(submitted)
StandardLayout = true, SmallCapsFigTabCaptions = false

9ECM
LOGO

Proofs, computational complexity, and games

Pavel Pudlák

Abstract.

This paper discusses the interplay between proof complexity, computational complexity, and combinatorial games. Classical proof theory typically concerns itself with the provability of sentences without considering the lengths of proofs. One of the basic results is that, for sufficiently rich theories, it is impossible to distinguish provable sentences from false sentences using an algorithm. In proof complexity, the lengths of proofs play a central role, hence, the question is how difficult it is to distinguish propositions with polynomial-size proofs from false; in particular, whether this is possible using an algorithm running in polynomial time. This problem is represented using the concept of canonical pairs of proof systems, which is closely connected with the concept of interpolation pairs of proof systems. Interpolation pairs allow us to connect proof systems to computational complexity problems and, for some proof systems, even to prove exponential lower bounds on the lengths of proofs. Our goal is to extend this approach to lower bounds to stronger systems, in particular, to constant depth Frege proof systems. To this end, we have characterized the interpolation pairs of constant-depth Frege systems using certain combinatorial games. In this paper, we will define these games and explain their connection to some other concepts studied in proof complexity, specifically, the communication complexity versions of **TFNP** classes. Our aim is to show that the three concepts studied in logic and computational complexity theory, *playing combinatorial games, computing,* and *solving problems using communication of two players,* are different facets of the same thing.

Mathematics Subject Classification 2020: 03D15 (primary); 68Q15 (secondary).
Keywords: proofs, complexity, games.

1. Introduction

Classical *computability theory* studies questions asking which functions are computable, which problems are decidable, etc. Here, computability means that some algorithm produces an answer in a finite amount of time. Knowing that a problem can be solved in a finite time certainly provides an essential piece of information, but more precise information is needed in practice. *Computational complexity* therefore considers resources required to perform computations. These are mostly time and space (by which we mean memory), but also others, such as randomness, nondeterminism, etc. While this may look like a part of applied science, many mathematicians are attracted to this field by deep theoretical problems, such as the **P** vs. **NP** question, which is considered by many as one of the most difficult mathematical problems.

Similarly, classical *proof theory* studies the provability of sentences in principle, i.e., disregarding the lengths of proofs, while *proof complexity* is concerned with the lengths of proofs. There are many parallels between these research areas; one is that several fundamental problems in proof complexity resist all attempts to be solved, like the fundamental problems in computational complexity.

There are problems that belong both to proof theory and computability theory. The most important of these is: for a given formal system, say a theory with a computable set of axioms, how difficult is it to distinguish provable sentences from false sentences? In this paper, I will deal with a similar problem in proof and computational complexity: for a given propositional proof system, how difficult is it to distinguish propositions with polynomial-length proofs from false propositions? This question is formalized by the concept of the *canonical pair* of a proof system. Canonical pairs are specific pairs of disjoint **NP** sets, and the question is how complex a set we need to separate such a pair.[1] The related concept of the *interpolation pair* of a proof system is connected with the method of *feasible interpolation* by which exponential lower bounds were proved for some proof systems. By studying these pairs, we want to determine limits of this method.

In [4], we characterized the canonical pair of Resolution, which is the depth-0 Frege system. This also characterizes the interpolation pair of the depth-1 Frege system. In [25], I gave a combinatorial characterization of interpolation pairs for all levels of the bounded depth Frege hierarchy, which is also a characterization of canonical pairs because the canonical pair of depth d Frege system is equivalent to the interpolation pair of depth $(d + 1)$-Frege system. These characterizations are based on certain combinatorial games[2] introduced in that paper. This paper will introduce these games and, for depth 0 and 1, show how they naturally emerge from the

[1] Even if both sets are **NP**-complete, it may be possible to separate them using a set in **P**.
[2] One can also say *one* game with the parameter d.

communication complexity **TFNP** classes associated with the proof systems. Moreover, I will briefly mention games with more than two players and their connection to multiparty-communication complexity and a couple of combinatorial games that have been studied in connection with constant depth Frege systems.

2. Propositional proof systems

When I started with logic, I considered propositional logic absolutely uninteresting. I wondered why some logicians pay so much attention to the laws and rules of propositional logic when there is a simple procedure to test whether a propositional formula is a tautology. This view conformed with the widespread opinion that mathematics starts with infinity because finite problems are trivial. We can indeed check a tautology simply by testing all truth assignments to its variables, but when we consider *the complexity of testing*, things become interesting (or if we consider non-classical logics, which I will not do in this paper). Many tautologies have simple proofs, while others, for example, random formulas in a suitable probability distribution, seem hard.

But why should a mathematician study such problems? Isn't it just a problem for practical computations? The answer is yes and no. Yes, because propositional proofs are used in automated reasoning (including the now popular AI systems). No, because the lengths of propositional proofs are connected with provability in first-order theories. Lower bounds on the lengths of propositional proofs can, in principle, be used to show the unprovability of Π_1-sentences in first-order theories axiomatizing fragments of arithmetic, a fact discovered by S. A. Cook [10]. We are able to prove superpolynomial lower bounds on propositional proofs only in very special instances, yet there are several interesting applications of this method.

Another reason for studying propositional proof complexity is that in this area, there are deep problems similar to the fundamental problems in computational complexity. For instance, the problem, whether it is possible to prove every tautology in a Frege proof system using a polynomial number of formulas, seems as hard as the problem of proving superpolynomial lower bounds on circuits computing explicitly defined Boolean functions. (The latter is still a widely open problem after 75 years of research.)

When discussing the lengths of propositional proofs, we have to specify what we mean by proofs. This is done by defining the important concept of *propositional proof systems*. In the rest of the paper, I will only say "proof system" because I will not consider proof systems for other logics.

The general concept of a proof system P only requires three properties:

(1) P is sound (proves only tautologies),

(2) P is complete (proves all tautologies), and

(3) it is decidable in polynomial time whether a given string is a proof of a given formula.

Here, I will only consider proof systems that are fragments of a standard proof system based on axiom schemas and deduction rules. There are many such formalizations that differ in the choice of the language, axioms, and rules, but they are all equivalent in a well-defined sense. The official name for this class of proof system is *Frege Systems.* In order to define bounded depth fragments of Frege Systems, we restrict the language to the connectives $\wedge, \vee, \neg$ and allow negations only at variables. Then we say that a formula ϕ has depth d if the number of alternations of ANDs and ORs is d. The *Depth-d Frege System,* F_d, is the subsystem in which only depth d formulas are allowed.

I will define more precisely the two systems at the bottom of this hierarchy, F_0 and F_1. The first one is usually presented as a *refutation system* and is called *Resolution*. In a refutation system one starts with the negation of the assumptions and tries to derive a contradiction. Resolution is used to refute CNF formulas, which are conjunctions of disjunctions. The disjunctions are presented as sets of literals and called *clauses*, where *literals* are variables and negated variables. The CNF to be refuted is presented as set of clauses. A refutation is a sequence of clauses derived from the clauses of the CNF formula and previously derived clauses. There is only one derivation rule, which is called the *resolution rule,* which is usually stated as follows

$$\frac{C \cup \{x\} \quad D \cup \{\neg x\}}{C \cup D},$$

which means that from the two assumptions $C \cup \{x\}, D \cup \{\neg x\}$, conclusion $C \cup D$ follows. The aim is to derive the empty clause $\emptyset$ that represents a contradiction.

The system F_1 is an extension in which clauses may contain not only literals, but also conjunctions of literals. It has an axiom $\{x, \neg x\}$ for every variable x, and rules

$$\frac{C \cup \{\alpha \wedge z\} \quad D \cup \{\neg z\}}{C \cup D}, \qquad \frac{C \cup \{\alpha\} \quad D \cup \{z\}}{C \cup D \cup \{\alpha \wedge z\}}, \qquad \frac{C}{C \cup D},$$

where α is a, possibly empty, conjunction of literals.

3. Canonical and interpolation pairs of proof systems

Definition 1 (Razborov, [27]). *The* canonical pair *of a proof system P is the pair of two disjoint* ***NP*** *sets*

- $C_P^A = \{(\phi, 1^n)\ ;\ \textit{there a } P\textit{-refutation of } \phi \textit{ of length } \leq n\}$

- $C_P^B = \{(\phi, 1^n)\ ;\ \phi \text{ is satisfiable}\}$

where ϕ is a propositional formula (a CNF w.l.o.g.) and 1^n, a string of 1s of length n, is a limit on the lengths of proofs given in unary.

I will denote by $C_P := (C_P^A, C_P^B)$.

Informally, think of C_p^A as the set of formulas with short P-refutations and C_p^B as the set of satisfiable formulas. The "padding" 1^n is a convenient way to avoid saying what short means. It must also be in C_P^B because otherwise, it would be easy to distinguish elements of C_P^A from elements of C_P^B.

The concept of canonical pairs is a way to connect a proof system with a computational problem—instead of asking about the lengths of proofs, we can ask what is the computational complexity of separating these disjoint **NP**-pairs. The stronger the proof system is, the more difficult it is to separate the pairs. The intuition is that it is easier to separate two things that are far apart than those that are close to each other. In the canonical pairs, the second set is always the same, while the size of the first increases with the increasing strength of the proof system because more tautologies have short proofs.

We do not know whether the canonical pairs of F_d can be separated by a set in **P**. Building on our previous result [21], Bonet et al [5] showed that for large depth d, the canonical pairs of F_d cannot be separated by a set in **P** if one assumes some strong conjecture from cryptography. There are no results of this kind based on standard conjectures about complexity classes such as **P**≠**NP**.[3] The most interesting problem is the complexity of the canonical pair of F_0 (Resolution). Essentially, it is a win-win question: a polynomial separation of this pair would have application in SAT-solving and automated theorem proving, where Resolution is used; on the other hand, a superpolynomial lower bound may lead to new lower bounds in proof complexity.

Since it is unclear whether cryptographic hypotheses could help prove inseparability for low levels of Frege hierarchy, we can try the opposite: to find applications of the hypotheses that some canonical pairs are not polynomially separable in cryptography. The problem is, however, that cryptography needs sets in **NP**∩**coNP**, not just a pair of disjoint **NP** sets.

The connection of canonical pairs with lower bounds on proofs is more general. To this end, I will introduce another concept.

Definition 2 ([24]). *The* interpolation pair *of a proof system P is the pair of two disjoint **NP** sets*

[3]The famous result about nonautomatibility of Resolution [2] implies that C_{Res}^A is **NP**-complete. The fact that both sets in a disjoint **NP**-pair are **NP**-complete, however, does not imply that they cannot be separated by a set in **P**.

- $I^A_P = \{(\phi_1, \phi_2, \Pi)\ ;\ Var(\phi_1) \cap Var(\phi_2) = \emptyset,$
  Π *is a* P*-refutation of* $\phi_1 \wedge \phi_2$, *and* ϕ_1 *is satisfiable*$\}$,

- $I^B_P = \{(\phi_1, \phi_2, \Pi)\ ;\ Var(\phi_1) \cap Var(\phi_2) = \emptyset,$
  Π *is a* P*-refutation of* $\phi_1 \wedge \phi_2$, *and* ϕ_2 *is satisfiable*$\}$,

where $Var(\phi)$ *denotes the set of variables present in* ϕ.
I will denote by $I_P := (I^A_P, I^B_P)$.

Informally, this concept captures the complexity of the problem of deciding which of the formulas ϕ_1 or ϕ_2 is satisfiable with the promise that one of them is. It is intended to formalize the question of whether the method of feasible interpolation applies to a given proof system. This method works as follows. Let X_1 and X_2 be two disjoint **NP** sets. We can formalize the sets by a sequences of propositional formulas ϕ^n_1 and ϕ^n_2, $n = 1, 2, \ldots$. Then their disjointness means that $\phi^n_1 \wedge \phi^n_2$ is unsatisfiable. If I^A_P and I^B_P can be separated by a set in **P**, but X_1 and X_2 cannot, then there cannot be polynomial size P-refutations of $\phi^n_1 \wedge \phi^n_2$.

Of course, we do not know how to prove that some disjoint **NP** pair is not polynomially separable, but often one show that refutations of suitable formulas imply separation by *monotone* Boolean circuits and then use known lower bounds on monotone Boolean circuits [26]. Furthermore, there are stronger monotone computation models which can be used to prove lower bounds using this method, one of which is monotone circuits that compute with real numbers [13, 18, 23]. My ultimate goal is to use combinatorial games, by which some interpolation pairs are characterized, as a sort of generalized monotone Boolean circuits to prove lower bounds for slightly stronger systems.

The two concepts defined above are closely related. For the Frege system, for example, the pairs are equivalent with respect to polynomial-time reductions. For bounded depth Frege systems, we have: $I_{\mathsf{F}_{d+1}}$ is equivalent to C_{F_d} for $d \geq 0$, see [4].

4. Games and circuits

The interpolation pairs, hence also canonical pairs, of F_d have been characterized using certain combinatorial games. To show how games can be used to characterize interpolation pairs, I will start with the simplest case, which is F_0 formalized as the Resolution refutation system.

A *basic game* is defined by

(1) directed acyclic graph with a single source,

(2) the non-leaf nodes are labeled by A and B,

(3) leaves are labeled by a and b.[4]

The game starts at the root and, at a node labeled A, Alice decides which arrow they will follow, and, at a node labeled B, it is Bob who makes the decision. We define that the leaves labeled by a are winning positions for Alice and those labeled by b are winning position for Bob.

The interpolation pair of F_1 is characterized by basic games in the following sense. Let G^A (resp. G^B) be the games in which Alice (Bob) has a winning strategy. Then the pair I_{F_1} is polynomially equivalent to the pair (G^A, G^B).[5].

In basic games, always one of the players has a winning strategy according to what is usually referred to as Zermelo's Theorem. This simple fact is proved by showing by induction that at each node v one of the players has a winning strategy when the games starts at v. The induction starts at the leaves and goes in the reverse order of the arrows. Note that this gives us a polynomial time algorithm to compute which player has a winning strategy. In fact, this algorithm can be presented as a circuit that computes with values a and b. To connect it with Boolean circuits, let $a := 0$ and $b := 1$. Then the circuit becomes a *monotone Boolean circuit* in which Alice's nodes are conjunctions and Bob's nodes are disjunctions. Clearly, the converse is also true: given a monotone circuit (with gates AND and OR) and an assignment to its inputs, we can interpret it as a basic game.

Game schemas. In order to identify basic games with monotone Boolean circuits, we should consider games in which leaves are left without labels and use strings of leaf labels as inputs. Let's call such structures *game schemas.* One can use this paradigm also for other games in which the winner is determined by a label on the terminal node. Thus we can view game schemas as *generalized circuits*. In all these cases, the monotonicity of the computed functions is an intrinsic property—if Alice has a winning strategy and we enlarge the set of leaves labeled with a, then Alice obviously still has a winning strategy.

Positional strategies. Let's furthermore observe that if a player has a winning strategy, then they also have a *positional* winning strategy, which means that the instruction how to play at a node v only depends on v, i.e., it is independent on the history of the play so far. The concept of a positional strategy is needed in characterizing the interpolation pairs of proof systems higher in the bounded depth Frege system hierar-

[4]I call the nodes with no outgoing arrows *leaves*, although the graphs are not trees in general.

[5]This is essentially the main result in [17], where it is stated for Boolean circuits instead of games; see also [23] for a simpler proof; furthermore it is a special case of the main theorem in [25].

chy. These characterizations use more complex games, in which the above fact is not true—there are games in which no player has a positional winning strategy.

5. Circuits and PLS

The class **TFNP** is a class of relations $R(x, y)$ that satisfy the condition that for some polynomial p, for every x, there exists y such that $|y| \leq p(|x|)$ and $R(x, y)$ holds true. The best way to imagine a **TFNP** problem is as a *search problem* where a solution is guaranteed and can be checked in polynomial time. Unlike the typical complexity classes such as **NP**, it seems that there are no complete problems in **TFNP**. In the seminal paper [15] several natural subclasses have been identified, which do have complete problems, and more classes have been defined later.

One of the classes introduced in [15] is **PLS**, Polynomial Local Search, which is connected with Boolean circuits. A problem in **PLS** is given by a directed acyclic graph with bounded outdegree and a single source node. The nodes of the graph are numbers and if there is an arrow from i to j, then $i < j$. Furthermore, there is a *feasibility predicate*, a property of the nodes. The source is feasible and for every feasible node which is not a leaf one of the successors is feasible. A solution is a feasible leaf. If the graph is given explicitly and the feasibility condition is computable in polynomial time, then the solution can be found in polynomial time. Therefore, to make the problem nontrivial, we assume that the graph is given *succinctly*. This means that the graph is of exponential size, but the successors and the feasibility condition of nodes are computable in polynomial time.

There are two more ways to make the problem nontrivial: using *communication complexity* or using *query complexity*, a.k.a. *black box model*. These two computational models are connected (see, e.g., [12]), but here, I will focus only on the communication complexity versions of the **TFNP** classes.

In the communication complexity version, instead of assuming that the graph is defined succinctly, we restrict the complexity of computing the primitives using bounds on communication complexity. So we will assume that there are two players, Alice and Bob, such that they are able to compute the feasibility of a node and the successors of the node by exchanging only a logarithmic number of bits. Thus, we get the class *communication-complexity*-**PLS**, which I will abbreviate by **ccPLS**. Another name for this kind of communication complexity is *DAG-like* communication complexity, which is used to differentiate it from the standard communication complexity model in which the possible communication paths form a tree.[6]

[6]DAG is an abbreviation for *directed acyclic graph*.

6. From ccPLS to games

In communication complexity we have two players like in basic games, but otherwise the setup seems to be very different from basic games. In particular

- in games the players compete, while in communication complexity they cooperate to find a feasible leaf;
- in communication complexity the players have their own strings that they use, while there is nothing like that in games;
- and what should the criterion for winning be in a game corresponding to the communication protocol?

The key is to focus on one player, say, Alice. Alice has an input string x and this completely determines her actions. Let S_x be the protocol that she follows and let us view it as a *strategy*. So x is a code of a strategy.

Concerning the winning positions, note that we can always make S_x a *winning* strategy. The trivial way is to label all leaves with a. More interestingly, there is a minimum assignment of a's that makes S_x a winning strategy. Such an assignment u_x is defined by putting a on those leaves that can be reached when Alice plays her strategy S_x and Bob tries all possible ways to play the game.

We can, certainly, do the same for Bob. Thus for each input y for Bob, we get his strategy T_y and the minimum assignment v_y to leaves that makes T_y winning. When Alice and Bob both play according to the communication protocol on the inputs x and y, they end up on a leaf that is labeled a in u_x and b in v_y.

Thus we naturally arrived at the problem that was used to characterize monotone circuit depth by Karchmer and Wigderson in [16], and circuit size by Razborov in [27]. In the setup of these papers, a monotone Boolean function f and two inputs x and y such that $f(x) = 0$ and $f(y) = 1$ are given. The goal of the players is to find an index such that $x_i = 0$ and $y_i = 1$. Let's call the problem they are solving *Karchmer-Wigderson problem*. In the theory of Boolean functions, the minimal weight strings for which a monotone function is zero are called *minterms*; the dual concept is called *maxterms*. Our u_x correspond to the minterms and v_y to the maxterms.

To get games (or circuits), we need to extend the communication graph. In the DAG-like communication protocol, the players decide collectively to which node to proceed from the current one by communicating $O(\log n)$ bits. From a node U, they can get to polynomially many nodes V_i. The problem is not that this is more than two, but the fact that the decision is not made by one player. We represent this communication protocol by a labeled tree T_U^N (N stands for "neighborhood"). The tree represents communication of the players in which always one player sends one bit, hence decides to which successor in T_U^N the computation proceeds. If the feasibility of nodes were computed by exchanging a single bit it would suffice to insert such a tree for every

node of the communication graph to get a basic game. In general the players need $O(\log n)$ bits to compute the feasibility predicate. Therefore the transformation of the graphs is more complex.

For each node U, there is also a $\log n$-depth communication protocol for the players to check the feasibility of U. Let T^F_U be the labeled tree representing this protocol and let L_U be its leaves. In the new DAG, node U will be replaced by elements of L_U, i.e., it will be split into $|L_U|$ nodes. The graph will be constructed as follows. The root V_0 is always feasible, so the players do not need to communicate, which means that the tree $T^F_{V_0}$ is a single point and it is also the unique leaf l_0. We attach $T^N_{V_0}$ to l_0. Let l be a leaf in $T^N_{V_0}$ and let U be the node on which the players have agreed when they reached l. Then we take T^F_U and attache it to l. In this way we connect l to the nodes that represent U. In general, if V is not a root and is represented by several nodes, i.e., $|L_V| > 1$, we do the same for each node in L_V. In this way, we get a communication protocol in which the decision to which node to continue is always made by one player and the feasibility predicate can be checked by exchanging a single bit. The latter fact requires an argument, which I leave to the reader with the following hint. The feasibility predicate at node $k \in L_u$ is

$$\bigwedge_{e \in E(x)} P_e(x) \wedge \bigwedge_{h \in H(y)} P_h(y),$$

where $E(x)$ are the edges decided by Alice on the branch in T^F_U leading to k, $H(y)$ are the edges decided by Bob on the same branch, and $P_e(x)$ is the predicate saying that Alice will use e, while $P_f(y)$ is the predicate saying that Bob will use f.

Since we have obtained a DAG with nodes assigned to players, we can interpret it as a game schema G, but this game schema may not relate to the problem the communication protocol solved. Suppose the protocol solved the Karchmer-Wigderson problem for a monotone function f. Let $X := \{x | f(x) = 1\}$ and $Y := \{y | f(y) = 0\}$. Assume X are inputs given to Alice, and Y are inputs to Bob. Let α_x, for $x \in X$, be Alice's strategies for playing with given x, and β_y, for $y \in Y$, be Bob's strategies for playing with y. The fact that the communication game solved the Karchmer-Wigderson problem ensures that for $x \in X$, Alice's strategy α_x beats all Bob's strategies β_y, $y \in Y$, in game (G, x) (G with the winning positions determined by 1s in the vector x). But this does not imply that α_x is a winning strategy in game (G, x) because Bob may play a strategy that is not among the strategies β_y. Nevertheless, it suffices to omit some arrows and nodes from G to get a game schema G' such that α_x is a winning strategy in (G', x). This may look surprising because in G', Bob is not forced to play one of the strategies β_y. The reason why it works is a property of the feasibility predicates. When we extend the communication protocol by trees, the players only need to send one bit to each other in order to check the feasibility predicate $F_v(x, y)$

at a given node v. This implies that the predicate is a conjunction of two predicates $F_v^A(x) \wedge F_v^B(y)$, which can be computed separately by Alice and Bob. Consequently, if x, x' are two inputs for Alice and y, y' for Bob, we have

$$F_v(x,y) \wedge F_v(x',y') \;\Rightarrow F_v(x,y') \wedge F_v(x',y). \tag{6.1}$$

In terms of strategies, $F_v(x,y)$ is equivalent to the condition that the path determined by strategies α_x and β_y leads to a leaf u that is labeled a in x and b in y.

Lemma 1. *Let G' be the game schema that results from G by omitting all arrows that do not occur on any path defined by a pair of strategies α_x, β_y for $x \in X, y \in Y$.*[7] *Then α_x is a winning strategy in (G', x) for every $x \in X$, and β_y is a winning strategy in (G, y) for every $y \in Y$.*

Proof. Because of symmetry, it suffices to consider $x \in X$. I will show that every node u in G' satisfies the following condition.

- ⋆ If u is feasible for (x, y), then u is a winning position for Alice in (G, x) with α_x being a winning strategy.

The proof is by induction, starting from the leaves and going towards the root.

Let u be a (x, y)-feasible leaf for some $y \in Y$. Since the communication protocol solves the Karchmer-Wigderson problem for f, the value of x on u must be a. Hence u is a winning position in (G, x).

Let u be a non-leaf. Suppose u belongs to Alice and let v be the node to which Alice will go playing α_x. If u is (x, y)-feasible, then so is v. By the induction assumption, v is a winning position for Alice with the winning strategy α_x, so the same holds for u.

Let u belong to Bob and v, w be its successors in G'. Suppose u is (x, y)-feasible. Then either v or w is (x, y)-feasible; let it be v. Since the arrow $u \to w$ has not been deleted, there is a path defined by some pair $\alpha_{x'}, \beta_{y'}$ that has the arrow $u \to w$. By property (6.1), u is also (x, y')-feasible. Since strategy $\beta_{y'}$ points to w, w is (x, y')-feasible. Hence, by the induction assumption, both v and w are winning positions for Alice, and, consequently, v is also a winning position.

I leave the case when u belongs to Bob and has only one successor to the reader. ∎

What I have done is a proof of Razborov's result, mentioned before, presented in terms of games.

[7] To formally satisfy the definition of a basic game, we should also delete all nodes that become not reachable from the root.

Theorem 1 (Razborov, [27]). *The minimum size of a monotone circuit computing a monotone Boolean function f is equal, up to a polynomial factor, to a minimum size of graph in a DAG-like communication protocol solving the Karchmer Wigderson problem for f.*

7. More players

I will now digress to show a potential application of a generalization of basic games to more players. In communication complexity, protocols with more players are considered, called *multiparty communication complexity*. There are two main versions called *number in hand* and *number on the forehead*. In both versions, the input is split into k parts, where k is the number of players. In the number-in-hand model each player sees only one part that is assigned to them. In the number-on-the-forehead each player sees all parts except theirs.

We can study both versions using games. For the sake of simplicity I will consider the number-in-hand model with three players, Alice, Bob, and Cindy. The generalization of the basic game to three players is straightforward. The inner nodes are now labeled by A, B and C, the leaves by a, b and c. The corresponding game schema is without labels on the leaves.

With more than two players, Zermelo's theorem does not hold true. Nevertheless, we can still compute who has a winning strategy, if there is a player who has such a strategy. To this end, we will need four values a, b, c, and 0, where 0 denotes that at the node nobody has a winning strategy. Thus we can define circuits corresponding to these game schemas. The following proposition characterizes the problems that can be studied using this concept.

Proposition 1. *Let $W_A, W_B, W_C \subseteq \{a, b, c\}^n$. Then there exists a game schema G such that Alice has a winning strategy when we label the leaves with W_A, Bob has a winning strategy when we label the leaves with W_B, and Cindy has a winning strategy when we label the leaves with W_C, if and only if the following condition holds true:*

(Δ) *for all $\alpha \in W_A, \beta \in W_B, \gamma \in W_C$, there exists $i \in [n]$ such that*

$$\alpha(i) = a \ \wedge \ \beta(i) = b \ \wedge \ \gamma(i) = c. \tag{7.1}$$

Clearly, if $\alpha \in W_A$ (similarly if $\beta \in W_B$ or $\gamma \in W_C$), then only the values on indices i such that $\alpha(i) = a$ matter. Therefore it is more natural to use *sets* of indices instead of vectors in $\{a, b, c\}^n$.

A natural generalization of the Karchmer-Wigderson problem is, for given strings $\alpha \in W_A, \beta \in W_B, \gamma \in W_C$, find an index i such that (7.1) holds. Then we can also

generalize the Karchmer-Wigderson-Razborov theorem as follows (not only for three players).

Theorem 2. *If a number-in-hand DAG-like communication protocol P solves the Karchmer-Wigderson problem for sets $W_A, W_B, W_C \subseteq \{a,b,c\}^n$ satisfying (Δ), then one can construct in polynomial time a game schema G for three players A, B, C such that A, resp. B, resp. C, has a winning strategy in (G, W^A), resp. (G, W^B), resp. (G, W^C).*

To get Boolean circuits, we can represent the game schema G by a triple of circuits C^A, C^B, C^C, where in C^A the nodes of A are labeled by $\vee$ and the nodes of B and C are labeled by $\wedge$, and the circuits C^B and C^C are defined accordingly. Pavel Hrubeš discovered this connection of communication protocols and triples (in general k-tuples) of circuits many years ago, but he did not write it up.

Intuitively, one should get larger lower bounds when there are more players, because the players know less about the input and hence must communicate more. So far this idea has been used only in some special cases.

Problem 1. *Prove, for explicitly given sets W_A, W_B, W_C, a larger lower bound on the size of these circuits than we have for monotone Boolean circuits!*

Example. In this example we will have k players. Let $n := 2^k + 1$. The input vectors will be edge colorings by k colors of the complete graph on n vertices. For $j = 1, \ldots, k$, the winning set of the jth player W_j is the set of colorings such that the graph defined by the edges with color j contains two cliques that cover all vertices. One can easily see that these sets satisfy the condition from Proposition 1 above, namely, for any string of colorings $(\kappa_1, \ldots, \kappa_k)$, $\kappa_1 \in W_1, \ldots, \kappa_k \in W_k$, there exists an edge e such that $\kappa_1(e) = 1, \ldots, \kappa_k(e) = k$. By the generalization of Proposition 1, there exists a game in which sets W_i are winning set of players i for all i. Proving a lower bound on the size of such games maybe a way to get better lower bounds for the well-known tautology called *BitPHP*. And indeed, a few days before the deadline for this paper, I received a draft by P. Beame and M. Whitmeyer that uses multiparty-communication complexity to prove better bounds on $BitPHP$ for tree-like cutting-planes proofs [3].[8] To prove such a bound for DAG-like Resolution proofs is still an open problem.

Lower bounds on the number-on-forehead multiparty communication complexity are notoriously hard, but could resolve important open problems both in circuit

[8]The cutting-planes proof system is stronger than Resolution, but incomparable with higher fragments of constant depth Frege systems. Tree-like versions of these systems are weaker than the DAG-like ones.

complexity and proof complexity. Therefore it would be more interesting to prove lower bounds on the size of the corresponding games. In game theory, the number on forehead model corresponds to the same games except that instead of asking about winning strategies of single players, we ask about winning strategies of coalitions of all players except of one.[9]

8. The interpolation pair of F_1

Recall that F_1 is a generalization of Resolution in which we allow conjunctions in the clauses. Furthermore, the interpolation pair of F_1 is equivalent to the canonical pair of Resolution. The first combinatorial characterization of this pair appeared in [4] and was based on the *point-line game* introduced in that paper.

The setup of a point line game consists of

- the board, which is a directed acyclic graph with one source, and all nodes of outdegree 2, except for leaves;
- the nodes of the graph are containers with slots – *the points*;
- the source node is empty, each leaf node has one point;
- if $U \to V$ are nodes connected by an arrow, then there is a partial mapping from the points in V to the points in U – *the lines*;
- there are two players, Red and Blue, and each node is owned by one player and marked by the color of the owner.

The game is played as follows:

- starting at the source, the players follow the arrows and the color of the node decides who will choose the arrow;
- they move red and blue pebbles along the lines;
- when they move from a red node U to some node V and there are points in V that are not connected to points in U, they will be covered by blue pebbles; similarly for a blue node U, the free points in V will be covered by red pebbles; thus all points in the currently used nodes are filled with pebbles;
- the player whose pebble appears in the leaf wins.

A *positional strategy* of the Red player is a set of arrows, one for each red node. Similarly for the Blue player.

[9]Note that a player i does not have a winning strategy if and only if the coalition of all players j, $j \neq i$, has a winning strategy. So these problems are, in a sense, dual to each other.

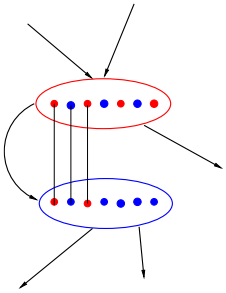

Figure 1. Red decided to move to the blue node. The first three pebbles are moved along the lines, the rest of the bottom node is covered by blue pebbles.

Lemma 2. *Given a set of arrows S, it is decidable in polynomial time whether S is a winning positional strategy of the Red player. Hence the set of games with winning positional strategies for the Red player is in* ***NP****. The same for the Blue player.*

Theorem 3 (Beckmann, P., Thapen, [4]). *Let S^R (respectively, S^B) be the set of games in which the Red (Blue) player has positional winning strategies. Then the interpolation pair of F_1 and (S^R, S^B) are polynomially reducible to each other.*

The condition that the strategies must be positional seems to be ad hoc in order to get **NP** pairs, however it naturally appears in the proof of the theorem and we will see another reason for it in a moment.

The point-line game without pebbles. It is important to realize that the point-line game can be played without pebbles. The difference is that such a game does not end at a leaf that the players reach, because they still have to determine the winner. To this end the players go back along the path by which they arrived at the leaf and follow the predecessors of the unique point on the leaf. Eventually they reach a node such that the point does not have a predecessor in the preceding node. This is the place where the color of the pebble would be decided if they played with pebbles. If the preceding node is red the color of the pebble would be blue, and vice versa. This pebble would go all the way to the leaf.

Note that in such a game the players have to record the path by which they arrived to a leaf. If they use pebbles, they do not have to do that because the information that they need is encoded by the color of pebbles.

The point-line game schema. To use colors of certain nodes as inputs in order to compute a partial Boolean function is rather awkward and it would be difficult to analyze such a computational model. A more natural way is to allow points in the source node, and use them to put different strings of colored pebbles. The simulation

of the original games by this modification is easy and I leave it to the reader. So *the point-line game schema* is like the point-line game except that we allow points in the source node and these points are used as placeholders for input bits represented by pebbles of two colors.

9. The point-line game and the class CPLS

The class *Colored Polynomial Local Search,* **CPLS**, is an extension of **PLS** introduced by Krajíček, Skelley and Thapen in [22]. This class is connected with the point line game in the same way as **PLS** is connected the basic game. To describe this connection I will present **CPLS** as a point-line game played by *one player* on an *exponentially large board* and *without pebbles*. The goal of the game is to find a point without a predecessor in a predecessor node. In more detail:

The setup is

- the nodes and the points are given by numbers $\leq 2^n$ (n is the input size parameter); arrows go from a smaller number to a larger ones; node 0 has no point; each leaf has a unique point;
- the arrows, lines and points are given by polynomial time algorithms as follows:
  - (1) for a node U, an algorithm computes the successors of U, or outputs *U is a leaf,*
  - (2) for two nodes $U \to V$ and a point $a \in V$, an algorithm computes the predecessor of a in U, if there is any, otherwise outputs *"no predecessor in U",*
  - (3) for a leaf of the graph S, an algorithm computes the unique point in S.

The goal is:

- to find a triple $U \to V$, $a \in V$ such that a has no predecessor in U.

These triples are, moreover, associated with some elements, which are outputs. The outputs are not relevant for what I want to do because I want only to transform the problem to a game schema.

To transform **CPLS** into a communication complexity class, we proceed in the same way as in the case of **PLS**; i.e., we use a polynomial domain instead of the exponential one, and let the two players compute the arrows, lines, and points using small communication protocols.

The basic idea of the transformation of **ccCPLS** into a game is the same as for **ccPLS**; i.e., we interpret the protocols for players as their winning strategies. The transformation of the graphs of the communication protocol into a graph of the game is, however, more complicated because we have, in fact, two graphs: one whose vertices are, what we call, nodes and another whose vertices are points and edges are lines.

The graph of nodes can be treated the same way as we did with **ccPLS**. Thus for every node U, we replace the outgoing arrows with a tree T_U that represents possible communication paths to determine the successor.

Concerning the points and lines, it is more difficult to simulate communication by playing a point-line game. The problem is that, for two nodes $U \to V$ and a point $a \in V$, we need to compute a neighbor of a in U, and this is in the opposite direction. In the point-line game without pebbles, players also go in the opposite direction, but they just follow the line and do not make any decisions. Therefore, I will consider a generalization of the game in which, for every $a \in V$, there is a basic game G_a with the root at a and leaves being some points of U. Then the simulation is easy: use the the communication trees as these basic games.

Such a generalization differs from point-line games considerably. We can interpret the nodes of G_a as points in V, but then we will have lines inside of V. To make it more alike the original game, we can do the following. Let us call the nodes of the basic games G_a points and arrows lines. Insert a chain of nodes between U and V, $V_0 = U, V_1, \ldots, V_{k-1}, V_k = V$, where k is the maximal depth of the trees that are the graphs of G_a, and let the arrows only be $U \to V_1 \to \cdots \to V_{k-1} \to V$. Then put the points of all G_a into the nodes of the chain so that the lines are only between the consecutive nodes. One may need to add additional points and lines when the branches are shorter than k. Then, it remains to deal with the markings that associate players with points in G_a. We can assume, w.l.o.g, that in all G_a the same player starts and they alternate regularly. So, we can assign colors to nodes V_i so that they correspond to the colors of the points in these games. As a result, we can remove the markings of the points.

To complete the full circle from point-line games, to **CPLS**, to communication complexity **CPLS**, and back to point-line games, one would need to go on and prove a result similar to Razborov's Theorem 1. I believe it is possible, but for now, I leave it for future research.

Remark 1. One can check that such games are essentially the same as the Turing machine games of depth 2 that I will define in the next section. Since Turing machine games of depth 2 characterize the same interpolation pair as point-line Games, they are polynomially reducible to each other. Thus we get point-line games from **ccCPLS**.

Remark 2. One thing to note is that in this way we necessarily get *positional* strategies, which justifies the condition that we imposed on strategies when defining the point-line game. Being positional in general means that the strategy only uses "local" information. The locality is an intrinsic property of the communication complexity versions of **TFNP** problems. Therefore, we always get positional strategies from **TFNP** problems.

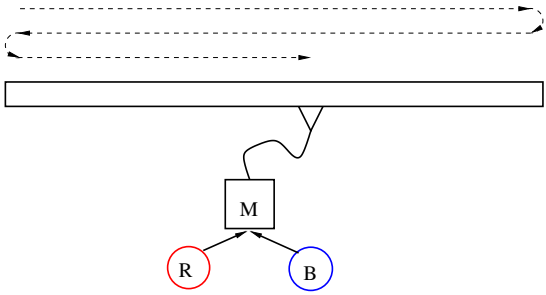

Figure 2. Turing machine M, controlled by players R and B, in the middle of the 3rd round moves to the right.

10. Games for all levels of Bounded Depth Frege Hierarchy

I will present a characterization of the interpolation pairs of F_d for all d from [25]. It is based on games, which I called simply depth-d games. Here I will call them *Turing machine games of depth d*.

Definition 3. *A* Turing machine game of depth d *is given by* (M, n, A, R, B)*, where*

- M *is a nondeterministic Turing machine,*
- n *is the length of a finite tape,*
- A *is a finite, but possibly large, alphabet used by the machine,*
- (R, B) *is a partition of* A *into winning positions for Red and Blue players respectively.*

The game is played as follows:

- *initially the tape is blank,*
- *starting at the leftmost square the machine moves forth and back from one end to the other d times while reading and rewriting the symbols on the tape;* [10]
- *two players, Red and Blue, alternate in deciding which action the nondeterministic machine takes (i.e., what it prints on the tape and which state the automaton takes on),*
- *the last printed symbol decides who is the winner.*

Admittedly, the machine in the definition is not what we usually call a Turing machine because in the standard definition the size of the alphabet and the number of states of the automaton are bounded by a constant. It is possible to slightly change the definition to achieve the constant bound, but the resulting concept would not be natural.

[10] A machine which moves its head independently of the content of the tape and its state is called *oblivious*, so the machine in the definition is oblivious.

Definition 4. *The* size of a TM-game *is the number of states of* $M + |A| + n$. *A* positional strategy *for a TM game is a strategy that only uses the number of the round, the state of* M*, and the symbol on the tape read by* M.

Lemma 3 ([25]). *For every* $d \geq 1$*, there exists a polynomial time algorithm (polynomial in the size of* M*) to decide if a given positional strategy is a winning strategy.*

Hence the existence of a positional strategy for given player is an **NP** predicate. Unfortunately, my proof of this lemma only gives algorithms in which the exponent of the polynomial bound grows with increasing d. Thus I am not able to use these games to characterize the interpolation pair of the unbounded depth Frege system.

Let W_d^R (W_d^B) denote the set of depth-d TM-games with winning positional strategies for the Red (respectively, Blue) player. The characterization of the interpolation pairs is given by the following theorem.

Theorem 4 ([25]). *For all* $d \geq 1$*, the interpolation pairs of* F_d *(depth-d Frege System) and* (W_d^R, W_d^B) *are polynomially reducible to each other.*

It is not difficult to see that depth-1 TM-game is the basic game: given the machine, the nodes of the basic game are triples (x, s, i), where i is the position of the head, x is the symbol on the ith square, and s is the state. To reduce the point-line game to the depth 2 TM game is also easy if we take the version with all pebbles black. The machine going in one direction follows the arrows of the graph; going in the opposite direction follows the lines. The converse reduction can be done using the generalized version of Section 9.

The paradigm used in the proof of Theorem 4 is the search of a falsified clause in a Resolution refutation when the variables are split between two players. In such a situation, always one of the players can control the step in the search procedure because the assumptions of the resolution rule contain only one variable which is not in the conclusion. So the player who knows the value of this variable can decide which of the assumptions is false assuming the conclusion is false. For proofs of depth $d > 0$, I use the key idea of Skelley and Thapen [28] that one can define a calculus which is equivalent to F_d, and in which the only essential rule is the resolution rule. Such a calculus cannot be the standard sequent calculus with cuts restricted to propositional variables because this restriction would make it very weak. What they actually did was to use *deep inference rules*. These rules allow one to derive a new formula by changing a given formula inside; in other words, the rules can be applied *inside of formulas*.

In my proof I use a different calculus with two deep inference rules that I call resolution, which is the usual resolution rule applied inside of formulas, and *dual*

resolution. Both rules can be applied inside of formulas:

$$\frac{(A \vee p) \wedge (B \vee \neg p)}{A \vee B}, \qquad \frac{A \wedge B}{(A \wedge p) \vee (B \wedge \neg p)}.$$

The Turing machine, roughly speaking, searches for unsatisfied formulas of the highest depth d, which is the depth of the proof. Then it reverses the order of search and searches their unsatisfied subformulas of depth $d - 1$ and so on until it finds an unsatisfied initial clause. When the TM goes in one direction, it uses instances of the resolution rule to search for an unsatisfied formula, the same way as it is done when we use interpolation for Resolution; when it goes in the opposite direction, it uses dual resolution.

11. Some related combinatorial games

Various games have been studied in game theory. Here I want to mention only two that are related to the games presented in this article. In contrast to our games, these games are played on directed graphs that are in general not acyclic; thus the number of rounds is infinite.

Parity Games. A parity game is given by a directed graph whose set of nodes is $\{0, 1, 2, \ldots, n\}$, and for each node, there is at least one outgoing edge. The set of nodes is divided into two parts, V_0 and V_1, which we associate with two players, Player 0 and Player 1. W.l.o.g., we can assume that V_0 are even numbers and V_1 are odd. A play starts with a pebble on 0 and then players move the pebble along the arrows of the graph. When the pebble is on a node in V_i, then Player i moves the pebble. After they play infinitely many rounds, the winner is decided by the least node v that was visited infinitely many times: if $v \in V_i$, then Player i wins.

E. A. Emerson showed that in a parity game, always one of the players has a *positional* winning strategy [11]. In such a strategy, for every node, the player always uses the same arrow. This can be used to define the games in which Player i has a winning strategy by an **NP** condition. In 2017, C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan found a quasi-polynomial algorithm to decide who has a winning strategy in parity games [7].

Simple Stochastic Games. These games were introduced by A. Condon [9] as a special case of stochastic games studied before. The game is given by a directed graph whose nodes are divided into three sets $V_{min}, V_{max}, V_{ave}$ and with three distinguished nodes s, v_0, v_1. Starting at node s, players Min and Max move the pebble if it is at a node assigned to them. If the pebble is on a node $v \in V_{ave}$, the pebble is moved to a child of v randomly. We assume that the outdegree of every node is at least 1, except

for the sinks v_0, v_1. The aim of Min (resp. Max) is to reach v_0 (resp. v_1). Given a pair of positional strategies σ for Min and τ for Max, we can compute the probability that they reach v_1. The *value of the game* is inf sup of this probabilities over all σ, τ. The decision problem for simple stochastic games is to determine whether the value is $> 1/2$.

Condon [9] proved that the decision problem for simple stochastic games is in **NP**∩**coNP**. In 2011, K. Chatterjee and N. Fijalkow strengthened this result to **UP**∩**coUP** [8]. No truly subexponential algorithms are known.

In [4], we showed that the decision problem for parity games can be reduced to the interpolation pair of F_1, hence also to point-line games and Turing machine games of depth 2. Strengthening the results of A. Atserias and E. Maneva [1] for mean-payoff games,[11] L. Huang and T. Pitassi [14] proved that the decision problem for simple stochastic games is reducible to the interpolation pair of F_2. In [4], we gave a more direct proof of this result. We observed that our proof does not use the full power of F_2, but we were not able to improve it to F_1. Below, we restate these reductions in terms of Turing machine games.

Theorem 5. *The decision problem for parity games is polynomially reducible to deciding who has a* positional *winning strategy in depth 2 Turing machine games. The decision problem for simple stochastic games is polynomially reducible to deciding who has a* positional *winning strategy in depth 3 Turing machine games.*

12. Conclusions and open problems

In this paper, I have focused on the relation of games and communication complexity versions of two **TFNP** classes. These games characterize the interpolation pairs of F_0 and F_1 and the **TFNP** classes that correspond to theories T^1_2 and T^2_2. It seems that this connection can be extended to the interpolation pairs of all F_i and the game-induction principles GI_i introduced in [28] that are associated with theories T^i_2. This raises the natural question of whether one can get in such a way a game from any **TFNP** class, for which the communication complexity version makes good sense. The answer is that one always gets *some* game, but it may not be what we will expect. In the two examples above, **ccPLS** and **ccCPLS**, we were able to incorporate the small trees representing the communications at nodes into the original graph. Adding these trees may essentially distort the principle for **TFNP** classes that are not based on DAGs.

There are more connections between proof systems and various computational models. In particular, the black-box (a.k.a. query complexity) versions of **TFNP** have

[11] I will not define these games here.

been associated with all important proof systems studied in proof complexity, and for several proof systems also classes of generalized monotone circuits have been found [6, 12, 20]. An approach to feasible interpolation for bounded depth Frege system, different from the one here, was proposed in [19]. This suggests a lot of questions, but I confine myself to a few relevant to this paper's topic.

(1) What is the computational complexity of deciding who has a positional winning strategy in a given depth d TM-game, assuming that one player has a positional winning strategy?

(2) For $d > 1$, prove a lower bound on the complexity of depth d TM-games computing an explicitly given partial *monotone* Boolean function.

(3) In particular, for some $d > 1$ (ideally for all), prove that (W^R_{d+1}, W^B_{d+1}) cannot be polynomially reduced to (W^R_d, W^B_d) by a *monotone* projection.

(4) Characterize the canonical pair of the unbounded depth Frege system.

Furthermore, I presented connections between games and communication complexity rather informally. My goal was to show that games offer an alternative view of open problems in proof complexity, but more precise definitions and proofs are needed, which I will do in subsequent publications.

Acknowledgments. I would like to thank Emil Jeřábek, Jan Krajíček, and Neil Thapen for their comments on a draft of this paper.

Funding. This work was supported by the Czech Academy of Science (RVO 67985840) and GAČR grant 23-04825S.

References

[1] A. Atserias and E. Maneva, Mean-payoff games and propositional proofs. Inform. and Comput. **209(4)**, (2011), 664–691.

[2] A. Atserias, M. Müller: *Automating Resolution is NP-Hard.* Proc. 60th Annual IEEE Symp. on Foundations of Computer Science, (2019), 498-509.

[3] P. Beame and M. Whitmeyer, Multiparty communication complexity of collision-finding. arXiv (2024), http://arxiv.org/abs/2411.07400v1.

[4] A. Beckmann, P. Pudlák, N. Thapen, Parity games and propositional proofs. *ACM Transaction on Computational Logic,* **15(2)** (2014), article 17.

[5] M. L. Bonet, T. Pitassi and R. Raz, On Interpolation and Automatization for Frege Systems. *SIAM Journal on Computing* **29(6)** (2000), 1939–67.

[6] S. Buss, N. Fleming, and R. Impagliazzo, TFNP Characterizations of Proof Systems and Monotone Circuits. In: *Proc. Innovations in Theoretical Computer Science (ITCS),* LIPIcs volume 251, 30:1-30, (2023).

[7] C. S. Calude, S. Jain, B. Khoussainov, W. Li, F. Stephan, Deciding Parity Games in Quasi-polynomial Time. SIAM J. Comput. **51(2)**, (2022), 17-152.

[8] K. Chatterjee and N. Fijalkow, A reduction from parity games to simple stochastic games. GandALF 2011: 74-86.

[9] A. Condon, On algorithms for simple stochastic games. In Advances in computational complexity theory. DIMACS Ser. Discrete Math. Theoret. Comput. Sci. **13**. Amer. Math. Soc., Providence, RI, (1993), 51-–71.

[10] S. A. Cook, Feasibly constructive proofs and the propositional calculus. In *Proc. seventh annual ACM symposium on Theory of computing*, ACM New York, (1975), 83–97.

[11] E. A. Emerson. 1985. Automata, tableaux, and temporal logics. In *Logics of programs* (Brooklyn, N.Y., 1985). Lecture Notes in Comput. Sci. **193**. Springer, Berlin, 79–88.

[12] M. Göös, P. Kamath, R. Robere, and D. Sokolov, Adventures in Monotone Complexity and TFNP. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019).* Leibniz International Proceedings in Informatics (LIPIcs), **124**, (2019) 38:1–38:19.

[13] P. Hrubeš and P. Pudlák, A note on monotone real circuits. *Information Processing Letters* **131** (2018), 15–19.

[14] L. Huang and T. Pitassi, Automatizability and simple stochastic games. In *Automata, languages and programming. Part I.* Lecture Notes in Comput. Sci. **6755**. Springer, 2011, 605–617.

[15] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, How Easy is Local Search? *Journal of Computer System Sciences* **37** (1988), 79–100.

[16] M. Karchmer, A. Wigderson, Monotone Circuits for Connectivity Require Super-Logarithmic Depth. *SIAM J. Discret. Math.* **3(2)** (1990), 255–265.

[17] J. Krajíček, Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic", *J. of Symbolic Logic,* **62(2)**, (1997), pp.457–486.

[18] J. Krajíček, Interpolation by a game, *Mathematical Logic Quarterly*, **44(4)**, (1998), pp.450-458.

[19] J. Krajíček, A form of feasible interpolation for constant depth Frege systems. *J. of Symbolic Logic,* **75(2)**, (2010), 774-784.

[20] J. Krajíček, Randomized feasible interpolation and monotone circuits with a local oracle. *J. of Mathematical Logic,* **18(2)**, 1850012 (2018).

[21] J. Krajíček, P. Pudlák, Some consequences of cryptographical conjectures for S^1_2 and EF. In *Logic and Computational Complexity, (proceedings of meeting held in Indianapolis 1994),* Ed. D. Leivant, Springer LNCS 960 (1995), 210–220.

[22] J. Krajíček, A. Skelley and N. Thapen, NP search problems in low fragments of bounded arithmetic. *Journal of Symbolic Logic* **72(2)** (2007), 649–672.

[23] P. Pudlák, Lower bounds for resolution and cutting planes proofs and monotone computations. *J. of Symb. Logic* **62(3)** (1997), 981–998.

[24] P. Pudlák, On reducibility and symmetry of disjoint NP-pairs. *Theor. Comput. Science* **295** (2003), 323–339

[25] P. Pudlák, The canonical pairs of bounded depth Frege systems. *Annals of Pure and Applied Logic* **172(2)** (2021), Article 102892.

[26] A. A. Razborov, Lower bounds for the monotone complexity of some Boolean functions. *Soviet. Math. Dokl.,* **31** (1985), 354-–357.

[27] A. A. Razborov, Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya: Mathematics,* **59(1)**, (1995), 205-–227.

[28] A. Skelley and N. Thapen, The provably total search problems of bounded arithmetic. *Proc. London Math. Soc*, **103(1)** (2011), 106–138.

Pavel Pudlák
Institute of Mathematics, Czech Academy of Sciences, Žitná 25, 11567 Prague, Czech Republic; pudlak@math.cas.cz

Dear Author,

This is not part of your paper, it serves for your checking that all your data is correct. Please check the accuracy of each field and kindly provide the missing ones (if they apply).
In particular, please consider the following points:

- Are first and last names entered properly? Are there further names or initials missing?
- If applicable, please provide your Mathematical Reviews ID from MathSciNet and your ORCID. (The MR ID can be checked even without MatSciNet access in three easy steps:
  (1) Copy the bibliographic data of any published paper (co-)authored by you in the search field at https://mathscinet.ams.org/mathscinet/freetools/mref
  (2) Click your name in the search result
  (3) Find your MR Author ID in the first row.)
- Please check your institutional affiliations (department and institution) and use their official titles.
- Are there any parts missing from the addresses, like postal code, PO Box, street names, etc.?
- Please provide email addresses in lowercase characters. If you provided a non-institutional email address like Gmail, consider also adding your institutional one.

In addition, please also check if there is any funding or other information that you would like to include.

Thankfully, the EMS Press team

Personal data (Author 1)	
given name(s)	Pavel
surname	Pudlák
MR ID	142555
ORCID	0000-0002-2696-070X
Affiliation 1 of Author 1	
department	Institute of Mathematics
organisation	Czech Academy of Sciences
ROR ID	02tv1yf50
street address or PO Box	Žitná 25
zip code	11567
city	Prague
country	Czech Republic
email	pudlak@math.cas.cz