Abstract

We survey lower-bound results in complexity theory that have been obtained via newly found interconnections between propositional proof complexity, boolean circuit complexity, and query/communication complexity. We advocate for the theory of total search problems (TFNP) as a unifying language for these connections and discuss how this perspective suggests a whole programme for further research.

1 Introduction

In recent years there has been a large number of new results in both propositional proof complexity and boolean circuit complexity. These results include: optimal $2^{\Omega(n)}$ lower bounds on the size of monotone boolean formulas computing an explicit boolean function in NP [PR17], the refinement of the mon-AC^i hierarchy from the mon-NC^i hierarchy and new tradeoffs for cutting planes proofs [dRNV16], a new family of techniques for proving lower bounds on cutting planes proofs and monotone circuit size [GGKS20], and exponential lower bounds on the size of cutting planes proofs for random CNF formulas [FPPR17, HP17]. All of these new results have been enabled, either directly or indirectly, by two tools:

1. The discovery of new connections between proof complexity and query complexity, as well as between boolean circuit complexity and communication complexity.

2. The development of query-to-communication lifting theorems which show that, for certain tasks, query complexity lower bounds can be “lifted” to communication lower bounds.

Taken together, these tools reveal new connections between proof complexity and monotone circuit complexity. In particular, many standard propositional proof systems (e.g., resolution, cutting planes, Nullstellensatz) have been shown to have corresponding “partner” models of monotone computation (resp. boolean circuits, real circuits, span programs) such that the lower bounds on proof complexity in a proof system imply lower bounds on complexity of computation in the “partner” model.

1Department of Computer Science, Lund University, Lund, Sweden. susanna.rezende@cs.lth.se. Supported by ELLIIT and Knut and Alice Wallenberg grant KAW 2021.0307.

2School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland. mika.goos@epfl.ch.

3School of Computer Science, McGill University, Montreal, QC, Canada. robere@cs.mcgill.ca. Supported by NSERC.
A central theme in these results is the important role played by certain total search problems in both proof complexity and boolean circuit complexity. On the one hand, for many proof systems the complexity of refutations of an unsatisfiable CNF formula \( F \) is closely related to the complexity of the so-called falsified clause search problem \( \text{Search}(F) \), where we are given an assignment \( x \) to the variables of \( F \) and need to find a clause in \( F \) that is false under \( x \). Meanwhile, for boolean models of computation, the right total search problem is the famous Karchmer–Wigderson game [KW90], which was introduced to understand circuit complexity using communication complexity. It has turned out to be fruitful to place these total search problems on center stage when trying to understand proof systems and circuit models. In fact, recent research has shown that the classical (Turing machine) theory of total search problems (TFNP) can serve as a general “organizing principle” for proof systems and circuit classes in this way [GKRS18].

Outline. This survey explores the aforementioned themes starting from base principles. In Section 2 we will give a gentle introduction to the classical connections between tree-like resolution and decision trees, as well as boolean formulas and deterministic communication protocols. In Section 3, we will extend the results from Section 2 to dag-like models. Then, in Section 4, we describe applications of these ideas to the semi-algebraic cutting planes proof system. Finally, in Section 5, we will discuss how these classical connections can be viewed as part of a more general framework, in analogy to the theory of TFNP, and how this perspective suggests a programme for further research.

2 Tree-like Resolution Proofs and Monotone Boolean Formulas

A boolean variable is a variable \( x \) taking values in \( \{0,1\} \), where we interpret 0 as “False” and 1 as “True”. A boolean literal is a boolean variable \( x_i \) or its negation \( \overline{x}_i \). A boolean formula (also called a propositional formula or just a formula) is an expression composed out of boolean variables and \( \land, \lor \) and \( \neg \). A clause is a disjunction of boolean literals: for example, \( x_1 \lor x_2 \lor \overline{x}_3 \) is a clause, as is \( x_1 \). The empty clause, denoted \( \bot \), has no literals in it, and is always false. A boolean formula \( F \) is in conjunctive normal form (CNF) if it is a conjunction of clauses; for example,

\[
x_1 \land (\overline{x}_1 \lor x_2) \land (\overline{x}_2 \lor x_3) \land \overline{x}_3.
\]

2.1 Resolution and decision trees

Our starting point is a well-known connection between tree-like resolution and decision trees.

Definition 2.1. Let \( A, B \) be clauses of boolean literals, and let \( \ell \) be any boolean literal not occurring in \( A \). The resolution rule is the deduction rule

\[
\frac{A \lor \ell \quad B \lor \overline{\ell}}{A \lor B}.
\]

The weakening rule is the deduction rule

\[
\frac{A}{A \lor \ell}.
\]

Note that the resolution rule is sound: if \( x \) is an assignment that satisfies both \( A \lor \ell \) and \( B \lor \overline{\ell} \) then \( x \) also satisfies \( A \lor B \).
Let $F = C_1 \land \cdots \land C_m$ be a boolean formula in CNF. A resolution proof of a clause $C$ from $F$ is a sequence of clauses

$$D_1, D_2, \ldots, D_s,$$

where $D_s = C$, and for each $i$ either $D_i$ is a clause from $F$ or is deduced from earlier clauses by the resolution rule or the weakening rule. The width of the proof is the maximum number of literals in any clause appearing in the proof. The proof is called a refutation of $F$ if $C = \bot$, the empty clause. It is natural to associate a proof dag (directed acyclic graph) with a resolution refutation, where the nodes of the dag are the clauses in the proof; the sources (no incoming edges) of the dag are clauses in $F$ and each internal node has incoming edges from the at most two clauses used to deduce it. A resolution refutation is tree-like if every clause is used at most once as an input to the resolution rule; it is easy to see that the proof dag of a tree-like proof is a tree.

The main recurring theme that we will see in this survey is the use of total search problems to understand problems in both proof and circuit complexity. We now define a central family of total search problem that captures the complexity of proofs in many proof systems. Given a CNF formula $F = C_1 \land \cdots \land C_m$ on $n$ variables, define the relation $\text{Search}(F) \subseteq \{0, 1\}^n \times [m]$ by

$$(x, i) \in \text{Search}(F) \iff C_i(x) = 0.$$  

We think of this relation as defining a search problem: given $x \in \{0, 1\}^n$ as input, find an $i \in [m]$ such that $(x, i) \in \text{Search}(F)$. It is easy to see that this search problem is total (that is, $\forall x \exists i : (x, i) \in \text{Search}(F)$) if and only if the CNF formula $F$ is unsatisfiable. Moreover, it turns out that the complexity of the smallest decision tree solving $\text{Search}(F)$ captures exactly the complexity of tree-like resolution. Indeed, it turns out that decision trees for $\text{Search}(F)$ and tree-like resolution refutations of $F$ are essentially the same object!

**Theorem 2.2.** Let $F$ be an unsatisfiable CNF formula. There is a size-$s$, depth-$d$ tree-like resolution refutation of $F$ if and only if there is a size-$s$, depth-$d$ decision tree solving $\text{Search}(F)$.

### 2.2 Boolean formulas and Karchmer–Wigderson games

Next we switch gears and recall a second famous result in complexity theory: the relationship between boolean circuit complexity and communication complexity due to Karchmer and Wigderson [KW90] (independently discovered by Yannakakis [KPPY84]). We first define boolean circuits.

**Definition 2.3.** A boolean circuit on $n$ input variables is given by a sequence of functions $g_1, g_2, \ldots, g_s$ such that, for each $i \in [s]$, either

- $g_i = z_i$ or $g_i = \overline{z_i}$ for some input variable $z_i$, or
- $g_i = g_j \circ g_k$ where $\circ \in \{\land, \lor\}$ and $j, k < i$.

The boolean circuit has a natural underlying dag; we say the circuit is a formula if the underlying dag is a tree. The size of the circuit is $s$, the number of gates, and we say that the circuit computes the boolean function $f$ if $g_s = f$. Finally, we say that the boolean circuit is monotone if the negated input variables $\overline{z_i}$ are not used.

In this language, one should think of boolean circuits as being analogous to resolution proofs, and boolean formulas as analogous to tree-like resolution proofs. Both are, in some sense, “deductive” or “bottom-up”: we start from “atoms” (either input clauses of the formula $F$, or variables of the underlying boolean function $f$), and deduce new intermediate objects from old ones using some predefined rules.
Extending this analogy, then, what corresponds to decision trees in the world of boolean circuit complexity? Karchmer and Wigderson [KW90] supplied an answer: communication protocols solving Karchmer–Wigderson games.

**Definition 2.4.** Let \( f : \{0,1\}^n \to \{0,1,*\} \) be a (possibly partially defined) boolean function. The Karchmer–Wigderson game of \( f \) is the relation \( \text{KW}(f) \subseteq f^{-1}(1) \times f^{-1}(0) \times [n] \) defined by \((x,y,i) \in \text{KW}(f)\) if and only if \( x_i \neq y_i \). If \( f \) is monotone (meaning that for all \( x,y \in \{0,1\}^n \) if \( x \leq y \) then \( f(x) \leq f(y) \)), then define the monotone Karchmer–Wigderson game of \( f \), denoted \( \text{mKW}(f) \subseteq f^{-1}(1) \times f^{-1}(0) \times [n] \), by \((x,y,i) \in \text{mKW}(f)\) if and only if \( x_i = 1, y_i = 0 \).

As in the case of \( \text{Search}(F) \) for unsatisfiable CNF formulas \( F \), Karchmer–Wigderson games are also total search problems, when we view the pair \((x,y)\) as the input and \( i \in [n] \) as the output. Indeed, one should think of \( \text{KW}(f) \) as playing the same role as \( \text{Search}(F) \) does in proof complexity: it is a total search problem that in some sense captures the complexity of computing \( f \).

Karchmer and Wigderson proved that this is true for boolean formulas. Indeed, in the same way that decision trees and tree-like resolution refutations are essentially identical objects, one can show that boolean formulas and communication protocols for \( \text{KW}(f) \) are also essentially identical.

**Theorem 2.5 ([KW90]).** Let \( f : \{0,1\}^n \to \{0,1,*\} \) be a partial boolean function. There is a depth-\( d \) boolean formula computing \( f \) if and only if there is a depth-\( d \) deterministic communication protocol solving \( \text{KW}(f) \). Moreover, if \( f \) is monotone, then there is a depth-\( d \) monotone boolean formula computing \( f \) if and only if there is a depth-\( d \) communication protocol solving \( \text{mKW}(f) \).

The proof of the previous theorem is essentially identical to the proof of Theorem 2.2, with objects in the world of resolution substituted for objects in the world of boolean circuits appropriately. In Figure 1, we record a dictionary of the related objects in this correspondence. This analogy is the lynchpin underlying the connections between proof and circuit complexity.

| Resolution proofs | Decision trees | Boolean circuits | Communication protocols |
|-------------------|----------------|-----------------|-------------------------|
| unsatisfiable CNF \( F \) | \( \text{Search}(F) \) | boolean functions \( f \) | \( \text{mKW}(f) \) |
| clauses \( C \) | subtrees | intermediate functions \( g_i \) | rectangles \( R \) |
| deduction rules | variable queries | circuit gates | communicated bits |

**Figure 1:** The corresponding objects in resolution proofs and boolean circuits.

### 2.3 Feasible interpolation and lifting theorems

Next we show how to use the above analogy to systematically relate the two worlds (proof and circuit complexity) together. In order to do this, we will show how to directly relate the falsified clause search problem \( \text{Search}(F) \) to the monotone Karchmer–Wigderson game \( \text{mKW}(F) \).

Let us first answer a simple question. Suppose \( F = C_1 \land \cdots \land C_m \) is an unsatisfiable CNF formula on \( n \) variables \( Z = \{z_1, z_2, \ldots, z_n\} \). How can we turn the search problem \( \text{Search}(F) \subseteq \{0,1\}^n \times [m] \) into a two-party communication task? The obvious way is to take a partition of the variables \( Z = X \cup Y \), and then give the assignment to the \( X \)-variables to Alice and the assignment to the \( Y \)-variables to Bob. Let us call this two-party search problem \( \text{Search}^{(X,Y)}(F) \), and observe that an efficient decision tree for \( \text{Search}(F) \) immediately implies an efficient communication protocol for \( \text{Search}^{(X,Y)}(F) \) simply by simulating the decision tree.

This is progress, but while \( \text{Search}^{(X,Y)}(F) \) is a total search problem it is not a Karchmer–Wigderson game. It turns out, however, that we can interpret it as the monotone Karchmer–Wigderson game for some partial boolean function. The idea for this goes back to Razborov [Raz90],
although it has been reintroduced in the recent works on lifting theorems [GP18, GPW18, FPPR17], as well as independently by Hrubeš and Pudlák [HP17] as a generalisation of the classic theory of feasible interpolation introduced by Krajíček [Kra97]. We will describe the definition introduced by Hrubeš and Pudlák. If $C$ is a clause and $X$ is a subset of its variables we let $C^X$ denote the subclause of $C$ containing only literals over $X$.

**Definition 2.6.** Let $F = C_1 \land \cdots \land C_m$ be an unsatisfiable CNF formula and let $(X, Y)$ be any partition of its variables. The unsatisfiability certificate associated with $F$ and $(X, Y)$ is the partial monotone function $\text{cert}_F^{(X,Y)} : \{0,1\}^m \to \{0,1,*\}$ defined as

$$\text{cert}_F^{(X,Y)}(z) := \begin{cases} 1 & \text{if } \bigwedge_{i:z_i=0} C_i^X \text{ is satisfiable}, \\ 0 & \text{if } \bigwedge_{i:z_i=1} C_i^Y \text{ is satisfiable}, \\ * & \text{otherwise}. \end{cases}$$

We may suppress the partition $(X,Y)$ or the underlying CNF formula $F$ in $\text{cert}_F^{(X,Y)}$ when it is clear from context.

Indeed, one can show that $mKW(\text{cert}_F^{(X,Y)})$ is communication reducible to $\text{Search}^{(X,Y)}(F)$ and vice-versa. Using the unsatisfiability certificate, we can therefore prove the following “feasible interpolation” theorem relating the boolean formula depth of the unsatisfiability certificate to the resolution depth of the underlying CNF formula.

**Theorem 2.7.** Let $F$ be an unsatisfiable CNF formula, let $(X, Y)$ be any partition of the variables of $F$, and let $\text{cert}_F$ denote the unsatisfiability certificate with respect to this variable partition. If there is a resolution proof of depth $d$ refuting $F$ then there is a monotone boolean formula of depth $d$ computing $\text{cert}_F$.

First note that the previous theorem implies that upper bounds on proofs imply upper bounds on circuits; thus, conversely, lower bounds on monotone circuit complexity implies lower bounds on proof complexity. Second, although we have stated the previous theorem in terms of depth, the argument can be easily modified to capture the size of the two systems as well.

Finally, it is natural to ask if a converse to the previous result can hold. That is, are there unsatisfiable CNF formulas $F$ such that the decision tree complexity of solving $\text{Search}(F)$ is a lower bound on the communication complexity of solving $mKW(\text{cert}_F)$? The answer to this is an emphatic yes, thanks to well-known query-to-communication lifting theorems, which have been proved in a long line of work starting with Raz and McKenzie [RM99]. These powerful and general results allow us to generate entire families of formulas $F$ for which the decision tree complexity of $\text{Search}(F)$ and the communication complexity of $mKW(\text{cert}_F)$ are tightly related.

The basic idea of a lifting theorem is simple: we start with an unsatisfiable CNF formula $F$, and replace each variable $z_i$ in $F$ with a gadget function $g(x_i, y_i)$ on new “fresh” inputs $x_i, y_i$. In this way, we obtain a new formula $F \circ g^n = F(g(x,y))$, which we can transform back into a CNF formula using syntactic manipulations. Perhaps the most often used example of a gadget function $g$ is the index gadget $\text{IND}_{[]} : \ell \times \{0,1\}^\ell \to \{0,1\}$, which takes in a string $y \in \{0,1\}^\ell$ and a pointer $x \in [\ell]$ to a bit in the string and outputs $\text{IND}_y(x,y) = y_x$. Raz and McKenzie [RM99] (further developed by [GPW18, CKLM19]) proved the following lifting theorem that relates the query complexity of $\text{Search}(F)$ to the communication complexity of $mKW(\text{cert}_{F\circ \text{IND}_[]})$.

**Theorem 2.8.** Let $F$ be an unsatisfiable CNF formula on $n$ variables, and let $m = n^C$ for a large enough constant $C \geq 1$. The size of any monotone boolean formula computing $\text{cert}_{F\circ \text{IND}_[]} \circ \text{IND}_[]$ is at least $n^{\Omega(D(F))}$, where $D(F)$ is the minimum depth of any (tree-like) resolution refutation of $F$. 
This result allows us to deduce lower bounds on monotone formula complexity (a difficult task) automatically from resolution depth lower bounds (a relatively simple task). Moreover, by choosing the underlying unsatisfiable CNF formulas $F$ to be specially structured, we can obtain lower bounds for many standard functions considered in circuit complexity, such as the $st$-connectivity, clique and generation functions.

The $st$-connectivity function $\text{stConn}_n$ takes as input an $n$-vertex directed graph with two distinguished vertices $s$ and $t$, and outputs 1 if there is a directed path from $s$ to $t$ and 0 otherwise. Karchmer and Wigderson [KW90] proved, for the case of undirected graphs, that any monotone boolean formula computing $\text{stConn}_n$ must have size $n^{\Omega(\log n)}$, and alternative proofs were given by [GS95, Pot17, Rob18] for the case of directed graphs. A similar lower bound follows from Theorem 2.8 by considering $\text{Clique}$ by checking all $\binom{n}{k}$ potential $k$-cliques. Razborov [Raz85a] proved the first superpolynomial size lower bound for monotone circuits precisely for the clique function. This was later improved in [AB87, Weg87] to an $\Omega(\sqrt{n})$ lower bound for $k \leq n^{2/3-\varepsilon}$ and more recently to an $\Omega(k)$ lower bound for $k \leq n^{1/3-\varepsilon}$ [CKR20]. For monotone formulas, a weaker lower bound but with a simpler proof was presented in [GH92] and this was improved to an $\exp(\Omega(k))$ lower bound for $k \leq \frac{1}{3} n + 1$ in [RW92]. Raz and McKenzie [RM99] obtained an $n^{\Omega(k)}$ size lower bound for monotone formulas for $k \leq n^{\varepsilon}$ by applying Theorem 2.8 with $F$ being a certain formulation of the pigeonhole principle.

The generation function $\text{Gen}_{n,k}$ was introduced by Raz and McKenzie [RM99] in order to separate the monotone NC hierarchy. It is defined as follows. Given a set $T \subseteq [n]^3$, we say that $T$ generates a point $w \in [n]$ if $w = 1$, or if there is a triplet $(u, v, w) \in T$ such that $T$ generates $u$ and $v$. The function $\text{Gen}$ takes as an input a set $T \subseteq [n]^3$ and outputs 1 if $T$ generates $n$ and 0 otherwise. By applying Theorem 2.8 to the so-called pebbling formulas, Raz and McKenzie [RM99] proved an $\exp(\Omega(n^\varepsilon))$ monotone formula size lower bound for $\text{Gen}_n$. Since $\text{Gen}_n$ is computable by a polynomial-size monotone circuit, this implies that $\text{mon-NC}^i \neq \text{mon-P}$. By a similar argument, they also showed that $\text{mon-NC}^i \neq \text{mon-NC}^{i+1}$. A later extension of this lifting theorem in [dRNV16], building on [GPW18], completed the picture of the relation between the $\text{mon-AC}$ and $\text{mon-NC}$ hierarchies, which looks like $\text{mon-NC}^i \nsubseteq \text{mon-AC}^i \nsubseteq \text{mon-NC}^{i+1}$.

We summarise the main formula size lower bounds that follow from Theorem 2.8 as follows.

**Corollary 2.9.** Let $\varepsilon$ be a small enough constant. The size of any monotone boolean formula computing:

1. $\text{stConn}_n$ is at least $n^{\Omega(\log n)}$;
2. $\text{Clique}_{n,k}$ for $k \leq n^{\varepsilon}$ is at least $n^{\Omega(k)}$; and
3. $\text{Gen}_n$ is at least $\exp(n^{\varepsilon})$.

Recent extensions of Theorem 2.8 [CKLM19, LMM*20] in particular improve the size of the gadget and as a consequence the parameter $\varepsilon$ above need not be very small. We also note that the best known monotone formula lower bounds for these functions follow from another lifting theorem [PR17, dRMN*20].

Let us conclude with three main takeaway lessons from this section.

- First, the structure of tree-like resolution proofs and boolean formulas are extremely similar, and this shared structure can be exploited in a nice way using the appropriate total search problems.
• Second, by exploiting this analogy, we can easily deduce generic feasible interpolation results, which allow us to translate circuit lower bounds into proof lower bounds.

• Third, by using composition, one can prove a lifting theorem that is essentially a converse to feasible interpolation. In concrete terms this means that, for certain tasks, lower bounds on proofs imply lower bounds on boolean circuits.

In the remainder of this survey we will see several variations on this theme that have occurred in the recent literature. Next, we will examine how to move from tree-like proofs and circuits to more general dag-like models.

### 3 Dag-Like Resolution Proofs and Monotone Boolean Circuits

In order to capture dag-like models, we will need to introduce dag-like versions of decision trees and communication protocols that capture resolution and boolean circuits, respectively. Both of our dag-like models will share the following definition.\(^4\)

**Definition 3.1.** Let \(S \subseteq I \times O\) be a total search problem with input \(I\) and output \(O\), and let \(F\) be a family of functions from \(I \rightarrow \{0,1\}\). An \(F\)-dag solving \(S\) is given by a directed acyclic graph of fan-out at most 2 such that each node \(v\) in the graph is associated with a function \(f_v \in F\) satisfying the following:

- **Root.** There is a unique root node \(r\) with fan-in 0 associated with the (trivial) function 1.
- **Internal Consistency.** For each internal node \(v\) with children \(u, w\) we have that
  \[f_v^{-1}(1) \subseteq f_u^{-1}(1) \cup f_w^{-1}(1).\]
- **Solutions.** Each leaf node \(v\) is labelled with an \(o \in O\) such that for all \(x \in f_v^{-1}(1)\), \((x, o) \in S\).

The *size* of an \(F\)-dag is the number of nodes in the graph.

First we instantiate this model and consider a dag-like version of decision trees, which we call *conjunction-dags* following [GGKS20]. This model and close variations on it have been studied in the literature in a number of separate works prior to [GGKS20]; notably, by Pudlák [Pud00], and then by Atserias and Dalmau [AD08], under the name *Prover-Adversary games*.

**Definition 3.2.** Let \(F\) be an unsatisfiable CNF formula. A *conjunction-dag* solving \(\text{Search}(F)\) is an \(F\)-dag solving \(\text{Search}(F)\) where \(F\) is the family of all conjunctions over boolean literals in \(F\).

A few remarks on this definition are in order. First, it is not hard to interpret the model as a dag-like version of a decision tree (indeed, a decision tree is a tree-like conjunction-dag!).

Given an input \(x \in \{0,1\}^n\), the “search algorithm” starts by examining the root node, at which \(x\) trivially satisfies the associated conjunction. By the internal consistency property, if \(x\) satisfies the conjunction at a node \(v\), then it also satisfies the conjunction at one of the children of \(v\), and thus the algorithm can proceed from \(v\) to the satisfied child. Finally, once the algorithm reaches a leaf node, the final property guarantees that the leaf will be labelled with a solution to \(\text{Search}(F)\).

Second, it is also easy to see how this model corresponds to resolution. If we are given a resolution proof \(\Pi\), then we can simply replace each clause in \(\Pi\) with the negation of that clause (i.e., a conjunction); it is easy to verify that the resulting dag satisfies the three properties of a conjunction-dag. The converse direction is not much harder; we refer to [GGKS20] for a proof.

\(^4\)For those who are familiar, the next definition is closely related to the notion of a “semantic derivation”, introduced by Krajíček [Kra97].
Theorem 3.3. Let $F$ be an unsatisfiable CNF formula. There is a size-$s$ resolution refutation of $F$ if and only if there is a size-$s$ conjunction-dag solving $\text{Search}(F)$.

Now, we can similarly define a dag-like model of two-party communication that corresponds to boolean circuits. The model of dag-like communication was first introduced by Razborov [Raz95], where it was shown that dag-like communication captures boolean circuit size in the same way that deterministic communication captures boolean formula size. Razborov’s original definition used the language of TFNP, particularly as a “communication version” of the complexity class PLS (see Section 5 for more details on this). The model was subsequently studied by Krajíček [Kra97] and simplified by Sokolov [Sok17], although we will use the following version from [GGKS20]. As we will be primarily interested in monotone circuit complexity, we define it purely for the mKW game.

Definition 3.4. Let $f : \{0, 1\}^n \rightarrow \{0, 1, *\}$ be a (possibly partial) monotone boolean function. A rectangle-dag solving $\text{mKW}(f) \subseteq f^{-1}(1) \times f^{-1}(0) \times [n]$ is an $F$-dag solving $\text{mKW}(f)$ where $F$ is the family of all indicator functions of combinatorial rectangles over $f^{-1}(1) \times f^{-1}(0)$.

It is again easy to observe that standard deterministic communication protocols for $\text{mKW}(f)$ are rectangle-dags where the underlying graph is a tree, and furthermore that one can associate these objects with a top-down “search algorithm” as we described for conjunction-dags above.

Second, as we stated above, it is known that the size of rectangle-dags captures boolean circuit size. While we will state it for monotone circuits, a similar result also holds for non-monotone circuits.

Theorem 3.5 ([Raz95, Sok17]). Let $f$ be a monotone boolean function. There is a size-$s$ monotone boolean circuit computing $f$ if and only if there is a size-$s$ rectangle-dag solving $\text{mKW}(f)$.

Finally, just as in the previous section, one can use these models to prove a simple “feasible interpolation”-style result for resolution proofs in terms of monotone boolean circuits. This feasible interpolation result was originally observed for certain structured formulas by Krajíček [Kra97] and Pudlák [Pud97] using direct arguments; one of the benefits of our abstract setup is that we get a similar feasible interpolation result in terms of the unsatisfiability certificate immediately and transparently from the definitions.

Theorem 3.6. Let $F = C_1 \land \ldots \land C_m$ be an unsatisfiable CNF formula, let $(X, Y)$ be any partition of the variables of $F$, and let $\text{cert}_F$ denote the corresponding unsatisfiability certificate. If there is a resolution refutation of $F$ of size $s$ then there is a monotone boolean circuit computing $\text{cert}_F$ of size $O(s)$.

As in the case of tree-like proofs and boolean formulas, it is natural to consider whether or not a converse to the previous result is possible. If so, this would allow us to translate lower bounds for resolution proofs (which is a very well-developed theory, e.g., [Hak85, BW01]) into lower bounds for monotone boolean circuits, for which we essentially have only one technique, the method of approximations with its variations [Raz85a, Juk97, HR00, BU99]. Indeed, such a result was proven by Göös, Garg, Kamath and Sokolov [GGKS20], who crucially used the search viewpoint described above in their proof.

Theorem 3.7 ([GGKS20]). Let $F$ be an unsatisfiable CNF formula on $n$ variables, and let $m = n^C$ for a sufficiently large constant $C$. Then the size of any monotone boolean circuit computing $\text{cert}_{F \land \text{IND}_m}$ is at least $n^{\Omega(w(F))}$, where $w(F)$ is the minimum width of any resolution refutation of $F$. 

Note that the measure being “lifted” here is resolution refutation width, in contrast to Theorem 2.8 where it was depth. This difference is indeed necessary, since both $\text{stConn}$ and $\text{Gen}$ have polynomial size monotone circuit. Given that there is a width lower bound for refuting the pigeonhole principle, by applying Theorem 3.7 in the same way as done in [RM99], we get a monotone circuit lower bound for $\text{Clique}$. This lower bound, as is the case for most other clique lower bounds, actually holds for (any extension of) the partial function $\text{clique-colouring}$. The function $\text{Clique-Col}_{n,k}$ receives as input an $n$-vertex graph and outputs 1 if it has a $k$-clique, 0 if it is $(k-1)$-colourable and can output anything if neither is the case. Tardos [Tar88] proved that there is an extension of $\text{Clique-Col}$ that is monotone and is in $\mathbb{P}$, and therefore this gives an exponential separation between monotone and non-monotone circuits.

Another interesting application of Theorem 3.7, observed in a follow-up work [GKRS18], is that it improves this separation to an exponential separation between $\text{NC}^2$ and mon-$\mathbb{P}$. This is done by considering the monotone function $3\text{Xor-Sat}_n$, which receives as input an indicator vector for a set of $3\text{Xor}$ constraints over $n$ variables and outputs 1 if the set is unsatisfiable and 0 otherwise. By lifting so-called Tseitin formulas, which are known to require large resolution width to be refuted [Urq87], they obtain a monotone circuit lower bound for $3\text{Xor-Sat}$. Since $3\text{Xor-Sat}$ is in $\text{NC}^2$ [Mul87], this improves on the exponential monotone vs. non-monotone separation.

**Corollary 3.8.** Let $\varepsilon$ be a small enough constant. The size of any monotone boolean circuit computing:

1. $\text{Clique}_{n,k}$ for $k \leq n^\varepsilon$ is at least $n^{\Omega(k)}$; and
2. $3\text{Xor-Sat}_n$ is at least $\exp(n^\varepsilon)$.

As we will discuss in the next section, Theorem 3.7 can also be extended to obtain lower bounds for monotone real circuits, and therefore cutting planes proofs. With regards to monotone circuit lower bounds, however, it is natural to wonder how the lower bounds that follow from the previous theorem compare to the known results provable via the method of approximations. It seems that the techniques are ultimately incomparable, but recent work of Lovett et al. [LMM+20] gave an alternative proof of Theorem 3.7 using the sunflower lemma, which is the same central combinatorial tool used in the method of approximations.

## 4 Cutting Planes and Real Monotone Circuits

Cutting planes is a proof system that captures integer programming reasoning, based on the cutting planes method of Gomory [Gom63] and Chvátal [Chv73]. In the same way as the basic unit in resolution are clauses, in cutting planes they are integer linear inequalities. Let $\mathcal{I}$ be a system of unsatisfiable integer linear inequalities over variables $x_1, \ldots, x_n$. A cutting planes refutation of $\mathcal{I}$ is a sequence of linear inequalities $(L_1, \ldots, L_\ell)$ such that $L_\ell$ is $0 \geq 1$ and each $L_i$ is either in $\mathcal{I}$ or can be inferred from previous inequalities by one of the two following deduction rules:

\[
\begin{aligned}
\text{linear combination} & \quad \sum_i a_i x_i \geq A, & \sum_i b_i x_i \geq B, \\
\text{or division} & \quad \sum_i (\alpha a_i + \beta b_i) x_i \geq \alpha A + \beta B
\end{aligned}
\]

where $a_i, b_i, A, B, \alpha, \beta$ are integers and $\alpha, \beta$ are positive. The length of the refutation is $\ell$. We say that the refutation has bounded coefficients if all coefficients are bounded in absolute value by some polynomial in $n$. As before, we say the refutation is tree-like if the underlying proof dag is a tree.

For most of what we will discuss here, the exact rules of derivation are not particularly important, as long as they are sound over integers and refutationally complete. It is convenient, therefore,
to define a **semantic cutting planes refutation** of \( \mathcal{I} \) as a sequence of linear inequalities \( (L_1, \ldots, L_t) \) such that \( L_t \) is 0 \( \geq \) 1 and each \( L_i \) is either in \( \mathcal{I} \) or is **semantically implied** (over boolean variables) by two previous inequalities \( L_j \) and \( L_k \) where \( j, k < i \), i.e., all boolean points that satisfy both \( L_j \) and \( L_k \) also satisfy \( L_i \).

In order to use cutting planes to refute unsatisfiable CNF formulas, we must consider the translation of clauses into linear inequalities. A clause \( \bigvee_{i \in P} x_i \lor \bigwedge_{i \in N} \overline{x}_i \) is translated to \( \sum_{i \in P} x_i + \sum_{i \in N} (1 - x_i) \geq 1 \), where we interpret 0 as false and 1 as true, and a CNF formula \( F \) over variables \( x_1, \ldots, x_n \) is translated into a system of linear inequalities consisting of the translation of each clause in \( F \) and the inequalities \( x_i \geq 0 \) and \( x_i \leq 1 \) for \( i \in \{1, \ldots, n\} \) to guarantee variables take boolean values. We often identify \( F \) with this system of linear inequalities.

As was the case in resolution, we can view a semantic cutting planes refutation in a top-down manner, as an analogue of decision trees. The difference is that, in order to capture cutting-planes, we need dags where nodes are associated to halfspaces.

**Definition 4.1.** Let \( F \) be an unsatisfiable CNF formula. A **halfspace-dag** solving \( \text{Search}(F) \) is an \( \mathcal{F} \)-dag solving \( \text{Search}(F) \) where \( \mathcal{F} \) is the family of all halfspaces over the boolean variables in \( F \).

Following precisely the same argument that was sketched in Section 3 for resolution and conjunction-dags, it is possible to prove an equivalence between cutting planes refutations of \( F \) and halfspace-dags solving \( \text{Search}(F) \).

**Theorem 4.2.** Let \( F \) be an unsatisfiable CNF formula. There is a size-\( s \) semantic cutting planes refutation of \( F \) if and only if there is a size-\( s \) halfspace-dag solving \( \text{Search}(F) \).

### 4.1 Monotone real circuits and triangle-dags

In order to extend Krajíček’s interpolation technique [Kra97] to cutting planes, Pudlák [Pud97] defined **monotone real circuits** as a generalisation of monotone boolean circuits where each gate is allowed to compute any non-decreasing real function of its inputs, but the inputs and the output of the circuit are boolean.

Soon after, Krajíček [Kra98] defined **monotone protocols** with the goal of extending the interpolation technique to a wider class of proof systems. Here we define a subclass of these monotone protocols that we will refer to as **triangle-dags**\(^5\) as coined in [GGKS20].

We say \( T \subseteq \mathcal{X} \times \mathcal{Y} \) is a **combinatorial triangle** if for all \((x, y) \in T \) and \((x', y') \in T \) it holds that either \((x, y) \in T \) or \((x', y) \in T \) (or both). In other words, there is some labelling of the rows \( a_T : \mathcal{X} \to \mathbb{R} \) and of the columns \( b_T : \mathcal{Y} \to \mathbb{R} \) such that \( T = \{(x, y) : a_T(x) < b_T(y)\} \). In particular, note that every combinatorial rectangle is a triangle.

**Definition 4.3.** Let \( f : \{0, 1\}^n \to \{0, 1, \ast\} \) be a (possibly partial) monotone boolean function. A **triangle-dag** solving \( \text{mKW}(f) \subseteq f^{-1}(1) \times f^{-1}(0) \times [n] \) is a \( \mathcal{F} \)-dag solving \( \text{mKW}(f) \) where \( \mathcal{F} \) is the family of all indicator functions of combinatorial triangles over \( f^{-1}(1) \times f^{-1}(0) \).

We once again have a tight relation between a circuit model and an \( \mathcal{F} \)-dag.

**Theorem 4.4** ([HP18]). Let \( f \) be a monotone boolean function. There is a monotone real circuit of size \( s \) computing \( f \) if and only if there is a triangle-dag of size \( s \) solving \( \text{mKW}(f) \).

\(^5\)In [Kra98] this subclass is called **monotone protocols** whose **real communication complexity** is 1; in [Sok17] **real communication games** and in [HP18] **real protocols**.
Indeed, just as we thought of rectangles as the communication “analogue” of clauses, we think of triangles as the communication “analogue” of linear inequalities. The fact that small monotone real circuits for $f$ give small triangle-dags for the $mKW$-game on $f$ was already observed by Krajíček [Kra98]. The other direction is less obvious and was shown more recently by Hrubeš and Pudlák [HP18].

Some of the known lower bounds for monotone boolean circuits were generalised to monotone real circuits: clique-colouring [Pud97], broken mosquito screen [HC99], perfect matching [Fu97], st-connectivity [Joh98], generation [BEGJ00]. Jukna [Juk99] gave a general lower bound criterion for monotone boolean circuits and then showed that the same criterion also holds for monotone real circuits.

These results might suggest that monotone real circuits cannot be much more powerful than monotone boolean circuits. Rosenbloom [Ros97], however, proved the contrary. He showed that monotone real circuits can be exponentially stronger than even non-monotone boolean circuits. The proof is simple: by a counting argument, most slice functions cannot be computed by boolean circuits of subexponential size, whereas any slice function can be computed by a linear-size, logarithmic-depth monotone real formula. This implies, together with the clique-colouring lower bound [Pud97] and the fact that there is a monotone function in $\mathsf{P/poly}$ that extends clique-colouring [Tar88], that monotone real circuits and non-monotone boolean circuits are incomparable.

### 4.2 Feasible interpolation and lifting theorems

Krajíček’s interpolation technique [Kra97] gives a method of obtaining resolution lower bounds from monotone circuit lower bounds. The technique is actually more general and implies lower bounds also for cutting planes with bounded coefficients. Pudlák [Pud97] was able to extend these results to unrestricted cutting planes by showing that cutting planes refutations can be interpolated by monotone real circuits. The classical method of interpolation only applies to certain structured formulas but, as we have discussed in Section 2, they can be generalised to other formulas by considering unsatisfiability certificates [HP17, FPPR17].

**Theorem 4.5.** Let $F$ be an unsatisfiable CNF formula with $m$ clause, let $(X, Y)$ be any partition of the variables of $F$, and let $\text{cert}_F$ denote the corresponding unsatisfiability certificate. If there is a cutting planes refutation of $F$ of length $s$ then there is a monotone real circuit computing $\text{cert}_F$ of size $O(s + m^2)$.

Pudlák’s interpolation theorem [Pud97] implied the first superpolynomial lower bound for unrestricted cutting planes and for two decades monotone interpolation was the only known method for proving cutting planes lower bounds. Some of the formulas for which we can obtain cutting planes lower bound via this monotone interpolation theorem include clique-colouring [Pud97], random $k$-CNF formulas for $k = \Theta(\log n)$ [HP17, FPPR17], and weak bit pigeonhole principle [HP17].

Again we can consider if a converse of this theorem holds. Can we obtain monotone real circuit lower bounds from proof complexity lower bounds? The answer is once again yes. The dag-like lifting theorem by Göös et al. [GGKS20] holds also for triangle-dags and therefore for monotone real circuits. Note that the proof complexity measure that is being lifted is still resolution width.

**Theorem 4.6 ([GGKS20]).** Let $F$ be an unsatisfiable CNF formula on $n$ variables, and let $m = n^C$ for a sufficiently large constant $C$. Then the size of any monotone real circuit computing $\text{cert}_{F_{\text{ind}}}^m$ is at least $n^{2w(F)}$, where $w(F)$ is the minimum width of any resolution refutation of $F$.

This implies that the lower bounds for monotone boolean circuits that can be obtained via this lifting theorem also hold for monotone real circuits. The main motivation for proving this
lifting theorem was not, however, to obtain monotone real circuit lower bounds, but to obtain cutting planes lower bounds. The lifting theorem, as stated in [GGKS20], gives a lower bound for triangle-dags. It is straightforward to see that every halfspace-dag is, in particular, a triangle-dag. Therefore, in order to obtain a cutting planes lower bound it is sufficient to reduce the mKW game on cert_{F_{\text{IND}_{\text{m}}}} to Search(F') for some formula F'. This can be done via syntactic manipulations, giving rise to the so-called lifted formulas, or with some more clever reductions, which can give lower bounds for more natural formulas such as the clique-colouring formula. Lifting theorems significantly increase the families of formulas for which we have cutting planes lower bounds: any formula for which we can prove a strong-enough resolution width lower bound can be transformed into a formula that is hard for cutting planes.

**Corollary 4.7** ([GGKS20]). Given any unsatisfiable k-CNF formula F on n variables, there is a polynomial-time constructible unsatisfiable 2k-CNF formula F' on \(n^{O(1)}\) variables such that any cutting planes refutation of F' must have length at least \(n^{\Omega(w(F))}\), where \(w(F)\) is the minimum width of any resolution refutation of F.

Prior to [GGKS20], lifting theorems were proven for a related model called real communication protocols [Kra98, BEGJ00]. In this setting, in order to solve a relation or search problem \(S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}\), Alice and Bob communicate via a referee that acts as a greater-than oracle: at each round, Alice and Bob each send a real number to the referee and the referee replies to both player with a bit indicating whether Alice’s number is greater than Bob’s number.

Such a protocol can be viewed as a binary tree where every non-leaf node v is labelled by two functions \(f^A_v : \mathcal{X} \rightarrow \mathbb{R}\) and \(f^B_v : \mathcal{Y} \rightarrow \mathbb{R}\), and every leaf node u is labelled by some output \(o_u \in \mathcal{O}\). For an input \((x, y) \in \mathcal{X} \times \mathcal{Y}\), the output of the protocol is defined by constructing a path from the root to a leaf according to the following rule: at node v if \(f^A_v(x) > f^B_v(y)\) then the path continues to the left child of v, otherwise the path continues to the right child of v. When the path reaches a leaf u of the tree, then the output is \(o_u\). We say the protocol computes \(S\) if for every \((x, y) \in \mathcal{X} \times \mathcal{Y}\) the output \(o\) of the protocol is such that \((x, y, o) \in S\). The depth of the protocol is the length of the longest root-to-leaf path and the size is the number of nodes in the tree.

It is not hard to see that this model can be quite powerful: the equality function—which can only be solved by exponential-size triangle-dags—can be trivially solved by constant-size real communication protocols. Nevertheless, Bonet et al. [BEGJ00] showed that it is possible to prove lower bounds for this model by extending [RM99]. Note that this in particular implies a tree-like version of the lifting theorem in [GGKS20].

**Theorem 4.8** ([BEGJ00]). Let F be an unsatisfiable CNF formula on \(n\) variables, and let \(m \geq n^C\) for a sufficiently large constant \(C\). Any real communication protocol that solves Search\((F) \circ \text{IND}_m\) must have depth at least \(\Omega(D(F) \cdot \log m)\), where \(D(F)\) is the minimum depth of any resolution refutation of F.

We note that Search\((F) \circ \text{IND}_m\) is communication reducible to the mKW game for cert_{F_{\text{IND}_{\text{m}}}}. Thus, following Raz and McKenzie [RM99], this lifting theorem implies that the mKW game for the generation function—which can be solved by polynomial-size rectangle-dags—requires exponential-size real communication protocols. From this result, Bonet et al. [BEGJ00] use the fact [Kra98, Joh98] that a lower bound for real communication protocols of the mKW game of a function \(f\) implies a lower bound for monotone real formulas of \(f\) to show that monotone boolean circuits can be exponentially stronger than monotone real formulas (the opposite separation follows from [Ros97]).

Bonet et al. [BEGJ00] then use this result to show that resolution can be exponentially stronger than tree-like cutting planes (the opposite separation is witnessed by the pigeonhole principle).
Their proof uses their circuit separation and Pudlák’s interpolation theorem [Pud97], but it is possible to obtain the cutting planes lower bound directly from the lower bound for real communication protocols.

In the wake of Göös et al. [GPW18], de Rezende et al. [dRNV16] extended this lifting theorem to a round-preserving setting. This was used to obtain length-space trade-offs for cutting planes. We note that, in contrast to [BEGJ00], for this application it is indeed important that the communication model of the lifting theorem is real communication protocols and not triangle-dags.

5 Perspectives and Open Problems

In this final section, we discuss directions for further research inspired by the themes of this survey. Our main conceptual message is this: We advocate to foster the interplay between the different subfields of complexity theory—circuit complexity, proof complexity, query/communication complexity, and the theory of total search problems. Firstly, can we find more connections between these theories? To discover such interconnections, we adhere to the following philosophy.

Guiding philosophy: A natural concept introduced in one theory should have a natural counterpart in another theory.

Secondly, given such connections, can we bring the techniques of one field to bear on the open questions of another field?

5.1 Query/communication TFNP

In the context of classical (Turing machine) complexity theory, the study of total NP search problems (TFNP) was initiated by Megiddo and Papadimitriou [MP91, Pap94]. This theory aims to characterise the complexities of search problems that have a solution for every input and where a given solution can be efficiently checked for validity. By now, this theory has flowered into a sprawling jungle of widely-studied complexity classes (such as PLS [JPY88], PPA/PPAD/PPP [Pap94], CLS [DP11]) that serve to classify the complexities of many important search problems. For instance, one of the most celebrated results in the field is that computing a Nash equilibrium (every game must have one) is complete for PPAD [DGP09, CDT09].

Example. As a running example, we recall the class PLS (Polynomial Local Search) which embodies the combinatorial principle “every dag has a sink”. Namely, PLS consists of all search problems that can be reduced (in polynomial-time) to the Iteration problem, defined as follows. The input to Iteration consists of two circuits $S, P : \{0,1\}^n \rightarrow \{0,1\}^n$ (here $S$ is the successor circuit and $P$ the potential circuit) that implicitly define a dag $G = (V, E)$, $V = \{0,1\}^n$, where $(u, v) \in E$ iff the successor of $u$ is $v$ (i.e., $S(u) = v$) and the potential decreases from $u$ to $v$ (i.e., $P(v) < P(u)$ where $P$’s output is interpreted as an $n$-bit number). The goal is to output the name of any node in $G$ that is a sink (no successor).

In this survey, we have discussed total search problems in the context of more restricted models of computation: proof systems can be viewed as a model of computation solving Search($F$), and communication protocols can solve KW($f$) games. How are these connected to the classical TFNP theory?

A couple of connections have already been made formal. Given any subclass of TFNP—take PLS for instance—one may naturally define its query (decision tree) analogue PLS$^{dt}$ and its communication analogue PLS$^{cc}$. For instance, to define PLS$^{dt}$ we start with the query analogue of the Iteration
problem, namely, $\text{Search}(F)$ where $F$ is any natural low-width encoding of the contradiction stating that a dag has no sink; for example, one can take as $F$ the famous pebbling formulas. Then $\text{PLS}^{\text{dt}}$ is defined as consisting of all query search problems that can be reduced, via shallow decision-tree reductions, to the query-Iteration problem. (See [GKRS18] for a more formal definition.) What is striking is that in some cases the query analogue of a TFNP class can be captured by a natural propositional proof system, and the communication analogue can be captured by a natural circuit model (via the KW game). In particular, we have the following classical characterisations discussed in this survey:

(FP) The class $\text{FP}$ consists of all search problems soluble in polynomial time on a deterministic Turing machine. By analogy, Karchmer–Wigderson games involve deterministic communication protocols, that is, the communication analogue $\text{FP}^{\text{cc}}$ of $\text{FP}$. We may summarise the Karchmer–Wigderson theorem (Theorem 2.5) by saying that $\text{FP}^{\text{cc}}$ captures formulas. Moreover, the deterministic lifting theorem (Theorem 2.8) can be viewed as showing lower bounds for $\text{FP}^{\text{cc}}$ from its query analogue $\text{FP}^{\text{dt}}$, which in turn corresponds to the depth of tree-like resolution proofs.

(PLS) Razborov [Raz95] showed that the circuit complexity of a function $f$ is captured by the least cost of a “$\text{PLS}$-protocol” solving the KW$(f)$ game. Here a $\text{PLS}$-protocol is a natural communication analogue of PLS, which turns out to be equivalent to rectangle-dags, which we discussed in Section 3. We may summarise Razborov’s result by saying that $\text{PLS}^{\text{cc}}$ captures circuits. The dag-like lifting theorem Section 3 can then be viewed as showing lower bounds for $\text{PLS}^{\text{cc}}$ from its query analogue $\text{PLS}^{\text{dt}}$, which turns out to correspond to the width of resolution proofs.

Both of these connections were found in the 90s. A more recent work has identified a third connection of the above type: $\text{PPA}^{\text{cc}}$ captures $\mathbb{F}_2$-span programs, and $\text{PPA}^{\text{dt}}$ captures the $\mathbb{F}_2$-Nullstellensatz proof system [GKRS18]. These characterisations, too, could have been discovered in the 90s: The class $\text{PPA}$ (embodying the principle “every graph with an odd degree vertex has another”) is one of the original TFNP subclasses defined by Papadimitriou [Pap94]. Similarly, span programs are a well-studied circuit model introduced in 1993 [KW93] with fundamental connections to quantum computing and secret sharing in cryptography.

Currently, around a dozen interesting subclasses of TFNP have been identified. A natural research question, following our philosophy, is to systematically study the interconnections that arise out of the Turing-machine/query/communication variants of TFNP subclasses. Figure 2 summarises our current understanding of the landscape for communication search problem classes.

**Open problem 1.** Complete the picture in Figure 2.

In particular, the following questions remain open.

(Q1) Find circuit models captured by classic classes such as $\text{PPAD}^{\text{cc}}$, $\text{PPADS}^{\text{cc}}$, $\text{PPP}^{\text{cc}}$. What complexity class captures comparator circuits (only an upper bound of $\text{SOML}^{\text{cc}}$ is known)?

(Q2) Find proof systems captured by $\text{PPAD}^{\text{dt}}$, $\text{PPADS}^{\text{dt}}$, $\text{PPP}^{\text{dt}}$.

(Q3) In query complexity, the relative complexities of classical TFNP subclasses are nearly completely understood [BCE*98, BM04, Mor05]. In communication complexity, by contrast, there are huge gaps in our understanding. For example, there are no lower bounds against classes $\text{PPADS}^{\text{cc}}$ and $\text{PPP}^{\text{cc}}$ for an explicit search problem.
(Q4) Prove more class separations. If we have two classes, which both admit an associated circuit model, then separating the two classes is often equivalent to separating the two circuit models in the monotone setting. For example, can we show SOML$^{cc}$ $\not\subseteq$ PPA$^{cc}$? This is closely related to whether monotone comparator circuits can be more powerful than monotone $\mathbb{F}_2$-span programs (no separation is currently known).

(Q5) Query-to-communication lifting theorems are known for FP [RM99], PLS [GGKS20], and PPA [PR18]. Prove more. (This is one way to attack (Q3) if proved for PPADS or PPP.)

Building a theory of the intricate relationships between communication search problem classes would further elucidate the relative power of the underlying combinatorial principles and (if available) of the associated circuit models. Moreover, such a theory would likely have applications to all three involved subfields, which is exactly what already happened in the case of the best-understood classes FP, PLS, PPA.

5.2 Cross-fertilisation of lower-bound methods

Besides building interconnections between theories, another research direction is to put these connections to use and prove concrete lower-bound results. We propose several problems that can benefit from such cross-fertilisation.

**Proof complexity.** The progress on the central project of proof complexity—proving lower bounds against increasingly powerful proof systems—has been slow. While the limitations of resolution are well understood after decades worth of research, we still lack lower bounds even for mild generalisations of resolution. Two of the simplest such generalisations are resolution...
over cutting planes, denoted $R(\text{CP})$ [Kra98], and resolution over $\mathbb{F}_2$-linear equations, denoted $R(\text{LIN}_2)$ [RT08, IS14]. These are dag-like proof systems that reason using disjunctions of linear inequalities in case of $R(\text{CP})$ or disjunctions of linear equations in case of $R(\text{LIN}_2)$.

**Open problem 2.** Show that $R(\text{CP})$ and $R(\text{LIN}_2)$ are not polynomially bounded (i.e., there are unsatisfiable formulas that the systems cannot refute with a polynomial-size proof).

We believe the time is ripe for a breakthrough here. Krajíček, in his new textbook on proof complexity, writes [Kra19, §22.2]:

In my view the most pressing open problem is to extend some of the lower bound methods from the A-level to systems like $R(\text{CP})$ or $R(\text{LIN})$. Feasible interpolation seems to be best positioned for that.

Lifting theorems discussed in this survey have deepened our understanding of the interpolation technique and present a new avenue of attack towards Open problem 2. In fact, these techniques come frustratingly close to resolving it: Both $R(\text{CP})$ and $R(\text{LIN}_2)$ involve lines that can be evaluated using a randomized communication protocol. Recent work has already produced lifting theorems both for dag-like proofs whose lines are computed by deterministic protocols [GGKS20] and for tree-like proofs whose lines are computed by randomized protocols [GPW20]. Seemingly, one only needs to find a way to combine the techniques of these two papers! See also [Fol22] for a discussion of this problem.

Another foremost open problem in proof complexity (highlighted in Razborov’s survey [Raz16]) concerns semi-algebraic proof systems that manipulate low-degree polynomials, e.g., the ubiquitous *Sum-of-Squares* system. Can we prove lower bounds on their dag-like proof size? Since degree-$d$ polynomials can be efficiently computed by multi-party number-on-forehead (NOF) protocols, one might hope to approach this question by developing a lifting theory for NOF protocols. However, our understanding of NOF protocols is lacking even in the tree-like setting.

**Circuit complexity.** An outstanding open problem in circuit complexity concerns matchings:

**Open problem 3.** Show that perfect matching requires exponential-size monotone circuits.

The best lower bound so far is quasi-polynomial as shown by Razborov [Raz85b]. Matching principles have been studied extensively in proof complexity [DR01, Urd03, Ale04] and one could hope that these results could be lifted to monotone circuit complexity. There has been an analogous challenge in the study of linear programming formulations of the matching polytope. While there are lifting theorems to prove LP formulation lower bounds [CLRS16, KMR21], there is no lifting-based proof for Rothvoß’s [Rot14] exponential lower bound for the matching polytope. This is one reason why it is still open to show an SDP formulation lower bound for matching; the only lower-bound technique known is a lifting theorem [LRS15]. There also remains the lurking prospect that Razborov’s lower bound is, in fact, tight! There is a recent cautionary tale this effect: In a surprising breakthrough Dadush and Tiwari [DT20] showed that Tseitin formulas can be refuted in quasi-polynomial size by the cutting planes proof system, contradicting a conjecture of an exponential lower bound widely believed since the introduction of cutting planes in 1987 [CCT87].

**Total search problems.** Finally, in the theory of TFNP, a timely opportunity is to use proof complexity to understand the power of some of the newly introduced TFNP subclasses; examples include the “continuous local search” class CLS [DP11], “end of metered line” class EOML [HY17] and the “unique end of potential line” class UEPL [FGMS20]. Last year saw a surprising breakthrough result by Fearnley et al. [FGHS21] who showed that CLS, which was introduced as a
natural subclass of the not-so-natural $\text{PLS} \cap \text{PPAD}$, in fact coincides with $\text{PLS} \cap \text{PPAD}$! That such a class collapse was missed for a decade gives an impetus for finding black-box separations (aka relativised oracle separations) for the newly introduced classes. Can one show $\text{CLS}^\text{dt} \neq \text{UEOPL}^\text{dt}$, that is, $\text{CLS} \neq \text{UEOPL}$ relative to an oracle? Capturing $\text{CLS}^\text{dt}$ and $\text{UEOPL}^\text{dt}$ in the language of proof complexity is a first step.

References

[AB87] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. Combinatorica, 7(1):1–22, March 1987. doi:10.1007/bf02579196.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. Journal of Computer and System Sciences, 74(3):323–334, May 2008. doi:10.1016/j.jcss.2007.06.025.

[Ale04] Michael Alekhnovich. Mutilated chessboard problem is exponentially hard for resolution. Theoretical Computer Science, 310(1-3):513–525, January 2004. doi:10.1016/s0304-3975(03)00395-5.

[BCE98] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. Journal of Computer and System Sciences, 57(1):3–19, August 1998. doi:10.1006/jcss.1998.1575.

[BEGJ00] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. SIAM Journal on Computing, 30(5):1462–1484, January 2000. Preliminary version in FOCS ’98. doi:10.1137/s0097539799352474.

[BM04] Joshua Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In Proceedings of the 19th Conference on Computational Complexity (CCC ’04), pages 54–67, 2004. doi:10.1109/CCC.2004.1313795.

[BP98] Samuel R. Buss and Toniann Pitassi. Good degree bounds on Nullstellensatz refutations of the induction principle. Journal of Computer and System Sciences, 57(2):162–171, October 1998. Preliminary version in CCC ’96. doi:10.1006/jcss.1998.1585.

[BU99] Christer Berg and Staffan Ulfberg. Symmetric approximation arguments for monotone lower bounds without sunflowers. Computational Complexity, 8(1):1–20, June 1999. doi:10.1007/s000370050017.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. Journal of the ACM, 48(2):149–169, March 2001. Preliminary version in STOC ’99. doi:10.1145/375827.375835.

[CCT87] William Cook, Collette Coullard, and György Turán. On the complexity of cutting-plane proofs. Discrete Applied Mathematics, 18(1):25–38, September 1987. doi:10.1016/0166-218x(87)90039-4.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. Journal of the ACM, 56(3):1–57, May 2009. doi:10.1145/1516512.1516516.
[FPRP17] Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random Θ(log n)-CNFs are hard for cutting planes. In Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS ’17), October 2017. doi:10.1109/focs.2017.19.

[Fu97] Xudong Fu. Lower bounds on sizes of cutting plane proofs for modular coloring principles. In DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 135–148. American Mathematical Society, November 1997. Preliminary version in STOC ’96. doi:10.1090/dimacs/039/08.

[GGKS20] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. Theory of Computing, 16(1):1–30, 2020. Preliminary version in STOC ’18. doi:10.4086/toc.2020.v016a013.

[GH92] Mikael Goldmann and Johan Håstad. A simple lower bound for monotone clique using a communication game. Information Processing Letters, 41(4):221–226, March 1992. doi:10.1016/0020-0190(92)90184-w.

[GKRS18] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS ’19), volume 124 of Leibniz International Proceedings in Informatics (LIPIcs), pages 38:1–38:19, January 2018. doi:10.4230/LIPIcs.ITCS.2019.38.

[Gom63] Ralph E. Gomory. An algorithm for integer solutions to linear programs. Recent advances in mathematical programming, 64(260-302):14, 1963.

[GP18] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. SIAM Journal on Computing, 47(5):1778–1806, January 2018. doi:10.1137/16M1082007.

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. SIAM Journal on Computing, 47(6):2435–2450, January 2018. Preliminary version in FOCS ’15. doi:10.1137/16m1059369.

[GPW20] Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. SIAM Journal on Computing, 49(4):FOCS17–441–FOCS17–461, January 2020. Preliminary version in FOCS ’17. doi:10.1137/17m115339x.

[GS95] Michelangelo Grigni and Michael Sipser. Monotone separation of logarithmic space from logarithmic depth. Journal of Computer and System Sciences, 50(3):433–437, June 1995. Preliminary version in SCT ’91. doi:10.1006/jcss.1995.1033.

[Hak85] Armin Haken. The intractability of resolution. Theoretical Computer Science, 39:297–308, 1985. doi:10.1016/0304-3975(85)90144-6.

[HC99] Armin Haken and Stephen A. Cook. An exponential lower bound for the size of monotone real circuits. Journal of Computer and System Sciences, 58(2):326–335, April 1999. doi:10.1006/jcss.1998.1617.

[HP17] Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer
[HP18] Pavel Hrubeš and Pavel Pudlák. A note on monotone real circuits. Information Processing Letters, 131:15–19, March 2018. doi:10.1016/j.ipl.2017.11.002.

[HR00] Danny Harnik and Ran Raz. Higher lower bounds on monotone size. In Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC '00), pages 378–387, 2000. doi:10.1145/335305.335349.

[HY17] Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '17), pages 1352–1371, January 2017. doi:10.1137/1.9781611974782.88.

[IS14] Dmitry Itsykson and Dmitry Sokolov. Lower bounds for splittings by linear combinations. In Proceedings of the 39th Mathematical Foundations of Computer Science (MFCS '14), volume 8635 of Lecture Notes in Computer Science, pages 372–383. Springer, 2014. doi:10.1007/978-3-662-44465-8_32.

[Joh98] Jan Johannsen. Lower bounds for monotone real circuit depth and formula size and tree-like cutting planes. Information Processing Letters, 67(1):37–41, July 1998. doi:10.1016/s0020-0190(98)00079-9.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? Journal of Computer and System Sciences, 37(1):79–100, August 1988. doi:10.1016/0022-0000(88)90046-3.

[Juk97] Stasys Jukna. Finite limits and monotone computations: The lower bounds criterion. In Proceedings of the 12th Annual IEEE Conference on Computational Complexity (CCC '97), pages 302–313, June 1997. doi:10.1109/CCC.1997.612325.

[Juk99] Stasys Jukna. Combinatorics of monotone computations. Combinatorica, 19(1):65–85, January 1999. doi:10.1007/s004930050046.

[KMR21] Pravesh K. Kothari, Raghu Meka, and Prasad Raghavendra. Approximating rectangles by juntas and weakly exponential lower bounds for LP relaxations of CSPs. SIAM Journal on Computing, pages STOC17–305–STOC17–332, May 2021. Preliminary version in STOC ’17. doi:10.1137/17m1152966.

[KPPY84] Maria Klawe, Wolfgang J. Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth. In Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC '84), pages 480–487, December 1984. doi:10.1145/800057.808717.

[Kra97] Jan Krajiček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. Journal of Symbolic Logic, 62(2):457–486, June 1997. doi:10.2307/2275541.

[Kra98] Jan Krajiček. Interpolation by a game. Mathematical Logic Quarterly, 44(4):450–458, 1998. doi:10.1002/malq.19980440403.

[Kra19] Jan Krajiček. Proof Complexity. Cambridge University Press, 2019.

[KW90] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require
super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, May 1990. doi:10.1137/0403021.

[**KW93**] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the 8th Structure in Complexity Theory Conference (SCT '93)*, pages 102–111, May 1993. doi:10.1109/SCT.1993.336536.

[LMM*20] Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. Lifting with sunflowers. Technical Report TR20-111, Electronic Colloquium on Computational Complexity (ECCC), 2020. URL: https://eccc.weizmann.ac.il/report/2020/111/.

[**LRS15**] James Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the 47th Annual Symposium on Theory of Computing (STOC '15)*, pages 567–576, 2015. doi:10.1145/2746539.2746599.

[Mor05] Tsuyoshi Morioka. *Logical Approaches to the Complexity of Search Problems: Proof Complexity, Quantified Propositional Calculus, and Bounded Arithmetic*. PhD thesis, University of Toronto, 2005. URL: https://eccc.weizmann.ac.il/resources/pdf/morioka.pdf.

[**MP91**] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, April 1991. doi:10.1016/0304-3975(91)90200-L.

[**Mul87**] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, March 1987. doi:10.1007/bf02579205.

[**Pap94**] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, June 1994. doi:10.1016/s0022-0000(05)80063-7.

[**Pot17**] Aaron Potechin. Bounds on monotone switching networks for directed connectivity. *Journal of the ACM*, 64(4):1–48, September 2017. doi:10.1145/3080520.

[**PR17**] Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC '17)*, June 2017. doi:10.1145/3055399.3055478.

[**PR18**] Toniann Pitassi and Robert Robere. Lifting Nullstellensatz to monotone span programs over any field. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18)*, pages 1207–1219, 2018. doi:10.1145/3188745.3188914.

[**Pud97**] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, September 1997. doi:10.2307/2275583.

[**Pud00**] Pavel Pudlák. Proofs as games. *The American Mathematical Monthly*, 107(6):541–550, June 2000. doi:10.1080/00029890.2000.12005233.

[Raz85a] Alexander A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. *Soviet Mathematics Doklady*, 31(2):354–357, 1985. English translation of a paper in *Doklady Akademii Nauk SSSR*.

[Raz85b] Alexander A. Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical notes of the Academy of Sciences of the USSR*, 37(6):485–493,
June 1985. doi:10.1007/BF01157687.

[Raz90] Alexander A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. Combinatorica, 10(1):81–93, March 1990. doi:10.1007/bf02122698.

[Raz95] Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. Izvestiya: Mathematics, pages 201–224, February 1995. doi:10.1070/im1995v059n01abeh000009.

[Raz16] Alexander A. Razborov. Proof complexity and beyond. SIGACT News, 47(2):66–86, June 2016. doi:10.1145/2951860.2951875.

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. Combinatorica, 19(3):403–435, March 1999. Preliminary version in FOCS '97. doi:10.1007/s004930050062.

[Rob18] Robert Robere. Unified lower bounds for monotone computation. PhD thesis, University of Toronto (Canada), 2018. URL: https://hdl.handle.net/1807/92007.

[Ros97] Arnold Rosenbloom. Monotone real circuits are more powerful than monotone Boolean circuits. Information Processing Letters, 61(3):161–164, February 1997. doi:10.1016/s0020-0190(97)00007-0.

[Rot14] Thomas Rothvoß. The matching polytope has exponential extension complexity. In Proceedings of the 46th Annual Symposium on Theory of Computing (STOC '14), pages 263–272, 2014. doi:10.1145/2591796.2591834.

[RT08] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. Annals of Pure and Applied Logic, 155(3):194–224, October 2008. doi:10.1016/j.apal.2008.04.001.

[RW92] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. Journal of the ACM, 39(3):736–744, July 1992. doi:10.1145/146637.146684.

[Sok17] Dmitriy Sokolov. Dag-like communication and its applications. In Proceedings of the 12th International Computer Science Symposium in Russia (CSR '17), volume 10304 of Lecture Notes in Computer Science, pages 294–307. Springer, June 2017. doi:10.1007/978-3-319-58747-9_26.

[Tar88] Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. Combinatorica, 8(1):141–142, March 1988. doi:10.1007/bf0122563.

[Urq87] Alasdair Urquhart. Hard examples for resolution. Journal of the ACM, 34(1):209–219, January 1987. doi:10.1145/7531.8928.

[Urq03] Alasdair Urquhart. Resolution proofs of matching principles. Annals of Mathematics and Artificial Intelligence, 37(3):241–250, March 2003. doi:10.1023/a:1021231610627.

[Weg87] Ingo Wegener. The complexity of Boolean functions. Wiley-Teubner, 1987.