Fine-Grained Complexity of Regular Expression Pattern Matching and Membership

Philipp Schepper
CISPA – Helmholtz Center for Information Security, Saarbrücken, Germany
Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus, Saarbrücken, Germany
philipp.schepper@cispa.saarland

Abstract

The currently fastest algorithm for regular expression pattern matching and membership improves the classical $\mathcal{O}(nm)$ time algorithm by a factor of about $\log^{3/2} n$. Instead of focussing on general patterns we analyse homogeneous patterns of bounded depth in this work. For them a classification splitting the types in easy (strongly sub-quadratic) and hard (essentially quadratic time under SETH) is known. We take a very fine-grained look at the hard pattern types from this classification and show a dichotomy: few types allow super-poly-logarithmic improvements while the algorithms for the other pattern types can only be improved by a constant number of log-factors, assuming the Formula-SAT Hypothesis.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases Fine-Grained Complexity, Regular Expression, Pattern Matching, Dichotomy

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.80

Related Version A full version of this paper is available at `https://arxiv.org/abs/2008.02769`. All presented lower bounds and an alternative proof of the upper bounds for pattern matching using the polynomial method are contained in the author's Master's thesis.

Funding Supported by the European Research Council (ERC) consolidator grant No. 725978 SYSTEMATICGRAPH.

Acknowledgements I thank Karl Bringmann for the supervision during the research for my Master's Thesis which this paper is based on and especially the pointer to Batch-OV which simplified the upper bounds extremely.

1 Introduction

Regular expressions with the operations alternative $|$, concatenation $\circ$, Kleene Plus $+$, and Kleene Star $\star$ are used in many fields of computer science. For example to search in texts and files or to replace strings by other strings as the unix tool `sed` does. But they are also used to analyse XML files [17, 18], for network analysis [12, 24], human computer interaction [13], and in biology to search for proteins in DNA sequences [16, 20].

The most intuitive problem for regular expressions is the *membership* problem. There we ask whether a given text t can be generated by a given regular expression p, i.e. is $t \in \mathcal{L}(p)$? We also call p a pattern in the following. A similar problem is the *pattern matching* problem, where we are interested whether some substring of the given text t can be matched by p. To simplify notation we define the matching language of p as $\mathcal{M}(p) \coloneqq \Sigma^* \mathcal{L}(p) \Sigma^*$. Then we want to check whether $t \in \mathcal{M}(p)$. The standard algorithm for both problems runs in time $\mathcal{O}(nm)$ where n is the text length and m the pattern size [22].

Based on the "Four Russians" trick Myers showed an algorithm with running time $\mathcal{O}(nm/\log n)$ [19]. This result was improved to an $\mathcal{O}(nm \log\log n/\log^{3/2} n)$ time algorithm by Bille and Thorup [5]. Although for several special cases of pattern matching and

Table 1 Hard pattern types that have to be considered.

Pattern matching / Membership	$\circ\star$	$\circ\vert\circ$	$\circ\vert+$	$\circ+\circ$	$\circ+\vert$	$\vert\circ\vert$ / $+\vert\circ\vert$	$\vert\circ+$ / $+\vert\circ+$	$\vert+\vert\circ$

membership improved sub-quadratic time algorithms have been given [3, 11, 14], it remained an open question whether there are truly sub-quadratic time algorithms for the general case. The first conditional lower bounds were shown by Backurs and Indyk [4]. They introduced so-called homogenous patterns and classified their hardness into easy, i.e. strongly sub-quadratic time solvable, and hard, requiring essentially quadratic time assuming the Strong Exponential Time Hypothesis (SETH). This classification of Backurs and Indyk was completed by a dichotomy for all homogeneous pattern types by Bringmann, Grønlund, and Larsen [8]. They reduced the hardness of all hard pattern types to the hardness of few pattern types of bounded depth. By this it was sufficient to check few cases instead of infinitely many.

To understand what a homogeneous pattern is, we observe that one can see patterns as rooted and node labeled trees where the inner nodes correspond to the operations of the pattern. Then a pattern is homogenous if the operations on each level of the tree are equal. The type of the pattern is the sequence of operations from the root to the leaves. See Section 2 for a formal introduction.

But as SETH rules out only polynomial improvements, super-poly-logarithmic runtime improvements are still feasible. Such improvements are know for Orthogonal Vectors (OV) [2, 9], for example, although there is a known conditional lower bound based on SETH. But for pattern matching and membership no faster algorithms are known. By a reduction from Formula-SAT Abboud and Bringmann showed that in general pattern matching and membership cannot be solved in time $\mathcal{O}(nm/\log^{7+\epsilon} n)$ under the Formula-SAT Hypothesis [1].

For Formula-SAT one is given a De Morgan formula F over n inputs and size s, i.e. the formula is a tree where each inner gate computes the AND or OR of two other gates and each of the s leaves is labeled with one of the n variables or their negation. The task is to find a satisfying assignment for F. While the naive approach takes time $\mathcal{O}(2^n s)$ to evaluate F on all possible assignments, there are polynomial improvements for formulas of size $s = o(n^3)$ [10, 15, 21]. But despite intense research there is currently no faster algorithm known for $s = n^{3+\Omega(1)}$. Thus it seem reasonable to assume the following hypothesis:

▶ **Hypothesis 1.1** (Formula-SAT Hypothesis (FSH) [1]). *There is no algorithm that can solve Formula-SAT on De Morgan formulas of size $s = n^{3+\Omega(1)}$ in $\mathcal{O}(2^n/n^\epsilon)$ time, for some $\epsilon > 0$, in the Word-RAM model.*

Although the new lower bound of $\mathcal{O}(nm/\log^{7+\epsilon} n)$ is quite astonishing since before only polynomial improvements have been ruled out, the bound is for the general case. It remained an open question whether it also holds for homogeneous patterns of bounded depth. Using the results by Bringmann, Grønlund, and Larsen [8] relating the hardness of different pattern types to each other, it suffices to check the pattern types in Table 1 for the corresponding problem.

We answer this last question and give a dichotomy for these hard pattern types: For few pattern types we give the currently fastest algorithm for pattern matching and membership. For the remaining patterns we show improved lower bounds of the form $\Omega(nm/\log^c n)$.

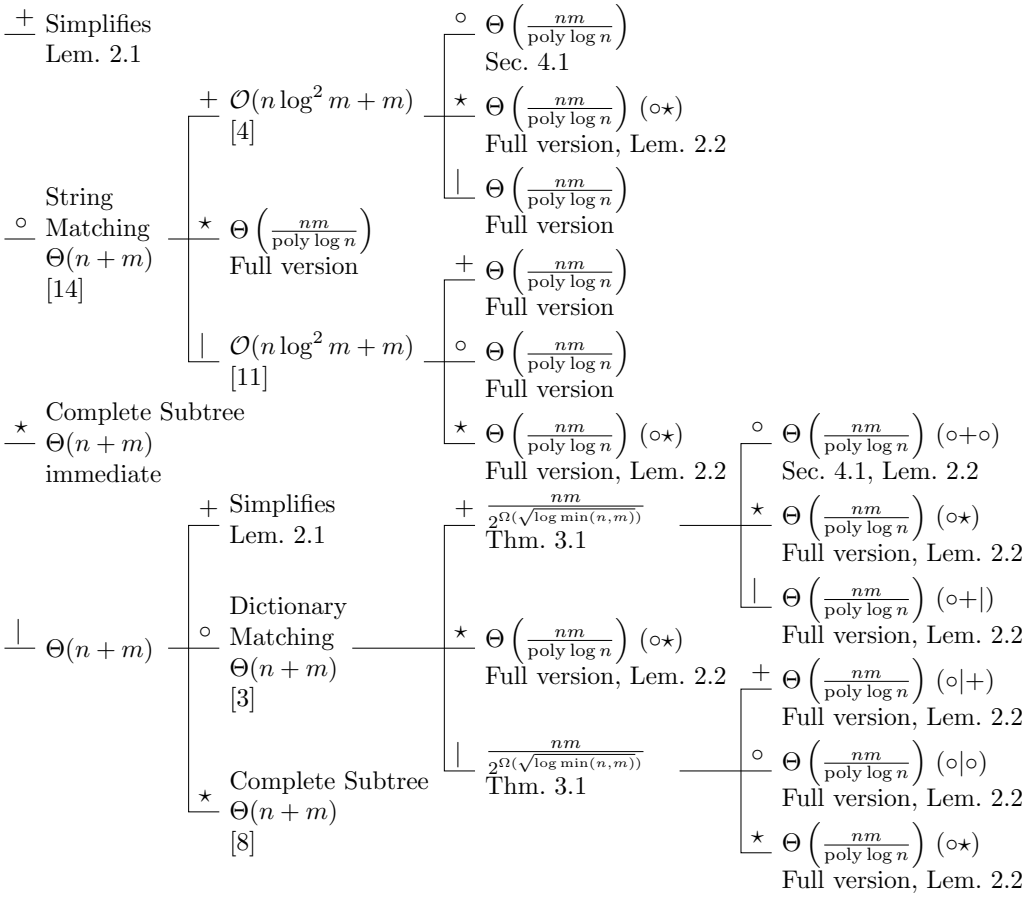


Figure 1 The classification of the patterns for pattern matching.

▶ **Theorem 1.2.** *For texts of length n and patterns of size m we have the following time bounds for the stated problems:*

- $nm/2^{\Omega(\sqrt{\log \min(n,m)})}$ *for* $|\circ|$*- and* $|\circ+$*-pattern matching, and* $+|\circ|$*- and* $+|\circ+$*-membership*
- $\Theta(nm/\text{poly}\log n)$ *for pattern matching and membership with types* $\circ+|$*,* $\circ|+$*,* $\circ+\circ$*,* $\circ|\circ$*, and* $\circ\star$ *and for* $|+|\circ$*-membership, unless FSH is false.*

This dichotomy result gives us a simple classification for the hard pattern types. Depending on the pattern type one can decide if there is super-poly-logarithmic algorithm, or if even the classical algorithm is optimal up to a constant number of log-factors. See Figure 1 for an overview of the results for pattern matching. Further, the dichotomy shows that the type of a pattern has a larger impact on the hardness than the depth. The alternative as outer operation of the "easier" patterns allows us to split the pattern into independent sub-patterns. This is crucial for the speed-up since pattern matching for $\circ+$ and $\circ|$ is near-linear time solvable [4, 11]. Contrary almost all hard pattern types have a concatenation as outer operation which does not allow this decomposition into independent problems. Further, the length of the matched texts can vary largely. The pattern $(a \mid aba)(b \mid bca)(a \mid ab)$, for example, can match strings of length 3 to 8. We exploit both properties in our reductions, especially to encode a boolean OR.

In Section 2 we give a formal definition of homogeneous patterns and state the problems we start reducing from and the ones we reduce to. We show the algorithms for the upper bounds in Section 3. In Section 4 we give the improved lower bounds for pattern matching while the ones for membership are given in Section 5.

2 Preliminaries

Regular expressions. Recall, that patterns over a finite alphabet Σ are build recursively from other patterns using the operations $|$, $\circ$, $+$, and $\star$. We construct the patterns and the language of each pattern (i.e. the set of words matched by the pattern) as follows. Each symbol $\sigma \in \Sigma$ is a pattern representing the language $\mathcal{L}(\sigma) = \{\sigma\}$. Let in the following p_1 and p_2 be two patterns. For the alternative operation we define $\mathcal{L}(p_1 \mid p_2) = \mathcal{L}(p_1) \cup \mathcal{L}(p_2)$. For the concatenation we define $\mathcal{L}(p_1 \circ p_2) = \{w_1 w_2 \mid w_1 \in \mathcal{L}(p_1) \wedge w_2 \in \mathcal{L}(p_2)\}$. For the Kleene Plus we set $\mathcal{L}(p_1^+) = \{w \mid \exists k \geq 1 : \exists w_1, \ldots, w_k \in \mathcal{L}(p_1) : w = w_1 \ldots w_k\}$. With ε as the empty word we have $\mathcal{L}(p_1^\star) = \mathcal{L}(p_1^+) \cup \{\varepsilon\}$ for the Kleene Star.

Based on this construction it is easy to see patterns as rooted and node-labeled trees where each inner node is labeled by an operation and the leaves are labeled by symbols. We call this tree the *parse tree* of a pattern in the following. Then each node is connected to the node representing the sub-pattern p_1 and also for p_2 in the case of the binary operations $\circ$ and $|$. We extend the definition of the alternative and the concatenation in the natural way to more than two sub-patterns. To simplify notation we omit the symbol $\circ$ from the patterns in the following. We define the *size* of a pattern to be the number of inner nodes plus the number of leaves in the parse tree.

We call a pattern *homogeneous* if for each level of the parse tree, all inner nodes are labeled with the same operation. We define the *type* of a homogeneous pattern p to be the sequence of operations from the root of the parse tree of p to the deepest leaf. The *depth* of a pattern is the depth of the tree, which is equal to the number of operations in the type. For example, the pattern $[(abc \mid c)(a \mid dc)c(db \mid c \mid bd)]^+$ is of type $+\circ|\circ$ and has depth 4.

Relations between pattern types. Backurs and Indyk showed in [4] the first quadratic time lower bound for several homogeneous patterns based on SETH. This classification was completed by the dichotomy result of Bringmann, Grønlund, and Larsen in [8]. As there are infinitely many homogeneous pattern types, they showed linear-time reductions between different pattern types. By these reductions lower bounds also transfer to other (more complicated) pattern types and faster algorithms also give improvements for other (equivalent) pattern types.

▶ **Lemma 2.1** (Lemma 1 and Lemma 8 in the full version of [8]). *For any type T, applying any of the following rules yields a type T' such that both are equivalent for pattern matching and membership under linear-time reductions, respectively:*

- *For pattern matching: remove prefix $+$ and replace prefix $|+$ by $|$.*
- *For membership: replace any substring $+|+$ by $+|$ and replace prefix $r\star$ by $r+$ for any $r \in \{+, |\}^*$.*
- *For both problems: replace any substring pp, for any $p \in \{\circ, |, \star, +\}$, by p.*

We say that T simplifies *if one of these rules applies. Applying these rules in any order will eventually lead to an unsimplifiable type.*

▶ **Lemma 2.2** (Lemma 6 and Lemma 9 in the full version of [8]). *For types T and T', there is a linear-time reduction from T-pattern matching/membership to T'-pattern matching/membership if one of the following sufficient conditions holds:*

- *T is a prefix of T',*
- *we may obtain T' from T by replacing a $\star$ by $+\star$,*
- *we may obtain T' from T by inserting a $|$ at any position,*
- *only for membership: T starts with $\circ$ and we may obtain T' from T by prepending a $+$ to T.*

Together with the already known sub-quadratic time algorithms for various pattern types [3, 4, 8, 11, 14], it suffices to check the remaining cases in Table 1 to get a fine-grained dichotomy for the hard pattern types (i.e. the ones requiring essentially quadratic time under SETH).

Hypothesis. As mentioned in the introduction, we follow the ideas of Abboud and Bringmann in [1] and show reductions from Formula-SAT to pattern matching to prove lower bounds. Likewise as in their result, we also start from the intermediate problem Formula-Pair: Given a *monotone* De Morgan formula F with size s, that is a De Morgan formula where each leaf is labeled with a variable, i.e. no negation allowed, and each variable is used only once. Further, one is given two sets A, B of half-assignments to $s/2$ variables of F with $|A| = n$ and $|B| = m$. The task is to find a pair $a \in A, b \in B$ such that $F(a, b) = \mathsf{true}$.

There is an intuitive reduction from Formula-SAT to Formula-Pair as shown in [1]. Thus, FSH implies the following hypothesis, which we prove in Appendix A:

▶ **Hypothesis 2.3** (Formula-Pair Hypothesis (FPH)). *For all $k \geq 1$, there is no algorithm that can solve Formula-Pair for a monotone De Morgan formula F of size s and sets $A, B \subseteq \{0,1\}^{s/2}$ of size n and m, respectively, in time $\mathcal{O}(nms^k/\log^{3k+2} n)$ in the Word-RAM model.*

Batch-OV. For the upper bounds we transform texts and patterns into bit-vectors such that they are orthogonal if and only if the text is matched by the pattern. This gives us a reduction from pattern matching to Orthogonal Vectors (OV) ([9, 23]). But to improve the runtime we process many text simultaneously using the following lemmas.

▶ **Lemma 2.4** (Batch-OV, cf. [9]). *Let $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ and $d \leq 2^{c^{-1}\sqrt{\log n}}$ for some constant $c > 0$. We can decide for all vectors $a \in A$ whether there is a vector $b \in B$ such that $\langle a, b\rangle = 0$ in time $n^2/2^{\epsilon c\sqrt{\log n}}$ for sufficiently small $\epsilon > 0$.*

We generalize this balanced case to the unbalanced case which we use later:

▶ **Lemma 2.5** (Unbalanced Batch-OV). *Let $A, B \subseteq \{0,1\}^d$ with $|A| = n$ and $|B| = m$ and $d \leq 2^{c^{-1}\sqrt{\log \min(n,m)}}$ for some constant $c > 0$. We can decide for all vectors $a \in A$ whether there is a vector $b \in B$ such that $\langle a, b\rangle = 0$ in time $nm/2^{\epsilon c\sqrt{\log \min(n,m)}}$ for sufficiently small $\epsilon > 0$.*

Proof. If $n \leq m$, partition B into $\lceil m/n \rceil$ sets of size n and run the algorithm from Lemma 2.4 on every instance in time $\lceil m/n \rceil n^2/2^{\epsilon c\sqrt{\log n}} \approx nm/2^{\epsilon c\sqrt{\log n}}$. Analogously for $n > m$. ◀

3 Upper Bounds

For patterns p of type $|\circ|$ and $|\circ+$ let $p = (p_1 \mid p_2 \mid \ldots \mid p_k)$ be the pattern of size m. Likewise for the patterns with a Kleene Plus as additional outer operation. Let further $t = t_1 \cdots t_n$ be the text of length n. The main idea of the fast algorithm is to compute a set of matched substrings: $M = \{(i, j) \mid \exists \ell \in [k] : t_i \cdots t_j \in \mathcal{L}(p_\ell)\} \subseteq [n] \times [n]$. From M we construct a graph where the nodes correspond to different prefixes that can be matched. The tuples in M represent edges between these nodes. Then it remains to check whether the node corresponding to t is reachable.

▶ **Theorem 3.1** (Upper Bounds). *We can solve in time* $nm/2^{\Omega\sqrt{\log\min(n,m)}}$*:*

1. $|\circ|$*-pattern matching and* $+|\circ|$*-membership.*
2. $|\circ+$*-pattern matching and* $+|\circ+$*-membership.*

To compute M we split the patterns into large and small ones. For the large patterns we compute the corresponding values of M sequentially while for the small patterns we reduce to unbalanced Batch-OV and use the fast algorithm for this problem shown in Lemma 2.5.

3.1 Patterns of Type $+|\circ|$ and $|\circ|$

As mentioned in the beginning of this section, we compute the set M of matched substring by partitioning the sub-patterns into large and small ones.

▶ **Lemma 3.2.** *Given a text* t *of length* n *and patterns* $\{p_i\}_i$ *of type* $\circ|$ *such that* $\sum_i |p_i| = m$*. We can compute* M *in time* $nm/2^{\Omega(\sqrt{\log\min(n,m)})}$*.*

▶ **Lemma 3.3** (Large Sub-Patterns). *Given a text* t *of length* n *and patterns* $p_1, \ldots, p_\ell$ *of type* $\circ|$ *such that* $\sum_{i=1}^{\ell} |p_i| \leq m$*. We can compute* M *in time* $\mathcal{O}(\ell n \log^2 \min(n,m) + m)$*.*

Proof. From a result by Cole and Hariharan [11] we know that there is a $\mathcal{O}(n\log^2 \hat{m} + \hat{m})$ time algorithm for $\circ|$-pattern matching with patterns of size $\hat{m}$. We run this algorithm sequentially for every pattern. We can ignore all p_i with $|p_i| > |\Sigma|n$ since they match more than n symbols. We get $|p_i| \leq \min(|\Sigma|n, m) \leq \min(n^2, m) \leq \min(n^2, m^2)$. Since $\log\min(n^2, m^2) = 2\log\min(n,m)$, each iteration takes time $\mathcal{O}(n\log^2\min(n,m) + |p_i|)$ and the claim follows. ◀

▶ **Lemma 3.4** (Small Sub-Patterns). *Given a text* t *of length* n *and patterns* $p_1, \ldots, p_m$ *of type* $\circ|$*. There is a* $f \in 2^{\Omega(\sqrt{\log\min(n,m)})}$ *such that the following holds: If* $|p_i| \leq f$ *for all* $i \in [m]$*, then we can compute* M *in time* $nm/2^{\Omega(\sqrt{\log\min(n,m)})}$ *with small error probability.*

We postpone the proof of this lemma and first combine the results for small and large patterns to proof the main theorem.

Proof of Lemma 3.2. Choose $f \in 2^{\Omega(\sqrt{\log\min(n,m)})}$ as in Lemma 3.4 and split the patterns into large patterns of size $> f$ and small patterns of size $\leq f$.

For the at most m/f large patterns compute $M_>$ by Lemma 3.3 in time $\mathcal{O}(m/f \cdot n\log^2\min(n,m) + m) \in nm/2^{\Omega(\sqrt{\log\min(n,m)})}$. Duplicate the ℓ small-patterns m/ℓ times and compute $M_\leq$ for the m small patterns by Lemma 3.4 in the claimed running time. ◀

Proof of Theorem 3.1 Item 1. Construct M by Lemma 3.2. Check for $|\circ|$-pattern matching whether $M = \emptyset$ since any matched substring is sufficient.

For $+|\circ|$-membership we construct a graph G with nodes $v_0, \ldots, v_n$ where we put an edge from v_{i-1} to v_j if $(i,j) \in M$. Then v_n is reachable from v_0 iff there is a decomposition of t into substrings which can be matched by the p_is. This reachability check can be performed in time $\mathcal{O}(n + |M|)$ by a depth-first search starting from v_0. ◀

For the proof of Lemma 3.4 we proceed as follows. For the construction of M for small sub-patterns we define some threshold f and check for every substring of t of length at most f whether there is a pattern that matches this substring. This check is reduced to Batch-OV by encoding the substrings and patterns as bit-vectors.

For small alphabets with $|\Sigma| < f$ this encoding is rather simple since we can use a one-hot encoding of the alphabet. But for larger alphabets this does not work as the dimension of the vectors would increase too much and the fast algorithm for Batch-OV could not be used

anymore. Therefore, we define a randomised encoding χ to ensure that the final bit-vectors are not to large. For simplicity we can assume $|\Sigma| = \Theta(\min(n,m))$ by padding Σ with fresh symbols. The construction in the following lemma is based on the idea of Bloom-Filters [6].

▶ **Lemma 3.5** (Randomised Characteristic Vector). *For a finite universe Σ and a threshold $f \leq 2^{\mathcal{O}(\sqrt{\log|\Sigma|})}$ there is a randomised $\chi : \mathcal{P}(\Sigma) \to \{0,1\}^d$ with $d \in \mathcal{O}(f \log|\Sigma|)$ such that for all $\sigma \in \Sigma$ and $S \subseteq \Sigma$ with $|S| \leq f$ the following holds:*

- *If $\sigma \in S$, then $\chi(\sigma) \coloneqq \chi(\{\sigma\}) \subseteq \chi(S)$, i.e. $\forall i \in [d] : \chi(\{\sigma\})[i] = 1 \implies \chi(S)[i] = 1$.*
- *If $\chi(\sigma) \subseteq \chi(S)$, then $\sigma \in S$ with high probability, i.e. $\geq 1 - 1/\operatorname{poly}(|\Sigma|)$.*

Proof. We define χ element-wise and set for $S \subseteq \Sigma$: $\chi(S)[i] \coloneqq \bigvee_{s\in S} \chi(s)[i]$, i.e. the bitwise OR over $\chi(s)$ for $s \in S$. Hence, the first claim already holds by definition. For each $\sigma \in \Sigma$ we define $\chi(\sigma)$ independently by setting $\chi(\sigma)[i] = 1$ with probability $1/f$ for all $i \in [d]$. Let $S \subseteq \Sigma$ with $|S| \leq f$ and $\sigma \in \Sigma \setminus S$. For all $i \in [d]$:

$$\begin{aligned}
\Pr[\chi(\sigma)[i] \not\subseteq \chi(S)[i]] &= \Pr[\chi(\sigma)[i] = 1 \wedge \chi(S)[i] = 0] \\
&= \tfrac{1}{f}\Big(1 - \tfrac{1}{f}\Big)^{|S|} \geq \tfrac{1}{f}\Big(1 - \tfrac{1}{f}\Big)^{f} \geq \tfrac{\mathrm{e}^{-2}}{f} \\
\Pr[\chi(\sigma) \subseteq \chi(S)] &= \prod_{i=1}^{d} \Pr[\chi(\sigma)[i] \subseteq \chi(S)[i]] = \prod_{i=1}^{d} (1 - \Pr[\chi(\sigma)[i] \not\subseteq \chi(S)[i]]) \\
&\leq \prod_{i=1}^{d} \Big(1 - \tfrac{\mathrm{e}^{-2}}{f}\Big) = \Big(\Big(1 - \tfrac{\mathrm{e}^{-2}}{f}\Big)^{d}\Big).
\end{aligned}$$

Setting $d = fc\ln|\Sigma|$ for some arbitrary $c > \mathrm{e}^2$, we get:

$$= \Big(1 - \tfrac{\mathrm{e}^{-2}}{f}\Big)^{f\cdot c\ln|\Sigma|} \leq \mathrm{e}^{-1/\mathrm{e}^2 \cdot c\ln|\Sigma|} = |\Sigma|^{-c/\mathrm{e}^2} = 1/\operatorname{poly}|\Sigma|. \qquad \blacktriangleleft$$

Proof of Lemma 3.4. Define $f = 2^{\sqrt{\epsilon}/3\cdot\sqrt{\log\min(n,m)}}$ with ϵ as in Lemma 2.5 and let a be some fresh symbol we add to Σ. Let $\chi : \mathcal{P}(\Sigma) \to \{0,1\}^{f^2}$ be as in Lemma 3.5. For simplicity one can think of χ as the one-hot encoding of alphabet Σ.

We define $T_j \coloneqq \{t_i \cdots t_{i+j-1} \mid 1 \leq i \leq n-j+1\}$ and $P_j \coloneqq \{p_i \mid \mathcal{L}(p_i) \subseteq \Sigma^j\}$ for all $j \in [f]$. Then replace all symbols and sub-patterns of type $|$ by bit-vectors by applying χ. Finally, pad every vector in T_j and P_j by $f - j$ repetitions of $\chi(a)$ and flip all values of P_j bit-wise such that 1s become 0s and vice versa. Let T be the set of all $\leq nf$ modified texts and P be the set of all m transformed patterns.

We observe that a text-vector in T is orthogonal to a pattern-vector in P iff the original text was matched by the original pattern. Since $f \cdot f^2 \leq 2^{\sqrt{\epsilon}\sqrt{\log\min(n,m)}} \leq 2^{\sqrt{\epsilon}\sqrt{\log\min(nf,m)}}$, we can apply Lemma 2.5 for T and P:

$$\frac{nfm}{2^{(\epsilon/\sqrt{\epsilon})\sqrt{\log\min(nf,m)}}} \leq \frac{nm}{2^{(\sqrt{\epsilon}-\sqrt{\epsilon}/3)\cdot\sqrt{\log\min(n,m)}}} \in \frac{nm}{2^{\Omega(\sqrt{\log\min(n,m)})}}. \qquad \blacktriangleleft$$

3.2 Patterns of Type $+|\circ+$ and $|\circ+$

First observe that even for small patterns M can be too large to be computed explicitly. For $t = 0^n1^n$ and $p = 0^+1^+$ we have $M = [1,n] \times [n+1,2n]$ and thus cannot write down M explicitly in time $o(nm)$.

To get around this problem we first define the *run-length encoding* $r(u)$ of a text u as in [4]: We have $r(\varepsilon) = \varepsilon$. For a non-empty string starting with σ, let ℓ be the largest integer such that the first ℓ symbols of u are σ. Append the tuple (σ, ℓ) to the run-length encoding

and recurse on u after removing the first ℓ symbols. We use the same approach for patterns of type $\circ+$. But if there occurs a σ^+ during these ℓ positions, we add $(\sigma, \geq \ell)$ to the encoding, otherwise $(\sigma, = \ell)$. We write σ^ℓ for the tuple (σ, ℓ) and similar for $\sigma^{=\ell}$ and $\sigma^{\geq\ell}$ to shorten notation. For example, $r(aaa^+b^+bc) = a^{\geq 3}b^{\geq 2}c^{=1}$.

The idea is to compute a subset of M which only contains those (i, j) such that there is no distinct (i', j') in the subset with $i' \leq i$ and $j' \geq j$ and both substrings of t are matched by the same pattern p_ℓ. We augment each tuple with two boolean flags, indicating whether the first and last run of the pattern p_ℓ contains a Kleene Plus. From this set $M' \subseteq \{0, 1\} \times [n] \times [n] \times \{0, 1\}$ we can fully recover M. For our above example we get $M' = \{(1, n, n + 1, 1)\}$.

▶ **Lemma 3.6.** *Given a text t of length n and patterns $\{p_i\}_i$ of type $\circ+$ such that $\sum_i |p_i| = m$. We can compute M' in time $nm/2^{\Omega(\sqrt{\log \min(n,m)})}$.*

▶ **Lemma 3.7** (Large Sub-Patterns)**.** *Given a text t of length n and patterns $p_1, \ldots, p_\ell$ of type $\circ+$ such that $\sum_{i=1}^{\ell} |p_i| \leq m$. We can compute M' in time $\mathcal{O}(\ell n \log^2 \min(n, m) + m)$.*

Proof. We modify all patterns such that their first and last run is of the form $\sigma^{=\ell}$, i.e. we remove every Kleene Plus from these two runs. There is a $\mathcal{O}(n \log^2 \hat{m} + \hat{m})$ time algorithm for $\circ+$-pattern matching with patterns of size $\hat{m}$ shown in [4]. We run this algorithm sequentially for each altered pattern. For every tuple (i, j) the algorithm outputs, we add (f, i, j, e) to M' where f and e are set to 1 iff the first and last run of the pattern contain a Kleene Plus, respectively.

We can ignore all p_i with $|p_i| > |\Sigma|n$ because they match more than n symbols. Since $|p_i| \leq \min(|\Sigma|n, m) \leq \min(n^2, m) \leq \min(n^2, m^2) = 2 \log \min(n, m)$, each iteration takes time $\mathcal{O}(n \log^2 \min(n, m) + |p_i|)$ and the claim follows. ◀

▶ **Lemma 3.8** (Small Sub-Patterns)**.** *For a text t of length n and patterns $p_1, \ldots, p_m$ of type $\circ+$, there is a $f \in 2^{\Omega(\sqrt{\log \min(n,m)})}$ such that the following holds: If $|p_i| \leq f$ for all $i \in [m]$, then we can compute M' in time $nm/2^{\Omega(\sqrt{\log \min(n,m)})}$.*

We postpone the proof of this lemma and first show the final upper bound as the proof of Lemma 3.2 also works for Lemma 3.6.

Proof of Theorem 3.1 Item 2. Use Lemma 3.6 to construct M' and check for $|\circ+$-pattern matching whether $M' = \emptyset$.

For $+|\circ+$-membership we define a graph $G = (V, E)$. Instead of having nodes $v_0, \ldots, v_n$ as for $+|\circ|$-membership we have for each node v_i three versions, $V \coloneqq \{v_i^0, v_i^1, v_i^2 \mid 0 \leq i \leq n\}$. The versions correspond to the different ways a suffix or prefix of a run can be matched. For node v_i^0 we need that all symbols are explicitly matched by a pattern. For v_i^1 we need that the suffix of the run containing t_i has to be matched by a pattern starting with t_i^+. For v_i^2 we say that the prefix has to be matched by a pattern ending with t_{i-1}^+. Hence, we add edges for the runs simulating the σ^+ of a pattern: For each run σ^ℓ from position i to j in t with $\ell > 1$ we add the edges (v_{k-1}^1, v_k^1) and (v_k^2, v_{k+1}^2) to the graph for $i \leq k < j$. Further, we add edges (v_i^2, v_i^0) and (v_i^0, v_i^1) to change between the states for all $0 \leq i \leq n$. While this construction solely depends on the text, we add for each $(f, i, j, e) \in M'$ the edge (v_{i-1}^f, v_j^{2e}) to the graph. We claim that there is a path from v_0^0 to v_n^0 if and only if $t \in \mathcal{L}((p_1 \mid \cdots \mid p_k)^+)$. We prove this claim in Appendix B. See Figure 2 for an example of the construction.

The time for the construction is linear in the output size. The graph has $\Theta(n)$ nodes and $|M'| + \mathcal{O}(n)$ edges. As the DFS runs in linear time, the overall runtime follows. ◀

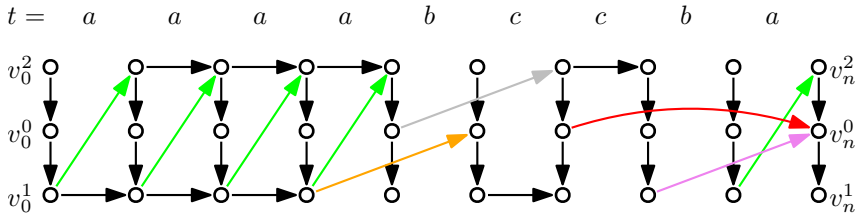


Figure 2 Graph for the pattern $(a^+ \mid a^+b \mid bc^+ \mid cba \mid b^+a)^+$ and text $aaaabccba$.

It remains to show how the set M' is constructed for small patterns.

Proof of Lemma 3.8. Set $f := 2^{\sqrt{\epsilon}/5\sqrt{\log \min(n,m)}}$ with ϵ as in Lemma 2.5 and consider all $\leq n/f^3$ many long runs of length $\geq f^3$ in t. Check for each long run by an exhaustive search whether there is a p_i such that the following holds: The run in the text is matched by one of the $\leq |p_i|$ runs in p_i and the remaining runs of p_i can match the contiguous parts of the text. This check can be performed in the following time for all large runs:

$$\frac{n}{f^3}\sum_{i=1}^{m}|p_i|^2 \leq \frac{n}{f^3}\sum_{i=1}^{m}f^2 \leq \frac{nm}{f}$$

Since a pattern can have at most f runs and each run matches now at most f^3 symbols, it remains to check substrings of t of length at most f^4. Hence, define $T = \{t_i \cdots t_{i+j-1} \mid \forall j \in [f^4], i \in [n-j+1]\}$ and ignore all substrings with more than f runs or runs longer than f^3. Convert these substrings and the patterns into bit-vectors by replacing the runs by the following bit-vectors of length $2\log|\Sigma| + 2f^3$:

$$c^r \mapsto \langle c\rangle\overline{\langle c\rangle}0^r1^{f^3-r}1^r0^{f^3-r} \qquad c^{=r} \mapsto \overline{\langle c\rangle}\langle c\rangle1^r0^{f^3-r}0^r1^{f^3-r} \qquad c^{\geq r} \mapsto \overline{\langle c\rangle}\langle c\rangle1^r0^{f^3-r}0^{f^3}$$

$\langle c\rangle$ denotes the unique binary representation of symbol c and $\overline{\langle c\rangle}$ its bit-wise negation. One can easily see that two such vectors are orthogonal if and only if the runs match each other. Thus, a text and a pattern vector resulting from this transformation are orthogonal iff the text is matched by the pattern. By padding the vectors with 1s we normalise their length but still preserve orthogonality between text and pattern vectors with the same number of runs. Let T' and P' be the resulting sets with $\leq nf^4$ and m elements, respectively.

From $\log|\Sigma| \leq \log \min(n,m) \leq f$ we get $f(2\log|\Sigma| + 2f^3) \leq f^5 \leq 2^{\sqrt{\epsilon}\sqrt{\log \min(nf^4,m)}}$ and hence can apply Lemma 2.5 for T' and P'. Actually we have to partition P' depending on whether a pattern has a Kleene Plus in its first and last run. Thus, we need four iterations but we can always duplicate patterns such that there are m patterns in each group.

$$\frac{nf^4m}{2^{\epsilon/\sqrt{\epsilon}\sqrt{\log \min(nf^4,m)}}} \leq \frac{nm}{2^{(\sqrt{\epsilon}-4/5\sqrt{\epsilon})\sqrt{\log \min(n,m)}}} \in \frac{nm}{2^{\Omega(\sqrt{\log \min(n,m)})}}.$$ ◀

4 Lower Bounds for Pattern Matching

Abboud and Bringmann showed in [1] a lower bound for pattern matching (and membership) in general of $\mathcal{O}(nm/\log^{7+\epsilon} n)$, unless FSH is false. We use this result and the corresponding reduction as a basis to show similar lower bounds for the remaining hard pattern types. But we also do not start our reductions directly from Formula-SAT but from Formula-Pair as defined in Section 2 and use the corresponding Formula-Pair Hypothesis from Hypothesis 2.3.

▶ **Theorem 4.1.** *There are constants $c_{\circ\star} = 76, c_{\circ+\circ} = c_{\circ|+} = 72, c_{\circ|\circ} = 81$, and $c_{\circ+|} = 27$ such that pattern matching with patterns of type $T \in \{\circ\star, \circ+\circ, \circ|\circ, \circ+|, \circ|+\}$ cannot be solved in time $\mathcal{O}(nm/\log^{c_T} n)$ even for constant sized alphabets, unless FPH is false.*

We show the lower bounds by a reduction from Formula-Pair to pattern matching:

▶ **Lemma 4.2.** *Given a Formula-Pair instance with a formula of size s, depth d, and sets A and B with n and $m \leq n$ assignments. (If $m > n$, swap A and B.) We can reduce this to pattern matching with a text t and a pattern p of type $T \in \{\circ\star, \circ+\circ, \circ|\circ, \circ+|, \circ|+\}$ over a constant sized alphabet in time linear in the output size.*

$|t| \in \mathcal{O}(n5^d s \log s)$ except for $\circ+|$, there we have $|t| \in \mathcal{O}(n2^d s \log s)$ Further, $|p| \in \mathcal{O}(mb_T^d s \log s)$ with $b_{\circ\star} = 6, b_{\circ+\circ} = b_{\circ|+} = 5, b_{\circ|\circ} = 8$, and $b_{\circ+|} = 1$.

Proof of Theorem 4.1. We show the result only for patterns of type $\circ+\circ$, the proof for the other types is analogous.

Let F be a formula of size s with two sets of n half-assignments each, and d be the depth of F. Applying the depth-reduction technique of Bonet and Buss [7] gives us an equivalent formula F' with size $s' \leq s^2$ and depth $d' \leq 6 \ln s$. By Lemma 4.2 we get a pattern matching instance with a text t and pattern p. Both of size $\mathcal{O}(n5^{d'} s' \log s') = \mathcal{O}(n5^{6 \ln s} s^2 \log s) = \mathcal{O}(ns^{6 \ln 5+2} \log s)$. Now assume there is an algorithm for pattern matching with the stated running time and run it on t and p:

$$\mathcal{O}\left(\frac{ns^{6 \ln 5+2} \log s \cdot ns^{6 \ln 5+2} \log s}{\log^{72}(ns^{6 \ln 2+2} \log s)}\right) \subseteq \mathcal{O}\left(\frac{n^2 s^{12 \ln 5+4} \log^2 s}{\log^{72} n}\right) \subseteq \mathcal{O}\left(\frac{n^2 s^{23.314}}{\log^{72} n}\right).$$

But this contradicts FPH which was assumed to be true. ◀

4.1 Proof of Lemma 4.2

Again we only give the proof for patterns of type $\circ+\circ$. The reduction for the other pattern types can be found in the full version of the paper. We first encode the evaluation of a formula on two half-assignments, then the encoding for finding such a pair. In the following we define the actual text t_g and pattern p_g. The universal text u_g and universal pattern q_g are needed for technical purposes and do not depend on the assignments.

4.1.1 Encoding the Formula

A formula of size s (i.e. s leaves) has $s-1$ inner gates and thus $2s-1$ gates in total. We assign every gate g a unique integer in $[2s-1]$, its ID, and write $\langle g \rangle$ for the binary encoding of the ID of gate g. We can always see $\langle g \rangle$ as a sequence of $\lfloor \log(2s-1) \rfloor + 1 \leq \lfloor \log s \rfloor + 2 = \Theta(\log s)$ bits padded with zeros if necessary. For a fixed gate g we define a separator gadget $G := 2\langle g \rangle 2$ with 2 as a new symbol.

INPUT Gate The text and the pattern depend on the variable that is read:

For $F_g(a, b) = a_i$ define $t_g := 0a_i1$ as the text and $p_g := 0^+11^+$ as the pattern.

For $F_g(a, b) = b_i$ define $t_g := 011$ as the text and $p_g := 0^+b_i1^+$ as the pattern.

Define $u_g := 0011$ as the universal text and $q_g := 0^+1^+$ as the universal pattern.

AND Gates We define: $t_g := t_1 G t_2$, $p_g := p_1 G p_2$, $u_g := u_1 G u_2$, and $q_g := q_1 G q_2$.

OR Gates The texts and the patterns for gate g are defined as follows where the parentheses are just for grouping and are not part of the text or pattern:

$$\begin{aligned} t_g &:= (u_1GGu_2)G(u_1GGu_2)G(t_1GGt_2)G(u_1GGu_2)G(u_1GGu_2) \\ u_g &:= (u_1GGu_2)G(u_1GGu_2)G(u_1GGu_2)G(u_1GGu_2)G(u_1GGu_2) \\ q_g &:= (u_1GGu_2)G(u_1GGu_2)G(q_1GGq_2)G(u_1GGu_2)G(u_1GGu_2) \\ p_g &:= (u_1GGu_2G)^+(q_1GGp_2)G(p_1GGq_2)(Gu_1GGu_2)^+ \end{aligned}$$

▶ **Lemma 4.3** (Correctness of the Construction). *For all assignments a, b and gates g:*

- $F_g(a,b) = \mathit{true} \iff t_g(a) \in \mathcal{L}(p_g(b))$
- $t_g(a) \in \mathcal{L}(q_g)$
- $u_g \in \mathcal{L}(q_g) \cap \mathcal{L}(p_g(b))$

Proof. The proofs of the second and third claim follow inductively from the encoding of the gates and especially because of the encoding of the INPUT gate. For the first claim we do a structural induction on the output gate of the formula.

INPUT Gate "⇒" Follows directly from the definition.

INPUT Gate "⇐" If the gate is not satisfied, then there are not enough 0s or 1s in the text than the pattern has to match.

AND Gate "⇒" Follows directly from the definition.

AND Gate "⇐" By the uniqueness of the binary encoding, the G in the middle of the text and the pattern have to match. Since the whole text is matched, we get $t_1 \in \mathcal{L}(p_1)$ and $t_2 \in \mathcal{L}(p_2)$ and $F_g(a,b)$ is satisfied by the induction hypothesis.

OR Gate "⇒" $F_g(a,b) = F_{g_1}(a,b) \vee F_{g_2}(a,b) = \mathsf{true}$. Assume w.l.o.g. that $F_{g_1}(a,b) = \mathsf{true}$, the other case is symmetric. Repeat $(u_1GGu_2G)^+$ only once to transform q_1GGp_2 into the second u_1GGu_2 by our third claim of the lemma. Now p_1GGq_2 matches t_1GGt_2 by the second claim and the assumption $t_1 \in \mathcal{L}(p_1)$. Finally, we match $Gu_1GGu_2Gu_1GGu_2$ by two repetitions of $(Gu_1GGu_2)^+$.

OR Gate "⇐" By the uniqueness of the binary encoding there are exactly 14 Gs in the text and the pattern can match 11 Gs when taking both repetitions once. Since each additional repetition increases the number by 3, exactly one repetition is taken twice. If the first repetition is taken once, the following q_1GGp_2 has to match the second u_1GGu_2 in the text. But then p_1 is transformed into t_1 showing that F_g is satisfied by the inductive hypothesis. The case for the second repetition is symmetric. ◀

Length of the text and the pattern. All texts and patterns for a specific gate only depend on the texts and patterns for the two sub-gates. Thus, we can compute the texts and patterns in a bottom-up manner and the encoding can be done in time linear in the size of the output. It remains to analyse the length of the texts and the size of the patterns:

▶ **Lemma 4.4.** $|u_r|, |t_r|, |p_r|, |q_r| \in \mathcal{O}(5^d s \log s)$.

Proof. p_g is obviously smaller than u_g. Since the sizes of u_g, t_g, and q_g are asymptotically equal, it suffices to analyse the length of u_g: $|u_g| \leq 5|u_1| + 5|u_2| + \mathcal{O}(\log s)$. Inductively over the $d(F_g)$ levels of F_g, i.e. the depth of F_g, this yields $|u_g| \leq \mathcal{O}(5^{d(F_g)} s \log s)$. The factor of $s \log s$ is due to the $\mathcal{O}(s)$ inner gates each introducing $\mathcal{O}(\log s)$ additional symbols. ◀

4.1.2 Final Reduction

In the first part of the reduction we have seen how to evaluate a formula on one specific pair of half-assignments. It remains to design a text and a pattern such that such a pair of half-assignments can be chosen. For this let $A = \{a^{(1)}, \dots, a^{(n)}\}$ be the first set and $B = \{b^{(1)}, \dots, b^{(m)}\}$ be the second set of half-assignments. Inspired by the reduction in Section 3.4 in the full version of [4] we define the final text and pattern as follows:

$$t \coloneqq \bigodot_{i=1}^{3n} \Big(33u_r3u_r3u_r3t(a^{(i)})3u_r3u_r3u_r3u_r\Big)$$

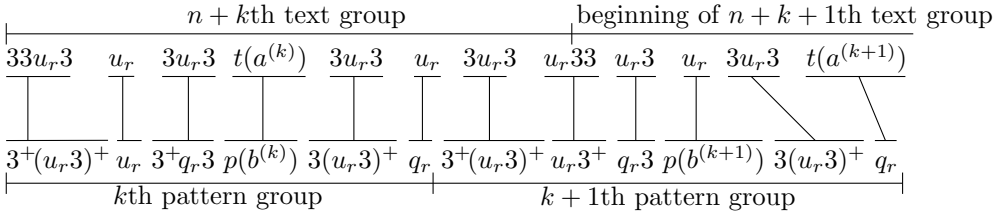
$$p \coloneqq 3u_r3u_r3u_r3u_r \bigodot_{j=1}^{m} \Big(3^+(u_r3)^+u_r3^+q_r3p(b^{(j)})3(u_r3)^+q_r\Big)3u_r3u_r3u_r3u_r.$$

Where we set $a^{(j)} = a^{(j \mod n)}$ for $j \in [n+1, 3n]$. We call the concatenations in t and p for each i and j the ith text group and the jth pattern group, respectively.

▶ **Lemma 4.5.** *If there are $a^{(k)}$ and $b^{(l)}$ such that $F(a^{(k)}, b^{(l)}) = \mathsf{true}$, then $t \in \mathcal{M}(p)$.*

Proof. Assume w.l.o.g. $a^{(k)}$ and $b^{(k)}$ satisfy F. Otherwise we have to shift the indices for the text and the pattern accordingly in the proof. We match the prefix of p to the suffix of the nth text group. Then we match the $n+i$th text group by the ith pattern group for $i = 1, \dots, k-1$: Both $(u_r3)^+$ are repeated twice. Then the remaining parts are matched in a straightforward way by transforming the q_rs into $t(a^{(i)})$ and u_r, and $p(b^{(i)})$ into u_r.

Then, we match the kth and $k+1$th pattern group to the $n+k$th text group and a part of the $n+k+1$th text group:



For the last step we shift the groups in the remaining text t' such that it becomes easier to prove which part of the text the remaining pattern matches:

$$\begin{aligned} t' =& 3u_r3u_r3u_r3u_r \bigodot_{i=n+k+2}^{3n} \Big(33u_r3u_r3u_r3t(a^{(i)})3u_r3u_r3u_r3u_r\Big) \\ =& \bigodot_{i=n+k+2}^{3n} \Big(3u_r3u_r3u_r3u_r33u_r3u_r3u_r3t(a^{(i)})\Big)3u_r3u_r3u_r3u_r. \end{aligned}$$

For each of the remaining pattern groups the first repetition is taken three times. With this the $n+i$th group of t' and the ith pattern group are matched in a straightforward way for $i = k+2, \dots, m$. The suffix of the pattern is matched to the start of the $n+m+1$th text group in the obvious way. ◀

▶ **Lemma 4.6.** *If $t \in \mathcal{M}(p)$, then there are $a^{(k)}$ and $b^{(l)}$ such that $F(a^{(k)}, b^{(l)}) = \mathsf{true}$.*

Proof. By the design of the pattern and the text, there must be a $j \leq n$ such that the prefix of the pattern is matched to the suffix of the $j-1$th text group. Likewise the suffix of the pattern has to match the same sequence in some other text group because nowhere else the

four $3u_r$ could be matched. Thus, not all text groups and pattern groups match each other precisely and there is a text group k and a pattern group l such that the pattern group does not match the whole text group or it matches more than this group. Choose the first of these groups, i.e. the pair with smallest k and l.

Since all prior groups have been matched precisely, the first repetition can be taken at most twice. Otherwise the following u_r could not be transformed into a part of the text. Now assume it is repeated exactly once. Then the following u_r matches the second u_r of the text group. Since 3 is a fresh symbol, q_r has to match the third u_r. But then $p(b^{(k)})$ has to be transformed into $t(a^{(l)})$ and Lemma 4.3 gives us a satisfying assignments.

It remains to check the case when $(u_r 3)^+$ is repeated twice. Then q_r is transformed into $t(a^{(l)})$ and $p(b^{(k)})$ is transformed into the fourth u_r. The second repetition has to be taken exactly twice in this case. Because otherwise the 33 from the beginning of the next text group could not be matched. But if the pattern $(u_r 3)^+$ is repeated twice, this pattern group is completely matched to a text group, contradicting our assumption. ◀

▶ **Lemma 4.7.** *The final text has length $\mathcal{O}(n5^d s \log s)$ and the pattern has size $\mathcal{O}(m5^d s \log s)$.*

By this we conclude the proof of Lemma 4.2 for this pattern type. ⌟

5 Lower Bounds for Membership

Instead of giving all reductions from scratch, we reduce pattern matching to membership and make use of the results in Lemma 4.2. By this we get the same bounds as for pattern matching given in Theorem 4.1. For the remaining pattern type $|{+}|\circ$ we give a new reduction from scratch which is necessary due to the missing concatenation as outer operation.

5.1 Reducing Pattern Matching to Membership

▶ **Lemma 5.1** (Reducing Pattern Matching to Membership). *Given a text t and a pattern p with type in $\{\circ\star, \circ{+}\circ, \circ|\circ, \circ{+}|, \circ|{+}\}$ over a constant sized alphabet.*

We can construct a text t' and a pattern p' of the same type as p in linear time such that $t \in \mathcal{M}(p) \iff t' \in \mathcal{L}(p')$. Further, $|t'| \in \mathcal{O}(|t|)$ and $|p'| \in \mathcal{O}(|t| + |p|)$, except for $\circ{+}|$, there we even have $|p'| \in \mathcal{O}(|p|)$.

Proof. Again we only show the proof for type $\circ{+}\circ$. The reductions for the other types can be found in the full version.

Let $\Sigma = \{1, \dots, s\}$ be the alphabet. We first encode every symbol such that we can simulate a universal pattern (i.e. matching any symbol) by some gadget U of type $\circ{+}$. Let $f : \Sigma \to \Sigma^{s+1}$ be this encoding with $f(x) = 1 \cdots (x-1)xx(x+1) \cdots s$. Since we can extend f in the natural way to texts by applying it to every symbol, we can also modify patterns of type $\circ{+}\circ$ by applying f to every symbol *without* changing the type. After applying f we still have $t \in \mathcal{M}(p) \iff f(t) \in \mathcal{M}(f(p))$.

For the step from pattern matching to membership we set $U \coloneqq 1^+2^+ \cdots s^+$ and $R \coloneqq 12 \cdots s$. Obviously $R \in \mathcal{L}(U)$ and $f(\sigma) \in \mathcal{L}(U)$ for all $\sigma \in \Sigma$. But we also get $R \notin \mathcal{L}(f(\sigma))$ since R does not contain a repetition of σ. Finally, we define $t' \coloneqq R^{|t|+1} f(t) R^{|t|+1}$ and $p' = R^+ U^{|t|} f(p) U^{|t|} R^+$. We claim $t \in \mathcal{M}(p) \iff t' \in \mathcal{L}(p')$.

"$\Rightarrow$" If $t \in \mathcal{M}(p)$, then there is a substring $\hat{t}$ of t matched by p. By the above observations, $f(p)$ matches $f(\hat{t})$ which is a substring of $f(t)$. Then we use $U^{|t|}$ to match the not matched suffix and prefix of $f(t)$ and a part of $R^{|t|+1}$. The remaining repetitions of R are matched by the R^+ in the beginning and the end. "$\Leftarrow$" If $t' \in \mathcal{L}(p')$, then $f(p)$ has to match some substring of $f(t)$ because R cannot be matched by the above observation. ◀

5.2 Patterns of Type $|+|\circ$

Even though the remaining hard pattern type $|+|\circ$ does not have a concatenation as outer operation, we can still show a similar lower bound as for the other types.

▶ **Theorem 5.2.** *$|+|\circ$-membership cannot be solved in time $\mathcal{O}(nm/\log^{17} n)$ even for constant sized alphabets, unless FPH is false.*

To proof the theorem it suffices to show that Formula-Pair can be reduced to membership with a text of length $\mathcal{O}(ns^2 \log s)$ and a pattern of size $\mathcal{O}(ms^3 \log s)$. Then the claim directly follows from the definition of FPH as for the other types.

Idea of the reduction. As for the other lower bounds, we first encode the evaluation of the formula on two fixed half-assignments. We define for each gate g a text t_g and two dictionaries D_g^M and D_g^S of words. The final dictionary for a gate g is defined as $D_g = \bigcup_{g' \in F_g} D_{g'}^S \cup D_{g'}^M$. The final pattern is D_r^+ where r is the root of F.

D_g^M corresponds to p_g and allows us to match the whole text t_g if the formula is satisfied. The texts of the sub-gates are then matched by the corresponding dictionaries. But for the OR gate we have to be able to ignore the evaluation of one sub-formula. For this we define the set D_g^S which corresponds to q_g and allows us to match the text independently from the assignments. As main idea we include the path from the root of the formula to the current gate in the encoding. This trace is appended to the text as a prefix and in reverse as suffix. The words in D_g^M for OR gates g allow us to jump to a gate in such a trace of exactly one sub-formula. Then we use corresponding words from D^S to propagate this jump to the sub-formulas. Because the included trace started at the root, we can proceed to the INPUT gates. There we add words to accept all evaluations of the gate. For the way back up we add the corresponding words in reverse to the dictionaries.

We make sure that these words are just used at one specific position by embedding the encoding of the corresponding gate in the trace. Since the gate number can be made unique these words can only be used at one specific position. This procedure allows us to write down the words as a set and not as a concatenation as for the other reductions.

Encoding the Formula. We identify each gate g with its ID, i.e. an integer in $[2s]$. Let $\langle g \rangle$ be the binary encoding of the gate ID with $\lfloor \log s \rfloor + 2 = \Theta(\log s)$ bits padded with zeros if necessary. Further, let $h_0 h_1 \dots h_d$ be the path from the root $r = h_0$ of F to the gate $g = h_d$ of depth $d \geq 0$. To simplify notation we define $h_i^g = 2\langle h_i \rangle \langle g \rangle 2$, i.e. the encoding of the gate on the path and the gate where the path ends.

INPUT Gates We set $D_g^S \coloneqq \{h_i^g \dots h_d^g 0 h_d^g \dots h_i^g, h_i^g \dots h_d^g 1 h_d^g \dots h_i^g \mid i \in [d]\}$.
For $F_g(a,b) = a_i$, we set $t_g \coloneqq h_0^g \dots h_d^g a_i h_d^g \dots h_0^g$ and $D_g^M \coloneqq \{h_0^g \dots h_d^g 1 h_d^g \dots h_0^g\}$
For $F_g(a,b) = b_i$, we set $t_g \coloneqq h_0^g \dots h_d^g 1 h_d^g \dots h_0^g$ and $D_g^M \coloneqq \{h_0^g \dots h_d^g b_i h_d^g \dots h_0^g\}$

AND Gate We define the text and the corresponding dictionaries as follows:

$$
\begin{aligned}
t_g &\coloneqq h_0^g \dots h_d^g t_1 t_2 h_d^g \dots h_0^g \\
D_g^M &\coloneqq \{h_0^g \dots h_d^g, h_d^g \dots h_0^g\} \\
D_g^S &\coloneqq \{h_i^g \dots h_d^g h_0^{g_1} \dots h_{i-1}^{g_1}, h_{i-1}^{g_1} \dots h_0^{g_1} h_0^{g_2} \dots h_{i-1}^{g_2}, h_{i-1}^{g_2} \dots h_0^{g_2} h_d^g \dots h_i^g \mid i \in [d]\}
\end{aligned}
$$

OR Gate We define the text and the additional dictionaries for g as:

$$\begin{aligned} t_g &:= h_0^g \dots h_d^g t_1 h_d^g t_2 h_d^g \dots h_0^g \\ D_g^M &:= \{h_0^g \dots h_d^g, h_d^g h_0^{g_2} \dots h_d^{g_2}, h_d^{g_2} \dots h_0^{g_2} h_d^g \dots h_0^g\} \\ &\quad \cup \{h_0^g \dots h_d^g h_0^{g_1} \dots h_d^{g_1}, h_d^{g_1} \dots h_0^{g_1} h_d^g, h_d^g \dots h_0^g\} \\ D_g^S &:= \{h_i^g \dots h_d^g h_0^{g_1} \dots h_{i-1}^{g_1}, h_{i-1}^{g_1} \dots h_0^{g_1} h_d^g h_0^{g_2} \dots h_{i-1}^{g_2}, h_{i-1}^{g_2} \dots h_0^{g_2} h_d^g \dots h_i^g \mid i \in [d]\} \end{aligned}$$

▶ **Lemma 5.3.** *For all assignments a, b and gates g:*

- $t_g(a) \in \mathcal{L}(h_0^g \dots h_{i-1}^g (D_g(b))^+ h_{i-1}^g \dots h_0^g)$ *for all* $i \in [d]$.
- $t_g(a) \notin \mathcal{L}(h_0^g \dots h_{i-1}^g (D_g(b))^+ h_{j-1}^g \dots h_0^g)$ *for all* $i \neq j \in [0, d]$, *where* $h_0^g h_{-1}^g$ *and* $h_{-1}^g h_0^g$ *denote the empty string.*

Proof. The first claim follows by a structural induction on the output gate using only words from $D_{g'}^S$ for the current gate g'. Likewise we show the second case by a structural induction on the output gate.

INPUT Gate The statement holds by the definition of the dictionary.

AND Gate Assume the claim is false for g. We can only match the "prefix" $h_i^g \dots h_d^g$ with the word $h_i^g \dots h_d^g h_0^{g_1} \dots h_{i-1}^{g_1}$. And analogously for the "suffix". The joining part of $t_1 t_2$ has to be matched by some $h_{k-1}^{g_1} \dots h_0^{g_1} h_0^{g_2} \dots h_{k-1}^{g_2}$ for $k \in [0, \dots, d]$ (possibly the empty string). Hence, $t_1 \in \mathcal{L}(h_0^{g_1} \dots h_{i-1}^{g_1} (D_{g_1}(b))^+ h_{k-1}^{g_1} \dots h_0^{g_1})$ and $t_2 \in \mathcal{L}(h_0^{g_2} \dots h_{k-1}^{g_2} (D_{g_2}(b))^+ h_{j-1}^{g_2} \dots h_0^{g_2})$. But from $i \neq j$ it follows that $k \neq i$ or $k \neq j$ and we have a contradiction to the induction hypothesis for g_1 and g_2.

OR Gate The "prefix" $h_i^g \dots h_d^g$ has to be matched by $h_i^g \dots h_d^g h_0^{g_1} \dots h_{i-1}^{g_1}$ and analogously for the "suffix". If the joining part of $t_1 h_d^g t_2$ was matched by $h_{k-1}^{g_1} \dots h_0^{g_1} h_d^g h_0^{g_2} \dots h_{k-1}^{g_2}$ for some $k \in [d]$, the same proof as for the AND gate applies. Otherwise, either $h_d^{g_1} \dots h_0^{g_1} h_d^g$ or $h_d^g h_0^{g_2} \dots h_d^{g_2}$ was used. Let it w.l.o.g. be the first one. Since $i \in [0, d]$, we have $i \neq d+1$ and hence a contradiction to the inductive hypothesis for t_1. ◀

▶ **Lemma 5.4** (Correctness of the Construction)**.** *For all assignments a, b and gates g:* $F_g(a, b) = \mathit{true} \iff t_g(a) \in \mathcal{L}((D_g(b))^+)$.

Proof. We proof the claim by an induction on the output gate.

INPUT Gate Follows directly from the construction of the text and the dictionary.

AND Gate "⇒" We can use D_1^+ and D_2^+ to match t_1 and t_2 by the induction hypothesis, respectively. The remaining parts are matched by the words in D_g^M.

AND Gate "⇐" The initial and last h_0^g of the text have to be matched. Since the gate g is part of the encoding, we can only use words from D_g^M for this. It follows directly that t_1 is matched by words from D_1 because the initial $h_0^{g_1}$ has to be matched too and the words in D_g^S are not eligible for this. The same argument shows that t_2 is matched by words from D_2. Hence, the claim follows by the induction hypothesis.

OR Gate "⇒" Assume w.l.o.g. that $F_{g_1}(a, b) = \mathsf{true}$, the other case is symmetric. We match the prefix of t_g in the obvious way by the corresponding word from D_g^M. By assumption we match t_1 with words from D_1. The prefix $h_0^{g_2} \dots h_d^{g_2}$ of t_2 is matched by the corresponding word in D_g^M. By the first claim of the previous lemma, we have $t_2 \in \mathcal{L}(h_0^{g_2} \dots h_d^{g_2} (D_{g_2})^+ h_d^{g_2} \dots h_0^{g_2})$ and the remaining suffix can be matched by the corresponding word from D_g^M.

OR Gate "⇐" By Lemma 5.3 the joining part of $t_1 h_d^g t_2$ has to be matched by either $h_d^g h_0^{g_2} \dots h_d^{g_2}$ or $h_d^{g_1} \dots h_0^{g_1} h_d^g$. Let it w.l.o.g. be the first one. Then t_1 has to be matched by words from D_1 again by the lemma. The inductive hypothesis gives us a satisfying assignment. ◀

▶ **Lemma 5.5.** *We have the following size bounds:*
- $|t_r| \in \mathcal{O}(sd \log s) \subseteq \mathcal{O}(s^2 \log s)$
- $|D_r| \in \mathcal{O}(sd) \subseteq \mathcal{O}(s^2)$
- $\forall x \in D_r : |x| \in \mathcal{O}(d \log s) \subseteq \mathcal{O}(s \log s)$

Proof. The lemma follows directly from the definitions and the observations that $|t_g| \leq |t_1| + |t_2| + \mathcal{O}(d \log s)$, $|D_g^M| \in \mathcal{O}(1)$, and $|D_g^S| \in \mathcal{O}(d)$. ◀

Outer OR. Let $A = \{a^{(1)}, \ldots, a^{(n)}\}$ be the first set and $B = \{b^{(1)}, \ldots, b^{(m)}\}$ be the second set of half-assignments. Again we encode A by the text and B by the pattern. For this we observe that the first step of the reduction produced a pattern of type $+|\circ$. Thus, we can use the outer alternative to encode the outer OR to select a specific $b^{(j)}$. To match the whole text, we blow up the text and the pattern and pad each symbol with three new symbols such that we can distinguish between the following three matching states: (1) ignore the padding and match a part of the original text to the original pattern, i.e. we evaluate the formula on two half-assignments. (2) Match an arbitrary prefix, i.e. the symbols before the actual match in state (1). (3) Match some arbitrary suffix, i.e. the symbols after the actual match from state (1). We allow a change between these states only at the end of a text group and require that we go through all three states if and only if the text can be matched by the pattern.

▶ **Definition 5.6** (Blow-Up of a Text). *Let $t = t_1 \cdots t_n$ be a text of length n and u be some arbitrary string. We define $t\Uparrow^u := ut_1ut_2 \cdots ut_n$ and extend it in the natural way to sets.*

Using this we define the final text and pattern as follows:

$$t := 563 \bigodot_{i=1}^{n} \Big(t(a^{(i)})3 \Uparrow^{456} \Big) 45$$

$$p := p_1^+ \mid p_2^+ \mid \cdots \mid p_m^+$$

$$p_j := 5604 \mid 5614 \mid 5624 \mid 5634 \mid 563 \mid D_r(b^{(j)}) \Uparrow^{456} \mid 456345 \mid 6045 \mid 6145 \mid 6245 \mid 6345$$

▶ **Lemma 5.7.** *If there are $a^{(k)}$ and $b^{(l)}$ such that $F(a^{(k)}, b^{(l)}) = \mathsf{true}$, then $t \in \mathcal{L}(p)$.*

Proof. It suffices to show that we can match t to p_l^+. The prefix of t and the first $k-1$ text groups are matched by repetitions of $56x4$ for values $x \in \{0, 1, 2, 3\}$ while the last three symbols of the $k-1$th group are matched by 563. This is possible by our blow-up with 456. By Lemma 5.4 and the definition of the blow-up we get $t(a^{(k)}) \Uparrow^{456} \in \mathcal{L}((D_r(b^{(l)}) \Uparrow^{456})^+)$. The following 456345 is matched by the corresponding pattern while the remaining symbols of the text are matched in a straight forward way by repetitions of $6x45$. ◀

▶ **Lemma 5.8.** *If $t \in \mathcal{L}(p)$, then there are $a^{(k)}$ and $b^{(l)}$ such that $F(a^{(k)}, b^{(l)}) = \mathsf{true}$.*

Proof. By the structure of the pattern we can already fix l. As there is no way to match the text just with words $56x4$ or $6x45$, the word 563 must have been used at the end of some group to switch to the first state. Hence, let the kth text group be the first group not matched by words of the form $56x4$. Observe that we cannot directly switch to an application of $6x45$ and thus get $t(a^{(k)}) \Uparrow^{456} \in \mathcal{L}((D_r(b^{(l)}) \Uparrow^{456})^+)$. Since the blow-up 456 always matches each other, we can ignore it and get $t(a^{(k)}) \in \mathcal{L}(D_r(b^{(l)})^+)$ proving the claim by Lemma 5.4. ◀

▶ **Corollary 5.9.** *The final text has length $\mathcal{O}(nsd \log s) \subseteq \mathcal{O}(ns^2 \log s)$ and the pattern has size $\mathcal{O}(msd^2 \log s) \subseteq \mathcal{O}(ms^3 \log s)$.*

This finishes the proof of Theorem 5.2. ⌟

References

1. Amir Abboud and Karl Bringmann. Tighter connections between formula-sat and shaving logs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. Full version: arXiv:1804.08978. doi:10.4230/LIPIcs.ICALP.2018.8.
2. Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 218–230. SIAM, 2015. doi:10.1137/1.9781611973730.17.
3. Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, 1975. doi:10.1145/360825.360855.
4. Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 457–466. IEEE Computer Society, 2016. Full version: arXiv:1511.07070. doi:10.1109/FOCS.2016.56.
5. Philip Bille and Mikkel Thorup. Faster regular expression matching. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2009. doi:10.1007/978-3-642-02927-1_16.
6. Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970. doi:10.1145/362686.362692.
7. Maria Luisa Bonet and Samuel R. Buss. Size-depth tradeoffs for boolean fomulae. *Inf. Process. Lett.*, 49(3):151–155, 1994. doi:10.1016/0020-0190(94)90093-0.
8. Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 307–318. IEEE Computer Society, 2017. Full version: arXiv:1611.00918. doi:10.1109/FOCS.2017.36.
9. Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255. SIAM, 2016. doi:10.1137/1.9781611974331.ch87.
10. Ruiwen Chen, Valentine Kabanets, and Nitin Saurabh. An improved deterministic #sat algorithm for small de morgan formulas. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25–29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2014. doi:10.1007/978-3-662-44465-8_15.
11. Richard Cole and Ramesh Hariharan. Verifying candidate matches in sparse and wildcard matching. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 592–601. ACM, 2002. doi:10.1145/509907.509992.
12. Theodore Johnson, S. Muthukrishnan, and Irina Rozenbaum. Monitoring regular expressions on out-of-order streams. In Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis, editors, *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 1315–1319. IEEE Computer Society, 2007. doi:10.1109/ICDE.2007.369001.
13. Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. Proton: multitouch gestures as regular expressions. In Joseph A. Konstan, Ed H. Chi, and Kristina Höök, editors, *CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA – May 05–10, 2012*, pages 2885–2894. ACM, 2012. doi:10.1145/2207676.2208694.

14 Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977. doi:10.1137/0206024.

15 Ilan Komargodski, Ran Raz, and Avishay Tal. Improved average-case lower bounds for demorgan formula size. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 588–597. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.69.

16 David Landsman. RNP-1, an RNA-binding motif is conserved in the DNA-binding cold shock domain. *Nucleic Acids Research*, 20(11):2861–2864, June 1992. doi:10.1093/nar/20.11.2861.

17 Quanzhong Li and Bongki Moon. Indexing and querying XML data for regular path expressions. In Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass, editors, *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 361–370. Morgan Kaufmann, 2001. URL: http://www.vldb.org/conf/2001/P361.pdf.

18 Makoto Murata. Extended path expressions for XML. In Peter Buneman, editor, *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*. ACM, 2001. doi:10.1145/375551.375569.

19 Eugene W. Myers. A four russians algorithm for regular expression pattern matching. *J. ACM*, 39(2):430–448, 1992. doi:10.1145/128749.128755.

20 Gonzalo Navarro and Mathieu Raffinot. Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. *Journal of Computational Biology*, 10(6):903–923, 2003. PMID: 14980017. doi:10.1089/106652703322756140.

21 Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 183–192. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.25.

22 Ken Thompson. Regular expression search algorithm. *Commun. ACM*, 11(6):419–422, 1968. doi:10.1145/363347.363387.

23 Richard Ryan Williams. The polynomial method in circuit complexity applied to algorithm design (invited talk). In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 47–60. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.47.

24 Fang Yu, Zhifeng Chen, Yanlei Diao, T. V. Lakshman, and Randy H. Katz. Fast and memory-efficient regular expression matching for deep packet inspection. In Laxmi N. Bhuyan, Michel Dubois, and Will Eatherton, editors, *Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems, ANCS 2006, San Jose, California, USA, December 3-5, 2006*, pages 93–102. ACM, 2006. doi:10.1145/1185347.1185360.

A FSH implies FPH

We use the following relation between Formula-SAT and Formula-Pair to show that FSH implies FPH:

▶ **Lemma A.1** (Weak version of Lemma B.2 in the full version of [1]). *An instance of Formula-SAT on a De Morgan formula of size s over n variables can be reduced to an instance of Formula-Pair with a monotone De Morgan formula of size $k = \mathcal{O}(s)$ and two sets of size $\mathcal{O}(2^{n/2})$ in linear time.*

Proof Idea. Let F be the formula for Formula-SAT on n variables and size s. We define F' to be the same formula as F but each leaf is labeled with a different variable and we remove the negations from the leaves.

For all half-assignments x to the first half of variables of F we construct a new half-assignment a_x for F' as follows: Let l be a leaf in F with a variable from the first half of inputs and let l' be the corresponding variable/leaf in F'. We set $a_x[l'] = \mathsf{true}$ if and only if l evaluates to true under x. We construct the set B analogous for the second half of inputs of F. Since F has n inputs this results in $2^{n/2}$ assignments for A and B. ◀

▶ **Lemma A.2.** *FSH implies FPH.*

Proof. Assume FSH holds and FPH is false for some fixed $k \geq 1$. Let F be a formula for Formula-SAT on N inputs and size $s = N^{3+1/(4k)} \in N^{3+\Omega(1)}$. By Lemma A.1 we transform F into a monotone De Morgan formula F' of size $s' = \mathcal{O}(s)$ and two sets with $n, m \in \mathcal{O}(2^{N/2})$ assignments. We run the algorithm for Formula-Pair on this instance to contradict FSH:

$$\mathcal{O}\left(\frac{n \cdot m \cdot s'^k}{\log^{3k+2} n} \log^{1+o(1)} 2^N\right) \subseteq \mathcal{O}\left(\frac{2^{N/2} 2^{N/2} s^k N^{1.25}}{\log^{3k+2} 2^{N/2}}\right) = \mathcal{O}\left(2^N \frac{N^{3k+0.25+1.25}}{N^{3k+2}(1/2)^{3k+2}}\right)$$
$$= \mathcal{O}\left(2^N \frac{N^{3k+1.5}}{N^{3k+2}}\right) = \mathcal{O}\left(\frac{2^N}{N^{0.5}}\right).$$

See the following paragraph for the additional factor of $N^{1+o(1)}$. ◀

As Abboud and Bringmann [1] we use the *Word-RAM* model as our computational model. The word size of the machine will be fixed to $\Theta(\log N)$ many bits for input size N. Likewise we assume several operations that can be performed in time $\mathcal{O}(1)$ (e.g. AND, OR, NOT, addition, multiplication, ...).

While this is sufficient for our reductions, we also need that the operations are robust to a change of the word size to state FPH. As in [1] we require that we can simulate the operations on words of size $\Theta(\log N)$ on a machine with word size $\Theta(\log \log N)$ in time $(\log N)^{1+o(1)}$.

In the above proof the input size increased from N to $n = 2^N$. Hence, we have to simulate the algorithm for Formula-Pair with word size $\log n = N$ on a machine with word size $\log N$ to get an algorithm for Formula-SAT. Thus, the running time slows down by a factor of $(\log n)^{1+o(1)} = N^{1+o(1)}$.

B Correctness of the Graph Construction for +|∘+-Membership

We show the correctness of the graph construction given in the proof of Theorem 3.1 Item 2.

▷ Claim B.1. If $t \in \mathcal{L}(p)$, then there is a path from v_0^0 to v_n^0.

Proof. Assume $p = (p_1 \mid \ldots \mid p_k)^+$. Since $t \in \mathcal{L}(p)$, we can decompose t into $t = \tau_1 \cdots \tau_\ell$ such that for all $l \in [\ell]$ $\tau_l \in \mathcal{L}(p_{k_l})$ for some $k_l \in [k]$. Define $\lambda_l = |\tau_1 \cdots \tau_l|$ as the length of the first l parts of t for all $l \in [\ell]$. We claim that if $\tau_1 \cdots \tau_l \in \mathcal{L}(p)$, then there is a path from v_0^0 to $v_{\lambda_l}^0$.

For $l = 0$, the claim is vacuously true as $\varepsilon \notin \mathcal{L}(p)$. Now assume the claim holds for arbitrary but fixed l. We define $i = \lambda_l + 1$ and $j = \lambda_{l+1}$ to simplify notation and get $\tau_{l+1} = t_i \cdots t_j$. From $\tau_{l+1} \in \mathcal{L}(p_{k_{l+1}})$ and Lemma 3.6 we know $(f, i', j', e) \in M'$ for some $i \leq i' \leq j' \leq j$. Further, f, e are set to 1 if and only if the first and last run of $p_{k_{l+1}}$ contains a Kleene Plus, respectively. Hence, $v_{j'}^{2e}$ is reachable from $v_{i'-1}^f$. Now it suffices to show that (1) $v_{i'-1}^f$ is reachable from v_{i-1}^0 and (2) v_j^0 is reachable from $v_{j'}^{2e}$. Then the claim follows inductively as v_{i-1}^0 is reachable from v_0^0.

We first show (1). If $f = 0$, we must have $i = i'$ and the claim holds. Thus assume $f = 1$. We know $\tau_{l+1} = t_i \cdots t_j \in \mathcal{L}(p_{k'})$ and $t_{i'} \cdots t_{j'} \in \mathcal{L}(p_{k'})$ for some $k' \in [k]$. As the first run of $p_{k'}$ contains a Kleene Plus, the symbols, $t_i, t_{i+1}, \ldots, t_{i'}$ are all equal. That is, they form a run from i to i'. By the construction of the graph, there are edges $(v^1_{i-1}, v^1_i), \ldots, (v^1_{i'-2}, v^1_{i'-1})$. But there is also the additional edge (v^0_{i-1}, v^1_{i-1}) proving (1).

By a symmetric argument one can show claim (2). ◁

▷ Claim B.2. If there is a path from v^0_0 to v^0_n, then $t \in \mathcal{L}(p)$.

Proof. First observe that it is not possible to reach v^0_n from v^0_0 without using edges introduced by tuples in M'. Now fix some path P from v^0_0 to v^0_n and let $P_1, \ldots, P_\ell$ be the edges on the path that are introduced by tuples in M'. Let $P_l = (v^{f_l}_{i'_l-1}, v^{2e_l}_{j'_l})$, i.e. $(f_l, i'_l, j'_l, e_l) \in M'$.

Assume $j'_0 = 0$ and $i'_{\ell+1} = n + 1$ in the following to simplify notation. For each tuple there are two indices i_l and j_l such that $j'_{l-1} \leq i_l - 1 \leq i'_l - 1$ and $j'_l \leq j_l \leq i'_{l+1} - 1$ and the path P goes through $v^0_{i_l-1}$ and $v^0_{j_l}$. These nodes exist, as every path from $v^{2e_{l-1}}_{j'_{l-1}}$ to $v^{f_l}_{i'_l-1}$ has to go through some node v^0_r. We have $j_l + 1 = i_{l+1}$ for all $l \in [0, \ell]$ with $j_0 = 0$ and $i_{\ell+1} = n + 1$ and hence, $t = t_{i_1} \cdots t_{j_1} t_{i_2} \cdots t_{j_2} \cdots t_{i_\ell} \cdots t_{j_\ell}$. Thus, it suffices to show that for every $l \in [\ell]$ there is a $k' \in [k]$ such that $t_{i_l} \cdots t_{j_l} \in \mathcal{L}(p_{k'})$.

We fix l in the following and omit it as index to simplify notation. By the construction of the graph we have $(f, i', j', e) \in M'$ and hence by Lemma 3.6 $t_{i'} \cdots t_{j'} \in \mathcal{L}(p_{k'})$ for some $k' \in [k]$. We extend this result and claim $t_i \cdots t_{j'} \in \mathcal{L}(p_{k'})$. Recall, that there is a path from v^0_{i-1} to $v^f_{i'-1}$ in P. If $f = 0$, then $i' = i$ and the claim follows. Otherwise, we know that the first run of $p_{k'}$ contains a Kleene Plus for some symbol α. As no edge resulting from a tuple in M' can be chosen, the edge (v^0_{i-1}, v^1_{i-1}) is contained in the path P. By the construction of the graph, the sequence $t_i \cdots t_{i'}$ is contained in some run β^c. But $\alpha = \beta$ and we get $t_i \cdots t_{i'-1} t_{i'} \cdots t_{j'} \in \mathcal{L}(p_{k'})$.

We can apply the symmetric argument to show that $t_i \cdots t_{j'} t_{j'+1} \cdots t_j \in \mathcal{L}(p_{k'})$ proving the claim. ◁