \\ x_{N/2} \mapsto b_{N/2} \\ x_{N/2+1} \mapsto ? \\ \ldots \\ x_N \mapsto ? \end{pmatrix}$$

$$a = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \ldots \\ 1 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ \ldots \\ 0 \end{pmatrix}$$

$$\langle a, b \rangle = 0$$

Reduction and its Correctness

$$\exists a \in A, b \in B, \langle a, b\rangle = 0 \implies \exists \alpha \vDash \psi$$

$$a, b : \langle a, b\rangle = 0$$

$$a = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \dots \\ 1 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ \dots \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \mapsto b_1 \\ \dots \\ x_{N/2} \mapsto b_{N/2} \\ x_{N/2+1} \mapsto b_{N/2+1} \\ \dots \\ x_N \mapsto b_N \end{pmatrix} \qquad \begin{pmatrix} x_1 \mapsto b'_1 \\ \dots \\ x_{N/2} \mapsto b'_{N/2} \\ x_{N/2+1} \mapsto b'_{N/2+1} \\ \dots \\ x_N \mapsto b'_N \end{pmatrix}$$

$$\alpha = \begin{pmatrix} x_1 \mapsto b_1 \\ \dots \\ x_{N/2} \mapsto b_{N/2} \\ x_{N/2+1} \mapsto b'_{N/2+1} \\ \dots \\ x_N \mapsto b'_N \end{pmatrix}$$

$$\alpha \vDash \psi$$

LOC

spec	proof	
466	756	Base.v
95	301	CNF.v
40	96	InnerProduct.v
207	790	ONotation.v
48	156	PartialAssign.v
219	738	PartialAssignOperations.v
72	182	SAT.v
53	256	SETH.v
8	11	OV.v
139	615	EvenSAT_to_OV.v
1347	3901	-> 5248

O-Notation

- Using this notion we can reformulate the old predicates

```
Definition L_computable_inParamTime
   {X Y} `{RX : registered X} `{RY : registered Y} (f : X -> Y) (F : ParamCompl X) :=
   exists (fT : X -> nat),
      fT ∈p F /\ is_computable_time
         (t:=TyArr (TyB X) (TyB Y)) f (fun x _ => (fT x,tt)).


Definition L_decidable_inParamTime
   {X} `{R : registered X} (P : X -> Prop) (F : ParamCompl X) :=
      exists (Pdec : X -> bool),
         L_computable_inParamTime Pdec F /\ forall x, P x <-> Pdec x = true.
```

Definition from the extraction framework

O-Notation

```
Context {X : Type} {Y : Type} (A : X -> Prop) (B : Y -> Prop).

Variable r : X -> Y.
Hypothesis H_r_is_correct : forall x, A x <-> B (r x).
Variable R : ParamCompl X.
Hypothesis H : L_computable_inParamTime r R.


Variable F : ParamCompl Y.
Hypothesis H_B_is_dec_in_FB: L_decidable_inParamTime B F.

Lemma hmmm: L_decidable_inParamTime A (O (R + (F ∘ r))).
```