Theoretical Computer Science 660 (2017) 75–85

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Negation-limited formulas ☆

Siyao Guo [a,1], Ilan Komargodski [b,*,2]

[a] Courant Institute of Mathematical Sciences, New York University, United States
[b] Weizmann Institute of Science, Israel

ARTICLE INFO

Article history:
Received 1 August 2016
Received in revised form 7 November 2016
Accepted 22 November 2016
Available online 24 November 2016
Communicated by J. Díaz

Keywords:
Formulas
Negations
Shrinkage
Lower-bounds

ABSTRACT

Understanding the power of negation gates is crucial to bridge the exponential gap between monotone and non-monotone computation. We focus on the model of formulas over the De Morgan basis and study the connection of negation-limited formulas with negation-limited circuits and with monotone formulas. We show the following results:

- We give an efficient transformation of formulas with t negation gates to circuits with $\log t$ negation gates. This transformation provides a generic way to cast results for negation-limited circuits to the setting of negation-limited formulas. For example, using a result of Rossman (CCC '15), we obtain an average-case lower bound for formulas of polynomial-size on n variables with $n^{1/2-\varepsilon}$ negations.
- We prove that every formula that contains t negation gates can be shrunk using a random restriction to a formula of size $O(t)$ with the shrinkage exponent of monotone formulas. As a result, the shrinkage exponent of formulas that contain a constant number of negation gates is equal to the shrinkage exponent of monotone formulas.

The above results follow from an efficient structural decomposition theorem for formulas that depends on their negation complexity which may be of independent interest.

1. Introduction

Understanding the complexity of classical computational models for Boolean functions is the holy grail of theoretical computer science. Two of the most well known computational models are *Boolean circuits* and *Boolean formulas*. Boolean circuits correspond to directed acyclic graphs in which every gate is labeled by an operation from the De Morgan basis {AND, OR, NOT}. Boolean formulas are Boolean circuits in which the fan-out of gates is at most 1. The size of a circuit

☆ A preliminary version of this work appeared in the *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, pages 850–866, 2015.

* Corresponding author.
E-mail addresses: sg191@nyu.edu (S. Guo), ilan.komargodski@weizmann.ac.il (I. Komargodski).

[1] Most of this work done in Chinese University of Hong Kong supported by RGC GRF grant CUHK 410111. Part of this work done while visiting IDC Herzliya, supported by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement N. 307952.

[2] Supported in part by a Levzion fellowship, a grant from the I-CORE Program of the Planning and Budgeting Committee, the Israel Science Foundation (4/11), BSF and the Israeli Ministry of Science and Technology.

http://dx.doi.org/10.1016/j.tcs.2016.11.027
0304-3975/

(resp. formula) is defined as the number of gates (resp. leaves) it contains. Monotone circuits (resp. formulas) correspond to circuits (resp. formulas) that do not contain NOT gates.

It turns out that the presence of NOT gates is the main bottleneck for understanding the power of the computational model. For example, for monotone Boolean circuits and formulas, exponential size lower bounds are known [32,1,7,16,14], whereas the best size lower bound for general circuits (resp. formulas) remains linear (resp. cubic) [25,19,15,38].[3] Bridging these gaps are major challenges since even a super-polynomial lower bound on the size of formulas (resp. circuit) for a function that is constructible deterministically (resp. non-deterministically) in polynomial-time would separate P from NC^1 (resp. NP from P).

Motivated by bridging these gaps, researchers quantitatively studied the power of negation gates by considering negation-limited computation, in which the number of negation gates is a-priori bounded.

Negation-limited circuits. The work on negation-limited circuits can be roughly divided into two categories. One category aims to upper bound the number of negation gates required for computing arbitrary functions. In 1958, Markov [26] showed a surprising result: any Boolean function on n inputs can be computed by a circuit with only $\log n$ negation gates! Later, Fischer [11] showed that the number of negation gates can be reduced to $\log n$ within a polynomial blowup in circuit size. Several follow-up works (e.g. [36,34,40,6]) further optimized the blowup. These results imply that in order to understand general circuits, it is sufficient to focus on circuits with only $\log n$ negations.

The second category aims to extend results for monotone circuits to negation-limited circuits. One such notable work is the recent super-polynomial circuit size lower bounds for all NC^1 with roughly $(1/2)\log n$ negations by Rossman [33] (improving upon an earlier work of Amano and Maruoka [3]).

The power of negations gates has been studied in other areas as well. Recently, Blais et al. [5] studied the possibility of learning circuits with few negations, and Guo et al. [13] studied the possibility of implementing cryptographic primitives using few negations. These works provide different perspectives and enrich our understanding about negation-limited circuits.

Negation-limited formulas. Negation-limited formulas, however, have received much less attention. In 1962, Nechiporuk [28] proved an analogue of Markov's result: $\lceil n/2 \rceil$ negation gates are sufficient to compute any Boolean function on n variables by a formula. Later, Morizumi [27] showed that any formula can be efficiently transformed into a formula that computes the same function but contains at most $\lceil n/2 \rceil$ negation gates.

One beautiful corollary of the above works [26,28] is that (roughly) formulas with t negations and circuits with $\log t$ negations compute the same class of functions, which is the class of functions whose "decrease complexity"[4] is at most t. Specifically, they showed that any function that is computable by a circuit (resp. formula) with t negations has decrease complexity at most 2^t (resp. t), and moreover, that any function whose decrease complexity is d is computable by a circuit (resp. formula) with $\log d$ (resp. d) negations. Hence, any circuit with t negations can be transformed into a formula with 2^t negations and vice versa. However, this transformation is inefficient: the resulting circuit and formula may be of exponential size.

This connection enables us to transform some results for negation-limited circuits to negation-limited formulas. For example, the recent results by Blais et al. [5] and Guo et al. [13] can be cast to hold for formulas with exponentially more negations. However, when the efficiency of the transformation is of concern (e.g. results like circuit size lower bounds), this connection is of no use. This motivates the following question:

Is there an efficient transformation between negation-limited formulas and negation-limited circuits?

Since formulas have more structural properties than circuits, another natural direction to pursue is to generically extend properties of monotone formulas to negation-limited formulas. Thus, we ask:

Is there a way to extend properties known for monotone formulas to the setting of negation-limited formulas?

Looking ahead, we will answer both questions affirmatively.

1.1. Our results

In this work, we give an efficient transformation from negation-limited formulas to negation-limited circuits and extend an interesting property of monotone formulas to negation-limited formulas. Our main tool, which is of independent interest, is a decomposition theorem which says that one can efficiently decompose formulas with t negations into t monotone components.

[3] Concretely, in the setting of Boolean formulas, there exists an explicit Boolean function on n inputs such that every formula that computes it must be of size $n^{3-o(1)}$ [15,38]. Moreover, there exists an explicit monotone function on n inputs such that every monotone formula that computes it must be of size $2^{\Omega(n/\log n)}$ [14].

[4] The decrease complexity of f, denoted by $d(f)$, is the maximal number of times a function $f: \{0,1\}^n \to \{0,1\}$ changes its value from 0 to 1 along a chain (i.e. a monotone sequence of strings) starting at 0^n and ending at 1^n.