Available online at www.sciencedirect.com
ScienceDirect
Information Processing Letters 101 (2007) 163–167
Information Processing Letters
www.elsevier.com/locate/ipl

Substitutions into propositional tautologies

Jan Krajíček [1,2]

Isaac Newton Institute, Cambridge, CB3 OEH, UK

Received 5 April 2006; received in revised form 1 September 2006; accepted 8 September 2006

Available online 9 October 2006

Communicated by K. Iwama

Abstract

We prove that there is a polynomial time substitution $(y_1, \ldots, y_n) := g(x_1, \ldots, x_k)$ with $k \ll n$ such that whenever the substitution instance $A(g(x_1, \ldots, x_k))$ of a 3DNF formula $A(y_1, \ldots, y_n)$ has a short resolution proof it follows that $A(y_1, \ldots, y_n)$ is a tautology. The qualification "short" depends on the parameters k and n.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Computational complexity; Proof complexity; Automated theorem proving

Let $A(y)$ be a 3DNF propositional formula in n variables $y = (y_1, \ldots, y_n)$ and assume that we want to prove that $A(y)$ is a tautology. By substituting $y := g(x)$ with $x = (x_1, \ldots, x_k)$ we get formula $A(g(x))$ which is, as long as g is computable in (non-uniform) time $n^{O(1)}$, expressible as 3DNF of size $n^{O(1)}$. The formula uses $n^{O(1)}$ auxiliary variables z besides variables x but only x are essential: We know a priori (and can witness by a polynomial time constructible resolution proof) that any truth assignment satisfying $\neg A(g(x_1, \ldots, x_k))$ would be determined already by its values at $x_1, \ldots, x_k$.

If $A(y)$ is a tautology, so is $A(g(x))$. In this paper we note that the emerging theory of proof complexity generators (Section 1) provides a function g with $k \ll n$ for which a form of inverse also holds (the precise statement is in Section 2):

For the following choices of parameters:

- $k = n^{\delta}$ *and* $s = 2^{n^{\varepsilon}}$, *for any* $\delta > 0$ *there is* $\varepsilon = \varepsilon(\delta) > 0$, *or*
- $k = \log(n)^c$ *and* $s = n^{\log(n)^{\mu}}$, *for* $c > 1$, $\mu > 0$ *specific constants,*

it holds:

There is a function g computable in time $n^{O(1)}$ extending k bits to n bits such that whenever $A(g(x))$ is a tautology and provable by a resolution proof of size at most s then $A(y)$ is a tautology too.

Unless you are an ardent optimist you cannot hope to improve the bound to s so that it would allow an exhaustive search over $\{0, 1\}^k$. In fact, it follows that unless $\mathcal{P} = \mathcal{NP}$ no automated provers (or SAT solvers) that are based on DPLL procedure [4,5], even augmented by clause learning [15] or restarts of the procedure [6]

E-mail address: krajicek@maths.ox.ac.uk (J. Krajíček).

[1] On leave from Mathematical Institute, Academy of Sciences and Faculty of Mathematics and Physics, Charles University, Prague.

[2] The paper was written while I was at the Isaac Newton Institute in Cambridge (program Logic and Algorithms), supported by an EPSRC grant # N09176. Also supported in part by grants A1019401, AV0Z10190503, MSM0021620839, 201/05/0124, and LC505.

0020-0190/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2006.09.002

can run in time subexponential ($2^{k^{o(1)}}$) in the number of essential variables, as their computations yield resolution proofs of size polynomial in the time [2], cf. Section 3. However, for the particular function g we use, the exhaustive search yields something (assuming the existence of strong one-way functions): If $A(g(x))$ is a tautology then there are at most $2^n/n^{\omega(1)}$ falsifying truth assignments to $A(y)$ (Section 3). This is a consequence of results of Razborov and Rudich [14].

Notation. $x, y, z, \ldots$ and $a, b, \ldots$ are tuples of variables and of bits, respectively, the individual variables or bits being denoted $x_i, y_j, \ldots$ and $a_i, b_j, \ldots$, respectively. $[n]$ is $\{1, \ldots, n\}$.

1. Proof complexity generators

A proof complexity generator is any function $g: \{0,1\}^* \to \{0,1\}^*$ given by a family of circuits[3] $\{C_k\}_k$, each C_k computing function $g_k : \{0,1\}^k \to \{0,1\}^{n(k)}$ for some injective function $n(k) > k$. (We want injectivity of $n(k)$ so that any string is in the range of at most one g_k.) We assume that circuits C_k have size $n(k)^{O(1)}$. Functions g of interest are those for which it is hard to prove that any particular string from $\{0,1\}^{n(k)}$ is outside of the range of g_k. This can be formalized as follows.

Assume $m(k)$ is the size of C_k. The set of τ-formulas corresponding to C_k is parameterized by $b \in \{0,1\}^{n(k)} \setminus Rng(g_k)$. Given such a b, construct propositional formula $\tau(C_k)_b$ (denoted simply $\tau(g)_b$ when C_ks are canonical) as follows: The atoms of $\tau(C_k)_b$ are $x_1, \ldots, x_k$ for bits of an input $x \in \{0,1\}^k$ and auxiliary atoms $z_1, \ldots, z_{m(k)}$ for bit values of subcircuits of C_k determined by the computation of C_k on x. The formula expresses in a DNF that if z_j's are correctly computed as in C_k with input x then the output $C_k(x)$ differs from b. The size of $\tau(C_k)_b$ is proportional to $m(k)$. The formula is a tautology as $b \notin Rng(g)$.

The τ-formulas have been defined in [7] and independently in [1], and their theory is being developed.[4] We now recall only few facts we shall use later.

The next definition formalizes the concept of "hard to prove" in two ways; the first one follows [13], the second one is from [9]. We apply these concepts only to resolution but they are well-defined for an arbitrary propositional proof system in the sense of [3].

[3] In general we could allow functions computable in $NTime(n(k)^{O(1)})/poly \cap coNTime(n(k)^{O(1)})/poly$.

[4] [8,12,9,13,10,11]; the reader may want to read the introductions to [9] or [13], to learn about the main ideas.

Definition 1.1. Let $s(k) \geqslant 1$ be a function, and let $g = \{g_k\}_k$ be a function as above.

- Function g is $s(k)$-hard for resolution if any formula $\tau(C_k)_b$, $b \in \{0,1\}^{n(k)} \setminus Rng(g)$, requires resolution proofs of size at least $s(k)$.
- g is $s(k)$-iterable for resolution iff all disjunctions of the form

$$\tau(C_k)_{B_1}(x^1) \vee \cdots \vee \tau(C_k)_{B_t}(x^1, \ldots, x^t)$$

  require resolution proofs of size at least $s(k)$. Here $t \geqslant 1$ is arbitrary, and $B_1, \ldots, B_t$ are circuits with $n(k)$ output bits such that:
  - x^i are disjoint k-tuples of atoms, for $i \leqslant t$.
  - B_1 has no inputs, and inputs to B_i are among $x^1, \ldots, x^{i-1}$, for $i \leqslant t$.
  - Circuits $B_1, \ldots, B_t$ are just substitutions of variables and constants for variables.

Note that the $s(k)$-iterability implies the $s(k)$-hardness, the latter being the iterability condition with $t = 1$. (The proof of Theorem 2.1 uses only hardness of the function but we need iterability to get a hard function computable in uniform polynomial time in Corollary 1.5.)

The disjunction from the definition of the iterability can be informally interpreted as follows. Assume that it is a tautology. Then it may be that already the first disjunct $\tau(C_k)_{B_1}(x^1)$ is a tautology, meaning that the string B_1 is outside of the range of g_k. If not, and $a^1 \in \{0,1\}^k$ is such that $g_k(a^1) = B_1$, then $B_2(a^1)$ is the next candidate for a string being outside of the range of g_k. If that fails (and a^2 is a witness) then we move on to $B_3(a^1, a^2)$, etc. The fact that the disjunction is a tautology means that in this process we find a string outside of the range of g_k in at most t rounds.

Exponentially hard functions for resolution do exists. A $\mathcal{P}/poly$-function, a linear map over $\mathbf{F}_2$ defined by a sparse matrix with a suitable "expansion" property, $2^{k^{\Omega(1)}}$-hard for resolution was constructed in [9, Theorem 4.2]. Razborov [13, Theorems 2.10, 2.20] gave an independent construction and he noticed that any proof of hardness utilizing only the expansion property of a matrix implies, in fact, $2^{k^{\Omega(1)}}$-iterability as well. We use a weaker statement than what is actually proved in [13].

Theorem 1.2. (Razborov [13].) *There exists a function $g = \{g_w\}_w$, with $g_w : \{0,1\}^w \to \{0,1\}^{w^2}$, computed by size $O(w^3)$ circuits, that is $2^{w^{\Omega(1)}}$-iterable for resolution.*

However, what we want is a function computed by a uniform algorithm (it is not known at present how to construct explicitly the matrices used in [9,13]) in order that our substitution is polynomial time computable too. Fortunately, we can get a uniform function from Theorem 1.2, using a result from [9].

Definition 1.3. Let $m \geqslant \ell \geqslant 1$. The truth table function $\mathbf{tt}_{m,\ell}$ takes as input m^2 bits describing[5] a size $\leqslant m$ circuit C with ℓ inputs, and outputs 2^ℓ bits: the truth table of the function computed by C.

$\mathbf{tt}_{m,\ell}$ is, by definition, equal to zero at inputs that do not encode a size $\leqslant m$ circuit with ℓ inputs.

Theorem 1.4. (Krajíček [9].) *Assume that there exists a $\mathcal{P}/poly$-function $g = \{g_w\}_w$, with $g_w : \{0,1\}^w \to \{0,1\}^{w^2}$, that is $2^{w^{\Omega(1)}}$-iterable for resolution.*

Then:

(1) *For any $1 > \delta > 0$, the truth table function $\mathbf{tt}_{2^{\delta\ell},\ell}$ is $2^{2^{\Omega(\delta\ell)}}$-iterable for resolution.*
(2) *There is a constant $c \geqslant 1$ such that the truth table function $\mathbf{tt}_{\ell^c,\ell}$ is $2^{\ell^{1+\Omega(1)}}$-iterable for resolution.*

The theorem (see [9, Theorem 4.2]) is proved by iterating the circuit computing g_w along an w-ary tree of depth t, suitable t. The two statements stated explicitly are just two extreme choices of parameters, but the proof yields an explicit trade-off for a range of parameters. We state this without repeating the construction from [9].

Let $c \geqslant 1$ and $\varepsilon > 0$ be arbitrary constants. Assume that there is a function $g = \{g_w\}_w$, with $g_w : \{0,\}^w \to \{0,1\}^{w^2}$, computed by size w^c circuits and that is 2^{w^ε}-iterable for resolution.

Then the truth function $\mathbf{tt}_{m,\ell}$ is s-iterable for the following choices of parameters, with $t \geqslant 1$ arbitrary:

1. $m := w^c \cdot t$,
2. $\ell := t \cdot \log(w)$,
3. $s := 2^{w^\varepsilon - t\log(w)}$.

Corollary 1.5.

(1) *For every $c > 1$ there are $\varepsilon > 0$ and a polynomial time computable function $g = \{g_k\}_k$,*

$g_k : \{0,1\}^k \to \{0,1\}^{k^c}$,

that is, 2^{k^ε}-hard for resolution.

(2) *There are $\varepsilon > \delta > 0$ and a polynomial time computable function $g = \{g_k\}_k$,*

$g_k : \{0,1\}^k \to \{0,1\}^{2^{k^\delta}}$,

that is, 2^{k^ε}-hard for resolution.

2. The substitution

Theorem 2.1.

(1) *For any $\delta > 0$ there are $\mu > 0$ and a polynomial time computable function $g = \{g_k\}_k$, extending $k = n^\delta$ bits to $n = n(k)$ bits such that for any* 3DNF *formula $A(y)$, $y = (y_1, \ldots, y_n)$, it holds*:
   - *If $A(g_k(x))$ has a resolution proof of size at most 2^{n^μ} then $A(y)$ is a tautology.*
(2) *There are $c > 1$, $\mu > 0$ and a polynomial time computable function $g = \{g_k\}_k$, extending $k = \log(n)^c$ bits to $n = n(k)$ bits such that for any* 3DNF *formula $A(y)$, $y = (y_1, \ldots, y_n)$, it holds*:
   - *If $A(g_k(x))$ has a resolution proof of size at most $n^{\log(n)^\mu}$ then $A(y)$ is a tautology.*

Proof. For Part 1, let $\delta > 0$ be arbitrary. Put $c := \delta^{-1}$, and take $\varepsilon > 0$ and the polynomial time function $g = \{g_k\}_k$ guaranteed by Corollary 1.5 (Part 1). Hence $g_k : \{0,1\}^{n^\delta} \to \{0,1\}^n$, for $k = n^\delta$.

Assume $A(y)$ is not a tautology and let $b \in \{0,1\}^n$ is a falsifying assignment. Then $\tau(g)_b$ can be proved in resolution by combining a size s proof of $A(g(x))$ with a size $n^{O(1)}$ proof of $\neg A(b)$. By the 2^{k^ε}-hardness of g, it must hold that

$$s + n^{O(1)} \geqslant 2^{n^{\delta\varepsilon}}.$$

Hence s must be at least 2^{n^μ}, for suitable $\mu < \delta\varepsilon$.

Part 2 is proved analogously, using Corollary 1.5 (Part 2). □

Note that if $g(x)$ is a hard proof complexity generator, so is function $(x,z) \to (g(x), z)$. Hence we may apply the substitutions from the theorem only to some variables y_i.

3. Remarks

We conclude by some remarks. First we substantiate the comment about automated theorem provers and SAT-solvers from the introduction.

Let $B(x,z)$ be the formula $A(g(x))$ with the auxiliary variables z also displayed. The k variables x are *essential* in B in the sense that there is a $O(|B|)$ size resolution proof of

$$B(x,z) \vee B(x,w) \vee z_j \equiv w_j$$

[5] $O(m\log(m))$ bits would suffice but we want simple formulas.

for all j. (In fact, such a proof is easily constructible once we have the algorithm for g.) Assume that it would be always possible to find a resolution proof of a formula whose size would be subexponential in the minimal number of essential variables and polynomial in the size of the formula; in our case $2^{k^{o(1)}}|A(g(x))|^{O(1)}$.

Taking g from Theorem 2.1 (Part 2) this would get a size $|A(g)|^{O(1)}$ proof of $A(g(x))$, which is bellow the required upper bound $n^{\log(n)^{\mu}}$. Hence we could interpret this as a new proof system R_g in the sense of Cook–Reckhow [3]: A proof in R_g of $A(y)$ is either a resolution proof or a size $|A(g(x))|^c$ (specific c) proof of $A(g(x))$. This proof system would allow for polynomial size proofs of all tautologies, hence $\mathcal{NP} = co\mathcal{NP}$.

The equality $\mathcal{NP} = co\mathcal{NP}$ followed only from assuming the existence of short resolution proofs. But automated provers (SAT-solvers) actually construct the proofs, or a proof can be constructed by a polynomial time algorithm from the description of any particular successful computation. Hence the existence of automated provers (SAT-solvers) running in time subexponential in the number of essential variables implies even $\mathcal{P} = \mathcal{NP}$ (or $\mathcal{NP} \subseteq \mathcal{BPP}$ if the prover is randomized).

Our second remark concerns the exhaustive search; in other words, what do we know about $A(y)$ if we only know that $A(g(x))$ is a tautology but we do not have a short proof of that fact.

Take for g the function from Theorem 2.1 (Part 1), or any $\mathbf{tt}_{m(\ell),\ell}$ with $m(\ell) = \ell^{\omega(1)}$. Let $n := 2^{\ell}$, and interpret strings $b \in \{0,1\}^n$ as truth tables of boolean functions in ℓ variables. Hence $b \notin Rng(g)$ implies that b is not computable by a circuit of size $\ell^{O(1)}$.

Assume $A(g(x))$ is a tautology while $A(y)$ is not. Define set $C \subseteq \{0,1\}^n$ by:

$$C := \{b \in \{0,1\}^n \mid \neg A(b)\}.$$

Then it satisfies:

(1) C is in $\mathcal{P}/poly$.
(2) $b \in C$ implies that b is not computable by a size $\ell^{O(1)}$ circuit (i.e. b is not in $\mathcal{P}/poly$).

Razborov and Rudich [14] defined the concept of a $\mathcal{P}/poly$-natural proof against $\mathcal{P}/poly$. It is a $\mathcal{P}/poly$ subset C of $\{0,1\}^n$ satisfying condition (2) above, and also condition

(3) The cardinality of C is at least $2^n/n^c$, some $c \geqslant 1$.

They proved a remarkable theorem (see [14]) that no such set exists, unless strong pseudo-random number generators do not exists (or, equivalently, strong one-way function do not exists).

In our situation this implies that (under the same assumption) there can be at most $2^n/n^{\omega(1)}$ assignments falsifying $A(y)$.

Let me conclude with an open problem: *Can the substitution speed-up proofs more than polynomially?* That is, are there formulas $A(y)$ having long resolution proofs but $A(g(x))$ having short resolution proofs? In yet another words, does R simulate the system R_g defined earlier?

Acknowledgements

I am indebted to Antonina Kolokolova (Simon Fraser U.) for discussions on related topics. I thank Klas Markström (Umea) for explaining me a few facts about automated theorem provers and SAT-solvers, and to Pavel Pudlák (Prague) for comments on the draft of the paper.

References

[1] M. Alekhnovich, E. Ben-Sasson, A.A. Razborov, A. Wigderson, Pseudorandom generators in propositional proof complexity, Electronic Colloquium on Computational Complexity, Rep. No. 23, 2000. Ext. abstract in: Proc. of the 41st Annual Symp. on Foundation of Computer Science, 2000, pp. 43–53.
[2] P. Beame, H. Kautz, A. Sabharwal, Towards understanding and harnessing the potential of clause learning, Journal of Artificial Intelligence Research (JAIR) 22 (2004) 319–351.
[3] S.A. Cook, A.R. Reckhow, The relative efficiency of propositional proof systems, Journal of Symbolic Logic 44 (1) (1979) 36–50.
[4] M. Davis, H. Putnam, A computing procedure for quantification theory, Journal of the ACM 7 (1) (1960) 201–215.
[5] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, Communications of the ACM 5 (7) (1962) 394–397.
[6] C.P. Gomes, B. Selman, H. Kautz, Boosting combinatorial search through randomization, in: 15th AAAI, 1998, pp. 431–437.
[7] J. Krajíček, On the weak pigeonhole principle, Fundamenta Mathematicae 170 (1–3) (2001) 123–140.
[8] J. Krajíček, Tautologies from pseudo-random generators, Bulletin of Symbolic Logic 7 (2) (2001) 197–212.
[9] J. Krajíček, Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds, Journal of Symbolic Logic 69 (1) (2004) 265–286.
[10] J. Krajíček, Diagonalization in proof complexity, Fundamenta Mathematicae 182 (2004) 181–192.
[11] J. Krajíček, Structured pigeonhole principle, search problems and hard tautologies, Journal of Symbolic Logic 70 (2) (2005) 619–630.
[12] A.A. Razborov, Resolution lower bounds for perfect matching principles, in: Proc. of the 17th IEEE Conf. on Computational Complexity, 2002, pp. 29–38.

[13] A.A. Razborov, Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution, Preprint, May'03.

[14] A.A. Razborov, S. Rudich, Natural proofs, Journal of Computer and System Sciences 55 (1997) 24–35.

[15] L. Zhang, C.F. Madigan, M.H. Moskewicz, S. Malik, Efficient conflict driven learning in a boolean satisfiability solver, in: Proc. of the 2001 IEEE/ACM International Conference on Computer-Aided Design, 2001, pp. 279–285.