Information Processing Letters 113 (2013) 666–671

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A characterization of tree-like Resolution size ☆

Olaf Beyersdorff [a,*], Nicola Galesi [b], Massimo Lauria [c]

[a] School of Computing, University of Leeds, UK
[b] Dipartimento di Informatica, Sapienza Università di Roma, Italy
[c] KTH Royal Institute of Technology, Sweden

ARTICLE INFO

Article history:
Received 23 August 2012
Received in revised form 31 May 2013
Accepted 3 June 2013
Available online 6 June 2013
Communicated by J. Torán

Keywords:
Computational complexity
Proof complexity
Prover–Delayer games
Resolution

ABSTRACT

We explain an asymmetric Prover–Delayer game which precisely characterizes proof size in tree-like Resolution. This game was previously described in a parameterized complexity context to show lower bounds for parameterized formulas (Beyersdorff et al. (2013) [2]) and for the classical pigeonhole principle (Beyersdorff et al. (2010) [1]). The main point of this note is to show that the asymmetric game in fact characterizes tree-like Resolution proof size, i.e. in principle our proof method allows to always achieve the optimal lower bounds. This is in contrast with previous techniques described in the literature. We also provide a very intuitive information-theoretic interpretation of the game.

1. Introduction

Resolution is one of the best-known and most-studied proof systems. It was developed by Blake [3], Davis and Putnam [6], and Robinson [15] and refutes unsatisfiable formulas in CNF by using the single inference rule $\frac{C\vee x \quad D\vee\neg x}{C\vee D}$. Due to its simplicity, Resolution is widely used in applications. In fact, most modern SAT solvers employ subsystems of Resolution as their underlying mechanism. One particularly important subsystem is *tree-like* Resolution where proofs are in the simple form of a tree, i.e. each derived clause can be used at most once. When we focus on unsatisfiable formulas, tree-like Resolution is equivalent to the execution of DLL algorithms and to decision trees for the *search problem* (i.e. to find a falsified clause under a given assignment).

A number of techniques have been developed to understand the complexity of Resolution and its fragments. Most notably, there is the size-width tradeoff [5], the feasible interpolation technique [12] and several game-theoretic methods [13,14]. All these techniques provide powerful tools for showing lower bounds to the size (and sometimes width and space) of Resolution refutations. One interesting question is how good these lower bounds are, i.e. if the techniques can be used to obtain optimal bounds. One nice result in this direction is the characterization of *tree-like Resolution space* by Esteban and Torán [9]. Their characterization uses the Prover–Delayer game devised by Pudlák and Impagliazzo [13] which is a simple and elegant method to obtain bounds on the *size* of tree-like Resolution refutations. But exactly for this characterization of Esteban and Torán, the game of Pudlák and Impagliazzo does not yield optimal size bounds for tree-like Resolution refutations.

Our contribution in this note is to explain a refined version of the Pudlák–Impagliazzo game, the *asymmetric Prover–Delayer game*. This game has been used previously in Parameterized Resolution [2] and for the classical pigeonhole principle [1]. Here we present a slightly simplified version for classical tree-like Resolution and observe that our asymmetric game precisely characterizes tree-like Resolution size. This result provides an interesting counterpart to the result of Esteban and Torán [9] in that we

☆ Research was supported by a grant from the John Templeton Foundation and by a DAAD grant.

* Corresponding author.
E-mail addresses: o.beyersdorff@leeds.ac.uk (O. Beyersdorff), galesi@di.uniroma1.it (N. Galesi), lauria@kth.se (M. Lauria).

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2013.06.002

now have elegant combinatorial characterizations of both tree-like Resolution space and size.

As mentioned above, the original symmetric game of [13] was also studied by Esteban and Torán [9], who prove that the tree-like Resolution *clause space complexity* of a formula is exactly equal to the largest number of points achievable by the Delayer in the symmetric game. The lower bound for proof length follows from the fact that a formula with clause space complexity s requires proof length at least 2^s (cf. [8]). This connection with clause space complexity limits the strength of the symmetric method, since there are formulas for which the above lower bound is not tight (e.g. the classical pigeonhole principle [11,7,1]). This is so because the clause space complexity of a formula F is s if and only if any proof tree for F contains a complete binary tree of height s. The gap between the size of such a minor and the size of the proof tree is exactly what the symmetric game fails to analyse. In contrast, our result here (Corollary 2) states that the asymmetric game precisely characterizes the proof size. As a consequence our result shows that for some formulas the tree-like size is larger than $2^{\text{tree-like space}}$.

The remaining part of this paper is organized as follows. In Section 2 we describe the asymmetric Prover–Delayer game. Section 3 collects some facts on the correspondence between tree-like Resolution refutations and boolean decision trees. Sections 4 and 5 then state the two directions of our characterization of tree-like Resolution size by the asymmetric game. We conclude in Section 6 with an example illustrating the advantage of the asymmetric game over the symmetric version.

2. Asymmetric Prover–Delayer games

The game starts with an unsatisfiable formula F in CNF, and it is played by two players called Prover (female) and Delayer (male). The Delayer brags that he knows a satisfying assignment for F, and the Prover wants to expose his lie. At each round of the game Prover asks Delayer for the value of one of the variables, and she continues to query until a clause of F is falsified. In each round the Delayer scores some points and indeed the objective of the Delayer is to maximize his score at the end of the game, before being eventually exposed by the Prover.

Let $F = \bigwedge_j C_j$ be the CNF they play on. We say that "$C_j(\alpha) = b$" when the partial assignment α forces clause C_j to evaluate to b. We say that $C_j(\alpha)$ is undefined otherwise.

The game is played in rounds, and while the two players interact they build a partial assignment as byproduct of their interaction. We denote by α_i and s_i the partial assignment and the Delayer score at the end of round i, respectively. At the beginning of the game $\alpha_0 = \emptyset$ and $s_0 = 0$. At round i:

1. Prover asks for a variable $x \notin dom(\alpha_{i-1})$;
2. Delayer assigns two weights p_0 and p_1 to the two possible answers; the weights must satisfy:

$$p_0 \geqslant 0, \qquad p_1 \geqslant 0, \qquad p_0 + p_1 = 1. \tag{1}$$

3. Prover chooses value b, and the status of the game is updated[1]:

$$\alpha_i := \alpha_{i-1} \cup \{(x, b)\}, \qquad s_i := s_{i-1} + \log \frac{1}{p_b};$$

The game ends at the first round i such that $C_j(\alpha_i) = 0$ for some clause C_j. The final score of the Delayer is s_i.

The game has been already used in the articles [1,2]. Here the game description is simpler, in particular two details are different. The first one regards the weights p_0 and p_1. The previous definition of the game separates the weight function and the Delayer strategy. Since both must be carefully chosen for proving a lower bound, we can assume that the Delayer himself decides the value of this function at each step of the game. The second difference is that in the previous definition the Delayer was allowed to answer Prover's query at the cost of not scoring anything in that round. In the new formulation the Delayer can simulate this behavior by choosing either $(0, 1)$ or $(1, 0)$ as weights.

If Delayer chooses $(1, 0)$ or $(0, 1)$ then the Prover cannot choose the value corresponding to weight 0; that would give an infinite amount of points to Delayer, and would be sub-optimal for the Prover since she can always enforce a finite score. Therefore in the case of integer weights, Prover's answer is determined by Delayer, who in turn does not get any points in that round.

This observation justifies the small changes to the game definition given in [1,2]. It is now more compact and elegant.

The Prover–Delayer game of Pudlák and Impagliazzo [13] can be seen as a special case of the game we describe here, and will be called the symmetric Prover–Delayer game in this paper. In the symmetric game the Delayer has only two options: either he decides the value of the queried variable himself, or asks the Prover to decide. In the former case he does not get any points, while in the latter he gets one point, regardless of Prover's choice. A moment's thought is sufficient to realize that limiting the (asymmetric) Delayer to choose the pair (p_0, p_1) among $\{(\frac{1}{2}, \frac{1}{2}), (0, 1), (1, 0)\}$ yields exactly the symmetric game. If Delayer chooses $(\frac{1}{2}, \frac{1}{2})$, then he gets one point whatever Prover decides. If Delayer chooses one of the extremely unbalanced pairs we already argued that he forces Prover's hand.

3. Tree-like Resolution and decision trees

This section collects some facts on the correspondence between tree-like Resolution and boolean decision trees. All this material is known (see e.g. [4]), but we give precise statements with proofs for completeness.

A *decision tree* for an unsatisfiable CNF $F = \bigwedge_i C_i$ is a binary tree where inner nodes are labeled by variables of F and leaves are labeled by clauses from F. Each path from the root to a leaf in the tree specifies a partial assignment.

[1] We consistently manage the extreme cases by setting $s_i = \infty$ whenever Prover chooses value b such that $p_b = 0$.

In addition, a decision tree for F must satisfy the condition that each path from the root to a leaf falsifies the clause at the leaf. Therefore, a decision tree for F solves the following search problem: given an assignment $\alpha \in \{0,1\}^n$ find i such that C_i is falsified by α.

It is a well-known fact that a tree-like Resolution refutation of F can be thought of as a decision tree for F. This follows by induction on the size of the refutation. If the size is 1 then the refutation is just the empty clause, so $C_i = \Box$ for some i and the search problem can be solved by the single vertex decision tree labeled by i. If the tree is bigger, let x be the last variable on which the refutation resolves. Consider the subtrees T_0 and T_1 of the refutation, inferring respectively x and $\neg x$. For $b \in \{0,1\}$ the tree-like Resolution inference T_b can be restricted to a refutation of $F\restriction_{x=b}$. By the inductive hypothesis the latter can be transformed into a decision tree D_b which solves the search problem for $F\restriction_{x=b}$. Clearly, the search problem for F is solved by a decision tree which queries x and if $x = b$ it applies the decision tree D_b. In this transformation from tree-like refutations to decision trees

- each inferred clause corresponds to a query node;
- each Resolution inference on x corresponds to a query for x;
- each occurrence of an initial clause C_i corresponds to a leaf labeled by i;
- each consistent[2] path from the root to a node in the decision tree corresponds to a partial assignment which falsifies the corresponding clause in the refutation.

The above transformation implies the following lemma.

Lemma 1. *Let F be an unsatisfiable CNF with a tree-like Resolution refutation T. Then there is a decision tree with the same tree structure as T which solves the search problem for F.*

Notice that the transformation can be reversed, even though it is not always possible to preserve the exact tree structure since decision trees leave more room for suboptimal choices with no equivalent representation in Resolution (e.g. unreachable nodes, search problem solved by a strict subtree, falsified clauses at an internal node). Nevertheless, it is easy to identify a decision tree without such redundant parts which is embedded in the original one.

Definition 1. An *embedding* of a rooted tree T' into a rooted tree T is an injective mapping f from the vertices of T' to the vertices of T, such that if u is parent of v then $f(u)$ is an ancestor of $f(v)$. We say that T' is *embeddable* into T if such embedding exists.

It is an easy observation that any tree obtained from another one by removing subtrees and/or collapsing edges connected by degree two vertices is embeddable into the original tree. A tree-like Resolution refutation is called *regular* when in every path from an initial clause to the empty clause no variable is resolved twice.

Lemma 2. *Consider an unsatisfiable formula F in CNF and a decision tree T for the search problem on F. Then there is a tree-like regular Resolution refutation of F with tree structure T' such that T' is embeddable into T.*

Proof. We assume that T has no unreachable nodes, i.e. never queries the same variable twice. Otherwise we remove the redundant queries and the corresponding unreachable subtrees. The new decision tree is embeddable into T, so this assumption is without loss of generality.

To get the refutation we essentially take the tree and flip it over: we label nodes with clauses in such a way that the clause labelling any internal node is deducible from the clauses labelling its children using a Resolution step.

Let ρ be the unique minimal partial assignment reaching a node u. There is a unique maximal clause D which is falsified by ρ. We label the node u with D. The root clause is the empty one by definition. If u is an internal node with query variable x, the two child nodes correspond to assignments $\rho \cup \{x = 0\}$ and $\rho \cup \{x = 1\}$, respectively. Thus the labelling clauses are $D \vee x$ and $D \vee \neg x$. If u is a leaf node outputting index i for clause C_i, then $D \supseteq C_i$. This is clearly a regular tree-like refutation of the leaf clauses with the same structure as T.

To obtain a proper refutation substitute each leaf clause D with an (arbitrary) initial clause $C_i \subseteq D$. This substitution must be propagated towards the root: each clause is substituted by a subclause. Consider the inference $\frac{A \vee x \quad B \vee \neg x}{C}$. The premises are mapped to A' and B' respectively. Either both $x \in A'$ and $\neg x \in B'$ or one of them (say A') does not contain variable x. In the former case C is substituted with the resolvent of A' and B', otherwise the inference tree of C is substituted by the one of A', reducing the length of the proof. □

The two constructions explained above lead to the following remarkable correspondence.

Corollary 1. *The smallest tree-like Resolution refutation of an unsatisfiable CNF F has exactly the same size as the shortest decision tree for the search problem on F.*

4. Decision trees as Prover strategies

Given the above correspondence between tree-like Resolution and decision trees, we can now start to explain our characterization of tree-like Resolution size by the asymmetric Prover–Delayer game. Loosely speaking, we interpret the weights chosen by Delayer as a way to define a distribution on the branching made in the decision tree. Under this view the Delayer's score at each step is just the entropy of the bit encoding the corresponding choice, according to Delayer's distribution. Since root-to-leaf paths are in bijection with leaves, this process induces a distribution on the leaves. Hence the entropy collected on the path is the entropy of the corresponding leaf choice. In

[2] If a path goes through two nodes which query the same variable it must take the same direction in both nodes. Otherwise it does not correspond to a legal input of the decision tree.

this interpretation, the asymmetric Prover–Delayer game becomes a challenge between a Prover, who wants to end the game giving up little entropy, and Delayer, who wants to get a lot of it. This means that the average score of the Delayer estimates the number of leaves in the tree. In our framework the decision tree determines Prover's queries, and the Delayer defines a distribution on paths.

The connection of this game to size of proofs in tree-like Resolution is given by the next theorem. A version of this result for tree-like *Parameterized* Resolution appeared already in [2].

Theorem 1. *Let F be an unsatisfiable CNF. If F has a tree-like Resolution refutation of size at most S, then there is a Prover strategy such that any Delayer gets at most $\log\lceil \frac{S}{2} \rceil$ points.*

Proof. Let F be a contradiction using variables $x_1, \ldots, x_n$. Choose any tree-like Resolution refutation of F of size S and interpret it as a boolean decision tree T for the search problem on F, according to Lemma 1. All internal vertices of T have two children, thus $S = 2L - 1$ where $L = \lceil \frac{S}{2} \rceil$ is the number of leaves of the tree.

The decision tree T completely specifies the query strategy for Prover: at the first step she will query the variable labelling the root of T. Whatever decision is made regarding the value of the queried variable, Prover moves to the root of the corresponding subtree and queries the variable which labels it. This process induces a root-to-leaf walk on T, and such walks are in bijection with the set of leafs.

To completely specify Prover's strategy we need to explain how Prover chooses the value of the variable x asked at that round. The most natural thing to do is to choose the value randomly as follows:

$$x = \begin{cases} 0 & \text{with probability } p_0, \\ 1 & \text{with probability } p_1, \end{cases}$$

where p_0 and p_1 are the weights determined by the Delayer.

In a game played between this randomized Prover and a specific Delayer D, we denote by $q_{D,\ell}$ the probability of such a game to end at leaf ℓ. We call π_D this distribution on the leaves. To prove the theorem the following observation is crucial:

Claim 1. *If the game ends at leaf ℓ, then Delayer D scores exactly $\log \frac{1}{q_{D,\ell}}$ points.*

Before proving this claim, we show that it implies the theorem. The expected score of Delayer D is

$$\sum_{\ell} q_{D,\ell} \log \frac{1}{q_{D,\ell}} = H(\pi_D)$$

which is the Shannon entropy of the distribution π_D. The support of π_D has size at most L, which implies that $H(\pi_D) \leqslant \log L$ since the entropy is maximized by the uniform distribution. By fixing the random choices of the Prover, we can force Delayer D to score at most $\log L$ points.

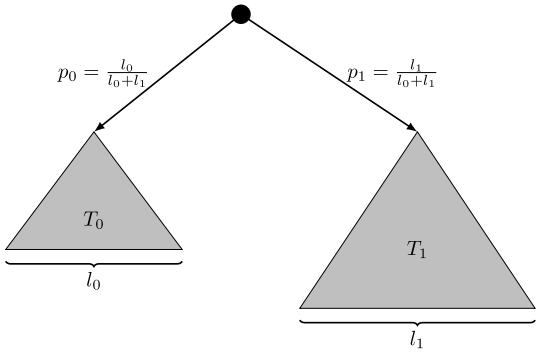


Fig. 1. The proof tree structure can be used to give different weights to the branches. Here l_0 and l_1 are the number of leaves of the left and right subtree, respectively.

To prove Claim 1 consider a leaf ℓ and the unique path that reaches it. Without loss of generality we assume that this path corresponds to the ordered sequence of assignments $x_1 = \epsilon_1, \ldots, x_m = \epsilon_m$. The probability of reaching the leaf ℓ is

$$q_{D,\ell} = q_1 q_2 \cdots q_m$$

where q_i is the probability of setting $x_i = \epsilon_i$ conditioned on the previous choices. The score of the corresponding game play is

$$\sum_{i=1}^{m} \log \frac{1}{q_i} = \log \frac{1}{\prod_{i=1}^{m} q_i} = \log \frac{1}{q_{D,\ell}}.$$

This concludes the proof of the claim and the theorem. □

The above theorem shows that lower bounds to the refutation size in tree-like Resolution can be obtained by choosing an appropriate distribution for the Delayer.

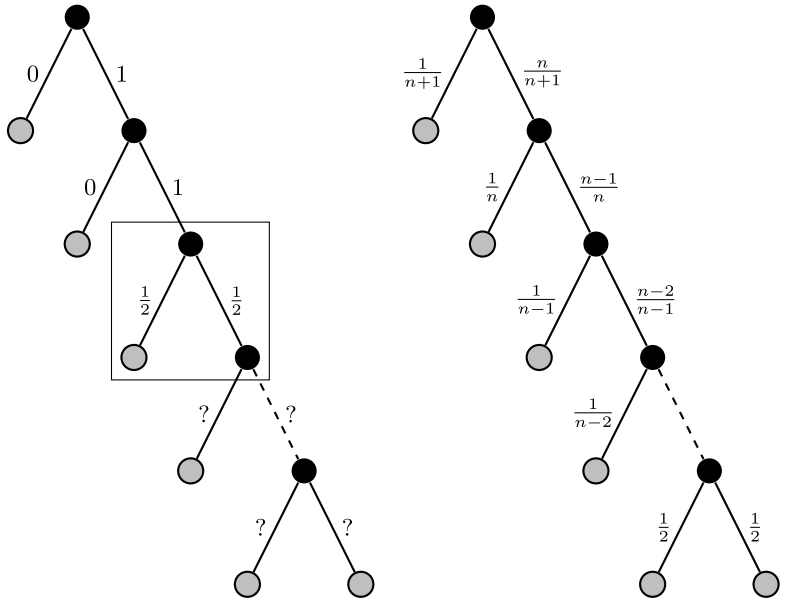
5. Distributions as Delayer strategies

So far we argued that the asymmetric Prover–Delayer game can be used to prove lower bounds for the size of tree-like Resolution refutations, but we would like to know how good this lower bound method is. Here we show that the method completely characterizes proof size, meaning that it is always possible (in principle) to define a Delayer strategy such that the implied lower bound is almost equal to the proof size. To be more precise this method characterizes the number of leaves in the shortest proof.

Theorem 2. *Let F be an unsatisfiable CNF with shortest tree-like Resolution refutation of size S. Then there is a Delayer strategy such that the Delayer scores at least $\log\lceil \frac{S}{2} \rceil$ points in any game on F against any Prover.*

Proof. We denote the number of leaves in the shortest tree-like refutation of a CNF F by $L(F)$, and we denote by $F\restriction_\alpha$ formula F restricted by the partial assignment α.

Delayer assigns weights according to the following rules, depending on the partial assignment α computed so far, and on the variable $x \notin dom(\alpha)$ queried by the Prover



(a) Symmetric (b) Asymmetric

Fig. 2. The probability associated to each leaf in a symmetric (a) and an asymmetric (b) Prover–Delayer game. In the symmetric game the Prover can end the game as soon as the Delayer gets one point, while in the asymmetric game the Delayer always gets $\log(n+1)$ points.

$$p_0 = \frac{L(F\restriction_{\alpha,x=0})}{L(F\restriction_{\alpha,x=0}) + L(F\restriction_{\alpha,x=1})} \quad \text{and}$$
$$p_1 = \frac{L(F\restriction_{\alpha,x=1})}{L(F\restriction_{\alpha,x=0}) + L(F\restriction_{\alpha,x=1})}.$$

Let n be the number of variables of the formula F. By induction on n we show that the Delayer wins at least $\log L(F)$ points. For the base case the formula has zero variables and is unsatisfiable, thus it contains the empty clause and $L(F) = 1$. The Delayer always score at least zero points, since Delayer's score is always non-negative. If $n > 0$ then two cases occur: either the formula already contains the empty clause, and then the reasoning above applies; or the Prover queries a variable x and chooses a value b. The score is $\log \frac{1}{p_b} + X$ where X is the score the Delayer wins in subsequent steps. By the induction hypothesis $X \geqslant \log L(F\restriction_{x=b})$, thus the total score is at least

$$\begin{aligned}
&\log \frac{1}{p_b} + \log L(F\restriction_{x=b}) \\
&\quad = \log\left(\frac{L(F\restriction_{x=0}) + L(F\restriction_{x=1})}{L(F\restriction_{x=b})}\right) + \log L(F\restriction_{x=b}) \\
&\quad = \log\big(L(F\restriction_{x=0}) + L(F\restriction_{x=1})\big) \geqslant \log L(F).
\end{aligned}$$

A refutation of size S has exactly $\lceil \frac{S}{2} \rceil$ leaves, so the theorem is proved. □

A comment on the previous proof is required. We already argued that the Delayer strategy defines a distribution on the root-to-leaf walks in the tree induced by the Prover strategy. The above proof is based on the fact that is possible to define a distribution which is uniform on the leaves of the shortest proof, and thus the entropy is exactly the logarithm of the size of the set of leaves (cf. Fig. 1).

Combining Theorems 1 and 2 we obtain the following tight characterization.

Corollary 2. *For any unsatisfiable CNF F, the maximum score achievable in an Asymmetric Prover–Delayer game by a Delayer is exactly $\log\lceil \frac{S_T(F)}{2} \rceil$ where $S_T(F)$ is the size of the shortest tree-like Resolution refutation of F.*

6. Advantage of the Asymmetric Prover–Delayer game

In this last section we briefly describe an example of the new characterization of tree-like Resolution size. Consider the formula

$$(x_1 \vee \cdots \vee x_n) \wedge \neg x_1 \wedge \cdots \wedge \neg x_n.$$

In the symmetric game, the Delayer will only earn 1 point (cf. Fig. 2). This only yields a trivial constant lower bound on the proof size. In contrast, the optimal Delayer in the asymmetric game will earn exactly $\log(n+1)$ points when using the distribution shown in Fig. 2. The formula is minimally unsatisfiable and has exactly $n+1$ critical assignments, namely to set exactly one of the n variables to 1 or to set all of them to 0. To win the game, Prover has to identify one of these critical assignments. And from an information-theoretic perspective, Prover needs exactly $\log(n+1)$ bits to specify one of these $n+1$ critical assignments. Consequently, the Delayer should earn exactly $\log(n+1)$ points. This is easily verified. Let k be the number of rounds that Prover needs to win the game. Then Delayer scores exactly

$$\underbrace{\sum_{i=1}^{k-1} \log \frac{n+1-(i-1)}{n+1-i}}_{\text{for the first } k-1 \text{ 0's}} + \underbrace{\log\bigl(n+1-(k-1)\bigr)}_{\text{for the last round}}$$
$$= \log \frac{n+1}{n+1-(k-1)} + \log\bigl(n+1-(k-1)\bigr)$$
$$= \log(n+1).$$

A more interesting example is the famous *pigeonhole principle* (PHP). Its complexity in Resolution was first determined by Haken's seminal exponential lower bound [10]. However, in tree-like Resolution the complexity of PHP is indeed $2^{\theta(n \log n)}$ as shown independently by Iwama and Miyazaki [11] and Dantchev and Riis [7]. The paper [1] provides an elegant proof of this optimal $n!$ lower bound for PHP via the asymmetric Prover–Delayer game. In contrast, the symmetric game only yields a lower bound of $2^{\Omega(n)}$, because the smallest tree-like Resolution refutations of PHP only contain full binary trees of height n.

References

[1] Olaf Beyersdorff, Nicola Galesi, Massimo Lauria, A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games, Inform. Process. Lett. 110 (23) (2010) 1074–1077.

[2] Olaf Beyersdorff, Nicola Galesi, Massimo Lauria, Parameterized complexity of DPLL search procedures, ACM Trans. Comput. Log. 14 (3) (2013), in press, http://tocl.acm.org/accepted.html, Conference version in: Proc. 14th International Conference on Theory and Applications of Satisfiability Testing, in: Lecture Notes in Comput. Sci., vol. 6695, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 5–18.

[3] A. Blake, Canonical expressions in boolean algebra, PhD thesis, University of Chicago, 1937.

[4] Eli Ben-Sasson, Russell Impagliazzo, Avi Wigderson, Near optimal separation of tree-like and general resolution, Combinatorica 24 (4) (2004) 585–603.

[5] Eli Ben-Sasson, Avi Wigderson, Short proofs are narrow — resolution made simple, J. ACM 48 (2) (2001) 149–169.

[6] Martin Davis, Hilary Putnam, A computing procedure for quantification theory, J. ACM 7 (1960) 210–215.

[7] Stefan S. Dantchev, Søren Riis, Tree resolution proofs of the weak pigeon-hole principle, in: Proc. 16th Annual IEEE Conference on Computational Complexity, 2001, pp. 69–75.

[8] Juan Luis Esteban, Jacobo Torán, Space bounds for resolution, Inform. and Comput. 171 (1) (2001) 84–97.

[9] Juan Luis Esteban, Jacobo Torán, A combinatorial characterization of treelike resolution space, Inform. Process. Lett. 87 (6) (2003) 295–300.

[10] Amin Haken, The intractability of resolution, Theoret. Comput. Sci. 39 (1985) 297–308.

[11] Kazuo Iwama, Shuichi Miyazaki, Tree-like resolution is superpolynomially slower than dag-like resolution for the pigeonhole principle, in: Proc. 10th International Symposium on Algorithms and Computation, in: Lecture Notes in Comput. Sci., vol. 1741, Springer-Verlag, Berlin, Heidelberg, 1999, pp. 133–142.

[12] Jan Krajíček, Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic, J. Symbolic Log. 62 (2) (1997) 457–486.

[13] Pavel Pudlák, Russell Impagliazzo, A lower bound for DLL algorithms for SAT, in: Proc. 11th Symposium on Discrete Algorithms, 2000, pp. 128–136.

[14] Pavel Pudlák, Proofs as games, Amer. Math. Monthly (2000) 541–550.

[15] John Alan Robinson, A machine-oriented logic based on the resolution principle, J. ACM 12 (1965) 23–41.