

# Fine-Grained Complexity and Formal Languages

---

Henning Fernau



Universität Trier - Germany

# Table of Contents

Introduction: Basic Problems in Fine-Grained Complexity

ETH and Finite Automata

Complexity Results for Finite Automata Based on SETH, OVH, 3SUM

Parsing

Automata Hypotheses for Fine-Grained Complexity

Summary

# Introduction: Basic Problems in Fine-Grained Complexity

---

# Classical Complexity Versus Fine-Grained Complexity

## Classical computational hardness

### Hardness hypothesis

SAT requires super-polynomial time.

Rather independent of machine model.

### Polynomial-time reduction

Purpose: If  $Q$  can be solved in poly-time, then SAT can (and any problem in NP).

### Conditional hardness

Conclusion:  $Q$  requires super-polynomial time (on any machine model).

More theoretical, more clean

## Fine-grained hardness

### Hardness hypothesis

Assume that hard problem  $H$  requires  $h(n)^{1-o(1)}$  time on a RAM.

### Fine-grained reduction

Purpose: If  $Q$  can be solved in time  $q(n)^{1-\varepsilon}$  on a RAM, then  $H$  can be solved in time  $h(n)^{1-\delta}$  on a RAM.

### Conditional hardness

Conclusion:  $Q$  requires  $q(n)^{1-o(1)}$  time on a RAM.

More practical, more dirty

# What Do YOU Believe?

## Common belief

SAT requires super-polynomial time. Or, more technically speaking:  $P \neq NP$ .

Technical interlude: size of a SAT formula (in CNF) measured in terms of

- number  $n$  of variables and
- number  $m$  of clauses.

Common belief: there is no polynomial  $p$  and no algorithm solving any SAT instance (of size  $n, m$ ) that runs in time  $p(n, m)$ .

However, such an instance can be solved in time  $\mathcal{O}(2^n nm)$ .

## Who believes

- SAT can be solved in time  $\mathcal{O}\left((n+m)^{\log(nm)}\right)$ , or in  $\mathcal{O}\left(2^{\text{polylog}(n+m)}\right)$ ?
- SAT can be solved in time  $\mathcal{O}\left(2^{\text{polylog}(n+m) \cdot \sqrt{n+m}}\right)$ ?
- 3-SAT can be solved in time  $\mathcal{O}\left(2^{\text{polylog}(n+m) \cdot \sqrt{n+m}}\right)$ ?

# ETH and SETH: A Simplified View

**Exponential Time Hypothesis, ETH** (Impagliazzo / Paturi 2001):

3-SAT instances (with  $n$  variables and  $m$  clauses) cannot be solved in time  $\mathcal{O}^*(2^{o(n)})$ .

**Important tool: *Sparsification Lemma*:**

If ETH holds, then 3-SAT instances cannot be solved in time  $\mathcal{O}^*(2^{o(n+m)})$ .

*If you believe that  $P \neq NP$ , then you might want to believe in ETH.*

Do you want to believe even more? **Strong Exponential Time Hypothesis, SETH:**

SAT instances on  $n$  variables cannot be solved in time  $\mathcal{O}^*((2 - \varepsilon)^n)$  for any  $\varepsilon > 0$ .

Relations to other complexity questions

**Known:** SETH implies ETH

ETH implies:  $FPT \neq W[1]$  (link to parameterized complexity)

There is a 1-1 correspondence between SUBEXP vs. EXP and FPT vs. XP

Motivation

(S)ETH bounds can be practically more relevant than pure NP-hardness assertions.

The running-time bounds are more concrete than just being non-polynomial.

# Further Hard Problems: Orthogonal Vectors

Model: word-RAM with words of  $\log n$  bits.

## ORTHOGONAL VECTORS (OV)

Input:  $n$  binary vectors of dimension  $d \in \omega(\log n)$

Problem: Are there vectors  $u, v$  s.t.  $u \cdot v = 0$ ?

## Algorithms for OV:

Easy:  $\mathcal{O}(n^2 d)$ -time,

can be improved to  
 $\mathcal{O}(n^{2-\Theta((\log d - \log \log n)^{-1})})$

Abboud, Williams, Yu 2015

**Orthogonal Vectors Hypothesis, OVH:** Solving OV requires  $\mathcal{O}(n^{2-o(1)})$  time.

**Theorem (Williams 2005 / Williams, Yu 2015)**

*SETH implies OVH.*



A. Abboud, R. R. Williams, and H. Yu.

**More applications of the polynomial method to algorithm design.**

In P. Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 218–230. SIAM, 2015.



R. Williams.

**A new algorithm for optimal 2-constraint satisfaction and its implications.**

*Theoretical Computer Science*, 348(2-3):357–365, 2005.



R. Williams and H. Yu.

**Finding orthogonal vectors in discrete structures.**

In C. Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 1867–1877, 2014.

# Further Hard Problems: All-Pairs-Shortest-Paths

Model: word-RAM with words of  $\log n$  bits.

## ALL-PAIRS-SHORTEST-PATHS (APSP)

Input: Edge-weighted graph with  $n$  vertices

Problem: Determine distance between any two vertices.

Algorithms for APSP:

Classical:  $\mathcal{O}(n^3)$ -time,

can be improved to

$\mathcal{O}(n^3/2^{\Omega(\sqrt{\log n})})$

**All-Pairs-Shortest-Paths Hypothesis, APSP:** Solving APSP requires  $\mathcal{O}(n^{3-o(1)})$  time.



S. Moran.

**A note on 'is shortest path problem not harder than matrix multiplication?'**

*Information Processing Letters*, 13(2):85–86, 1981.



F. Romani.

**Shortest-path problem is not harder than matrix multiplication.**

*Information Processing Letters*, 11(3):134–136, 1980.



R. Williams.

**Faster all-pairs shortest paths via circuit complexity.**

In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC*, pp.664–673. ACM, 2014.

# Further Hard Problems: 3SUM

Model: word-RAM with words of  $\log n$  bits, sometimes also REAL RAM.

## 3SUM





Input:  $n$  numbers

Problem: Are there numbers  $a, b, c$  s.t.  $a + b + c = 0$ ?

## Algorithms for 3SUM:

Hashtables:  $\mathcal{O}(n^2)$ -time,  
can be improved to  
 $\mathcal{O}(n^2/\log^2(n))$   
both for integers and reals

**3SUM Hypothesis:** Solving 3SUM requires  $\mathcal{O}(n^{2-o(1)})$  time. **Analogously:**  $k$ SUM.

-  I. Baran, E. D. Demaine, and M. Pătrașcu.  
**Subquadratic algorithms for 3SUM.**  
*Algorithmica*, 50(4):584–596, 2008.
-  T. M. Chan.  
**More logarithmic-factor speedups for 3SUM, (median, +)-convolution, and some geometric 3SUM-hard problems.**  
*ACM Transactions on Algorithms*, 16(1):7:1–7:23, 2020.
-  A. Gajentaan and M. H. Overmars.  
**On a class of  $\mathcal{O}(n^2)$  problems in computational geometry.**  
*Computational Geometry*, 5:165–185, 1995.
-  A. Grønlund and S. Pettie.  
**Threesomes, degenerates, and love triangles.**  
*Journal of the ACM*, 65(4):22:1–22:25, 2018.

# How Hard Is (Boolean) Matrix Multiplication?

## (BOOLEAN) MATRIX MULTIPLICATION (B)MM

Input: Two  $n \times n$  matrices

Problem: Compute their matrix product!

### Algorithms for MM:

Well-known:  $\mathcal{O}(n^3)$ -time,

also works in semirings,

can be improved? to  $\mathcal{O}(n^\omega)$

now:  $\omega < 2.37286, 2.371339$

- The mentioned “journal bound” on  $\omega$  was obtained by a specific *laser method*; it has been shown that this method is limited to  $\omega \geq 2.3725$ .
- Most of these improvements over the “highschool method” are impractical.
- They also depend on certain algebraic properties like “negation” that are not always available.

 J. Alman, R. Duan, V. Vassilevska Williams, Y. Xu, Z. Xu, and R. Zhou.

**More asymmetry yields faster matrix multiplication.**

In Y. Azar and D. Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2005–2039. SIAM, 2025.

 J. Alman and V. Vassilevska Williams.

**A refined laser method and faster matrix multiplication.**

*TheoretCS*, 3, 2024.

 A. Ambainis, Y. Filmus, and F. Le Gall.

**Fast matrix multiplication: Limitations of the Coppersmith-Winograd method.**

In R. A. Servedio and R. Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pp. 585–593. ACM, 2015.

# Reductions: Classical Versus Fine-Grained

## Classical Setting

- Mostly many-one reductions between decision problems,
- reductions are transitive,
- they are relatively “coarse”, e.g., poly-time reductions can “destroy” the degree of the polynomial.

## Fine-Grained Reductions

Let  $A, B$  be problems. Let  $a, b : \mathbb{N} \rightarrow \mathbb{N}$  time-constructible.  $A$  is  $(a, b)$ -reducible to  $B$  iff  $\forall \varepsilon > 0 \exists \delta > 0$  with an  $\mathcal{O}(a(n)^{1-\delta})$ -algorithm solving  $A$  on instances of size  $n$  by making  $k$  calls to an oracle for  $B$  with query lengths  $n_1, \dots, n_k$  such that  $\sum_{i=1}^k b(n_i)^{1-\varepsilon} < a(n)^{1-\delta}$ .

## Basic Facts on Fine-Grained Reductions

- If problem  $A$  is  $(a, b)$ -reducible to  $B$  and if  $B$  can be solved in time  $\mathcal{O}(b(n)^{1-\varepsilon})$ , then  $A$  can be solved in time  $\mathcal{O}(a(n)^{1-\delta})$ .
- If  $A$  is  $(a, b)$ -reducible to  $B$  and  $B$  is  $(b, c)$ -reducible to  $C$ , then  $A$  is  $(a, c)$ -reducible to  $C$ .

## ETH and Finite Automata

---

# Famous Problems on Finite Automata

**NON-UNIVERSALITY:** Given an automaton  $A$  with input alphabet  $\Sigma$ , is  $L(A) \neq \Sigma^*$ ?

**INEQUIVALENCE:** Given two automata  $A_1, A_2$ , is  $L(A_1) \neq L(A_2)$ ?

**INTERSECTION NON-EMPTINESS:** Given  $k$  automata  $A_1, \dots, A_k$ , is  $\bigcap_{i=1}^k L(A_i) \neq \emptyset$ ?

Classical status	DFA	NFA
NON-UNIVERSALITY	poly-time	PSPACE-complete
INEQUIVALENCE	poly-time	PSPACE-complete
INTERSECTION NON-EMPTINESS	PSPACE-complete	PSPACE-complete

↪ Focus on NFAs.

NFA	unary	binary
NON-UNIVERSALITY	NP-complete	PSPACE-complete
INEQUIVALENCE	NP-complete	PSPACE-complete
INTERSECTION NON-EMPTINESS	NP-complete	PSPACE-complete

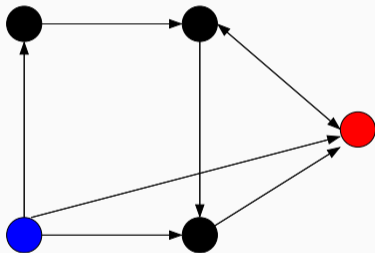
↪ Focus on unary (tally) languages.

# Tally Means Directed Graphs

Tally automata are nothing else than directed graphs (edge labels not interesting).

UNIVERSALITY hence models the following scenario:

Can somebody living in the **blue circle** visit the **red one** in  $k$  steps, for **any** number  $k$ ?



This question is trivial for undirected graphs...

So, possibly those mostly interested in graphs may bear with us ...

# NON-UNIVERSALITY for Tally NFAs: NP-hardness by Stockmeyer & Meyer 1973

Reduction from 3-SAT 😊 ( $n$  variables,  $m$  clauses)

**Idea:** Codify assignments by Chinese remainder.

Take the first  $n$  primes  $p_1, \dots, p_n$ ; Ex.:  $p_1 = 2, p_2 = 3, p_3 = 5$ .

$a^z$  encodes assignment  $\alpha$  if  $z \equiv \alpha(x_i) \pmod{p_i}$ ; Ex.:  $\alpha(x_1) = 0, \alpha(x_2) = 1, \alpha(x_3) = 1 \rightsquigarrow z = 16$ .

Recall:  $p_n \sim n \ln n$ .

$\exists$  NFA  $A_0$  for  $L_0 := \bigcup_{k=1}^n \bigcup_{j=2}^{p_k-1} \{a\}^j \{a^{p_k}\}^*$  with  $\leq np_n \sim n^2 \ln n$  many states.

$L_0$  collects words that do not encode assignments, i.e., **syntactic trash**. Ex.:  $aa \in L_0$ .

$L_j := \{a^{z_{k_j}}\} \cdot \{a^{p_{i_j(1)} \cdots p_{i_j(|c_j|)}}\}^*$  for **clause**  $c_j$ , with  $0 \leq z_{k_j} < p_{i_j(1)} \cdots p_{i_j(|c_j|)}$ , is uniquely determined by  $z_{k_j} \equiv \alpha(x_r) \pmod{p_{i_j(r)}}$  for  $r = 1, \dots, |c_j|$  s.t.  $\alpha$  falsifies  $c_j$ .

$i_j(\ell)$  is the index of the  $\ell^{\text{th}}$  variable in clause  $c_j$ .

As  $p_{i_j(1)} \cdots p_{i_j(|c_j|)} \leq p_n^3$  (3-SAT),  $L_j$  is accepted by a DFA with  $\leq p_n^3 \leq n^3 \cdot (\ln n)^3$  many states.

Observe:  $L_0 \cup L_1 \cup \dots \cup L_m = \{a\}^*$  iff given 3-SAT formula is unsatisfiable.

Altogether, the NFA has  $\leq mp_n^3 \approx n^4 (\log n)^3$  states (after sparcification): quite a many! 😞

**Cor.:** Unless ETH fails, for any  $\epsilon > 0$ , there is no  $\mathcal{O}^*(2^{o(q^{1/4-\epsilon})})$ -time algorithm for deciding, given a tally NFA  $A$  on  $q$  states, whether  $L(A) = \{a\}^*$ .

# What Are the “Correct Bounds”?

## The algorithmic side:

Textbook algorithm (conversion into DFAs) yields  $\mathcal{O}^*(2^q)$ -time algorithm.

Chrobak 1986 (conversion into DFAs): improvement to  $\mathcal{O}^*(2^{\Theta(\sqrt{q \log q})})$ .

Can we bring these bounds together?

**Thm.:** Unless ETH fails, there is no  $\mathcal{O}^*(2^{o(m)})$ -time algorithm for deciding if a given  $m$ -edge graph has a (proper) 3-coloring.

Standard proof ... / W.l.o.g.,  $m = O(n)$ .

Now, mimic proof of Stockmeyer / Meyer, reducing from 3-COLORING.





variables  $\approx$  vertices / clauses  $\approx$  edges

(a) Coloring condition has two vertices, not three variables  $\leadsto$  improvement

(b) Moreover, more efficient encoding  $\leadsto$  no  $\epsilon$ -term

**Thm.:** (F / Krebs 2017) Unless ETH fails, there is no  $\mathcal{O}^*(2^{o(q^{1/3})})$ -time algorithm for deciding, given a tally NFA  $A$  on  $q$  states, whether  $L(A) = \{a\}^*$ .

Czerwiński *et al.* 2025: algorithmic improvement to  $\mathcal{O}^*(2^{\Theta(\sqrt[3]{q \log q})})$ . **Nearly matching!**

-  M. Chrobak.  
**Finite automata and unary languages.**  
*Theoretical Computer Science*, 47:149–158, 1986.
-  W. Czerwiński, M. Dębski, T. Gogasz, G. Hoi, S. Jain, M. Skrzypczak, F. Stephan, and C. Tan.  
**Languages given by finite automata over the unary alphabet.**  
*Journal of Computer and System Sciences*, 151:103634, 2025.
-  H. Fernau and A. Krebs.  
**Problems on finite automata and the exponential time hypothesis.**  
*Algorithms*, 10:24:1–25, 2017.
-  L. J. Stockmeyer and A. R. Meyer.  
**Word problems requiring exponential time: Preliminary report.**  
In A. V. Aho, A. Borodin, R. L. Constable, R. W. Floyd, M. A. Harrison, R. M. Karp, and H. Raymond Strong, editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, STOC*, pp. 1–9. ACM, 1973.

# INTERSECTION NON-EMPTINESS

## Theorem (Tally case, F & Krebs 2017)

There is no algorithm that, given  $k$  tally DFAs (or NFAs)  $A_1, \dots, A_k$ , each with at most  $q$  states, decides if  $\bigcap_{i=1}^k L(A_i) \neq \emptyset$  in time  $\mathcal{O}^*(2^{\mathcal{O}(\min(k, q^{1/2}))})$  unless ETH fails.

Revisit previous constructions:  $k \approx n + m$ ,  $q \approx n^2$ .

The algorithmic side: UB  $\mathcal{O}^*(q^k)$  is terribly far off. 😞

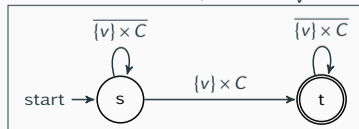
## Theorem

There is no algorithm that, given  $k$  DFAs  $A_1, \dots, A_k$  with unbounded input alphabet, each with at most 3 states, decides in time  $\mathcal{O}^*(2^{\mathcal{O}(k)})$  if  $\bigcap_{i=1}^k L(A_i) \neq \emptyset$  unless ETH fails.

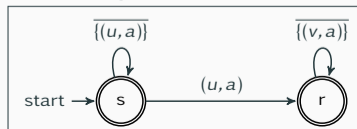
This matches the product automaton upper bound  $\mathcal{O}^*(3^k)$ . 😊

Reduction from 3-COLORING: Choose alphabet  $\Sigma = V \times C$ ,  $C = \{1, 2, 3\}$ .

For all vertices  $v$ , build  $A_v$ :



For all edges  $uv$ , colors  $a$ , build  $A_{uv,a}$ :



## Further ETH-based Automata Results: Synchronization

Given a deterministic finite semi-automaton, i.e., for each  $a \in \Sigma$ , a mapping  $f_a : Q \rightarrow Q$ , a set  $Q_{\text{sync}} \subseteq Q$ , a  $Q_{\text{sync}}$ -*synchronizing word*  $w \in \Sigma^*$  enjoys

$$\forall q, q' \in Q_{\text{sync}} : f_w(q) = f_w(q').$$

$Q_{\text{sync}}$ -SW: Is there a  $Q_{\text{sync}}$ -*synchronizing word*? PSPACE-complete.

$Q_{\text{sync}} = Q$ : Related to Černý's Conjecture.

Q-SW is "only" NP-complete (with length bound on synchr. word; else in P).

### Theorem

*There is an algorithm for solving  $Q_{\text{sync}}$ -SW on unbounded input alphabets that runs in time  $\mathcal{O}^*(2^q)$  for  $q$ -state deterministic finite semi-automata. (Power automaton construction)*

*Conversely, assuming ETH, there is no  $\mathcal{O}^*(2^{o(q)})$ -time algorithm for this task, even on bounded input alphabets.*

# Further ETH-based Automata Results: Pumping

Recall: The Pumping Lemma according to Kozen:

For any regular language  $L \subseteq \Sigma^*$ , there is a constant  $p_L$  such that: If  $w \in L$  and  $|w| \geq p_L$ , then there is a decomposition  $w = xyz$  with  $x, z \in \Sigma^*$  and  $y \in \Sigma^+$  such that  $xy^*z \subseteq L$ .

Although an upper bound on  $p_L$  is given in the proof, what about finding the smallest constant  $p_L$  satisfying the lemma?

**Theorem (Holzer and Rauch, 2023)**

*Given an  $s$ -state DFA with input alphabet  $\Sigma$  s.t.  $|\Sigma| \in \mathcal{O}(s)$ , the smallest pumping constant cannot be approximated*

- up to a factor of  $s^{1-\varepsilon}$  for any  $\varepsilon > 0$  unless  $P = NP$ , nor
- up to a factor of  $\omega\left(\frac{s \log \log s}{(\log s)^2}\right)$  unless ETH fails.

 A. Björklund, T. Husfeldt, and S. Khanna.

**Approximating longest directed paths and cycles.**

In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP*, volume 3142 of LNCS, pp. 222–233. Springer, 2004.

 H. Gruber, M. Holzer, and C. Rauch.

**On pumping preserving homomorphisms and the complexity of the pumping problem (extended abstract).**

In S. Z. Fazekas, editor, *Implementation and Application of Automata - 28th International Conference, CIAA*, volume 15015 of LNCS, pp. 153–165. Springer, 2024.

# Complexity Results for Finite Automata Based on SETH, OVH, 3SUM

---

## Theorem (F, Krebs 2017)

*There is no  $\mathcal{O}^*((|\Sigma| - \varepsilon)^\ell)$ -time algorithm solving  $Q$ -SW (for any  $\varepsilon > 0$ ) unless SETH fails, where  $\ell$  upper-bounds the length of a synchronizing word.*

Similar results for NON-UNIVERSALITY, INEQUIVALENCE.

## Theorem (Wehar 2016)

*There is no  $\mathcal{O}(n^{k-\varepsilon})$  algorithm solving DFA INTERSECTION NON-EMPTINESS (for any  $\varepsilon > 0$ ), where  $n$  is the sum of the number of states of all involved DFA, unless SETH fails.*



C. Allauzen, M. Mohri, and A. Rastogi.

**General algorithms for testing the ambiguity of finite automata and the double-tape ambiguity of finite-state transducers.**

*International Journal of Foundations of Computer Science*, 22(4):883–904, 2011.



K. Drabik, A. Dürr, F. Frei, F. Mazowiecki, and K. Węgrzycki.

**Fined-grained complexity of ambiguity problems on automata and directed graphs.**

Technical Report 2501.14725, ArXiv, Cornell University, USA, 2025.



M. Wehar.

***On the Complexity of Intersection Non-Emptiness Problems.***

PhD thesis, University at Buffalo, State University of New York, 2016.

## NFA UNAMBIGUITY (NFA-Unamb)

Input: NFA  $A$

Problem: Has every input word at most one accepting run?

Algorithm for

NFA-Unamb:

$\mathcal{O}(|A|^2)$ -time by

Allauzen *et al.* 2011

They conjecture optimality.

**Theorem (Drabik *et al.* 2025)**

*There is no  $\mathcal{O}(n^{2-\varepsilon})$  algorithm solving NFA-UNAMB (for any  $\varepsilon > 0$ ), unless OVH fails.*

The proof involves strengthening Wehar's theorem. It makes use of the  $k$ -Orthogonal Vector Hypothesis which generalizes OVH to selecting  $k$  orthogonal vectors;  $k$ -OVH is also implied by SETH.

**Theorem (Drabik *et al.* 2025)**

*There is no  $\mathcal{O}(n^{k-\varepsilon})$  algorithm solving DFA INTERSECTION NON-EMPTINESS (for any  $\varepsilon > 0$ ), where  $n$  is the sum of the number of states of all involved DFA, unless  $k$ -OVH fails.*

*Timed automata* are finite automata equipped with *clocks*. Configurations also contain clock values, and transitions change and depend on clock values.

The *dynamic acceptance problem* amounts to designing a data structure that can be initialized for a given timed automaton  $A$ , and afterwards, upon consuming consecutive elements of the data stream, efficiently maintains the information on whether the word read so far is accepted by  $A$ .

### Theorem

*If the REAL 3SUM Hypothesis holds, then there is a two-clock timed automaton  $A$  (with additive constraints) such that there is no data structure that, when initialized on  $A$ , supports dynamic acceptance in time  $\mathcal{O}(n^{1-\delta})$  for any  $\delta > 0$ , where  $n$  is the length of the consumed stream prefix.*










A. Grez, F. Mazowiecki, M. Pilipczuk, G. Puppis, and C. Riveros.

**Dynamic data structures for timed automata acceptance.**

*Algorithmica*, 84(11):3223–3245, 2022.

# Parsing

---

-  A. Abboud, A. Backurs, and V. Vassilevska Williams.  
**If the current clique algorithms are optimal, so is Valiant's parser.**  
In V. Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*, pp. 98–117. IEEE Computer Society, 2015.
-  K. Bringmann and P. Wellnitz.  
**Clique-based lower bounds for parsing tree-adjoining grammars.**  
In J. Kärkkäinen, J. Radoszewski, and W. Rytter, editors, *28th Annual Symposium on Combinatorial Pattern Matching, CPM*, volume 78 of *LIPIcs*, pp. 12:1–12:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
-  L. Lee.  
**Fast context-free grammar parsing requires fast Boolean matrix multiplication.**  
*Journal of the ACM*, 49(1):1–15, 2002.
-  A. Okhotin.  
**Parsing by matrix multiplication generalized to Boolean grammars.**  
*Theoretical Computer Science*, 516:101–120, 2014.
-  W. Rytter.  
**Context-free recognition via shortest paths computation: a version of Valiant's algorithm.**  
*Theoretical Computer Science*, 143(2):343–352, 1995.
-  G. Satta.  
**Tree-adjoining grammar parsing and Boolean matrix multiplication.**  
*Journal Computational Linguistics*, 20(2):173–191, June 1994.
-  L. G. Valiant.  
**General context-free recognition in less than cubic time.**  
*Journal of Computer and System Sciences*, 10(2):308–315, 1975.

# Parsing Like Valiant

Most computer scientists learn the CYK algorithm to parse context-free grammars in Chomsky normal form.

Valiant re-interpreted this as (Boolean) matrix multiplication BMM.

1. Re-define matrix multiplication: Let  $A, B$  be two  $n \times n$  matrices whose entries are subsets of the NT set  $N$ . (Finally,  $n = |w| + 1$ .) Define  $C = AB$  by

$$C[i, j] = \bigcup_{k=1}^n \underbrace{A[i, k] * B[k, j]}_{\{X \mid \exists Y \in A[i, k], Z \in B[k, j], X \rightarrow YZ \in P\}}$$

2. Via bitvectors, interpret this operation as BMM.

Hence, CFG parsing can be done as fast as BMM.

First Lee (huge grammar!) and then Abboud *et al.* proved that the converse holds.

These links transfer to generalized parsing, e.g., for Boolean grammars and for tree-adjointing grammars.

# Automata Hypotheses for Fine-Grained Complexity

---

# NFA Acceptance Hypothesis

## NFA ACCEPTANCE (NFA-Acc)

Input: An NFA  $A$  and an input word  $w$

Problem: Is  $w \in L(A)$ ?

## Algorithm for NFA-Acc:

DP:  $\mathcal{O}(|A| \cdot |w|)$ -time,

can be improved to

$$\mathcal{O}\left(|Q_A|^2 \cdot |w| / 2^{\Omega(\sqrt{\log(|Q_A|)})}\right)$$






Bringmann *et al.* 2024

**NFA-Acc Hypothesis:** Solving NFA-Acc requires  $\mathcal{O}\left((|A| \cdot |w|)^{1-o(1)}\right)$  time.

Support 1: NFA-Acc implies Online Matrix-Vector Multiplication Hypothesis (OMV) (Henzinger *et al.*)

Support 2: OVH implies NFA-Acc for sparse NFA, i.e.,  $|A| \in \mathcal{O}(|Q_A|)$

Support 3: NFA-Acc is equivalent to many other problems, e.g., in databases.

-  A. Backurs and P. Indyk.  
**Which regular expression patterns are hard to match?**  
In I. Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*, pp. 457–466. IEEE Computer Society, 2016.
-  K. Bringmann, A. Grønlund, M. Künnemann, and K. Green Larsen.  
**The NFA acceptance hypothesis: Non-combinatorial and dynamic lower bounds.**  
*TheoretCS*, 3, 2024.
-  K. Bringmann, A. Grønlund, and K. G. Larsen.  
**A dichotomy for regular expression membership testing.**  
In C. Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pp. 307–318. IEEE Computer Society, 2017.
-  K. Casel and M. L. Schmid.  
**Fine-grained complexity of regular path queries.**  
*Logical Methods in Computer Science*, 19(4), 2023.
-  M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak.  
**Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture.**  
In R. A. Servedio and R. Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pp. 21–30. ACM, 2015.

## Summary

---

# Fine-Grained Complexity and Formal Languages

- Both areas allow a fruitful interaction.  
This workshop should inspire and increase these interactions.
- Possible starting points:
  - Take your favorite computationally hard formal language problem and prove that well-known algorithms are conditionally optimal. It sometimes helps to re-analyze known reductions.
  - Take your favorite “base problem” from fine-grained complexity and look for “corresponding” formal language problems. Recall that automata are also graphs ...
- There are also a lot of naturally open questions that can be generated in this way.  
There are also lots of examples provided in this talk.