Proofs as games

Pavel Pudlák *

September 2, 1999

1 Introduction

Proving lower bounds on the complexity of languages and boolean functions is the hardest problem in complexity theory. We can prove lower bounds only in very special cases and the famous open problems, represented by the most important among them $\mathcal{P} = \mathcal{NP}$?, remain open. A closely related area in logic is concerned with proving lower bounds on the lengths of propositional proofs. There the situation is similar in that we can prove exponential lower bounds only for restricted proof systems. It is a hard open problem to prove nontrivial lower bounds on the lengths of proofs in the usual proof systems based on axiom schemas and rules.

In this paper I would like to show, using an example, how one can think of lower bounds on the lengths of proofs in terms of a two player game. In communicating and competing with other people we are unconsciously playing games every day. Our brain is well adapted to it. Thus presenting a mathematical problem in terms of a game often helps our intuition.

The idea of using games is not new. In computational complexity such an approach is known as *'adversary argument'* and it has been used many times. It is well know that it can potentially be used also in proof complexity. For the usual proof systems it has been shown that the length of the shortest proof is proportional to the logarithm of the length of a game where the adversary pretends that a given tautology is not true and the prover tries to find a contradiction in his answers [5]. This game can be viewed as a law suit where the defendant (= the adversary in the explanation above) tries to save himself by possibly using also lies. The other party, say an attorney who knows that the defendant is not

*Mathematical Institute, AV ČR, Žitná 25, 115 67 Praha 1, Czech Republic; e-mail: pudlak@math.cas.cz; partially supported by grant A1019901 of the Academy of Sciences of the Czech Republic and by a cooperative research grant INT-9600919/ME-103 from the NSF (USA) and the MŠMT (Czech Republic)

innocent, tries to pose questions in a way that leads quickly to an obvious contradiction in the testimony. A jury or a judge would not be impressed by a long proof, they need a simple contradiction.

Here I will consider *propositional resolution proofs*. These proofs are much simpler as they use only very special formulas. Their simplicity is an advantage for computer applications. Propositional resolution system is a part of a more general *resolution system* which was proposed for automated theorem proving by J.A. Robinson [6] and since then it has been a core of most automated theorem provers. The first exponential lower bound for propositional resolution proofs was proved by Armin Haken [2] in 1985. My main aim here is to present his proof in terms of a game.

2 A game

The tautology that Haken used and which we shall also consider is the simple statement that there cannot be a one-to-one mapping from an $n+1$-element set into an n-element set. This is known as the *Pigeon Hole Principle* or simply *PHP*, where we think of the $n+1$-element set as a set of pigeons and the n-element set as the set of holes. The game will be a restricted form of the game described above.

Let us define the game. There are two parties, *Prover* and *Adversary*. Adversary pretends that there is a one-to-one assignment of pigeons to holes, but of course, he has to cheat, while Prover is trying to convict him of lying. (Clearly, pretending that there is a one-to-one mapping from an $n+1$-element set (funds) into an n-element set (bills) is, perhaps, the best way of covering an embezzlement.) The communication proceeds by alternatively Prover asking a question and Adversary answering it. The prover may ask only questions of the form *'Does pigeon Alois go to hole Brno?'* and Adversary has to answer *'yes'* or *'no'*. In the first case we say that hole *Brno* has been assigned to pigeon *Alois,* in the second case we say that it has been prohibited. Prover keeps record of the answers, but he does not have to remember everything, he can erase some answers. Thus his record is an arbitrary set of triples of the form (*Attila,Budapest,yes*) and (*Alvarés,Barcelona,no*). Adversary watches Prover's record and thus, for example, he can give the opposite answer to a question that he has been asked before without being caught, because he sees that his previous answer has been erased. The game ends when Prover's record becomes *obviously* contradictory with the assumption that there is a one-to-one assignment of pigeons to holes. This has to be specified as, clearly, the assumption that there is a one-to-one mapping is contradictory. Precisely it means that according to his record either there have been opposite answers to the same question, or two pigeons have been assigned to the same hole, or for some pigeon all holes have been prohibited. Thus Adversary can cheat as long as Prover does not notice it, but when Prover has an evidence that Adversary cheats, the game ends.

The point of the game is not who wins, as Prover can always win by asking all possible questions and remembering all answers. The question is how much effort Prover must use in order to win. The most natural parameter used in such games is the number of rounds. This can be estimated easily, but it gives little information about resolution proofs. Our parameter will be space, ie., how much memory Prover needs. This is a more subtle parameter, since, by definition, Prover can record only a very special type of information bits. In order to define it precisely we have to talk about strategies for Prover. A strategy is a set of rules for Prover which determines completely his moves. In each round his move, ie., which part of the record he erases and the next question that he asks, is uniquely determined by his record and the answer of Adversary. We are only interested in a special kind of strategies, namely, *winning strategies*. These are strategies which enable Prover to win, no matter how Adversary plays. In the rest of the paper we shall omit the word 'winning', as we will not mention other strategies.

Suppose a strategy for Prover is fixed. Then we say that *the complexity of the strategy* is *the number of all different records that can appear* in all possible games where Prover plays this strategy. This is the same as the number of rules in the strategy, since the strategy has to say what to do in every situation that can possibly occur and does not have to have rules for impossible situations. It can be viewed as space complexity, since Prover can compress his records in binary strings. If the number of all records that he needs to consider is N, then he can use strings of length $\lceil \log_2 N \rceil$. Thus the memory requirement is the logarithm of the number of records. Let me stress that the complexity of Prover's strategy manifests only when many games are played. In each single game he needs only polynomially many records. (If he uses compression to binary strings, in a single game he may be lucky and he may need only logarithmic space, even if his strategy is of exponential complexity.)

As usual, we are interested in asymptotic bounds on the minimal complexity as a function of the input size. The size of the PHP tautology is polynomial in n (see the last section). Since we are mainly interested in the distinction between polynomial and exponential growth, it does not matter if we use n, a more convenient parameter, instead of the actual size of the tautology.

We would like to prove that every strategy of Prover has exponential complexity, therefore we think of Adversary forcing Prover to use exponentially many different records. Then a proof of a lower bound on the complexity of Prover's strategies can be viewed as a kind of *superstrategy* for Adversary. It is not a strategy in the usual sense, since, as we noted above, there is a simple strategy for Prover which in every single game uses only $(n+1)n$ different records. A superstrategy of Adversary is how to play a *series of games*. It can be presented as a parametrized set of ordinary strategies for single games. To force Prover to use an exponential number of records, Adversary tries all parameters when playing the series of games.

Now we are in a very favourable situation: to prove an upper bound we need to find a good strategy for Prover, and to prove a lower bound we need to find a good superstrategy for Adversary.

3 Haken's superstrategy

Haken's superstrategy is simple, but tricky. Before each game Adversary chooses a partial assignments of $n/4$ pigeons[1] to holes. They will play a series of games for all possible partial assignments of this form. Let the assignment be α. For a given α, Adversary's aim is to impose α upon Prover, ie., to force him to write it down. If Adversary succeeded each time to force Prover to record whole α, we would be done, since there are exponentially many incompatible α's. But Prover may ignore it and concentrate on other pigeons. Therefore Adversary has to play in a clever way so that at least some information about α is present in the record before the game ends. It will turn out that in most cases Prover will have to record a substantial part of α, which will give an exponential lower bound on the complexity of his strategy.

During the game Adversary will enlarge α to a bigger partial one-to-one assignment β. Later he may decrease it and increase it again, but he always keeps the original α as a subset of the current β, ie., he never deletes a pair from that part of β. Before saying how he answers questions of Prover, we state the rules for β.

1. If *Alexander* is a pigeon in the domain of α and α assigns *Brest* to him, the pair *(Alexander,Brest)* is always in β.

2. If a pigeon *Aldo* is not yet in the domain of β, he will receive a hole *Bologna* in the moment when the number of prohibited holes on Prover's record for him reaches $n/2$. (Recall that prohibited means that *(Aldo,Bologna,not)* appears on the record; the holes which belong to other pigeons are implicitly prohibited too, but we do not count them here.) The hole that will be assigned to *Aldo* will be the first which is free according to Adversary, which means, the first which is not prohibited by Prover's record for *Aldo,* nor does it belong to another pigeon of the current β. There may be no such hole and we will explain below what will happen then. The pair *(Aldo,Bologna)* will be added to β. The pair will be kept in β as long as (1) the number of prohibited holes in Prover's record is at least $n/2$ or (2) the pair *(Aldo,Bologna)* is on the current Prover's record. When a pigeon is discarded from β, he may receive a different hole next time he will appear there.

[1] In order to simplify computations we shall assume that n is divisible by 4.

The rest of the superstrategy is easy. If the question of Prover is about pigeon *Albert* and hole *Brussels*, Adversary says yes if and only if $\beta(Albert) = Brussels$. In particular, Adversary says always no, if *Albert* is not in the domain of β.

It may happen that at some point Adversary is not able to follow the superstrategy. The only reason for that can be that he is not able to extend β according to the rule 2. There may be a consistent move for him, but for the lower bound proof the rest of the game is irrelevant. Thus we may assume that he will give up in such a case anyway (by giving a contradictory information, for instance).

The idea behind is as follows. When we, as Adversaries, deal with pigeons that are not in the domain of α, it is not good to commit ourselves to particular holes early in the game. Answering *no* gives more freedom. On the other hand we cannot wait until only one hole is available for a pigeon, since this hole may be taken by another one and we run in a contradiction when we are asked about the former pigeon, which can happen in only two rounds.

This is a very rough explanation and we would have to study strategies of Prover in more detail in order to justify Haken's superstrategy for Adversary. Let us consider only one special case. Suppose the Adversary commits himself to an α which is defined for all but one pigeon. Then it may happen that Prover decides to ask all questions about the pigeon which was put aside. Adversary has to assign a hole to this pigeon. That hole was originally intended for another one, say pigeon *Anthony*. He has to modify his commitment and not to insist that *Anthony* goes to the hole $\alpha(Anthony)$. Next Prover may ask about pigeon *Anthony* and block another hole etc. In this way Prover may destroy completely the original plan of Adversary.

Let us return to Haken's superstrategy and prove that it forces Prover to produce exponentially many records. The key lemma says that before the game ends a record is produced where there are many pigeons about which there is much information. To state it precisely, let us define that a pigeon is *good* on a record of Prover if a hole is assigned to him or if there are at least $n/2$ holes prohibited for him in the record. Note that a pigeon outside the domain of α is good if and only if it is in the domain of β.

Lemma 1 *If Adversary uses the superstrategy above, then however Prover plays, before the game ends he must create a record with $n/4$ good pigeons.*

Proof. Observe that while Adversary follows his superstrategy there is no obvious contradiction on Prover's record. Adversary always assigns different holes to different pigeons and when there are $n/2$ holes prohibited for a pigeon on Prover's record, he has a hole reserved for that Pigeon. Hence before the games ends (an obvious contradiction appears on Prover's record), Adversary has to give up. This means that a situation is reached, in which he is not able to extend β according to the rules above. This may happen only in the following way. For some pigeon *Andrew* which is not in the domain of β, when the number of prohibited holes reaches $n/2$ all the remaining holes have already been reserved for other pigeons by the assignment β. Thus the number of pigeons in the domain of β plus the number of holes prohibited to *Andrew* is n. Since there are $n/2$ prohibited ones, the size of the domain of β is at least $n/2$. But there are only $n/4$ pigeons in the domain of α, hence there are at least $n/4$ good pigeons. As the number of good pigeons can change by at most one and initially there is none, there must be a record with exactly $n/4$ pigeons. □

Now we are ready to show that Haken's superstrategy for Adversary forces Prover to use exponentially many different records. We shall use a counting argument and as usual it is more intuitive to use the language of the probability theory. Let R be the set of all records that have the property stated in the lemma. Let α be a random partial assignment of the form that Adversary uses. We shall say that α belongs to r if r is obtained when Adversary used α. Our aim is to show that for a fixed $r \in R$ the probability that α belongs to r is exponentially small. In fact we shall upper bound this probability by the probability that α is consistent with r, since being consistent with is a weaker condition than belonging to. As every α belongs to some r, the size of R must be exponentially large. Here exponentially small means less than $2^{-\varepsilon n}$ for a positive constant ε, hence exponentially large means $2^{\varepsilon n}$.

Let G be the set of good pigeons of r, then, by the lemma, $|G| \geq n/4$. Thus the average size of the intersection of the domain of α and G will be at least $n/16$. If we fix some smaller fraction of n, say $n/32$, then the probability that the size of the intersection is smaller is exponentially small. This is because the distribution of the size of the intersection is very close to a binomial distribution. To prove it formally, observe that this probability is (fix one set and consider only the second one to be random):

$$\binom{n+1}{n/4}^{-1} \sum_{i=0}^{n/32} \binom{n/4}{i} \binom{n+1-n/4}{n/4-i}$$

This can be easily estimated by elementary means.

Then it remains to bound the probability that α belongs to r assuming the condition that the size of the intersection is at least $n/32$. Take a pigeon *Adam* from this intersection. Since it is good, a hole has been assigned to him or at least $n/2$ holes have been prohibited for him. The hole $\alpha(Adam)$ is chosen randomly. Thus in the first case the probability that it will coincide with the one from r will be $1/n$, and in the second case the probability that it will avoid the forbidden ones will be at most $1/2$. If the holes for pigeons from the domain of α were chosen completely independently we could conclude that the probability that α is consistent with r was at most $2^{-n/32}$ (assuming the condition about the intersection). This is not the case, as α is one-to-one, not a general mapping, thus we have to modify the computation. Think of randomly selecting the mapping in a stepwise process: select a hole for the first pigeon, then for the second etc. When selecting the hole for the $i+1$-th pigeon there are already i holes taken, thus the probability in the first case is at most $1/(n-i)$ and in the second at most $\frac{n/2}{n-i}$. In both cases the value is $< 3/4$. Hence a true estimate is $(3/4)^{n/32}$.

Thus the probability that α belongs to r is exponentially small. So we have proved:

Theorem 2 (Haken [2]) *There exists a positive ε such that the complexity of any strategy of Prover is at least $2^{\varepsilon n}$.*

4 Problems with too many pigeons

Not only big cities have problems with too many pigeons, but also lower bound provers do. A problem arises when we consider a more general set of tautologies, namely we take m pigeons and n holes, where $m > n$. The more pigeons we have the more absurd is the claim that they all can fit in n holes, hence the easier the proof should be. If one uses the same counting for larger m's the lower bound decreases with increasing m and it gives nothing when m is of the order of $n^2/\log n$ [1]. But this concerns only the proof of the lower bounds; we do not have upper bounds of this form. It is an open problem whether there are polynomial size resolution proofs of the generalized PHP for m larger than $n^2/\log n$.

Where is the problem? Maybe we could use Haken's superstrategy if we counted more precisely? The answer is no, we can *prove* that Haken's superstrategy fails.

Recall that our aim is to prove a lower bound on the complexity for every given strategy for Prover. To determine the complexity of a given strategy Adversary should play in all possible ways. But a superstrategy does not produce all possible games. In general it gives only a restricted set of Adversary's strategies. It may happen that for that particular set of Adversary's strategy there is a strategy of Prover which needs a small, say polynomial, number of records. We shall call such a strategy a *counter-strategy* for a given superstrategy. Let us stress that the existence of the counter-strategy only excludes the possibility of proving a lower bound in a certain way, but it does not solve the problem of the existence of Prover's strategy of small complexity. The counter-strategy may fail for another superstrategy. We will show such a counter-strategy against Haken's superstrategy for $m > (n^2 + 5n)/4$ pigeons and n holes.

Let us make a preliminary consideration. Suppose we used Haken's superstrategy with the empty partial assignment α. As there is no parameter, such a superstrategy is an ordinary strategy, ie., the game is played only once. For such a single case we know that Prover needs only a polynomial number of records. For instance he simply systematically asks about all pigeons, starting with the first one, and all holes, which gives $(n+1)n$ records. Let us call it the *straightforward strategy* of Prover.

Going back to Haken's superstrategy, the idea of the counter-strategy is to reduce it to the above case with $\alpha = \emptyset$ by concentrating on an interval of pigeons of length $n+1$ which does not contain a pigeon from the domain of α. Since the size of the domain of α is $n/4$ and $m > (n^2+5n)/4 = n/4 + n(n/4+1)$ there are such intervals (accidentally it is due to the Pigeon Hole Principle). Suppose Prover is able to guess such an interval. Then he plays his straightforward strategy which he has for the case of $n+1$ pigeons just shifted to this interval. That strategy uses only $(n+1)n$ records. Hence the total number of records that he may ever need will be this number times the number of possible shifts, which is $(n+1)n(m-n)$. Thus the number of records used is polynomial.

Now we only have to solve the problem how to find such an interval, since Prover does not know which α has been chosen by Adversary. The solution is simple. Prover will try all intervals $[1, n+1], [2, n+2], [3, n+3], \ldots$ one after another one. Each time he will play as if there was no pigeon from the domain of α in the interval. He will play as long as the answers of Adversary are consistent with this assumption. When he gets an answer which shows that his assumption was wrong he abandons all his record and moves to the next interval. Since he always starts with the first pigeon in the interval he knows from his record on which interval he is working. In this way he reaches an interval that he needs and creates only records which are shifted records of his straightforward strategy.

5 The weakness of positive thinking

Suppose Prover is willing to remember only positive information. We shall show that in such a case he needs exponentially many records (counted in the number of holes) whatever the number m of pigeons is ($m > n$). To this end we have to change the game a little. Instead of Prover's asking for a pair (*Alva,Boston*) whether pigeon *Alva* goes to hole *Boston*, he will ask only about a pigeon: Where does *Alva* go? Adversary has to select a hole for *Alva*, say *Berkeley* and Prover will record that *Alva* goes to *Berkeley*. This piece of positive information, of course, implies pieces of negative information, such as that other pigeons do not go to *Berkeley*. But the negative information will not be recorded.

Our superstrategy will be parametrized by colorings of pigeons by two colors, say, black and white. The rule used by Adversary is very simple. For a black pigeon he selects the first available hole among holes $1, \ldots, n/2$, for a white pigeon he takes the first available hole among holes $n/2+1, \ldots, n$.

The fact corresponding to Haken's lemma is also very simple:

Claim. *In each game played with this superstrategy there must be a record where either all holes $1, \ldots, n/2$ are filled, or all holes $n/2+1, \ldots, n$ are filled.*

The proof is obvious — as long as Adversary plays his strategy he does not create a contradiction. The only situation that forces him to abandon the strategy (or give up) is when one half of the holes is filled.

The claim implies that in each game there is a situation in which Prover knows either $n/2$ black pigeons or $n/2$ white pigeons. We will use it to prove that there must be exponentially many records. Let R be the set of records with the property stated in the claim above. Using R we can construct colorings of pigeons as follows. Take an $r \in R$. If all holes $1, \ldots, n/2$ are filled in r, color black the corresponding pigeons, then construct $2^{m-n/2}$ colorings by coloring arbitrarily the rest of the pigeons. If all holes $n/2+1, \ldots, n$ are filled in r, color white the corresponding pigeons, then construct $2^{m-n/2}$ colorings by coloring arbitrarily the rest of the pigeons. Thus we can get at most $|R|2^{m-n/2}$ colorings. By the claim, we should get all colorings in this way. Hence $|R|2^{m-n/2} \geq 2^m$, whence $|R| \geq 2^{n/2}$.

You can use the same idea for a superstrategy for the general game, but it does not give a superpolynomial lower bound then. It is a useful exercise to construct a counter-strategy against it, if you want to get a good feeling for the game.

6 Proofs

But where are the proofs? What is the connection of the game to resolution proofs? In fact you do not need to know anything about logic, since knowing the game you can prove bounds on the size of resolution proofs without referring to logic at all. However, if you are not interested in solving combinatorial problems, you may be more interested in seeing the connection between proofs and games, which I will briefly sketch in this section.

First I have to recall more about the resolution system. Resolution is a procedure to reduce the problem of finding a proof of a given first order sentence (ie., a sentence of predicate logic) true sentence to a more combinatorial problem, which is easier to implement on a computer. Because of the recursive undecidability of the existence of a proof in the first order logic, it is not possible to bound the length of proofs of true sentences by a computable function. Having a combinatorial characterization of proofs helps when programing, but the search problem remains equally difficult. On the other hand, for particular sentences some proof systems or search procedures may be substantially better than others. Therefore it is interesting to prove bounds for concrete sentences and particular proof systems or proof search procedures. Since it is very difficult to give precise bounds, most research is done on asymptotic bounds, ie., we are interested how fast the bounding function grows, when the length of the sentences grows to infinity. This is essentially the same as in the complexity theory, where we are interested in the asymptotic behaviour of, say, the time bound for an algorithm.

In this paper I have considered not first order logic, but only propositional logic. Resolution can be viewed as having a part which reduces first order logic to propositional logic and then applying a search procedure in propositional logic. Hence, if one proves a large lower bound for some propositions for the propositional part of resolution, it shows that the whole procedure is not efficient on these concrete examples. There are also deeper theoretical reasons for studying lengths of proofs in propositional logic. Namely, this problem is connected with famous open problems in complexity theory. The main open problem here is to prove or disprove the conjecture that there is no proof system (in a precisely defined sense) in which every propositional tautology has a proof of length at most polynomially longer than the length of the tautology. In fact, most researchers believe that the proofs of some tautologies have to be of exponential length. Resolution is one of concrete proof systems for which this conjecture has been verified. Because of lack of space, I will not say more about it; the reader can learn more for instance in the monograph [3] or in the survey [4].

Propositional resolution proofs are defined for a very special type of formulas. Let us fix a set of propositional variables. We call a *literal* a variable, or a negated variable. A clause is a finite set of literals which does not contain complementary literals. We interpret a clause as the disjunction of its literals. A set of clauses is interpreted as the conjunction of (the disjunctions corresponding to) clauses. Thus a set of clauses is essentially a CNF (a formula in the conjunctive normal form). Furthermore, the empty set is also considered a clause. It is called the empty clause and its meaning is the contradiction.

Let us express the negation of PHP with $n+1$ pigeons and n holes in this way. We need a propositional variable for each pair of a pigeon and a hole, so we shall index variables by two indices. Thus we take variables $x_{i,j}$ for $1 \leq i \leq n+1$ and $1 \leq j \leq n$. The negation of PHP is a set of clauses of two types. The first type are clauses saying for each pigeon that he has a hole. These are the following clauses (written as disjunctions)

$$\bigvee_j x_{i,j}$$

for every $1 \leq j \leq n$. The second type of clauses says that every hole contains at most one from each pair of pigeons, ie.,

$$\neg x_{i,j} \vee \neg x_{k,j}$$

for every $1 \leq i, k \leq n+1$.

Resolution proofs are proofs by contradiction. Thus we first express the negation of our tautology and then derive a contradiction. The only logical rule that we are allowed to use is the resolution rule which is the following. Given clauses C and D containing complementary literals, say, $x \in C$ and $\neg x \in D$, we may infer the clause $(C \setminus \{x\}) \cup (D \setminus \{\neg x\})$. Formally, a resolution proof is a sequence of clauses which starts with the given ones, then each next is derived by an application of the resolution rule and the last clause is empty. The resolution rule is clearly sound and it is not difficult to prove that there is a proof of contradiction from any contradictory set of clauses.

Note that a computer implementation requires also a search procedure for such proofs. But if we show that for some tautology there is no short proof, we prove that *every* search procedure based on resolution has long running time (because there is no short proof which it can generate).

The transformation of a proof into a strategy for Prover is very simple. In a resolution proof a clause may be derived from previous clauses in more than one way. To avoid this ambiguity, for each non-initial clause, we choose a pair of its parents. Then we reverse the order of clauses and in each clause we flip the literals. The flipped clauses will be the records of Prover. In fact, what we are doing is simply negating the disjunctions, the result being the conjunction of negated variables.

In particular the strategy starts with the empty record. To be consistent with the reversed order, the former parents will be called children now. The strategy of Prover is as follows. Seeing a record ρ it will look at its children ρ_1 and ρ_2 and ask the (unique) common variable x of the children. After receiving an answer from Adversary, which is the literal x or $\neg x$, Prover replaces ρ by the child containing the literal. Recall that both ρ_1 and ρ_2 are subsets of ρ except that they contain x or $\neg x$. Thus the only new information on Prover's record is about x. Eventually Prover arrives at the initial clauses. Their negations are records that are obviously in contradiction with the assumption that there is a one-to one mapping, so he wins.

Note that Adversary can steer the game so that it follows an arbitrary path through the proof. Thus he is able to force Prover to use the negation of every clause of the proof as his record, provided that they play a sufficiently large number of games. Hence the number of records Prover uses when using his strategy is simply the size of the resolution proof.

To construct a proof from a strategy for Prover, we simply reverse this transformation.

As you see, the transformation of a proof into a strategy and the converse one are quite straightforward. Still it helps understand lower bound proofs. It is because it is usually difficult to get intuition for a nonexistence proof. The game transforms the task of proving something negative into the positive task of forcing Prover to create a lot of records.

References

[1] S.R. Buss, Gy. Turán, Resolution proofs of generalized pigeonhole principles, Theoretical Computer Science 62 (1988), 311-317

[2] A. Haken, The intractability of resolution. Theoretical Computer Science 39 (1985), 297-308.

[3] J. Krajíček, Bounded Arithmetic, Propositional Logic, and Complexity Theory, Cambridge University Press, 1995.

[4] P. Pudlák, The lengths of proofs. Handbook of Proof Theory, S.A. Buss editor, North Holland 1998, 547-637

[5] P. Pudlák, S.R. Buss, How to lie without being (easily) convicted and the lengths of proofs in propositional calculus, in Computer Science Logic'94, Pacholski and Tiuryn eds., Springer-Verlag LNCS 933, 1995, 151-162

[6] J.A. Robinson, A machine-oriented logic based on the resolution principle, J. of ACM, 12(1), 1965, 23-41

[7] A. Razborov, A. Wigderson, A. Yao, Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus, Proc. Annual ACM Symp. on Theory of Computation 1997, 739-748