Diagonalization against Arithmetic Circuits

Pascal Koiran and Sylvain Perifel

July 11, 2008
LIP, École Normale Supérieure de Lyon
[Pascal.Koiran,Sylvain.Perifel]@ens-lyon.fr

1 Introduction

In this note we give an alternative proof of a lower bound due to Kabanets and Impagliazzo [4].

Theorem 1. *At least one of the following two statements must hold true:*

(i) $\mathsf{NEXP}^{\mathsf{RP}} \not\subseteq \mathsf{P/poly}$*;*
(ii) The permanent does not have polynomial size arithmetic circuits.

In the above result, the arithmetic circuits are constant-free (or equivalently can only use small integer constants). The proof given by Kabanets and Impagliazzo is a proof by contradiction. Assuming that $\mathsf{NEXP}^{\mathsf{RP}} \subseteq \mathsf{P/poly}$ and that the permanent has polynomial-size circuits, these authors show that the class EE (doubly exponential time) would have polynomial size Boolean circuits. It can be shown by diagonalization that this is not the case [5].

Our proof is not shorter than the original one, but we find it conceptually simple and it introduces a possibly new proof technique: diagonalization against arithmetic circuits. It remains to be seen whether this proof technique can yield new results.

Overview of our Proof – Arithmetic circuits naturally compute polynomials, and they can a fortiori be used to compute integers (which are constant polynomials). Some connections between theses two settings were established in [6,2]. For instance, it is natural to conjecture that the sequence $\lfloor 2^n \ln 2 \rfloor$ cannot be computed from the constant 1 within $(\log n)^{O(1)}$ arithmetic operations, but a proof of this conjecture would imply a superpolynomial lower bound for the permanent polynomial. The same connection holds for many "easily definable" integer sequences. For a pessimist, this indicates that obtaining good lower bounds on the complexity of explicit integer sequences is a hopeless problem. For an optimist, this means on the contrary that to obtain a lower bound for the permanent we "simply" have to prove a good lower bound for some explicit integer sequence.

Our sympathy goes to the optimist, but we will nevertheless use a more indirect strategy. Namely, assuming that the conclusion of Theorem 1 does not hold, we show that there is an "easily definable" integer sequence that is hard to compute. This is in contradiction with the results from [6] presented in the previous paragraph. In other words we design an integer sequence which is not computable by small arithmetic circuits, but which must be computable by small arithmetic circuits due to [6].

2 Integer Computations

A computation of length l of an integer n is a sequence $(n_{-1}, n_0, n_1, \ldots, n_l)$ of integers such that $n_{-1} = 1$, $n_0 = 2$, $n = n_l$ and for each $i \geq 1$ there exists $j, k < l$ and $\circ \in \{+, -, \times\}$ such that $n_i = n_j \circ n_k$. One sets $\tau(0) = \tau(1) = \tau(2) = 0$ and for $n \geq 3$, $\tau(n)$ is by definition [3] equal to the length of a shortest computation of n. In [1] the number 2 is not allowed as a "starting number", but the two corresponding complexity measures differ by at most 1 since 2 can be obtained from 1 in one arithmetic operation.

For technical reasons we will work with a related complexity measure $\tau^{1/2}$ where the rational constant $1/2$ is also allowed as a starting number. In this setting, a $1/2$-computation of length l of an integer n is a sequence $(n_{-2}, n_{-1}, n_0, n_1, \ldots, n_l)$ of *rational numbers* such that $n_{-2} = 1/2$, $n_{-1} = 1$, $n_0 = 2$, $n = n_l$ and for each $i \geq 1$ there exists $j, k < l$ and $\circ \in \{+, -, \times\}$ such that $n_i = n_j \circ n_k$. One sets again $\tau^{1/2}(0) = \tau^{1/2}(1) = \tau^{1/2}(2) = 0$ and for any integer $n \geq 3$, $\tau^{1/2}(n)$ is equal to the length of a shortest $1/2$-computation of n.

Definition 1. *A sequence (a_n) of integers is hard for computations of length n if $\tau^{1/2}(a_n) > n$ for all sufficiently large n.*

3 Nonexistence of Easily Definable Sequences that Are Hard to Compute

The main tool that we will use to show that certain "easily definable" sequences are (conditionally) not hard is essentially borrowed from [6].

Theorem 2. *Suppose that $n \mapsto p(n)$ is a polynomially bounded function, and that $p(n) \geq n$ for all $n \in \mathbb{N}$. Let (a_n) be an integer sequence such that for some integer b one can write:*

$$a_n = \sum_{j=0}^{p(n)} f(j, n) b^j \tag{1}$$

where the map $(j,n) \mapsto f(j,n)$ *is in* $\sharp\mathsf{P}/\mathsf{poly}$ *(here, we use binary encoding for* j *and* n*). If the permanent family can be computed by a family of polynomial size arithmetic circuits then* $\tau^{1/2}(a_n) \leq q(\log n)$ *for some polynomial* q*.*

The special role of the constant $1/2$ in this theorem is inherited from its role in the completeness proof of the permanent. One could state a similar theorem for the Hamiltonian instead of the permanent polynomial, with $\tau^{1/2}$ replaced by τ. Theorem 2 is an adaptation of Proposition 6.6 from [6] to the setting of $1/2$-computations. That proposition has a stronger conclusion ($\tau^{1/2}$ is replaced by τ) and a stronger hypothesis: in addition to the polynomial size hypothesis on arithmetic circuits for the permanent we need to assume a polynomial bound on their formal degree. In fact, Theorem 2 has a simpler proof than Proposition 6.6 from [6] since the only role of the formal degree bound in that proposition is to help get rid of the constant $1/2$. In the setting of $1/2$-computations this is not needed since that constant is given for free. The reader may consult [6] for more details.

We will apply Theorem 2 in the case where $f(j,n)$ is simply the bit of a_n of weight 2^j. To this end we associate to any integer sequence (a_n) the language

$$\mathsf{Bit}(a) = \{\langle n, j\rangle \mid \text{the bit of } a_n \text{ of weight } 2^j \text{ is equal to } 1\}. \tag{2}$$

Here, n and j are given in binary as in Theorem 2.

Definition 2. *Let* (a_n) *be a sequence of integers of bit size polynomial in* n*. We say that* (a_n) *is definable in time* $(\log n)^{O(1)}$ *with* $(\log n)^{O(1)}$ *bits of advice if* $\mathsf{Bit}(a)$ *is in* P/poly*.*

We then have the following immediate corollary to Theorem 2 (note that the condition $p(n) \geq n$ in Theorem 2 is not restrictive: if a_n has a smaller bit size, the high-order bits $f(j,n)$ will simply be equal to 0).

Corollary 1. *Let* (a_n) *be a sequence of integers definable in time* $(\log n)^{O(1)}$ *with* $(\log n)^{O(1)}$ *bits of advice. If the permanent family can be computed by a family of polynomial size arithmetic circuits then* (a_n) *cannot be hard for computations of length* n*.*

4 Existence of Easily Definable Sequences that Are Hard to Compute

4.1 Existence of Sequences that are Hard to Compute

It was shown in [3] that for every $\epsilon > 0$, almost all integers satisfy the property $\tau(n) \geq (\log n)/(\log(\log n))^{1+\epsilon}$. The improved lower bound $\tau(n) \geq (\log n)/(\log(\log n))$, which holds again for almost all integers, was established in [7]. Like Shannon's lower bound in boolean complexity theory, these lower bounds are based on a counting argument. They are therefore unaffected by the availability of the constant $1/2$, which does not change the order of magnitude of the number of programs of a given size. For our purposes, the following fairly crude bound will suffice.

Proposition 1. *There exists a sequence (a_n) that is hard for computations of length n where a_n has bit size $O(n \log n)$.*

Proof. If an integer a satisfies $\tau^{1/2}(a) \leq n$ then it has a $1/2$-computation of length exactly n (pad by multiplications by 1 if necessary). There are at most $3^n(n+2)^{2n}$ $1/2$-computations of length n: for each arithmetic operation we must choose the nature of the operation ($+$, $-$ or $\times$) and the left and right operands (there are at most $n+2$ possible choices for each operand). Taking the log gives the required result. □

4.2 Their Definability

Let m_n be the smallest integer satisfying $\tau^{1/2}(m_n) > n$. By Proposition 1, m_n has bit size $O(n \log n)$. To this sequence we associate the language:

$$\mathsf{UBit}(m) = \{1^{\langle n,j \rangle} \mid \text{the bit of } m_n \text{ of weight } 2^j \text{ is equal to } 1\}. \quad (3)$$

Note that we use unary encoding instead of the binary encoding of (2).

Proposition 2. *The sequence (m_n) is definable in the polynomial hierarchy, in the sense that* $\mathsf{UBit}(m)$ *belongs to the polynomial hierarchy.*

Proof. Membership of $1^{\langle n,j \rangle}$ to $\mathsf{UBit}(m)$ can be expressed by the condition:

$$\exists a\ \forall C\ \forall b\ \exists C' \text{ such that } \begin{cases} \text{output}(C) \neq a; \\ b < a \Rightarrow \text{output}(C') = b; \\ \text{the bit of } a \text{ of weight } 2^j \text{ is equal to 1.} \end{cases}$$

Here C and C' denote $1/2$-computations of length at most n. The condition $\text{output}(C) \neq a$ asserts that $\tau^{1/2}(a) > n$. The second condition asserts

that a is the smallest such integer, and the third condition checks that the bit of a of weight 2^j has the right value.

Checking that the output of C is different from a is a problem in NP (the property $\text{output}(C) \neq a$ can be certified by evaluating C modulo a suitable polynomial size integer). Likewise, checking that $\text{output}(C') = b$ is a problem in coNP. We can therefore conclude from the above characterization of $\mathsf{UBit}(m)$ that this language belongs to the polynomial hierarchy. □

In section 3 we have shown (assuming that the permanent has polynomial size arithmetic circuits) that easily definable sequences that are hard to compute do not exist. In Proposition 4 we have shown that such a sequence does exist. Unfortunately we haven't reached a contradiction yet: the definability condition in Section 3 is much more stringent than in Proposition 4 since the former is based on (nonuniform) polynomial time computability using binary encoding, and the latter on computability in the polynomial hierarchy using unary encoding. In our proof of Theorem 1 we will close that gap by throwing in the additional assumption that $\mathsf{NEXP}^{\mathsf{RP}} \subseteq \mathsf{P/poly}$. First, we need a lemma from [4].

Lemma 1. *There is a randomized polynomial-time algorithm that tests whether a given arithmetic circuit computes the permanent. That is,*

$$\{(C, 1^n) \mid C(x_{i,j})_{1 \leq i,j \leq n} = \text{Per}_n(x_{i,j})_{1 \leq i,j \leq n}\} \in \mathsf{coRP}.$$

We are now ready for the proof of the Kabanets-Impagliazzo lower bound.

Proof (of Theorem 1). Assume that $\mathsf{NEXP}^{\mathsf{RP}} \subseteq \mathsf{P/poly}$ and that the permanent has polynomial size arithmetic circuits. We will show that the language $\mathsf{Bit}(m)$ is in $\mathsf{P/poly}$, a contradiction with Corollary 1. By our first assumption, it suffices to give a $\mathsf{NEXP}^{\mathsf{RP}}$ algorithm that decides whether a given input $\langle n, j \rangle$ belongs to $\mathsf{Bit}(m)$.

Since the language $\mathsf{UBit}(m)$ is in the polynomial hierarchy (Proposition 2), by Toda's theorem we can decide whether $\langle n, j \rangle \in \mathsf{Bit}(m)$ if we have access to circuits for permanents of size polynomial in n. Such circuits are of size polynomial in n by our second assumption. Within $\mathsf{NEXP}^{\mathsf{RP}}$ we can nondeterministically guess such a circuit, and check that it computes the permanent correctly thanks to Lemma 1. □

References

1. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1998.
2. P. Bürgisser. On defining integers in the counting hierarchy and proving lower bounds in algebraic complexity. In *Proc. STACS 2007*, pages 133–144, 2007. Full version: ECCC Report No. 113, August 2006.
3. W. De Melo and B. F. Svaiter. The cost of computing integers. *Proc. American Mathematical Society*, 124(5):1377–1378, 1996.
4. V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.
5. R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–56, 1982.
6. P. Koiran. Valiant's model and the cost of computing integers. *Computational Complexity*, 13:131–146, 2004.
7. C. Moreira. On asymptotic estimates for arithmetic cost functions. *Proc. American Mathematical Society*, 125(2):347–353, 1997.