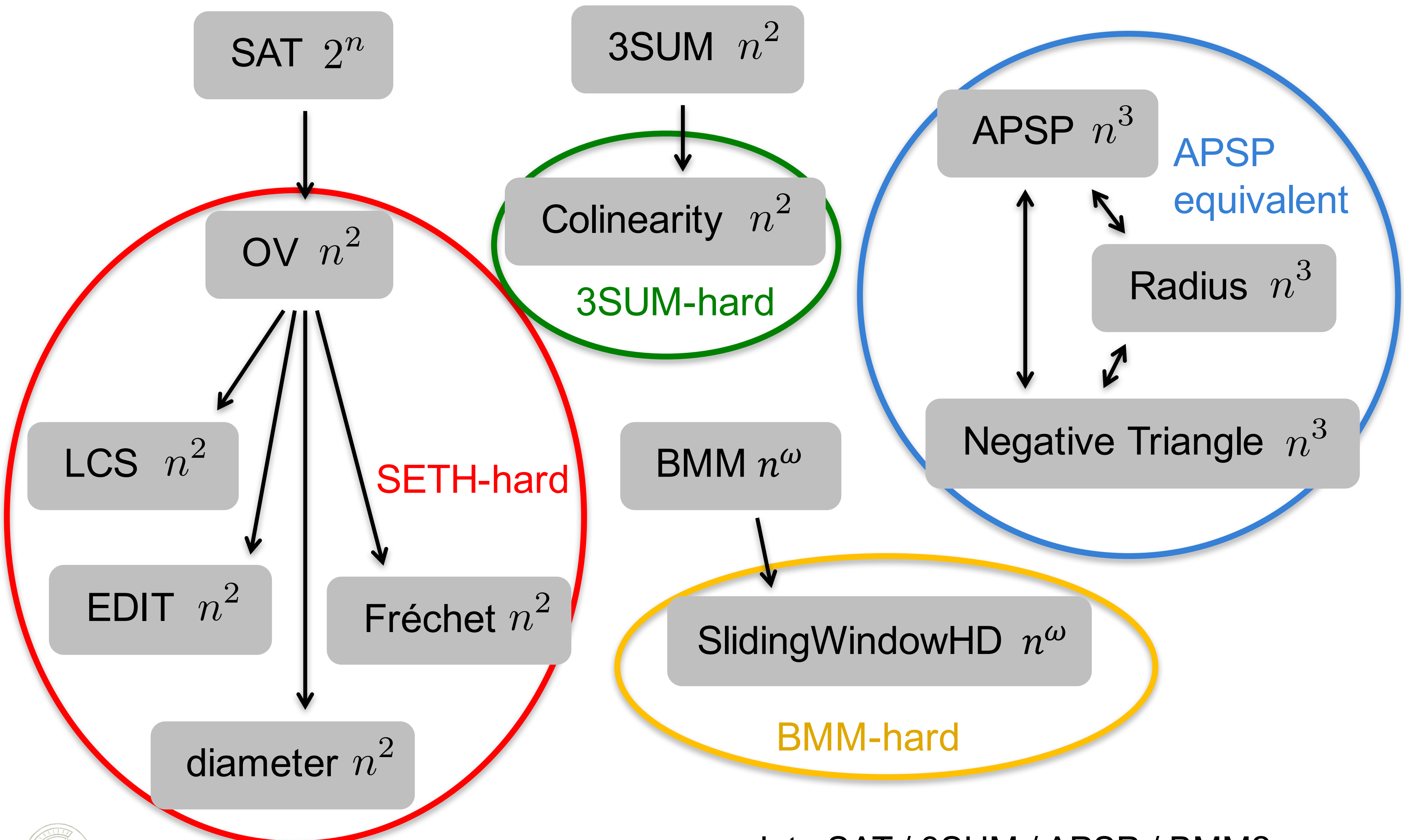
max planck institut
informatik

Complexity Theory of Polynomial-Time Problems

Lecture 11: Nondeterministic SETH

Karl Bringmann

Complexity Inside P



can we relate SAT / 3SUM / APSP / BMM?

Relating Hypotheses

only very weak relations are known

e.g. ETH implies that k-SUM has no $n^{o(k)}$ algorithm

OPEN: SETH implies that 3SUM has no $O(n^{2-\varepsilon})$ algorithm ?

today we will see a **barrier** for tighter connections

I. Nondeterministic SETH

Nondeterministic Algorithms

Turing machine: can choose any applicable transition at any point in time

RAM: operation **guess()** fills a cell with an integer

YES-instance: at least one accepting path **NO**-instance: all paths reject

guesses on an accepting path = **proof that we have a YES-instance**

NP: all problems solvable in polytime by a nondet. Turing machine

k-SAT algorithm: guess a satisfying assignment, check correctness $O(n+m)$

„guess a short proof of satisfiability and check it“

3SUM algorithm: guess $a, b, c \in A$ and check $a+b+c=0$ $O(1)/O(n)$

Co-Nondeterministic Algorithms

Turing machine: can choose any applicable transition at any point in time

RAM: operation **guess()** fills a cell with an integer

YES-instance: all paths accept   **NO**-instance: at least one path rejects

guesses on a rejecting path = **proof that we have a NO-instance**

co-NP: all problems solvable in polytime by a co-nondet. Turing machine

k-SAT: „guess a short proof of unsatisfiability and check it" – Is this possible ?

classic computational complexity:

if NP $\neq$ co-NP, then **k-SAT has no $O(\text{poly}(n))$ co-nondet. algorithm**

we believe that NP $\neq$ co-NP, since otherwise the polynomial hierarchy collapses

(Co-)Nondeterministic SETH

if NP $\neq$ co-NP, then k-SAT has no $O(\text{poly}(n))$ co-nondet. algorithm

not even a $O(2^{(1-\varepsilon)n})$ co-nondet. algorithm is known!

Nondeterministic SETH: k-SAT has no no $O(2^{(1-\varepsilon)n})$ co-nondet. algorithm

[CGIMPS'16]

do not allow randomization!

NSETH implies SETH (without randomization)

barely anyone believes that NSETH is true

but it formalizes a current **barrier**

NSETH can be used to conditionally **rule out reductions**

Potential Reduction from SAT to 3SUM

an *deterministic* algorithm A for k-SAT with **oracle** access to 3SUM s.t.:

3SUM

k-SAT

fomula ϕ

n variables,
$m \leq n^k$ clauses

reduction

$\vdots$

total time
$r(n)$

instance I_1
size n_1

$\vdots$

instance I_k
size n_k

Properties:

for any fomula ϕ, algorithm $A(\phi)$ correctly solves k-SAT on ϕ

A runs in time $r(n) = O(2^{(1-\gamma)n})$ for some $\gamma > 0$

for any $\varepsilon > 0$ there is a $\delta \in (0, \gamma)$ s.t. $\sum_{i=1}^{k} {n_i}^{2-\varepsilon} \leq 2^{(1-\delta)n}$

e.g. $k = 1$ and $n_1 = 2^{n/2} m^c$, then ${n_1}^{2-\varepsilon} \leq 2^{(1-\varepsilon/2)n} n^{ck} \leq 2^{(1-\varepsilon/3)n}$

Potential Reduction from SAT to 3SUM

an *deterministic* algorithm A for k-SAT with **oracle** access to 3SUM s.t.:

k-SAT

fomula ϕ

n variables,
$m \leq n^k$ clauses

reduction

⋮

total time
$r(n)$

3SUM

instance I_1
size n_1

⋮

instance I_k
size n_k

Properties:

for any fomula ϕ, algorithm $A(\phi)$ correctly solves k-SAT on ϕ

A runs in time $r(n) = O(2^{(1-\gamma)n})$ for some $\gamma > 0$

for any $\varepsilon > 0$ there is a $\delta \in (0, \gamma)$ s.t. $\sum_{i=1}^{k} {n_i}^{2-\varepsilon} \leq 2^{(1-\delta)n}$

$O(2^{(1-\delta)n})$ algorithm $\Longleftarrow$ $O(n^{2-\varepsilon})$ algorithm

max planck institut
informatik

Potential Reduction from SAT to 3SUM

an *deterministic* algorithm A for k-SAT with **oracle** access to 3SUM s.t.:

k-SAT

fomula ϕ

n variables,
$m \leq n^k$ clauses

reduction

$\vdots$

total time
$r(n)$

3SUM

instance I_1
size n_1

$\vdots$

instance I_k
size n_k

Properties:

for any fomula ϕ, algorithm $A(\phi)$ correctly solves k-SAT on ϕ

A runs in time $r(n) = O(2^{(1-\gamma)n})$ for some $\gamma > 0$

for any $\varepsilon > 0$ there is a $\delta \in (0, \gamma)$ s.t. $\sum_{i=1}^{k} {n_i}^{2-\varepsilon} \leq 2^{(1-\delta)n}$

nondet. $O(2^{(1-\delta)n})$ algorithm and co-nondet. $O(2^{(1-\delta)n})$ algorithm $\Longleftarrow$ nondet. $O(n^{2-\varepsilon})$ algorithm and co-nondet. $O(n^{2-\varepsilon})$ algorithm

max planck institut
informatik

Potential Reduction from SAT to 3SUM

an *deterministic* algorithm A for k-SAT with **oracle** access to 3SUM s.t.:

k-SAT

fomula ϕ

n variables,
$m \leq n^k$ clauses

reduction

⋮

total time
$r(n)$

3SUM

instance I_1
size n_1

⋮

instance I_k
size n_k

for each instance I_j: guess whether it is YES- or NO-instance

if we guessed YES: guess a proof π_j that I_j is a YES-instance

if we guessed NO: guess a proof π_j that I_j is a NO-instance

if we guessed correctly:

$\pi = (\pi_1, \dots, \pi_k)$ forms a proof that ϕ is satisfiable or unsatisfiable

algorithm A is the „proof checker“

Potential Reduction from SAT to 3SUM

an *deterministic* algorithm A for k-SAT with **oracle** access to 3SUM s.t.:

k-SAT

fomula ϕ

n variables,
$m \leq n^k$ clauses

reduction

⋮

total time
$r(n)$

3SUM

instance I_1
size n_1

⋮

instance I_k
size n_k

nondet. $O(2^{(1-\delta)n})$ algorithm and co-nondet. $O(2^{(1-\delta)n})$ algorithm $\Leftarrow$ nondet. $O(n^{2-\varepsilon})$ algorithm and co-nondet. $O(n^{2-\varepsilon})$ algorithm

no nondet. $O(2^{(1-\delta)n})$ algorithm or no co-nondet. $O(2^{(1-\delta)n})$ algorithm $\Rightarrow$ no nondet. $O(n^{2-\varepsilon})$ algorithm or no co-nondet. $O(n^{2-\varepsilon})$ algorithm

=NSETH

Ruling Out Reductions

If NSETH holds and

3SUM has a $O(n^{2-\varepsilon})$ co-nondeterministic algorithm

we will show this

then there is **no deterministic reduction from k-SAT to 3SUM**

either 3SUM has strongly subquadratic algorithms

or 3SUM is hard for a different reason than k-SAT

or NSETH fails

has drawbacks, but this is the **only tool** for negative results in this area

Co-Nondeterministic Algorithm for 3SUM

3SUM: given set A of integers in $\{-n^s, \ldots, n^s\}$, are there $a, b, c \in A$ s.t. $a + b + c = 0$?

Thm: 3SUM has a co-nondeterministic algorithm in time $\tilde{O}(n^{3/2})$

[CGIMPS'16]

$\tilde{O}$ hides polylogarithmic factors in n

$$\tilde{O}(f(n)) = O(f(n) \cdot \operatorname{polylog} n)$$

$$\tilde{O}(f(n)) = \bigcup_{c \geq 0} O(f(n) \log^c n)$$

Co-Nondeterministic Algorithm for 3SUM

3SUM: given set A of integers in $\{-n^s, \ldots, n^s\}$, are there $a, b, c \in A$ s.t. $a + b + c = 0$?

Thm: 3SUM has a co-nondeterministic algorithm in time $\tilde{O}(n^{3/2})$

[CGIMPS'16]

1) guess prime $p \leq n^{3/2} \log n$ $\tilde{O}(p)$

2) compute $t = |\{(a, b, c) \in A^3 \mid a + b + c = 0 \bmod p\}|$ $\tilde{O}(p)$

Recall: 3SUM for Small Numbers

3SUM is in time $O(n) + \tilde{O}(U)$ for numbers in $\{0, \ldots, U\}$

define polynomial $P(X) \coloneqq \sum_{a \in A} X^a$

has degree at most U

compute $Q(X) \coloneqq P(X) \cdot P(X) \cdot P(X) = (\sum_{a \in A} X^a)(\sum_{a \in A} X^a)(\sum_{a \in A} X^a)$

what is the coefficient of X^t in $Q(X)$? $(X^a \cdot X^b \cdot X^c = X^{a+b+c})$

it is the **number of (a, b, c) summing to t**

use efficient polynomial multiplication (via Fast Fourier Transform):

polynomials of degree d can be multiplied in time $\tilde{O}(d)$

Co-Nondeterministic Algorithm for 3SUM

3SUM: given set A of integers in $\{-n^s, \ldots, n^s\}$, are there $a, b, c \in A$ s.t. $a + b + c = 0$?

Thm: 3SUM has a co-nondeterministic algorithm in time $\tilde{O}(n^{3/2})$

[CGIMPS'16]

1) guess prime $p \leq n^{3/2} \log n$

2) compute $t = |\{(a,b,c) \in A^3 \mid a + b + c = 0 \bmod p\}|$ $\tilde{O}(p)$

let $B := \{a \bmod p \mid a \in A\}$ (in general B is a multi-set!)

$$\left.\begin{array}{l} \text{let } r_0 := |\{(a,b,c) \in B^3 \mid a + b + c = 0\}| \\ \text{let } r_1 := |\{(a,b,c) \in B^3 \mid a + b + c = p\}| \\ \text{let } r_2 := |\{(a,b,c) \in B^3 \mid a + b + c = 2p\}| \end{array}\right\} \text{universe size } U = p$$

then $t = r_0 + r_1 + r_2$

Co-Nondeterministic Algorithm for 3SUM

3SUM: given set A of integers in $\{-n^s, \dots, n^s\}$, are there $a, b, c \in A$ s.t. $a + b + c = 0$?

Thm: 3SUM has a co-nondeterministic algorithm in time $\tilde{O}(n^{3/2})$

[CGIMPS'16]

1) guess prime $p \leq n^{3/2} \log n$

2) compute $t = |\{(a,b,c) \in A^3 \mid a + b + c = 0 \bmod p\}|$ $\tilde{O}(p)$

3) if $t > \alpha \cdot n^{3/2} \log n$: accept (constant α to be fixed later)

4) guess distinct $(a_1, b_1, c_1), \dots, (a_t, b_t, c_t) \in A^3$ such that $a_i + b_i + c_i = 0 \bmod p$ $\forall i$

5) check that for all (a_i, b_i, c_i) we have $a_i + b_i + c_i \neq 0$

6) if everything works out: reject (otherwise accept)

✔ time $\tilde{O}(n^{3/2})$

✔ if we reject then we have a NO-instance

✔ **YES**-instance: all paths accept

NO-instance: at least one path rejects

max planck institut
informatik

Co-Nondeterministic Algorithm for 3SUM

3SUM: given set A of integers in $\{-n^s, \dots, n^s\}$, are there $a, b, c \in A$ s.t. $a + b + c = 0$?

NO-instance: at least one path rejects:

show that there exists a prime $p \leq n^{3/2} \log n$ such that

$$t = |\{(a,b,c) \in A^3 \mid a + b + c = 0 \bmod p\}| < \alpha \cdot n^{3/2} \log n$$

M := # tuples (a, b, c, p) with $a, b, c \in A$ and prime p s.t. $a + b + c = 0 \bmod p$

each $a + b + c$ is in $\{-3n^s, \dots, 3n^s\} \setminus \{0\}$, so it has at most $\log(3n^s)$ prime factors

thus $M \leq n^3 \log(3n^s) \leq 3s \cdot n^3 \log n$

by prime number theorem: there are at least $n^{3/2}/\beta$ primes $p \leq n^{3/2} \log n$

thus there is a prime p contained in at most $M/(n^{3/2}/\beta)$ tuples (a, b, c, p)

thus there is a prime p with $t \leq M/(n^{3/2}/\beta) \leq 3s \cdot \beta \cdot n^{3/2} \log(n)$

set $\alpha := 3s \cdot \beta$

Co-Nondeterministic Algorithm for 3SUM

3SUM: given set A of integers in $\{-n^s, \dots, n^s\}$, are there $a, b, c \in A$ s.t. $a + b + c = 0$?

Thm: 3SUM has a co-nondeterministic algorithm in time $\tilde{O}(n^{3/2})$

[CGIMPS'16]

1) guess prime $p \leq n^{3/2} \log n$

2) compute $t = |\{(a,b,c) \in A^3 \mid a + b + c = 0 \bmod p\}|$ $\tilde{O}(p)$

3) if $t > \alpha \cdot n^{3/2} \log n$: accept (constant α to be fixed later)

4) guess distinct $(a_1, b_1, c_1), \dots, (a_t, b_t, c_t) \in A^3$ such that $a_i + b_i + c_i = 0 \bmod p$ $\forall i$

5) check that for all (a_i, b_i, c_i) we have $a_i + b_i + c_i \neq 0$

6) if everything works out: reject (otherwise accept)

✔ time $\tilde{O}(n^{3/2})$

✔ if we reject then we have a NO-instance

✔ **YES**-instance: all paths accept

✔ **NO**-instance: at least one path rejects

I. Randomized Nondeterministic SETH

Randomized Nondeterministic SETH

Nondeterministic SETH: k-SAT has no no $O(2^{(1-\varepsilon)n})$ co-nondet. algorithm

what if we allow randomization?

then the hypothesis is wrong!

Thm: k-SAT has a randomized co-nondeterministic $O(2^{n/2}\text{poly}(n))$ algorithm with error probability $2^{-\Omega(n)}$

$\Uparrow$

[Williams'16]

Thm: OV has a randomized co-nondeterministic $O(n\ \text{poly}(d, \log n))$ algorithm with error probability $n^{-\Omega(1)}$

Tools: Basics on Polynomials

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

univariate polynomials $P(X) = \sum_{i=0}^{n} a_i X^i$, $Q(X) = \sum_{i=0}^{m} b_i X^i$, $m \leq n$

multiplication $P(X) \cdot Q(X)$: $\tilde{O}(n)$ (by FFT, without proof)

division with remainder: $\tilde{O}(n)$ (without proof)

$P(X) = S(X) \cdot Q(X) + R(X)$, where $R(X)$ has degree $< m$

we write $R(X) = P(X) \bmod Q(X)$

evaluate $P(X)$ at a given point x: $O(n)$

Horner's method: $P(x) = a_0 + x \cdot \left(a_1 + x \cdot \left(a_2 + x \cdot (\dots)\right)\right)$

Tools: Multipoint Evaluation on Polynomials

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

univariate polynomial $P(X) = \sum_{i=0}^{n} a_i X^i$

multipoint evaluation: $\tilde{O}(n)$

evaluate $P(X)$ at given points $X = x_1, \dots, x_n$

1) let $L(X) \coloneqq (X - x_1) \cdots (X - x_{n/2})$ and $R(X) \coloneqq (X - x_{n/2+1}) \cdots (X - x_n)$

2) let $P_L(X) \coloneqq P(X) \bmod L(X)$ and $P_R(X) \coloneqq P(X) \bmod R(X)$

3) recursively compute $P_L(x_1), \dots, P_L(x_{n/2})$ and $P_R(x_{n/2+1}), \dots, P_R(x_n)$

polynomial division: $P(X) = S(X) \cdot L(X) + P_L(X)$

$P(x_i) = S(x_i) \cdot L(x_i) + P_L(x_i) = P_L(x_i)$

$T(n) = 2T(n/2) + \tilde{O}(n) = \tilde{O}(n)$

$= 0$ for $i \le n/2$

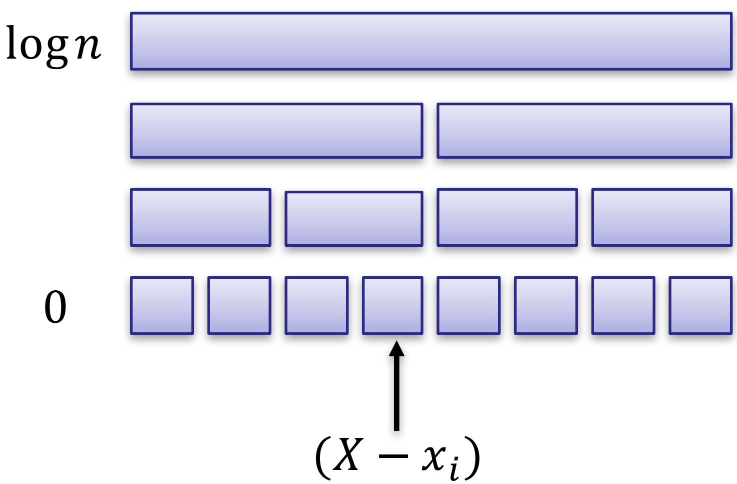
Tools: Multipoint Evaluation on Polynomials

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

univariate polynomial $P(X) = \sum_{i=0}^{n} a_i X^i$

computing $L(X) \coloneqq (X - x_1) \cdots (X - x_{n/2})$:

straight-forward binary tree



computes canonical polynomials

$$P_{s \cdot 2^t + 1, (s+1) 2^t}(X)$$

defined by

$$P_{i,j}(X) := \prod_{k=i}^{j} X - x_k$$

in layer i: $n/2^i$ multiplications of polynomials of degree 2^i

total time $\tilde{O}(n)$

Tools: Polynomial Interpolation

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

univariate polynomial $P(X) = \sum_{i=0}^{n} a_i X^i$

polynomial interpolation: $\tilde{O}(n)$

given pairs $(x_1, y_1), \dots, (x_n, y_n)$ find a polynomial $P(X)$ with $P(x_i) = y_i$ for all i

Lagrange's formula: $$P(X) = \sum_i y_i \cdot \prod_{j \neq i} \frac{X - x_j}{x_i - x_j}$$

Caveat: „division by x" in $\mathbb{Z}_p$ means multiplication with the inverse x^{-1}

extended Euclidean algorithm:

computes s, t with $s \cdot x + t \cdot p = \gcd(x, p) = 1$

modulo p: $s \cdot x = 1$

so $s = x^{-1}$ is the inverse of x

Tools: Polynomial Interpolation

1st goal: compute $\quad P(X) = \sum_i y_i' \cdot \prod_{j \neq i} X - x_j \quad$ (for $y_i' := y_i \cdot \prod_{j \neq i} \frac{1}{x_i - x_j}$)

let $L(X) \coloneqq (X - x_1) \cdots (X - x_{n/2})$ and $R(X) \coloneqq (X - x_{n/2+1}) \cdots (X - x_n)$

recursion:

$$\sum_i y_i' \cdot \prod_{j \neq i} X - x_j = \left(\sum_{1 \leq i \leq n/2} y_i' \cdot \prod_{\substack{1 \leq j \leq n/2 \\ j \neq i}} X - x_j \right) \cdot R(X)$$

$$+ \left(\sum_{n/2 < i \leq n} y_i' \cdot \prod_{\substack{n/2 < j \leq n \\ j \neq i}} X - x_j \right) \cdot L(X)$$

Tools: Polynomial Interpolation

2nd goal: compute factors $s_i = \prod_{j \neq i} \frac{1}{x_i - x_j}$

compute the derivative of

$$Q(X) := \prod_{i=1}^{n} X - x_i = \sum_{i=0}^{n} a_i X^i$$

$$Q'(X) = \sum_i \prod_{j \neq i} X - x_j = \sum_{i=0}^{n} i \cdot a_i X^{i-1}$$

can compute $Q'(X)$ in time $\tilde{O}(n)$

then we have

$$Q'(x_k) = \sum_i \prod_{j \neq i} x_k - x_j = \prod_{j \neq k} x_k - x_j = s_k{}^{-1}$$

compute all $Q'(x_k)$ for $k = 1, \dots, n$ in time $\tilde{O}(n)$ by multipoint evaluation

Tools: Polynomial Interpolation

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

univariate polynomial $P(X) = \sum_{i=0}^{n} a_i X^i$

polynomial interpolation: $\tilde{O}(n)$

given pairs $(x_1, y_1), \ldots, (x_n, y_n)$ find a polynomial $P(X)$ with $P(x_i) = y_i$ for all i

Lagrange‘s formula: $$P(X) = \sum_i y_i \cdot \prod_{j \neq i} \frac{X - x_j}{x_i - x_j}$$

can be computed in time $\tilde{O}(n)$!

Tools: Arithmetic Circuits

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

arithmetic circuits are a (succinct) representation of (multivariate) polynomials

input gates labeled with variables $X_1, \dots, X_k$ - **univariate** if $k = 1$

each other gate is a "$+$" or "$\times$" (unbounded fanin)

or a "$-$" (fanin 1)

or a **constant** (fanin 0)

fanout is unbounded

one **output** gate

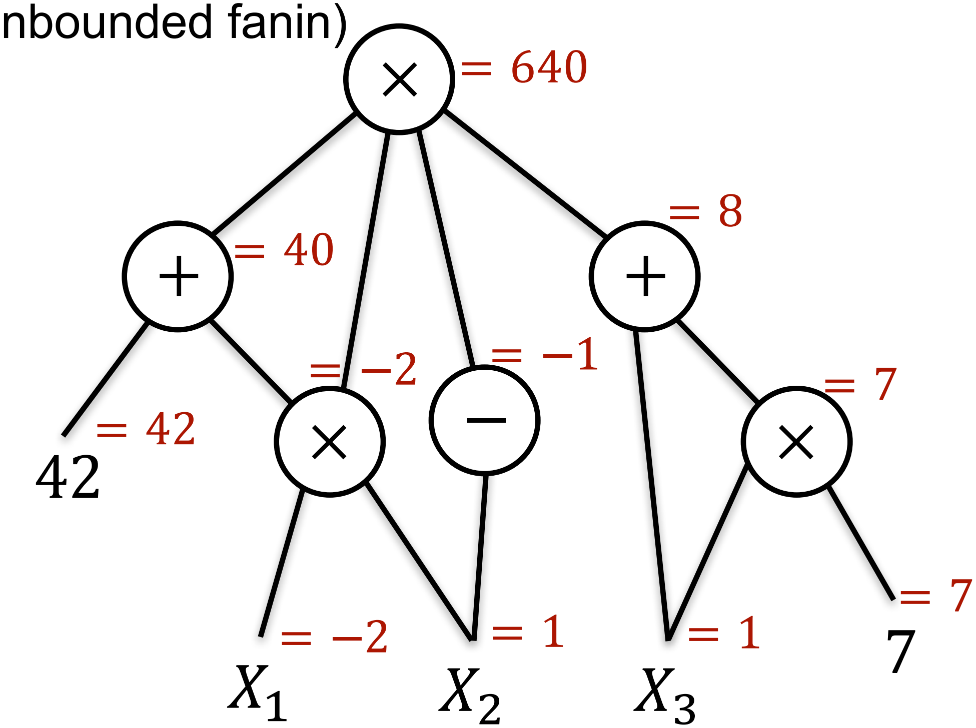
(no cyclic dependencies)

given an input $x_1, \dots, x_k \in \mathbb{Z}_p$

the output $C(x_1, \dots, x_k)$

is the number computed

by the output gate (in $\mathbb{Z}_p$)

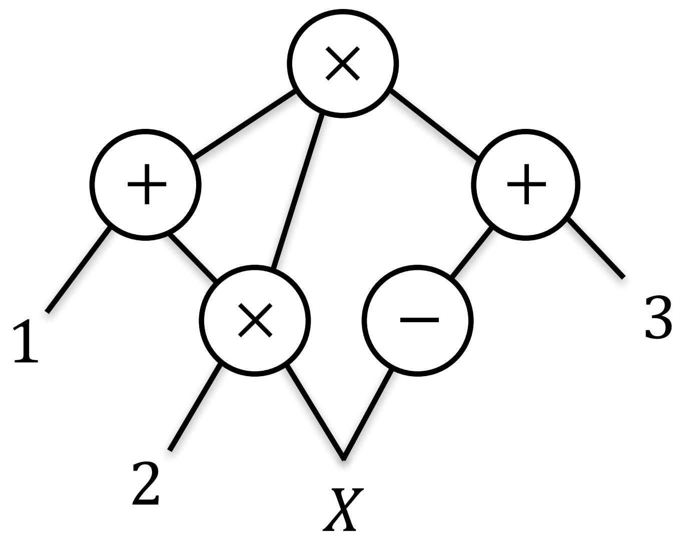


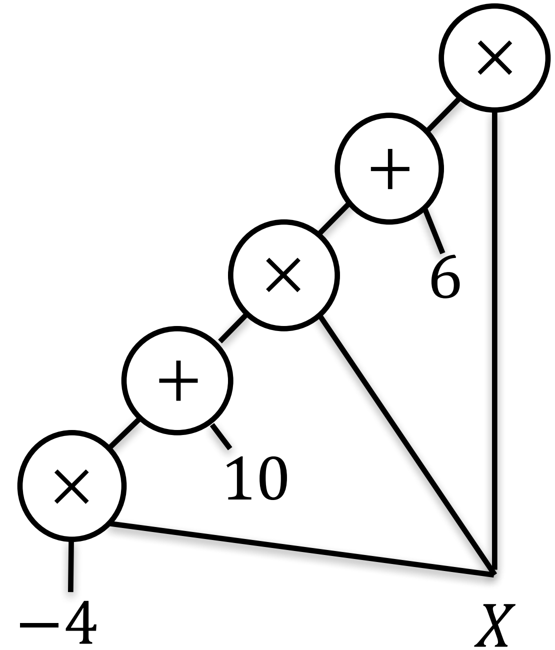
$$(42 + (X_1 X_2))(X_1 X_2)(-X_2)(X_3 + (X_3 \cdot 7))$$

Tools: Arithmetic Circuits

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

representation as circuit is not unique





$$(1+2X)(2X)(3-X) \quad = \quad -4X^3+10X^2+6X$$

circuit = unstructured, succinct

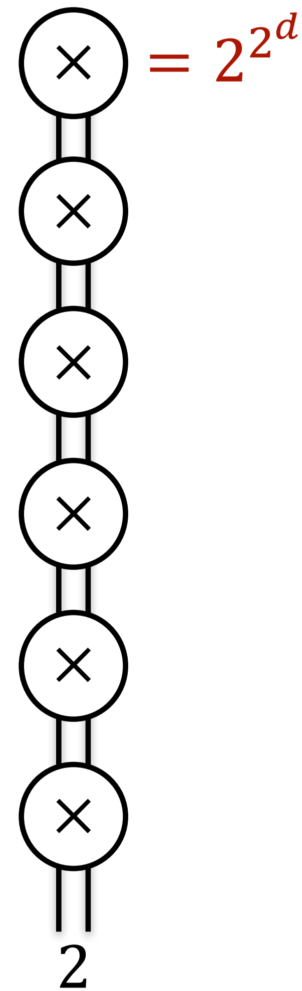
polynomial = structured, verbose

Tools: Arithmetic Circuits

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

working modulo p is necessary

over $\mathbb{Z}$ numbers can get very large:



Tools: Arithmetic Circuits

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

k inputs   size s = number if wires

degree d = „largest degree of any monomial assuming no cancelations"

degree(input gate) = 1

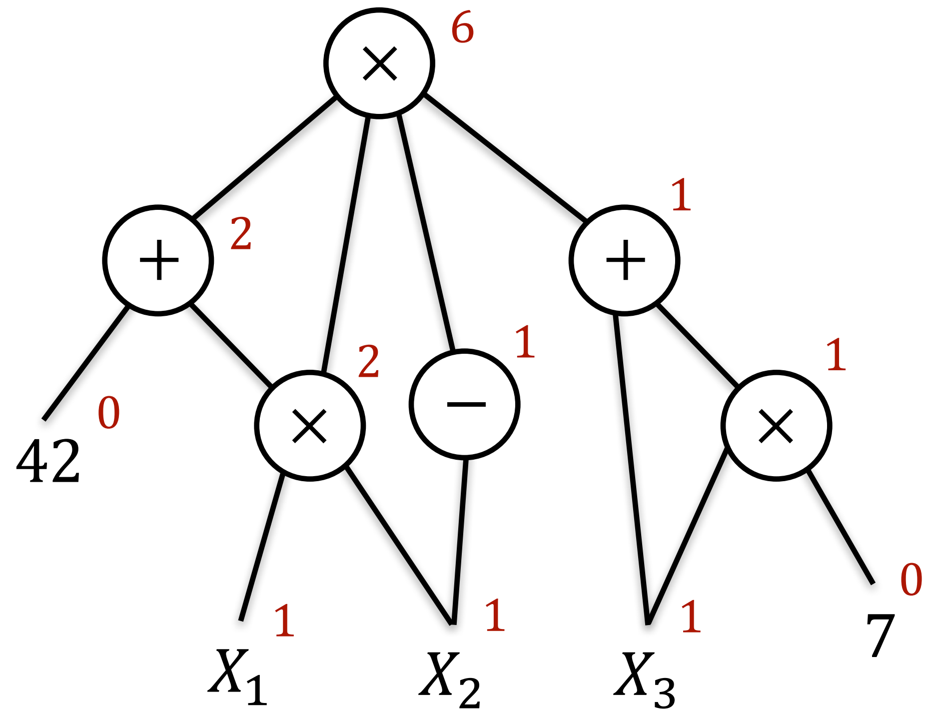
degree(constant gate) = 0

degree("−" gate) = degree of child

degree("+" gate) =
maximum of degrees of children

degree("×" gate) =
sum of degrees of children

degree of circuit =
degree(output gate)



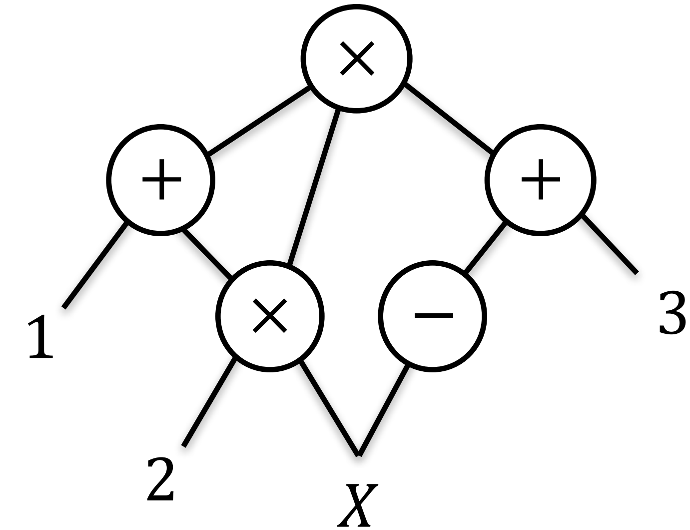
$$(42 + (X_1 X_2))(X_1 X_2)(-X_2)(X_3 + (X_3 \cdot 7))$$

Tools: Evaluation of Circuits

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

k inputs   size s = number if wires   degree d

evaluating a circuit at given input: $O(s)$



$(1+2X)(2X)(3-X)$

Tools: Identity Testing

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

k inputs size s = number if wires degree d

given two circuits C_1, C_2, do they represent the **same polynomial** over $\mathbb{Z}_p$?

this problem is called **polynomial identity testing**

assume that C_1, C_2 are **univariate** and assume $p \geq 2d$

Schwarz-Zippel-Lemma:

this has a randomized $\tilde{O}(s)$-time algorithm with error probability $1 - s^{-\Omega(1)}$

1) for $\log s$ rounds:

2) pick random $x \in \mathbb{Z}_p$

3) if $C_1(x) \neq C_2(x)$: return „not identical“

4) return „identical“

$C_1(X) - C_2(X)$ is a polynomial $Q(X)$ of degree d

if $C_1(X) \neq C_2(X)$ then $Q(X)$ is not the 0-polynomial and thus has at most d roots

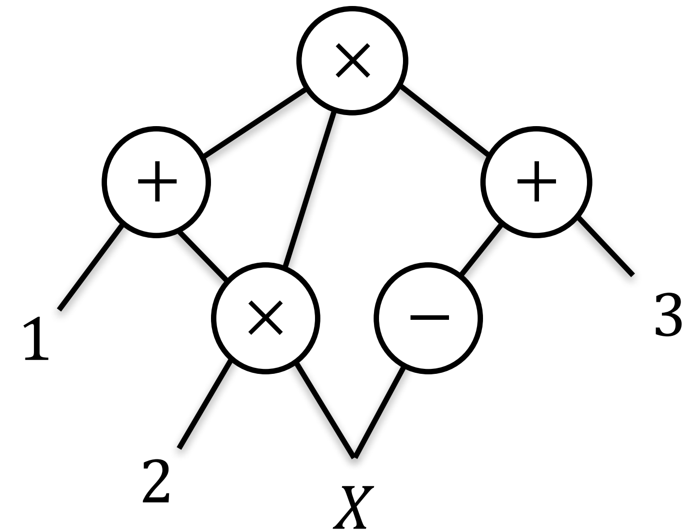
with probability $\geq 1/2$ we pick a non-root x

Tools: Evaluation of Circuits

fix field $\mathbb{Z}_p$ and assume that field operations can be performed in $O(1)$ time

k inputs size s = number if wires degree d

evaluating a circuit at given input: $O(s)$



$(1+2X)(2X)(3-X)$

multipoint evaluation (at n points):

no near-linear time algorithm known

trivial algorithm: $O(n \cdot s)$

converting a **univariate** circuit

to a polynomial: $\tilde{O}(s \cdot d)$

(write each gate as a degree polynomial)

conversion + multipoint evaluation for polynomials

= multipoint evaluation for univariate circuits in $\tilde{O}(s \cdot d + n + d)$

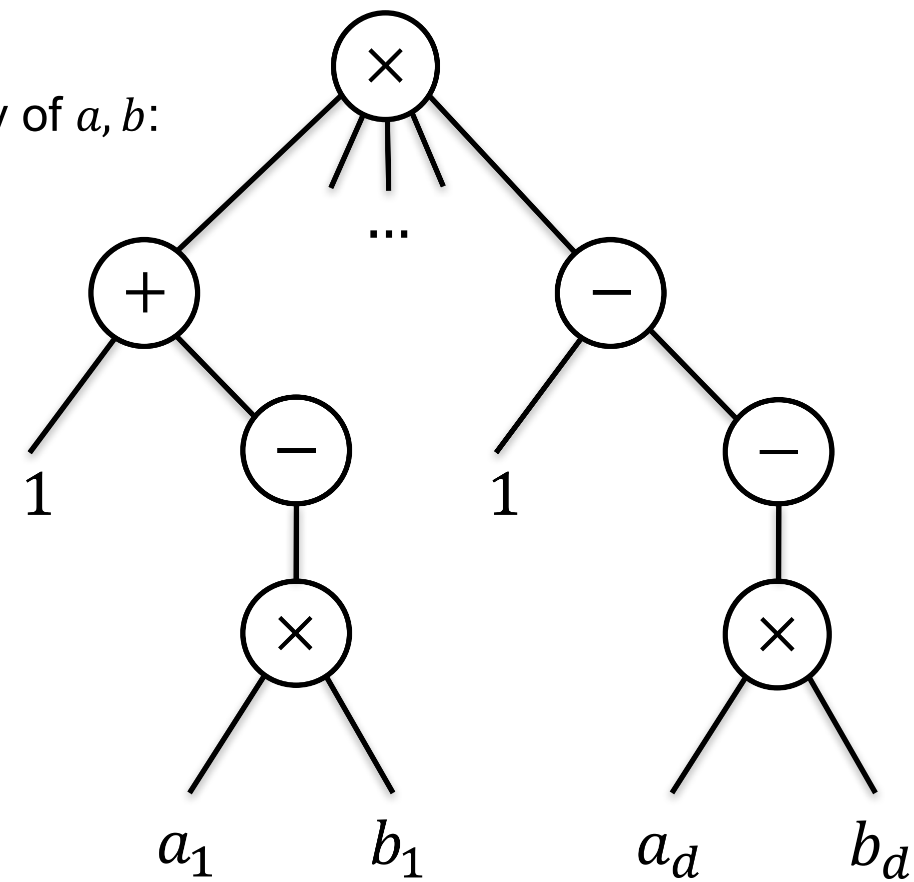
(for multivariate: degree d polynomial has up to $\binom{d+k+1}{k}$ monomials ☹)

OV as Multipoint Evaluation on Circuits

OV: Given sets $A, B \subseteq \{0,1\}^d$ of size n
Decide whether there are $a \in A, b \in B$ such that $a \perp b$

circuit $C(a,b)$ for testing orthogonality of a, b:

$$C(a,b) = \prod_{i=1}^{d} (1 - a_i b_i)$$

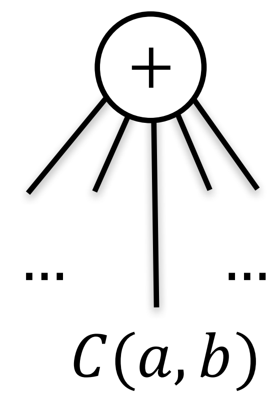


OV as Multipoint Evaluation on Circuits

OV: Given sets $A, B \subseteq \{0,1\}^d$ of size n
Decide whether there are $a \in A, b \in B$ such that $a \perp b$

circuit $C(a)$ for testing orthogonality of a with any $b \in B$:

$$C(a) = \sum_{b \in B} C(a,b) = \sum_{b \in B} \prod_{i=1}^{d} (1 - a_i b_i)$$



for each $b \in B$

- d inputs (for the coordinates of a)
- size $O(nd)$
- degree $\leq 2d$

- for any $a \in \{0,1\}^d$: $0 \leq C(a) \leq n$
- pick prime $p \geq n$ and work modulo p, i.e., over field $\mathbb{Z}_p$

(Co-)Nondet. Multipoint Evaluation on Circuits

Given circuit C on inputs $X_1, \dots, X_k$ with size s and degree d over $\mathbb{Z}_p$

Given inputs $z_1, \dots, z_n \in {\mathbb{Z}_p}^k$

want to evaluate C on each $z_j = (z_j[1], \dots, z_j[k])$

can assume $p \geq 2nd$, $p \leq n^{O(1)}$

1) compute polynomials $R_1(X), \dots, R_k(X)$ such that $R_i(j) = z_j[i]$ $\tilde{O}(kn)$

by polynomial interpolation

new goal: evaluate univariate circuit $C'(X) = C(R_1(X), \dots, R_k(X))$ on $X = 1, \dots, n$

C' has size $\leq O(s + kn)$ and depth $\leq dn$

2) guess a polynomial $Q(X)$ of degree at most dn $O(dn)$

3) check that $Q(X) = C(R_1(X), \dots, R_k(X))$

by "polynomial identity testing" $\tilde{O}(s + kn + dn)$

4) multipoint evaluate $Q(X)$ on $X = 1, \dots, n$ and return these values $\tilde{O}(dn)$

Co-Nondet. Algorithm for OV

use circuit $C(a)$ for testing orthogonality of a with any $b \in B \subseteq \{0,1\}^d$

evaluate $C(a)$ at each $a \in A$

d inputs, size $O(dn)$, degree $\leq 2d$

YES-instance: all paths accept **NO**-instance: at least one path rejects
...with high probability

1) compute polynomials $R_1(X), \ldots, R_k(X)$ such that $R_i(j) = z_j[i]$ $\tilde{O}(kn)$
by polynomial interpolation
new goal: evaluate univariate circuit $C'(X) = C(R_1(X), \ldots, R_k(X))$ on $X = 1, \ldots, n$
C' has size $\leq O(s + kn)$ and depth $\leq dn$

2) guess a polynomial $Q(X)$ of degree at most dn $O(dn)$

3) check that $Q(X) = C(R_1(X), \ldots, R_k(X))$ $\tilde{O}(s + kn + dn)$
by "polynomial identity testing", if not: ACCEPT

4) multipoint evaluate $Q(X)$ on $X = 1, \ldots, n$ and return these values $\tilde{O}(dn)$

5) ACCEPT if $\sum_{j=1}^{n} Q(j) \geq 1$ $\tilde{O}(dn)$

Co-Nondet. Algorithm for OV

NO-instance: If we correctly guess $Q(X)$ then the identity test works with prob. 1
and we correctly report non-existence of an orthogonal pair = REJECT

YES-instance: If we correctly guess $Q(X)$ then we ACCEPT
If we wrongly guess $Q(X)$ then identity test fails with prob. $1 - n^{-\Omega(1)}$
for any guess: we ACCEPT with probability $1 - n^{-\Omega(1)}$

1) compute polynomials $R_1(X), \dots, R_k(X)$ such that $R_i(j) = z_j[i]$ — $\tilde{O}(kn)$
by polynomial interpolation
new goal: evaluate univariate circuit $C'(X) = C(R_1(X), \dots, R_k(X))$ on $X = 1, \dots, n$
C' has size $\leq O(s + kn)$ and depth $\leq dn$

2) guess a polynomial $Q(X)$ of degree at most dn — $O(dn)$

3) check that $Q(X) = C(R_1(X), \dots, R_k(X))$
by "polynomial identity testing", if not: ACCEPT — $\tilde{O}(s + kn + dn)$

4) multipoint evaluate $Q(X)$ on $X = 1, \dots, n$ and return these values — $\tilde{O}(dn)$

5) ACCEPT if $\sum_{j=1}^{n} Q(j) \geq 1$ — $\tilde{O}(dn)$

max planck institut
informatik

Conclusion

Nondeterministic SETH: k-SAT has no no $O(2^{(1-\varepsilon)n})$ co-nondet. algorithm

No randomization allowed

If it holds, then there is no deterministic reduction from SETH to 3SUM,

since 3SUM has a $\tilde{O}(n^{3/2})$ co-nondeterministic algorithm

This is the only tool for ruling out reductions!

„Randomized Nondeterministic SETH“: is wrong!

We have seen a $\tilde{O}(dn)$ co-nondeterministic algorithm for OV

uses many tools for computing with polynomials and arithmetic circuits