max planck institut
informatik

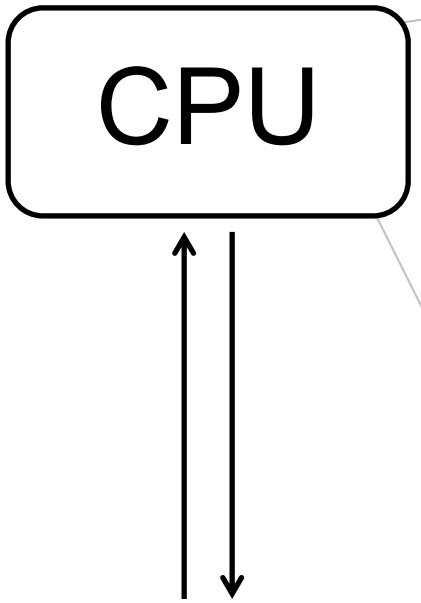
Fine-Grained Complexity - Hardness in P

Lecture 1: SETH and OV

Karl Bringmann

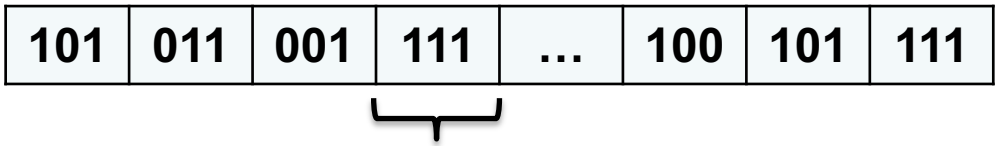
Machine Model

Random Access Machine (RAM):



- memory of $O(1)$ cells
- can perform all reasonable operations on two cells in $O(1)$ time (arithmetic + logical operations…)
- can read/write any cell in $O(1)$ time

storage:



cell / word

consisting of $\Theta(\log n)$ bits

can recognize **palindromes** in time $O(n)$

the details do not matter!

Reminder: SETH and OV

Problem k-SAT:

Given formula in k-CNF with n variables, is it satisfiable?

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_4 \vee x_5)$$

Strong Exponential Time Hypothesis: [Impagliazzo,Paturi'01]

$\forall \varepsilon > 0\ \exists k$: k-SAT has no $O(2^{(1-\varepsilon)n})$-time algorithm

SETH

⇓ [Williams'05]

LD-OVH

⇓

OVH

Problem Orthogonal Vectors:

Given sets $A, B \subseteq \{0,1\}^d$ of size n,
are any $a \in A, b \in B$ orthogonal? $(\sum_i a_i \cdot b_i = 0)$

OV-Hypothesis: (moderate dimension)

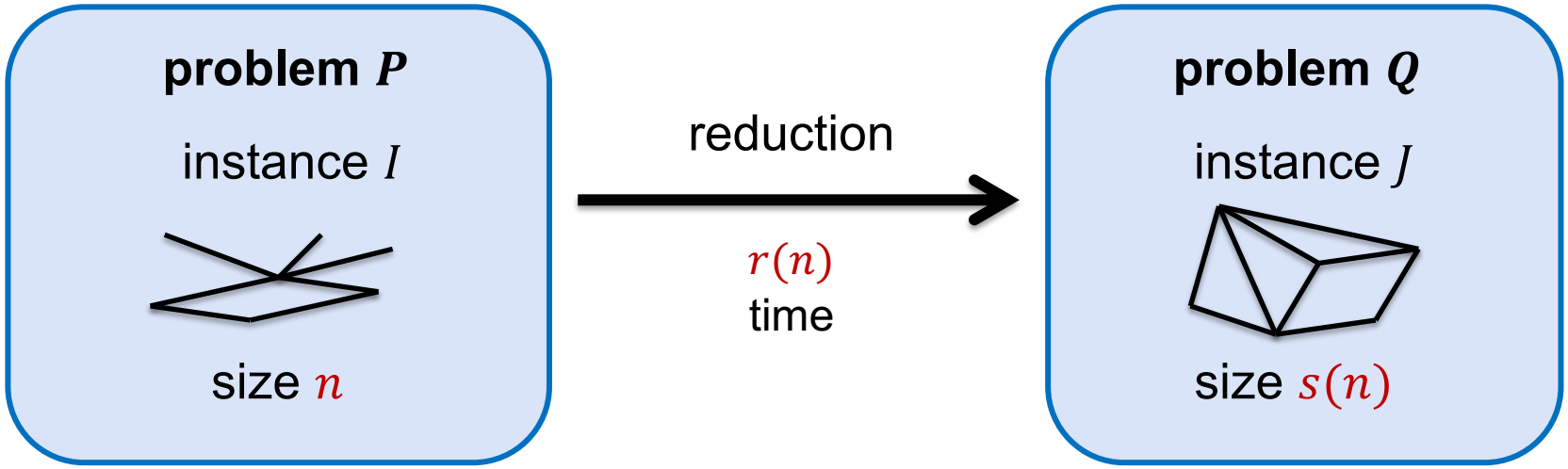
$\forall \varepsilon > 0$: OV has no $O(n^{2-\varepsilon} \cdot \mathrm{poly}(d))$-time algorithm

LD-OV-Hypothesis: (low dimension)

$\forall \varepsilon > 0\ \exists c > 0$: OV in $d = c \log n$ has no $O(n^{2-\varepsilon})$-time algorithm

Reminder: Fine-Grained Reductions

transfer hardness of one problem to another one by reductions



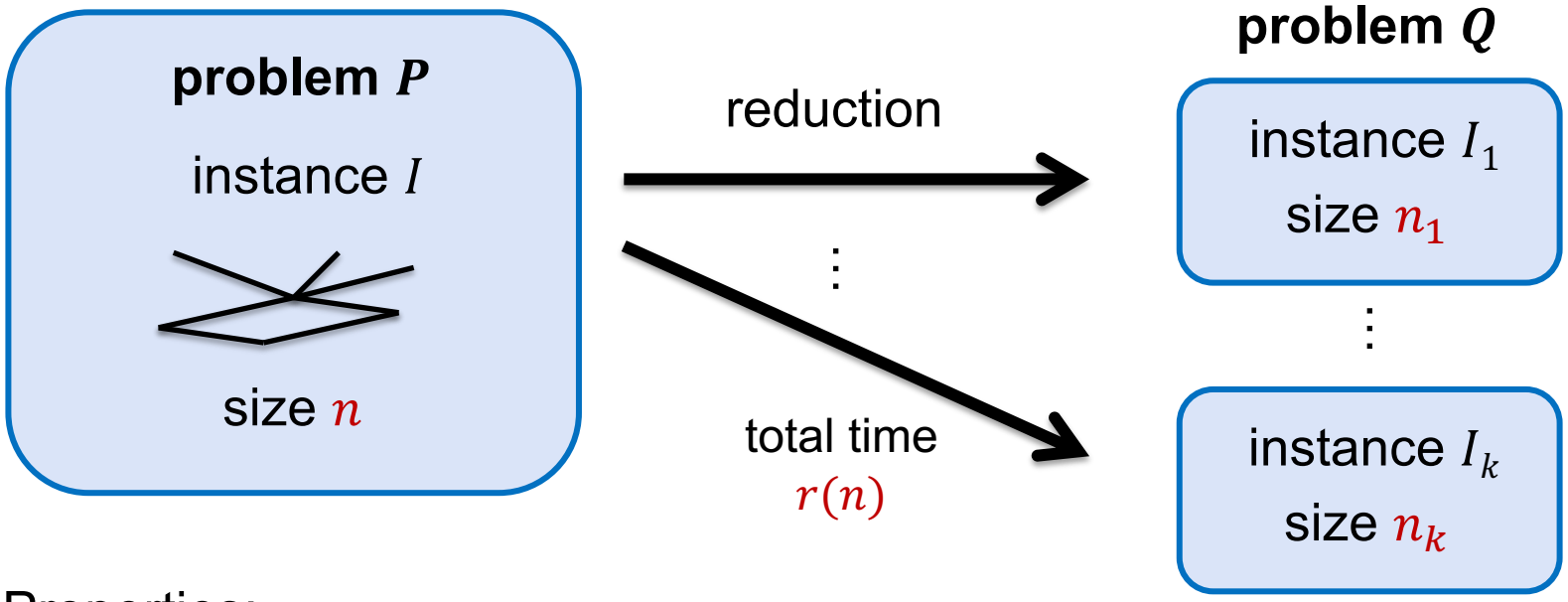
I is a 'yes'-instance $\iff$ J is a 'yes'-instance

$t(n)$ algorithm for Q implies a $r(n) + t(s(n))$ algorithm for P

if P has no $r(n) + t(s(n))$ algorithm then Q has no $t(n)$ algorithm

Reminder: Fine-Grained Reductions

A **fine-grained reduction** from (P, T) to (Q, T') is an algorithm A for P with **oracle** access to Q s.t.:



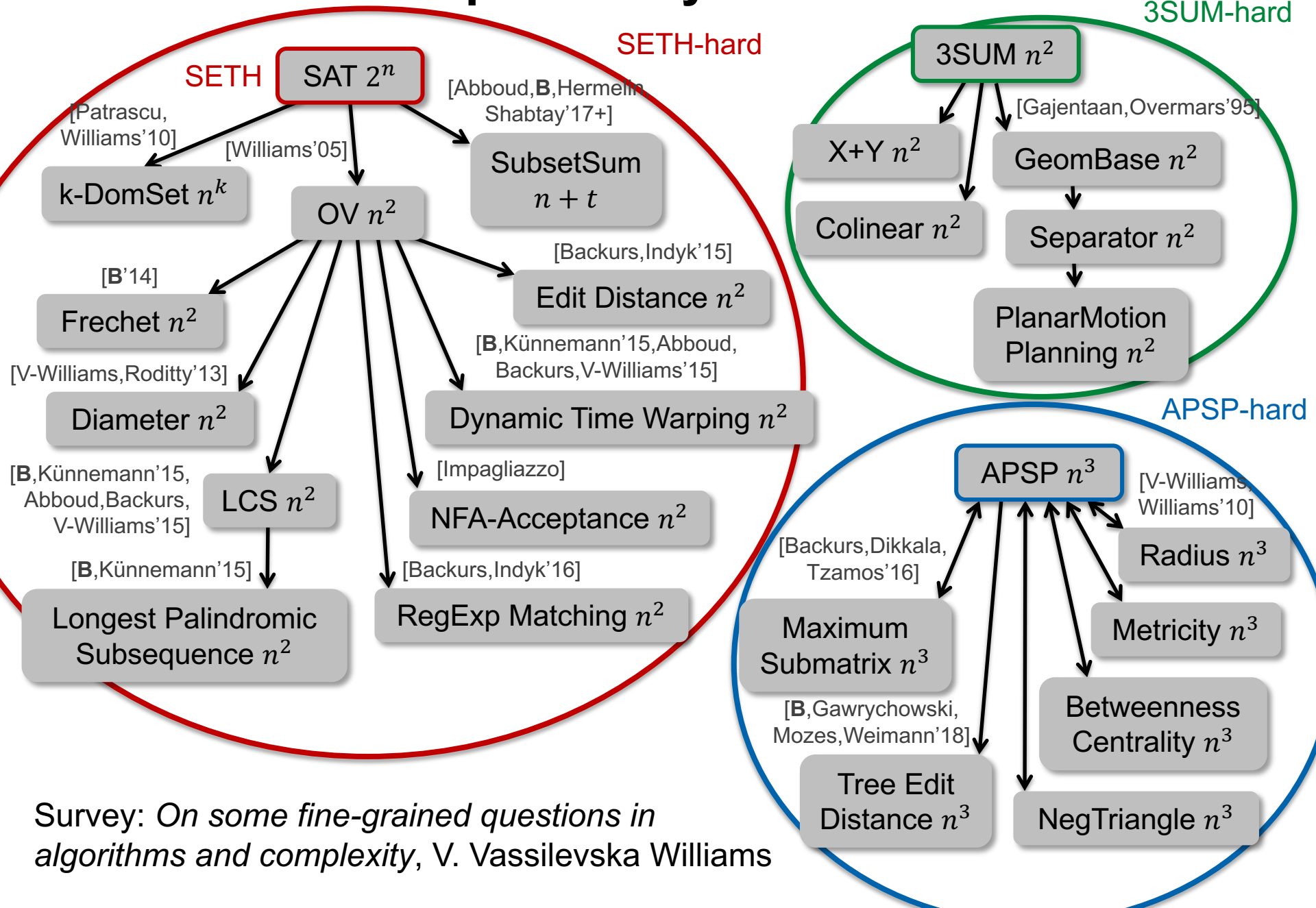
Properties:

for any instance I, algorithm $A(I)$ correctly solves problem P on I

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

for any $\varepsilon > 0$ there is a $\delta > 0$ s.t. $\sum_{i=1}^{k} T'(n_i)^{1-\varepsilon} \leq T(n)^{1-\delta}$

Landscape of Polytime Problems



Survey: *On some fine-grained questions in algorithms and complexity*, V. Vassilevska Williams

I. Easy Example

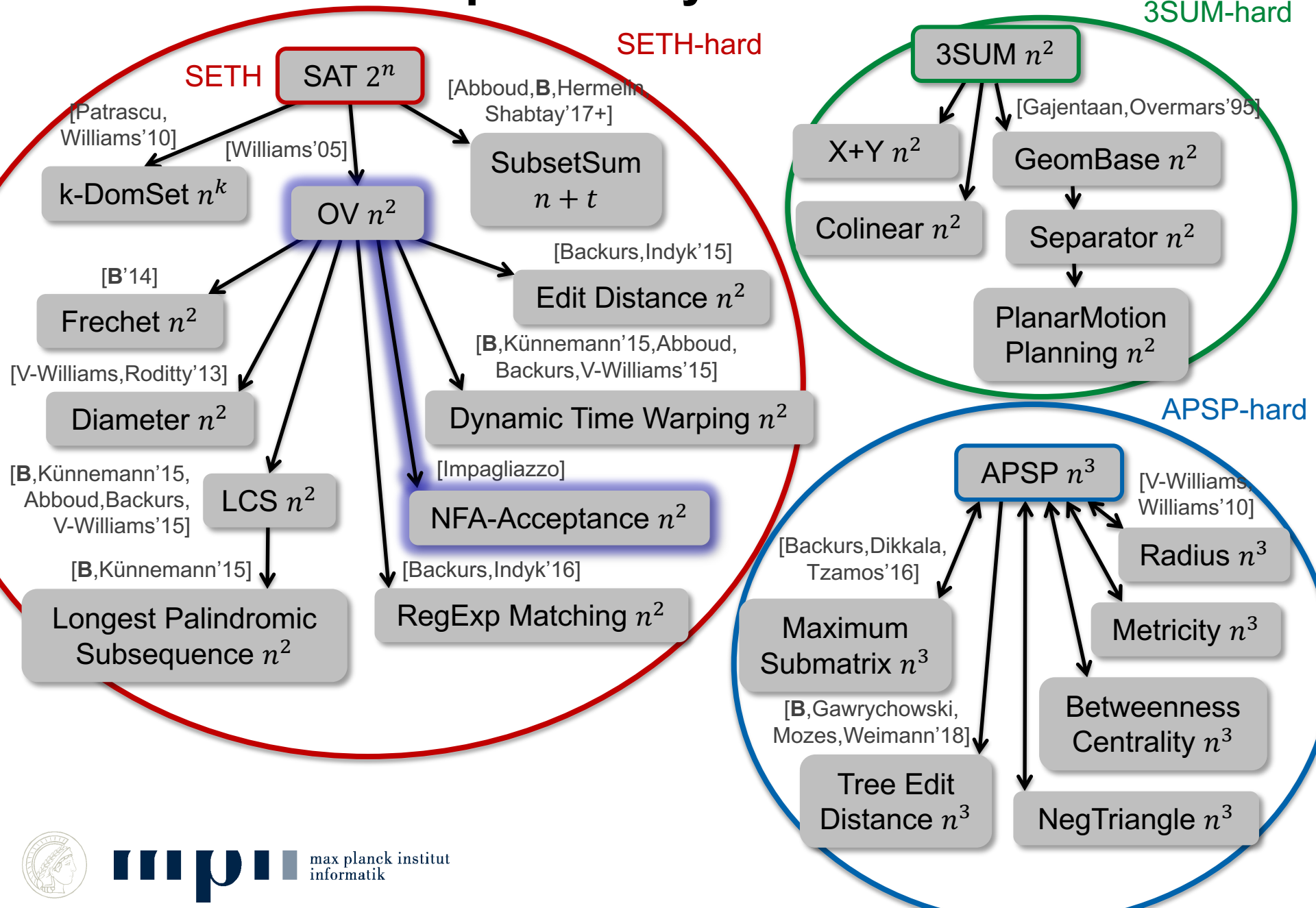
II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

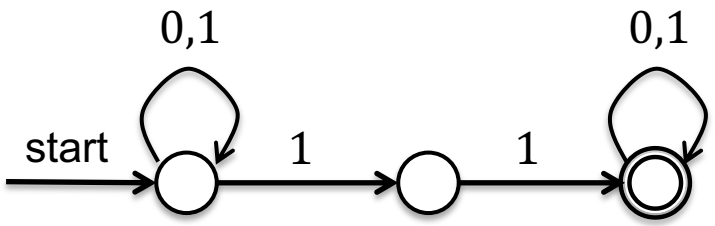
V. Summary

Landscape of Polytime Problems



NFA Acceptance Problem

nondeterministic finite automaton G accepts input string s if there is a walk in G from starting state to some accepting state, labelled with s



string: 01011010

dynamic programming algorithm in time $O(|s||G|)$:

$T[i] \coloneqq$ *set of states reachable via walks labelled with* $s[1..i]$

$T[0] \coloneqq \{\text{starting state}\}$

$T[i] \coloneqq \{v \mid \exists u \in T[i-1] \text{ and } \exists \text{ transition } u \to v \text{ labelled } s[i]\}$

OV-Hardness Result

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

time $O(dn)$

NFA acceptance

grammar G, string s

$|G| = O(dn)$

$|s| = O(dn)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Leftarrow$ $O((|s|\,|G|)^{1-\varepsilon})$ algorithm

Thm: NFA acceptance has no $O((|s|\,|G|)^{1-\varepsilon})$ algorithm unless OVH fails. [Impagliazzo]

Proof

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

time $O(dn)$

NFA acceptance

grammar G, string s

$|G| = O(dn)$

$|s| = O(dn)$

Proof:

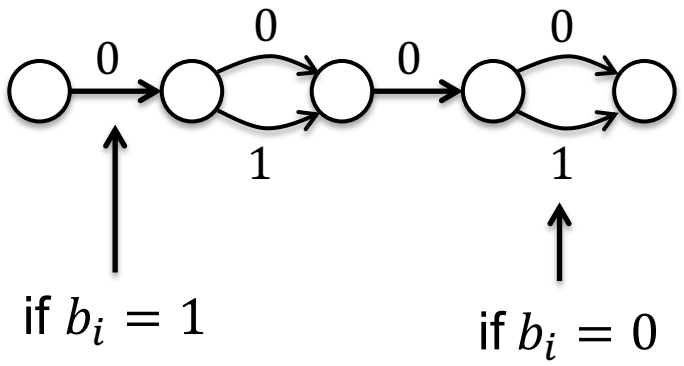
for any $a \in A$:

construct string:

$$0011$$
$$\uparrow$$
$$= a_1 a_2 \ldots a_d$$

for any $b \in B$:

construct part of NFA:



Proof

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

time $O(dn)$

NFA acceptance

grammar G, string s

$|G| = O(dn)$

$|s| = O(dn)$

Proof:

for any $a \in A$:

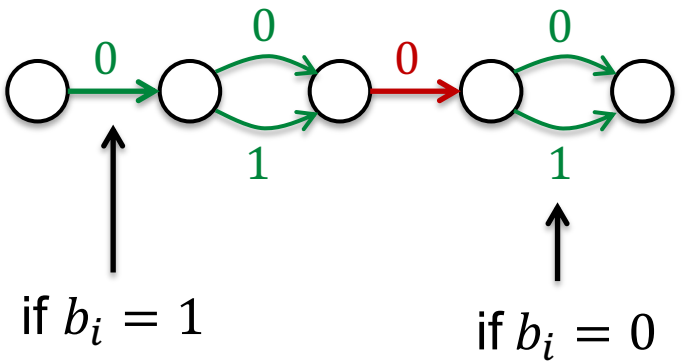
construct string:

0011

$= a_1 a_2 \ldots a_d$

for any $b \in B$:

construct part of NFA:



Proof

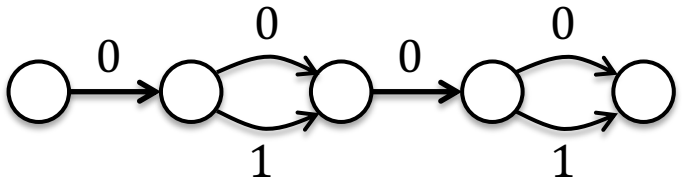
for any $a \in A$:

construct string:

0011

for any $b \in B$:

construct part of NFA:



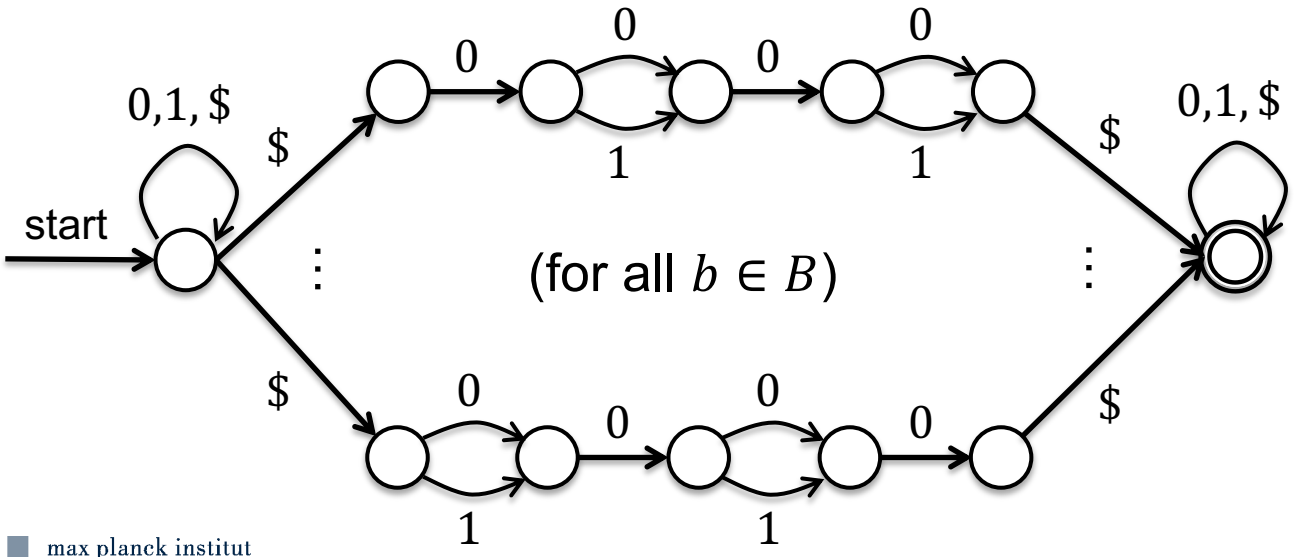
---

string $s = \$1100\$0110\$ \ldots \$0011\$$

(for all $a \in A$)

- equivalent to OV instance
- size $|s| = |G| = O(dn)$

NFA G:



OV-Hardness Result

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

time $O(dn)$

NFA acceptance

grammar G, string s

$|G| = O(dn)$

$|s| = O(dn)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Leftarrow$ $O((|s|\,|G|)^{1-\varepsilon})$ algorithm

Thm: NFA acceptance has no $O((|s|\,|G|)^{1-\varepsilon})$ algorithm unless OVH fails. [Impagliazzo]

I. Easy Example

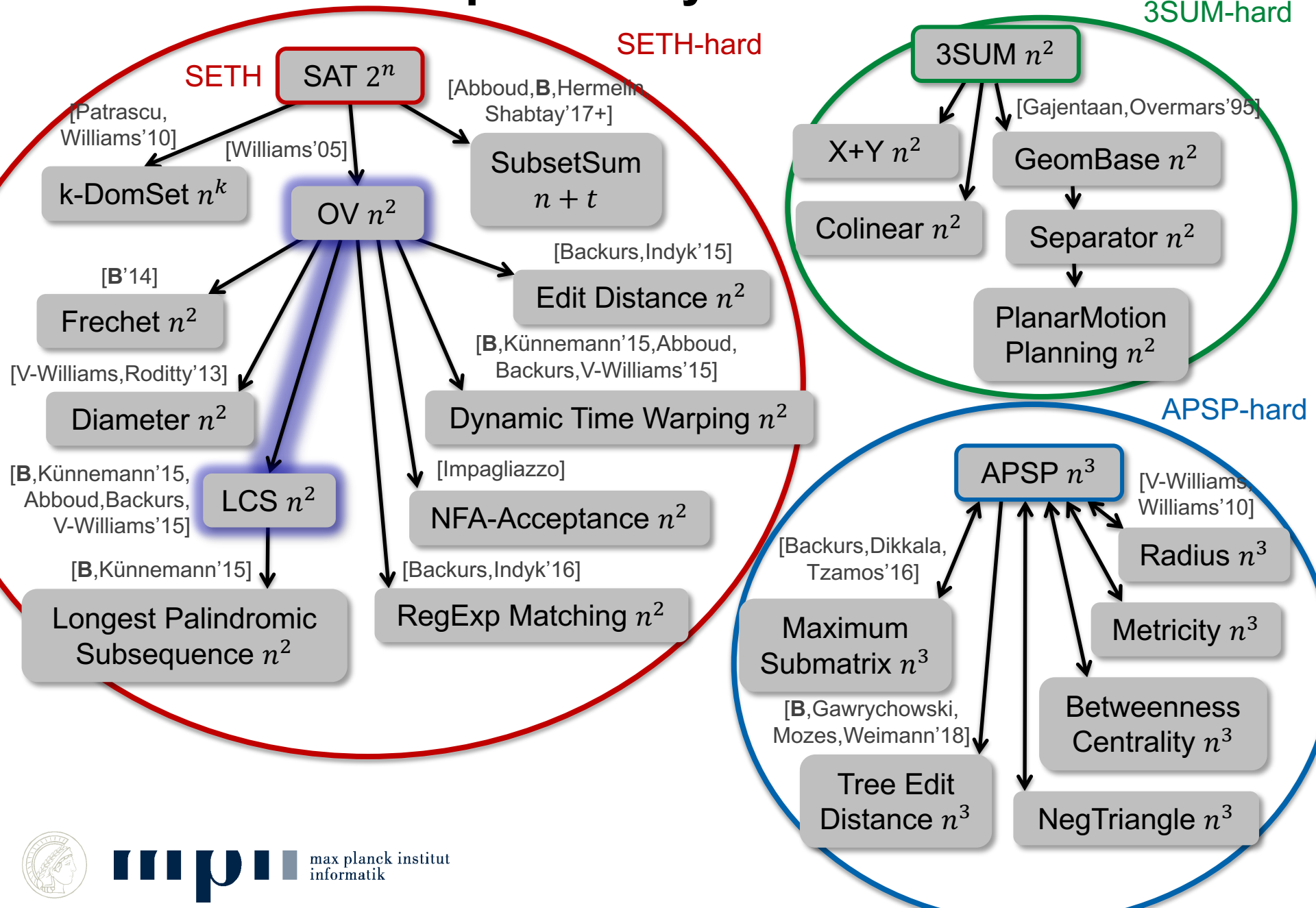
II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary

Landscape of Polytime Problems

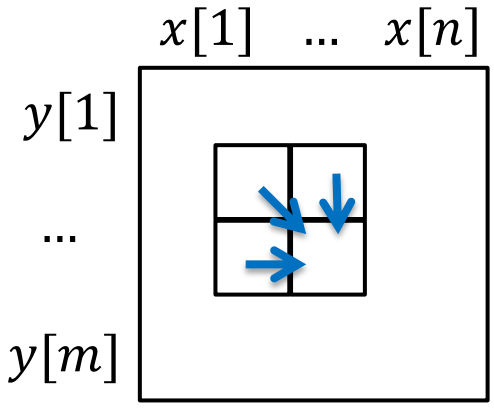


Longest Common Subsequence (LCS)

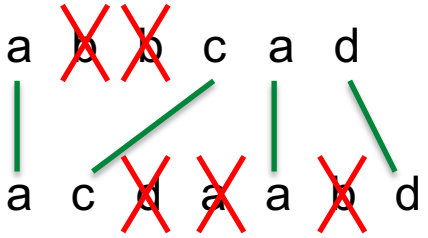
given strings x, y of length $n \geq m$, compute longest string z that is a subsequence of both x and y

write $LCS(x, y) = |z|$

natural dynamic program $O(n^2)$



$T[i,j] = LCS(x[1..i], y[1..j])$



$T[i,j] = \max\{T[i-1,j], T[i,j-1]\}$

if $x[i] = y[j]$:

$T[i,j] = \max\{T[i,j], T[i-1,j-1]+1\}$

logfactor improvement:

$O(n^2/\log^2 n)$

[Masek,Paterson'80]

→ *edit distance*

OV-Hardness of LCS

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

$\exists a \in A, b \in B: \forall i:$

$a_i \cdot b_i = 0$?

reduction

time $O(d^2 n)$

LCS

strings x, y

of length $O(d^2 n)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Longleftarrow$ $O(n^{2-\varepsilon})$ algorithm

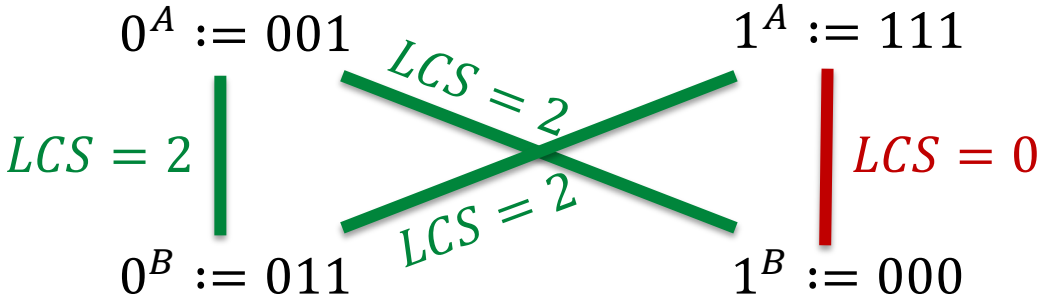
Thm: Longest Common Subsequence has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails.

[B.,Künnemann'15
Abboud,Backurs,V-Williams'15]

Proof: Coordinate Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

we want to simulate the **coordinates** $\{0,1\}$ and the behavior of $a_i \cdot b_i$



replace a_i by a_i^A and b_i by b_i^B

$LCS(a_i^A, b_i^B)$ can be written as $f(a_i \cdot b_i)$, with $f(0) > f(1)$

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$

in the picture: $d = 4$

concatenate $a_1^A, \dots, a_d^A$, padded with a new symbol 2

length $4d$

$$VG(a) := a_1{}^A\ 2 \dots 2\ a_2{}^A\ 2 \dots 2\ a_3{}^A\ 2 \dots 2\ a_4{}^A$$

$$VG(b) := b_1{}^B\ 2 \dots 2\ b_2{}^B\ 2 \dots 2\ b_3{}^B\ 2 \dots 2\ b_4{}^B$$

- no LCS matches symbols in $a_i{}^A$ with symbols in $b_j{}^B$ where $i \neq j$

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$

concatenate ${a_1}^A, \dots, {a_d}^A$, padded with a new symbol 2

length $4d$

$$VG(a) := {a_1}^A\ 2 \dots 2\ {a_2}^A\ 2 \dots 2\ {a_3}^A\ 2 \dots 2\ {a_4}^A$$

$$VG(b) := {b_1}^B\ 2 \dots 2\ {b_2}^B\ 2 \dots 2\ {b_3}^B\ 2 \dots 2\ {b_4}^B$$

- no LCS matches symbols in ${a_i}^A$ with symbols in ${b_j}^B$ where $i \neq j$
  - lose one block of 2's
  - cannot make up for it with symbols 0/1 since there are to few of them

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$

concatenate ${a_1}^A, \ldots, {a_d}^A$, padded with a new symbol 2

$$VG(a) := {a_1}^A\ 2 \ldots 2\ {a_2}^A\ 2 \ldots 2\ {a_3}^A\ 2 \ldots 2\ {a_4}^A$$

$$VG(b) := {b_1}^B\ 2 \ldots 2\ {b_2}^B\ 2 \ldots 2\ {b_3}^B\ 2 \ldots 2\ {b_4}^B$$

- no LCS matches symbols in ${a_i}^A$ with symbols in ${b_j}^B$ where $i \neq j$
- some LCS matches all 2‘s

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$

concatenate ${a_1}^A, \dots, {a_d}^A$, padded with a new symbol 2

$$VG(a) := {a_1}^A\ 2 \dots 2\ {a_2}^A\ 2 \dots 2\ {a_3}^A\ 2 \dots 2\ {a_4}^A$$

$$VG(b) := {b_1}^B\ 2 \dots 2\ {b_2}^B\ 2 \dots 2\ {b_3}^B\ 2 \dots 2\ {b_4}^B$$

- $LCS\big(VG(a), VG(b)\big) = (d-1)4d + \sum_{i=1}^{d} LCS({a_i}^A, {b_i}^B)$

#2's

$=f(a_i \cdot b_i)$

$LCS\big(VG(a), VG(b)\big) = C + 2$ if $a \perp b$

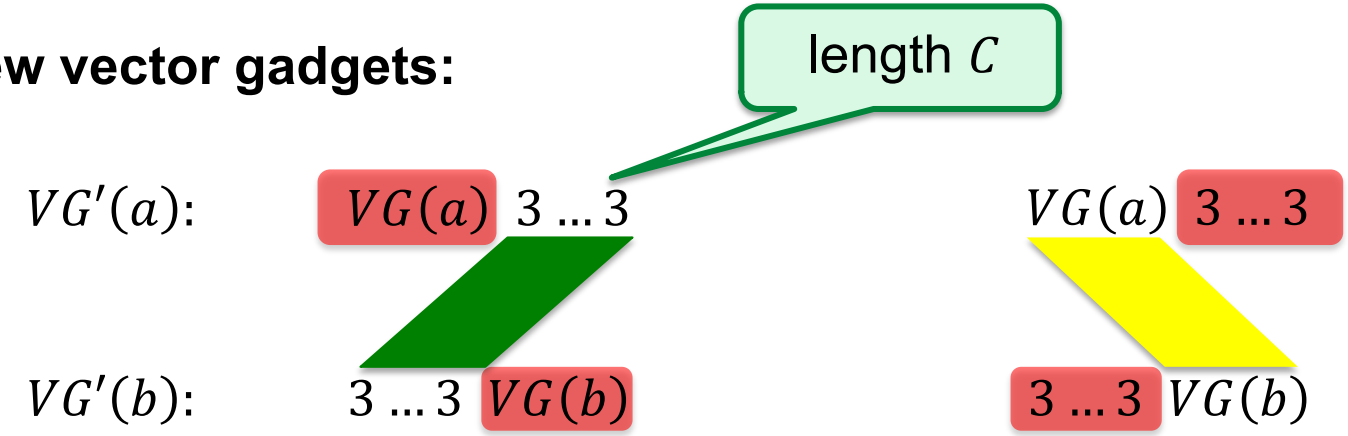
$LCS\big(VG(a), VG(b)\big) \leq C$ otherwise

for some constant C

Proof: Normalized Vectors Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i: \; a_i \cdot b_i = 0$

new vector gadgets:



$$LCS(VG'(a), VG'(b)) = \max\{LCS\big(VG(a), VG(b)\big), C\}$$

$$LCS(VG'(a), VG'(b)) = \begin{cases} C+2 & \text{if } a \perp b \\ C & \text{otherwise} \end{cases}$$

write VG for VG'

Proof: OR-Gadget

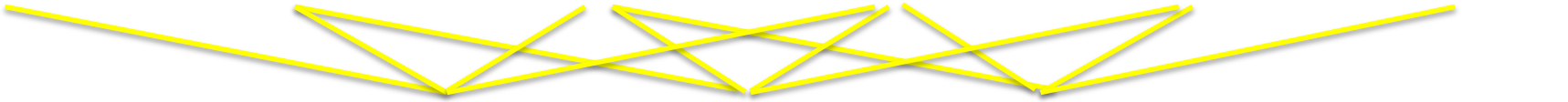
OV: Given $A, B \subseteq \{0,1\}^d$ of size n each

Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

fresh symbol 4, want to construct:

in the picture: $n = 3$

$VG(A[1])\ 4 \ldots 4\ VG(A[2])\ 4 \ldots 4\ VG(A[3])\ 4 \ldots 4\ VG(A[1])\ 4 \ldots 4\ VG(A[2])\ 4 \ldots 4\ VG(A[3])$



$4 \ldots \ldots \ldots \ldots \ldots 4\ VG(B[1])\ 4 \ldots 4\ VG(B[2])\ 4 \ldots 4\ VG(B[3])\ 4 \ldots \ldots \ldots \ldots \ldots 4$

length $100d^2$

length $100d^2 \cdot 2n$

Proof: OR-Gadget

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each

Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

fresh symbol 4, want to construct: in the picture: $n = 3$

can align $VG(B[j])$ with $VG(A[\Delta + j \bmod n])$ for any offset Δ

$$LCS \geq \underbrace{(2n-1)100d^2}_{\#4\text{'s in upper string}} + \max_{\Delta} \sum_{j=1}^{n} LCS(VG(A[\Delta + j \bmod n]), VG(B[j]))$$

maximize over offset

need normalization!

If there is an orthogonal pair, some offset Δ aligns this pair, and we get

$$LCS \geq (2n-1)100d^2 + nC + 2$$

Proof: OR-Gadget

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each

Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

fresh symbol 4, want to construct: in the picture: $n = 3$

if an orthogonal pair exists then $LCS \geq (2n-1)100d^2 + nC + 2$

Claim: otherwise: $LCS \leq (2n-1)100d^2 + nC$

this finishes the proof: - equivalent to OV instance

- length $O(d^2 n)$

Proof of Claim

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

Claim: if no orthogonal pair exists: $LCS \leq (2n-1)100d^2 + nC$

$VG(A[1])\ 4 \ldots 4\ VG(A[2])\ 4 \ldots 4\ VG(A[3])\ 4 \ldots 4\ VG(A[1])\ 4 \ldots 4\ VG(A[2])\ 4 \ldots 4\ VG(A[3])$

$4 \ldots \ldots \ldots \ldots \ldots 4\ VG(B[1])\ 4 \ldots 4\ VG(B[2])\ 4 \ldots 4\ VG(B[3])\ 4 \ldots \ldots \ldots \ldots \ldots 4$

consider how an LCS matches the $VG(B[j])$

- no crossings

Proof of Claim

Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

Claim: if no orthogonal pair exists: $LCS \leq (2n-1)100d^2 + nC$

$VG(A[1])\ 4 \ldots 4\ VG(A[2])\ 4 \ldots 4\ VG(A[3])\ 4 \ldots 4\ VG(A[1])\ 4 \ldots 4\ VG(A[2])\ 4 \ldots 4\ VG(A[3])$

$4 \ldots \ldots \ldots \ldots \ldots 4\ VG(B[1])\ 4 \ldots 4\ VG(B[2])\ 4 \ldots 4\ VG(B[3])\ 4 \ldots \ldots \ldots \ldots \ldots 4$

non-orthogonal

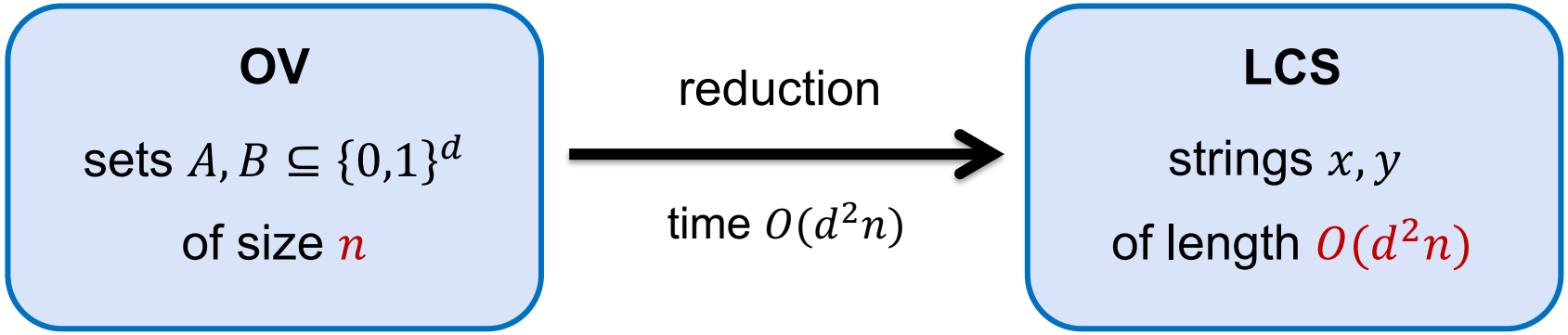
$$LCS \leq (2n-1)100d^2 + \sum_{j=1}^{n} \begin{cases} 0 & \text{if } VG(B[j]) \text{ is not matched} \\ C & \text{if } VG(B[j]) \text{ is matched to one} \\ |VG(B[j])| - |4 \ldots 4| & \text{if } VG(B[j]) \text{ is matched to } > 1 \end{cases}$$

#4's in upper string

could match VG completely, but loose many 4's

≤ 0

OV-Hardness of LCS



$O(n^{2-\varepsilon}\mathrm{poly}(d))$ algorithm $\Leftarrow$ $O(n^{2-\varepsilon})$ algorithm

Thm: Longest Common Subsequence has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails.

[B.,Künnemann'15
Abboud,Backurs,V-Williams'15]

Extensions

similar problems: edit distance, dynamic time warping, …

[Backurs,Indyk'15] [B.,Künnemann'15,Abboud,Backurs,V-Williams'15]

harder hardness: reductions from Formula-SAT / branching programs

[Abboud,Hansen,V-Williams,Williams'16] [Abboud,B.'18]

LCS-hard problems: longest palindromic subseq., longest tandem subseq.

[B.,Künnemann'15]

alphabet size: [B.,Künnemann'15]

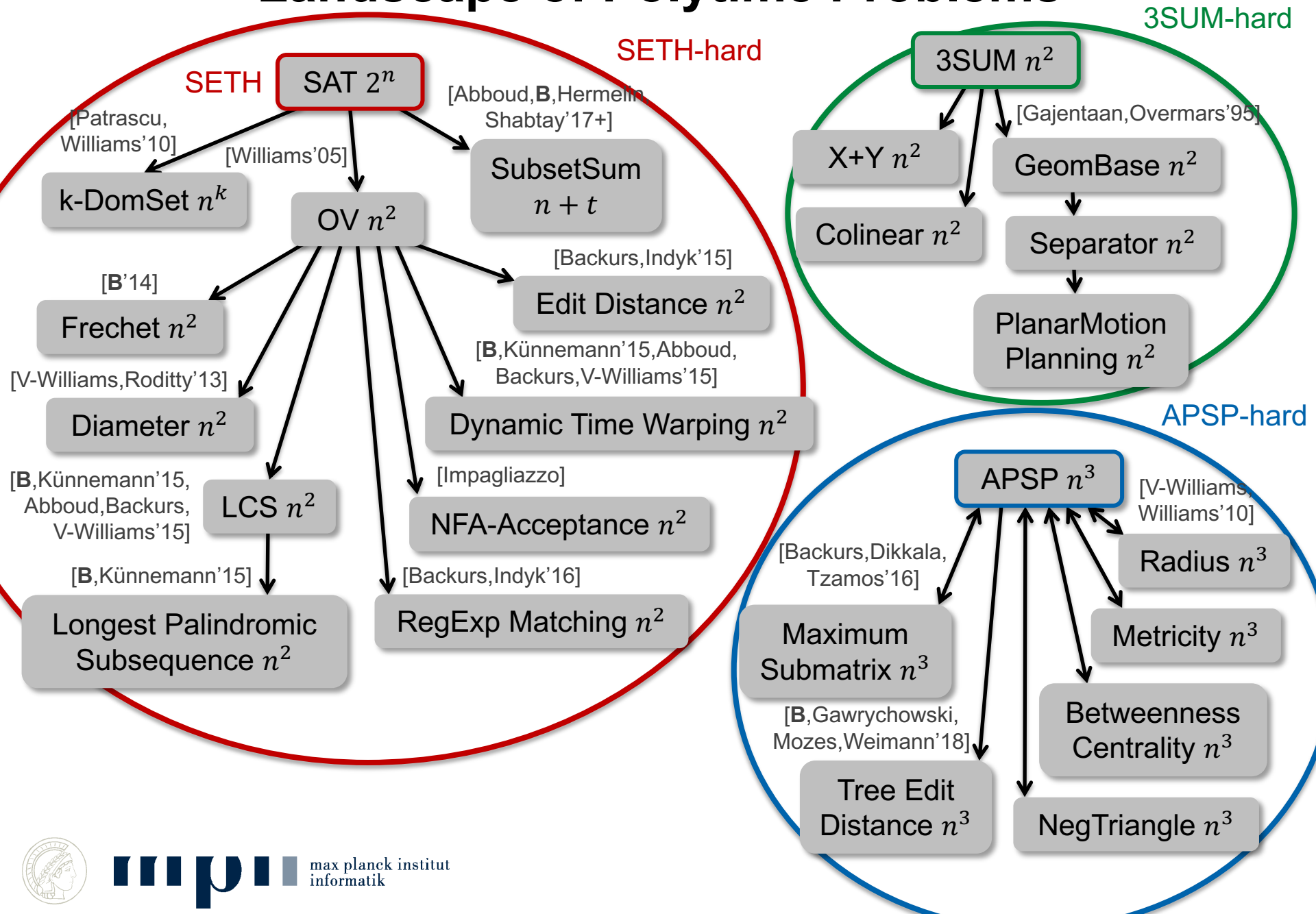
LCS and edit distance are hard on *binary* strings, i.e., alphabet $\{0,1\}$

k-LCS: LCS of k strings takes time $\Omega(n^{k-\varepsilon})$ [Abboud,Backurs,V-Williams'15]

Based on k-OV

max planck institut informatik

Landscape of Polytime Problems



I. Easy Example

II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary

Hardness of Approximation in P

„deterministic subquadratic-time approximation algorithms imply circuit lower bounds“

[Abboud,Backurs ITCS'17
Abboud,Rubinstein ITCS'18]

e.g. Longest Common Subsequence

„subquadratic-time approximation algorithms violate SETH“

e.g. Max-Inner-Product

[Abboud,Rubinstein,Williams FOCS'17
Rubinstein STOC'18
Chen CCC'18
Chen,Goldwasser,Lyu,Rothblum,Rubinstein Arxiv'18]

„FPT-approximation algorithms violate SETH“

[Chalermsook,Cygan,Kortsarz,Laekhanukit,Manurangsi,Nanongkai,Trevisan FOCS'17
Karthik,Laekhanukit,Manurangsi STOC'18]

Dominating Set

max planck institut
informatik

Hardness of Approximation in P

Problem Max-Inner-Product:

Given sets $A, B \subseteq \{0,1\}^d$ of size n,

compute $\max_{a \in A, b \in B} \langle a, b \rangle$

„optimization version of OV“

Is in time $O(n^2 d)$, but not in time $O(n^{2-\varepsilon} \cdot \mathrm{poly}(d))$ assuming OV-H

Thm: Let $\log n \ll d \leq n^{o(1)}$. [Chen CCC'18]

1) $\forall \delta > 0 \; \exists \varepsilon > 0$: A $(d/\log n)^{\delta}$-multiplicative approximation to Max-IP can be computed in time $O\left(n^{2-\varepsilon} \cdot \mathrm{poly}(d)\right)$

2) If $\exists \varepsilon > 0 \; \forall \delta > 0$: a $(d/\log n)^{\delta}$-multiplicative approximation to Max-IP can be computed in time $O\left(n^{2-\varepsilon} \cdot \mathrm{poly}(d)\right)$, then LD-OVH fails

max planck institut informatik

tight hardness of approximation!

Tool: Communication Complexity

Input: $x \in \{0,1\}^d$ Input: $y \in \{0,1\}^d$

want to compute $f(x, y)$

with as few communication as possible

e.g. $f = Disj$:

$$Disj(x, y) = \begin{cases} 1 & \text{if } \langle x, y \rangle = 0 \\ 0 & \text{otherwise} \end{cases}$$

deterministic communication complexity of $Disj$ is $\Theta(d)$

randomized communication complexity of $Disj$ is $\Theta(d)$

Communication Complexity Setup

non-deterministic
randomized
one-way
communication:

random-ness

R

B

M

In: $x \in \{0,1\}^d$

In: $y \in \{0,1\}^d$

accept/
reject

Merlin:

powerful,
untrusted,
sends advice,
knows x, y

Messages M, R, B
of bitlength m, r, b

We require:

Efficiency: Alice and Bob are efficiently computable

Completeness: If $f(x,y) = 1$:
$\exists M\colon \Pr_R[\text{Alice accepts}] = 1$

Soundness: If $f(x,y) = 0$:
$\forall M\colon \Pr_R[\text{Alice accepts}] \leq p$

Then we say that f has an (m, r, b, p)-*efficient protocol*

From CC to Reductions

Any (m, r, b, p)-efficient protocol for $Disj$ yields a reduction:

OV
n
d

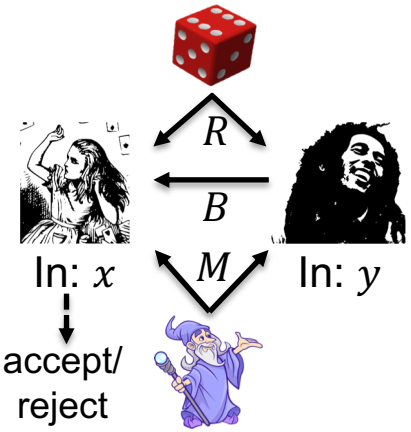
constructs 2^m instances of Max-IP with gap p

time $O(2^m \cdot n \cdot \text{poly}(d))$

Max-IP
$n' = n$
$d' = 2^{r+b}$

YES-instance $\Rightarrow$ Max-IP $\geq 2^r$ for some instance

NO-instance $\Rightarrow$ Max-IP $\leq p \cdot 2^r$ for all instances



Alice and Bob efficient

Complete: If $f(x, y) = 1$:
$\exists M$: $\Pr_R[\text{Alice accepts}] = 1$

Sound: If $f(x, y) = 0$:
$\forall M$: $\Pr_R[\text{Alice accepts}] \leq p$

From CC to Reductions

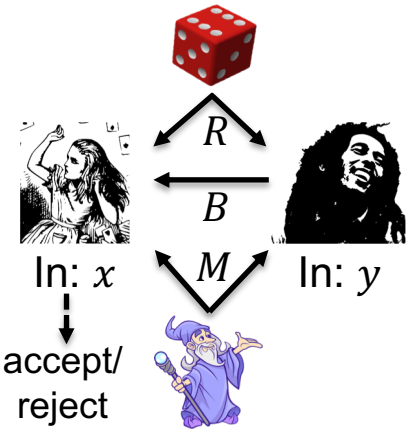
Any (m, r, b, p)-efficient protocol for $Disj$ yields a reduction:

OV
n
d

constructs 2^m instances of Max-IP with gap p

time $O(2^m \cdot n \cdot \text{poly}(d))$

Max-IP
$n' = n$
$d' = 2^{r+b}$



Alice and Bob efficient

Complete: If $f(x, y) = 1$:
$\exists M$: $\Pr_R[\text{Alice accepts}] = 1$

Sound: If $f(x, y) = 0$:
$\forall M$: $\Pr_R[\text{Alice accepts}] \leq p$

Proof: Given OV instance $X, Y \subseteq \{0,1\}^d$

Iterate over all advice strings $M \in [2^m]$:

Construct Max-IP instance X^M, Y^M:

$\forall x \in X$: $\forall R \in [2^r]$: create $x^{M,R} \in \{0,1\}^{2^b}$ with

$x_B^{M,R}$ = whether Alice accepts on input x, advice M, randomness R and Bob's message B

$\forall y \in Y$: $\forall R \in [2^r]$: create $y^{M,R} \in \{0,1\}^{2^b}$ with

$y_B^{M,R}$ = whether Bob sends message B on input y, advice M and randomness R

Then $\langle x^{M,R}, y^{M,R} \rangle$ = whether Alice accepts on inputs x, y, advice M, randomness R

From CC to Reductions

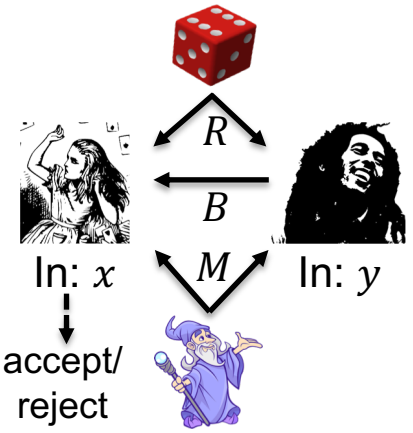
Any (m, r, b, p)-efficient protocol for $Disj$ yields a reduction:

OV
n
d

constructs 2^m instances of Max-IP with gap p

time $O(2^m \cdot n \cdot \text{poly}(d))$

Max-IP
$n' = n$
$d' = 2^{r+b}$



Alice and Bob efficient

Complete: If $f(x, y) = 1$:
$\exists M$: $\Pr_R[\text{Alice accepts}] = 1$

Sound: If $f(x, y) = 0$:
$\forall M$: $\Pr_R[\text{Alice accepts}] \leq p$

Proof: Given OV instance $X, Y \subseteq \{0,1\}^d$

Iterate over all advice strings $M \in [2^m]$:

Construct Max-IP instance X^M, Y^M:

$\forall x \in X$: create x^M by concatenating all $x^{M,R}$

$\forall y \in Y$: create y^M by concatenating all $y^{M,R}$

Then $\langle x^M, y^M \rangle = 2^r \cdot \Pr_R[\text{Alice accepts on inputs } x, y \text{ and advice } M]$

Then $\langle x^{M,R}, y^{M,R} \rangle =$ whether Alice accepts on inputs x, y, advice M, randomness R

From CC to Reductions

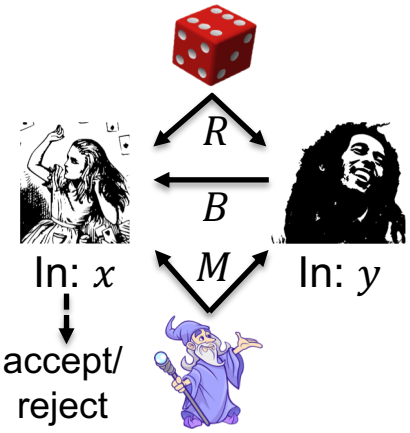
Any (m, r, b, p)-efficient protocol for $Disj$ yields a reduction:

OV
n
d

constructs 2^m instances of Max-IP with gap p

time $O(2^m \cdot n \cdot \text{poly}(d))$

Max-IP
$n' = n$
$d' = 2^{r+b}$



Alice and Bob efficient

Complete: If $f(x, y) = 1$:
$\exists M$: $\Pr_R[\text{Alice accepts}] = 1$

Sound: If $f(x, y) = 0$:
$\forall M$: $\Pr_R[\text{Alice accepts}] \leq p$

Proof: Given OV instance $X, Y \subseteq \{0,1\}^d$

Iterate over all advice strings $M \in [2^m]$:

Construct Max-IP instance X^M, Y^M:

$\forall x \in X$: create x^M by concatenating all $x^{M,R}$

$\forall y \in Y$: create y^M by concatenating all $y^{M,R}$

Then $\langle x^M, y^M \rangle = 2^r \cdot \Pr_R[\text{Alice accepts on inputs } x, y \text{ and advice } M]$

If $\exists x \in X, y \in Y$: $\langle x, y \rangle = 0$ then: $\exists M$: $\langle x^M, y^M \rangle = 2^r$

Otherwise: $\forall x \in X, y \in Y$: $\forall M$: $\langle x^M, y^M \rangle \leq p \cdot 2^r$

From CC to Reductions

Best known protocol for $Disj$:

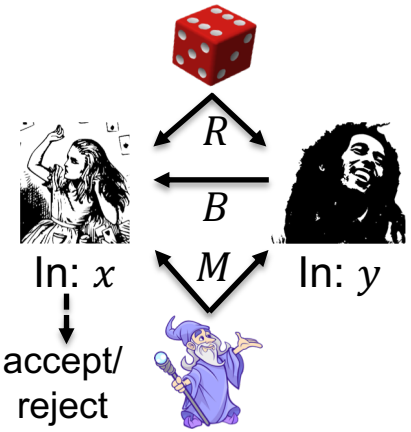
For any d, α and $p < 1/2$: [Chen CCC'18]

$\left(\frac{d}{\alpha}, \log_2 d + O\left(\log\frac{1}{p}\right), \mathrm{poly}(\alpha) \cdot \log\frac{1}{p}, p\right)$-efficient protocol

Complicated: algebraic codes, expander mixing lemma, ...

Easier: [Aaronson,Wigderson'09]

$(m, r, b, p) = \left(o(d), o(d), o(d), \frac{1}{2}\right)$-efficient protocol



Alice and Bob efficient

Complete: If $f(x,y) = 1$:
$\exists M$: $\Pr_R[\text{Alice accepts}] = 1$

Sound: If $f(x,y) = 0$:
$\forall M$: $\Pr_R[\text{Alice accepts}] \leq p$

Implies reduction:

OV
n
$d = c \log n$

constructs $n^{o(1)}$ instances of Max-IP with gap $1/2$

time $O(n^{1+o(1)} \cdot \mathrm{poly}(d))$

Max-IP
$n' = n$
$d' = n^{o(1)}$

$2^m, 2^{r+b} = n^{o(1)}$

[Abboud,Rubinstein,Williams'17]

So a 1.99-approximation of Max-IP in dimension $d = n^{o(1)}$ in time $O(n^{2-\varepsilon})$ violates LD-OVH

The Disj-Protocol

[Aaronson, Wigderson'09]

$(m, r, b, p) = \left(\tilde{O}(\sqrt{d}), O(\log d), \tilde{O}(\sqrt{d}), \frac{1}{2}\right)$-efficient protocol

Proof: Fix prime $p \in (4d, 8d]$, $T := \sqrt{d}$, assume that $T \in \mathbb{N}$

Given x, y compute polynomials of degree $< d/T$ over $\mathbb{Z}_p$:

$\psi_{x,t}(Z)$ s.t. $\psi_{x,t}(i) = x_{i \cdot T + t}$ for $0 \leq i < d/T$

$\psi_{y,t}(Z)$ s.t. $\psi_{y,t}(i) = y_{i \cdot T + t}$ for $0 \leq i < d/T$

Define $\Phi(Z) := \sum_{t=1}^{T} \psi_{x,t}(Z) \cdot \psi_{y,t}(Z)$ of degree $< 2d/T$

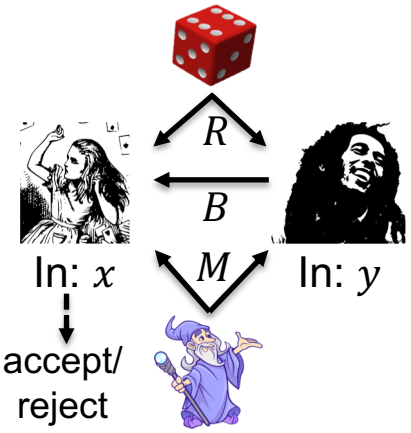
Protocol:

randomness: $R \in \mathbb{Z}_p$ uniformly at random

Bob: sends $\psi_{y,1}(R), \ldots, \psi_{y,T}(R)$

Merlin: sends polynomial $P(Z)$ of degree $< 2d/T$ over $\mathbb{Z}_p$
„good proof": $P(Z) = \Phi(Z)$

Alice: accepts iff $P(i) = 0$ for all $0 \leq i < d/T$ and $P(R) = \sum_{t=1}^{T} \psi_{x,t}(R) \cdot \psi_{y,t}(R)$



Alice and Bob efficient

Complete: If $f(x, y) = 1$:
$\exists M$: $\Pr_R[\text{Alice accepts}] = 1$

Sound: If $f(x, y) = 0$:
$\forall M$: $\Pr_R[\text{Alice accepts}] \leq p$

x, y disjoint iff
$\Phi(i) := 0$ for
all $0 \leq i < d/T$

Hardness of Approximation in P

Problem Max-Inner-Product:

Given sets $A, B \subseteq \{0,1\}^d$ of size n,

compute $\max_{a \in A, b \in B} \langle a, b \rangle$

„optimization version of OV“

Is in time $O(n^2 d)$, but not in time $O(n^{2-\varepsilon} \cdot \mathrm{poly}(d))$ assuming OV-H

Thm: Let $\log n \ll d \leq n^{o(1)}$. [Chen CCC'18]

1) $\forall \delta > 0 \; \exists \varepsilon > 0$: A $(d/\log n)^{\delta}$-multiplicative approximation to Max-IP can be computed in time $O(n^{2-\varepsilon} \cdot \mathrm{poly}(d))$

2) If $\exists \varepsilon > 0 \; \forall \delta > 0$: a $(d/\log n)^{\delta}$-multiplicative approximation to Max-IP can be computed in time $O(n^{2-\varepsilon} \cdot \mathrm{poly}(d))$, then LD-OVH fails

I. Easy Example

II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary

Mutivariate Algorithms for LCS

parameters for LCS: [B.,Künnemann'18]

$n = \|x\|$	.. length of longer string
$m = \|y\|$	.. length of shorter string
$L = LCS(x, y)$	.. length of LCS
$\|\Sigma\|$	.. size of alphabet Σ
$\Delta = n - L$	.. number of deletions in x
$\delta = m - L$	.. number of deletions in y
M	.. number of *matching pairs*
d	.. number of *dominant pairs*

multivariate algorithms: $\tilde{O}(n + \min\{d, \delta m, \delta\Delta\})$

under SETH, this is **optimal** for any relations $m = \Theta(n^{\alpha_m}), L = \Theta(n^{\alpha_L}), \ldots$

max planck institut informatik

Grammar-Compressed Strings

String T given by **Straight-Line Program**: [Abboud,Backurs,B.,Künnemann'17]

Nonterminals $S_1, \dots, S_n$

Each S_i is associated with a rule:

$S_i \to c$ for some alphabet symbol/terminal c

or

$S_i \to S_\ell S_r$ for some $\ell, r < i$

T is the string generated by S_n

compressed size n = number of nonterminals

uncompressed size N = length of T

$N \geq n$, and N can be as large as $2^{\Omega(n)}$

Computing the LCS-length of strings x, y of length N compressed to size n is in time $\tilde{O}(Nn)$…

[Tiskin'08] [Gawrychowski'12]

… and not in $O\big((Nn)^{1-\varepsilon}\big)$ time assuming SETH

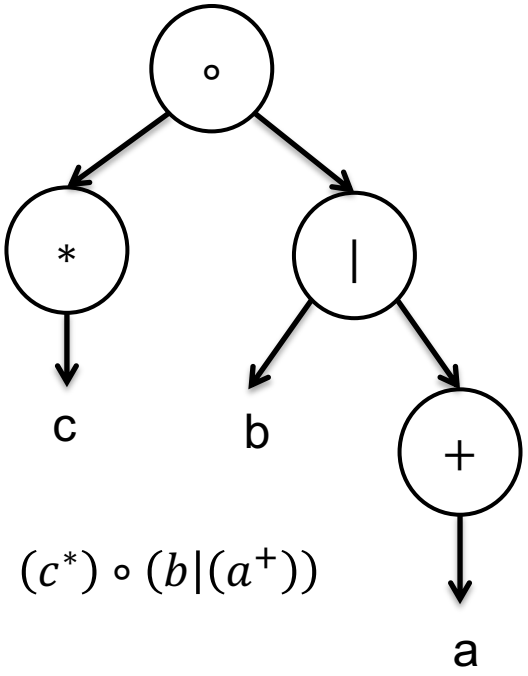
[Abboud,Backurs,B.,Künnemann'17]

Regular Expression Matching

describe patterns using the operations:

- concatenation ∘
- alternation | (OR)
- Kleene star $*$ (repetition with empty string)
- Kleene plus $+$ (repetition without empty string)

starting from alphabet symbols in Σ



$(c^*) \circ (b|(a^+))$

great expressiveness: describe regular languages

standard tool to describe pattern matching problems on strings

Regular Expression Matching

$n = |S|, m = |R|$

Reg-Exp *Pattern Matching*:
given reg-exp R and string S, does some substring of S match R?

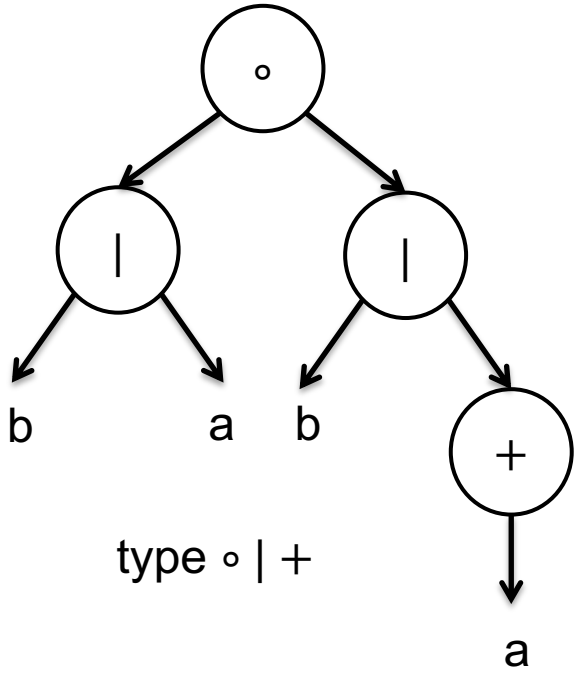
Reg-Exp *Membership Testing*:
given reg-exp R and string S, does S match R?

$O(nm)$	[Thompson'68]	$O(nm)$
$(nm)^{1-o(1)}$ under SETH	[Backurs,Indyk'16]	$(nm)^{1-o(1)}$ under SETH

Homogeneous Regular Expressions

reg-exp is **homogeneous of type t** if:

- any internal node on level i has operation t_i
- the depth is at most $|t|$
- leaves can be at any level



type ∘ | +

restrict to homogeneous reg-exps of type t:

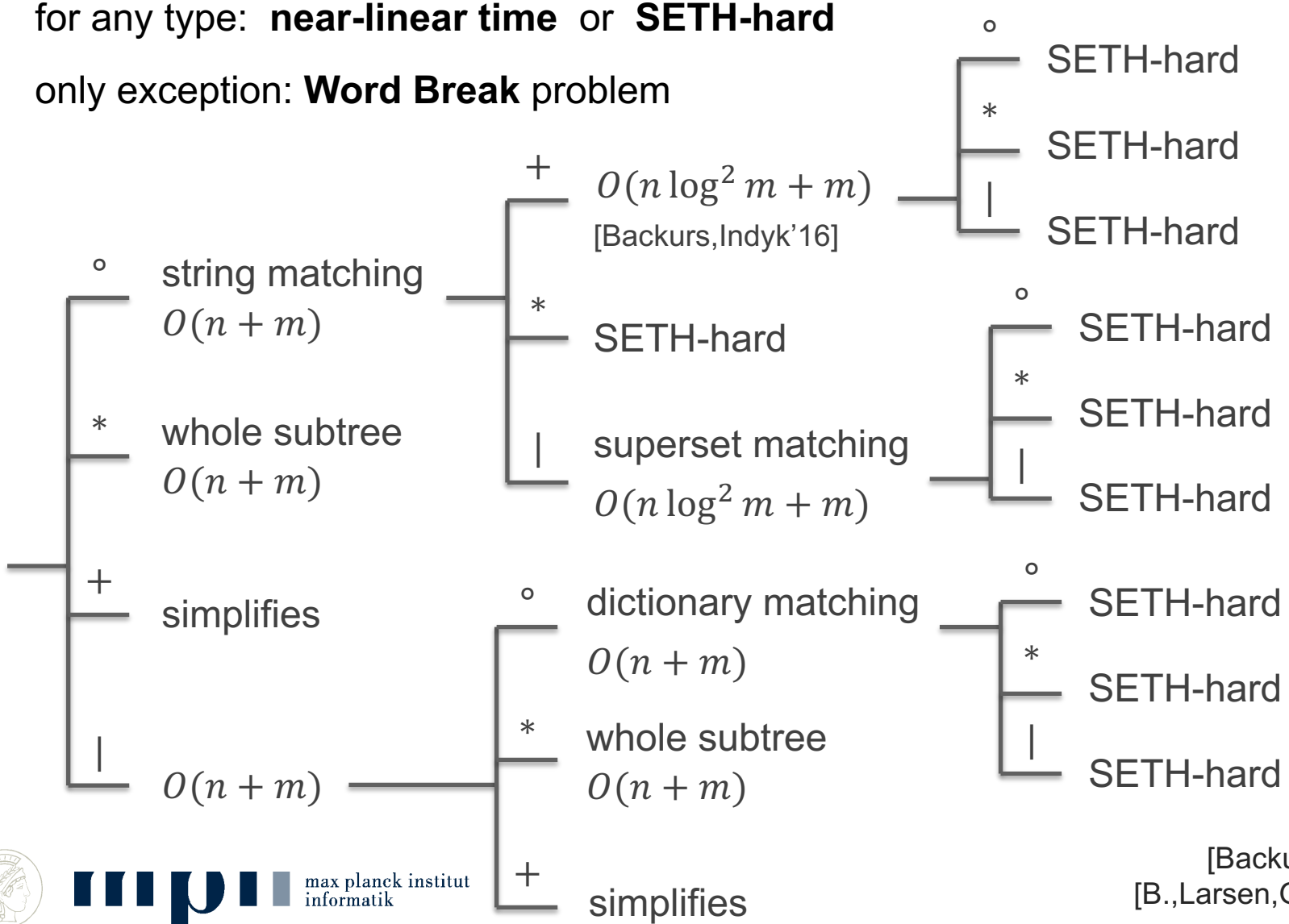
t-Pattern Matching, t-Membership Testing

t-Pattern Matching and t-Membership Testing

complete characterization!

for any type: **near-linear time** or **SETH-hard**

only exception: **Word Break** problem



[Backurs,Indyk'16]
[B.,Larsen,Grønlund'17]

Word Break Problem

Word Break = $+|\circ$-Membership Testing

given string S and dictionary D

can S be split into words in D?

$$n = |S|$$

$$m = \sum_{d \in D} |d|$$

$O(nm)$ trivial algorithm

$\tilde{O}(n\sqrt{m} + m)$ improved algorithm

$\tilde{O}(nm^{1/2-1/18} + m)$ [Backurs,Indyk'16]

$\tilde{O}(nm^{1/3} + m)$ [B.,Larsen,Grønlund'17]

matching conditional lower bound from k-*Clique for combinatorial algorithms*

Word Break is the only **intermediate** t-Membership problem!

I. Easy Example

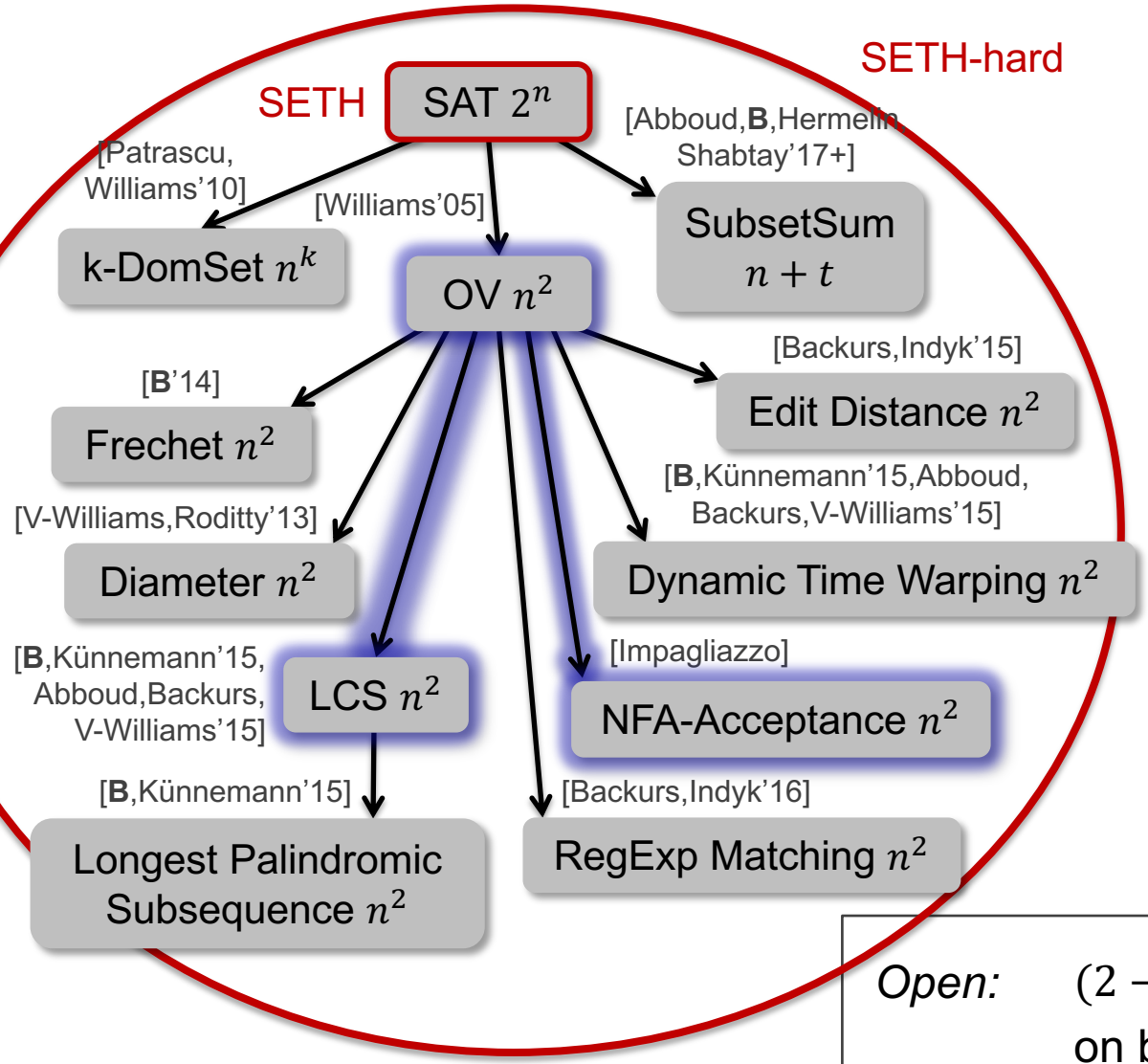
II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary

Conclusion



we have seen:

an involved lower bound

coordinate gadgets,
vector gadgets,
normalized vector gadgets

communication complexity
(non-deterministic,
randomized, one-sided)

hardness of
approximation in P

Open: $(2-\varepsilon)$-approximation of **LCS** on binary strings in time $O(n^{2-\varepsilon})$?

Hardness of approximation
for non-product-like problems