HAL open science

On Symmetric and Choiceless Computation

Anuj Dawar

To cite this version:

Anuj Dawar. On Symmetric and Choiceless Computation. 1st International Conference on Theoretical Computer Science (TTCS), Aug 2015, Tehran, Iran. pp.23-29, 10.1007/978-3-319-28678-5_2 . hal-01446261

HAL Id: hal-01446261

https://inria.hal.science/hal-01446261

Submitted on 25 Jan 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

On Symmetric and Choiceless Computation

Anuj Dawar

University of Cambridge Computer Laboratory, William Gates Building,
J.J. Thomson Avenue, Cambridge, CB3 0FD, UK.
E-mail: anuj.dawar@cl.cam.ac.uk.

Formal models of computation such as Turing machines are usually defined as performing operations on strings of symbols. Indeed, for most purposes, it suffices to consider strings over a two-letter alphabet $\{0, 1\}$. Decision problems are defined as sets of strings, and complexity classes as sets of decision problems. However, many natural algorithms are described on more abstract structure (such as graphs) because this is the natural level of abstraction at which to describe the problem being solved. Of course, we know that the abstract structures can be ultimately represented as strings (and, indeed, have to be in actual computational devices), but the representation comes at a cost. The same abstract structure may have many different string representations and the implementation of the algorithm may break the intended abstraction.

Research in the area of finite model theory and descriptive complexity (see [11, 13]) has, over the years, developed a number of techniques of describing algorithms and complexity classes directly on classes of relational structures, rather than strings. Along with this, many methods of proving inexpressiblity results have been shown, often described in terms of games. A key question that has been the focus of this research effort is whether the complexity class P admits a descriptive characterisation (see [10, Chap. 11]).

A recent paper [1] ties some of the logics studied in finite model theory to natural circuit complexity classes, and shows thereby that inexpressibility results obtained in finite model theory can be understood as lower bound results on such classes. In this presentation, I develop the methods for proving lower bound results in the form of combinatorial arguments on circuits, without reference to logical definability. The present abstract gives a brief account of the results and methods.

Symmetric Circuits.

We start with a brief introduction to the formalism of circuit complexity. A language $L \subseteq \{0, 1\}^*$ can be described as a family of *Boolean functions*: $(f_n)_{n\in\omega} : \{0, 1\}^n \to \{0, 1\}$. Each f_n can be represented by a *circuit* C_n which is a directed acyclic graph where we think of the vertices as gates suitably labeled by Boolean operators $\wedge, \vee, \neg$ for the internal gates and by inputs $x_1, \ldots, x_n$ for the gates without incoming edges. One gate is distinguished as determining the output. If there is a polynomial $p(n)$ bounding the size of C_n (i.e. the number of gates in C_n), then the language L is said to be in the complexity class $\mathsf{P/poly}$. If, in addition, the family of circuits is *uniform*, meaning that the function that takes

n to C_n is itself computable in polynomial time, then L is in P. For the definition of either of these classes, it does not make a difference if we expand the class of gates that we can use in the circuit beyond the Boolean basis to include, for instance, *threshold* or *majority* gates. The presence of such gates can make a difference for more restricted circuit complexity classes, for instance when we limit the depth of the circuit to be bounded by a constant, but not when we allow arbitrary polynomial-size circuits. Also, in the circuit characterization of P, it does not make a difference if we replace the uniformity condition with a stronger requirement. Say, we might require that the function taking n to C_n is computable in $\mathsf{DLogTime}$.

We are interested in languages that represent properties of relational structures such as graphs. For simplicity, let us restrict attention to directed graphs, i.e. structures in a vocabulary with one binary relation. A property of such graphs that is in P can be recognised by a family $(C_n)_{n\in\omega}$ of Boolean circuits of polynomial size and uniformity, as before, where now the inputs to C_n are labelled by the n^2 *potential edges* of an n-vertex graph, each taking a value of 0 or 1. Given an n-vertex graph G, there are many ways that it can be mapped onto the inputs of the circuit C_n, one for each bijection between $V(G)$—the vertices of G—and $[n]$. So, to ensure that the family of circuits is really defining a property of graphs, we require it to be *invariant* under the choice of this mapping. That is, each input of C_n carries a label of the form (i, j) for $i, j \in [n]$ and we require the output to be unchanged under any permutation $\pi \in S_n$ acting on the inputs by the action $(i, j) \mapsto (\pi(i), \pi(j))$. It is clear that any property of graphs that is invariant under isomorphisms of graphs and is in P is decided by such a family of circuits. Say that a circuit C_n is *symmetric* if any permutation $\pi \in S_n$ can be extended to an *automorphism* of C_n which takes each input (i, j) to $(\pi(i), \pi(j))$. Below, we do not distinguish notationally between the permutation π and its extension to an automorphism of C_n.

Lower Bounds for Symmetric Circuits

It is clear that symmetric circuits are necessarily invariant and it is not difficult to come up with examples that show that the converse is not true. That is, we can show that there are polynomial-time decidable properties of graphs that are not decided by polynomial-size families of Boolean circuits. Indeed, one can show that the property of a graph G having an even number of edges cannot be decided by any such family. It is rather more challenging to show that there are polynomial time decidable properties that are not decided by polynomial-size families of symmetric circuits with threshold gates. The proof is based on three ingredients, which are elaborated next.

Support Theorem The first is a combinatorial analysis of symmetric circuits establishing the so-called *bounded support property*. For a gate g in C_n, a symmetric circuit taking n-vertex graphs as input, we say that a set $X \subseteq [n]$ *supports* g if for every $\pi \in S_n$ such that $\pi(x) = x$ for all $x \in X$, we also have $\pi(g) = g$. The *support theorem* in [1] establishes that if $(C_n)_{n\in\omega}$ is a family of symmetric

circuits of polynomial size then there is a k such that all gates in C_n have a support of at most k elements.

Theorem 1 ([1]). *For any polynomial p, there is a k such that for all sufficiently large n, if C is a symmetric circuit on $[n]$ of size at most $p(n)$, then* every *gate in C has a support of size at most k.*

The theorem as proved in [1] establishes this in greater generality, allowing for circuit sizes that group super-polynomially and yielding support sizes that are then also non-constant. However, the simpler version suffices for the lower bounds state here. The proof is a combinatorial analysis of the action of permutations on a symmetric circuit and the interested reader should refer to [1] for details.

Bijection Games The second ingredient in the lower bound proof is the bijection game of Hella [12], which in combination with the support theorem, gives us a tool for showing that certain pairs of graphs are not distinguished by any circuit where all gates have bounded size supports. Hella defined this game as a characterization of equivalence in a fragment of first-order logic where the number of variables is limited, but we have *counting quantifiers*. The logic does not concern us here. We regard the game as defining a family of equivalence relations $\equiv^k$, parameterized by a positive integer k. The game is played on graphs G and H (or, more generally, finite relational structures) by two players called Spoiler and Duplicator using pebbles $a_1, \ldots, a_k$ on G and $b_1, \ldots, b_k$ on H. At any point in the game, the pebbles may be placed on vertices of the respective graphs and we do not distinguish notationally below between the pebble and the vertex on which it is placed. One move of the game proceeds as follows:

- Spoiler chooses a pair of pebbles a_i and b_i;
- Duplicator chooses a bijection $h : V(G) \to V(H)$ such that for pebbles a_j and b_j $(j \neq i)$, $h(a_j) = b_j$; and
- Spoiler chooses $a \in V(G)$ and places a_i on a and b_i on $h(a)$.

If, after this move, the map $a_1 \ldots a_k \mapsto b_1 \ldots b_k$ is not an isomorphism between the subgraphs of G and H induced by the pebbled vertices, the game is over and Spoiler wins, otherwise it can continue. We say that $G \equiv^k H$ if Duplicator has a strategy for playing forever. Clearly, if G and H are isomorphic, then $G \equiv^k H$ as Duplicator can always play the isomorphism as its choice of bijection. Conversely, if $k \geq n$ then Spoiler can force a win as long as G and H are not isomorphic. Thus $G \equiv^n H$ implies that G and H are isomorphic. For smaller values of k, the equivalence relation provides an approximation of ismorphism. The family of equivalence relations so defined is also known as the Weisfeiler-Lehman equivalences (see [5] for an account).

What links these games with symmetric circuits is the following.

Theorem 2. *If C is a symmetric circuit on n-vertex graphs such that every gate of C has a support of size at most k, and G and H are graphs such that $G \equiv^{2k} H$ then, C accepts G if, and only if, C accepts H.*

Proof. (Sketch) To prove this, we show that if C distinguishes G from H, it provides a winning strategy for Spoiler in the $2k$-pebble bijection game played on G and H, which guarantees a win in at most kd moves, where d is the depth of the circuit C. Specifically, Spoiler plays by maintaining a pointer to a gate g of C and a bijection $\alpha : V(G) \to [n]$ between the vertices of G and $[n]$ so that the following conditions are satisfied in any game position $(\bar{u}, \bar{v})$ that arises in the game:

- $\alpha(\bar{u})$ includes the support of g; and
- for any bijection $\beta : V(H) \to [n]$ such that $\beta^{-1}\alpha(\bar{u}) = \bar{v}$, we have $C_g(\alpha(G)) \neq C_g(\beta(H))$.

Here, $C_g(\alpha(G))$ denotes the value that the gate g takes in the evaluation of the circuit C when the inputs are assigned the edges of G according to the map α. Similarly, $C_g(\beta(H))$ denotes the value that the gate g takes in the evaluation of the circuit C when the inputs are assigned the edges of H according to the map β.

These conditions are initially satisfied by taking α to be any bijection and letting g be the output gate of C, since C is assumed to be symmetric and to distinguish G from H. The key step in the proof shows that, given that the conditions are satisifed, Spoiler can, within k moves, move the pointer to a child of g so that the conditions are still satisfied. Indeed, suppose $\gamma : V(G) \to V(H)$ is the bijection that Duplicator plays. Since, by assumption, g is evaluated differently under the assignments $\alpha(G)$ and $\alpha\gamma^{-1}(H)$, there must be a child h of g so that $C_h(\alpha(G)) \neq C_h(\alpha\gamma^{-1}(H))$. Spoiler aims to place pebbles on the vertices of G in $\alpha^{-1}(s)$ where s is the support of h, within k moves. At each move, Duplicator may change the bijection γ. However, since the elements corresponding to the support of g are pebbled, there is an automorphism of C that fixes g, corresponding to the changed bijection. This enables us to show that Spoiler can indeed force pebbles onto the elements $\alpha^{-1}(s)$ where s is the support of a suitable h, within k moves, while maintaining the conditions (1) and (2) above.

Thus, within kd moves, the conditions are satisfied with the pointer at a gate which is an input gate of C, say labelled with the input (i, j). The support of this gate is just the set $\{i, j\}$ so, by assumption, there are pebbles on $u = \alpha^{-1}(i)$ and $v = \alpha^{-1}(j)$ in G and corresponding pebbles on $\gamma(u)$ and $\gamma(v)$. The condition that $C_g(\alpha(G)) \neq C_g(\alpha\gamma^{-1}(H))$ tells us that one of these is an edge and the other is not, which means that Spoiler has won the bijection game.

Cai-Fürer Immerman Graphs The final ingredient in proving a lower bound is the construction of pairs of graphs that are not isomorphic, but are equivalent in the relation $\equiv^k$. Just such a construction is provided by Cai et al. [5]. To be precise, they show that there is sequence of pairs of graphs G_k and H_k ($k \in \mathbb{N}$) so that for each k we have $G_k \equiv^k H_k$ and there is a polynomial-time decidable property of graphs that includes all the G_k and excludes all H_k. In particular, it follows from our discussion above that this polynomial-time property is not

decided by any polynomial-size family of *symmetric* circuits, even in the presence of threshold gates.

The graphs G_k and H_k are obtained from a single graph G by replacing the vertices and edges of G by suitably defined gadgets. In particular, each edge of G is replaced in G_k and H_k by a pair of parallel edges. Swapping the endpoints of one such pair of edges distinguishes G_k from H_k. It can then be shown that if the graph G we start from is sufficiently well connected, in particular if it has tree-width at least k, then $G_k \equiv^k H_k$. For details of the construction, including the connection to tree-width and pebble games, we refer the reader to [9].

Reductions and Complete Problems.

Having established a super-polynomial lower bound for symmetric threshold circuits for one (artificial) problem, we are able to tranfer such lower bounds to other problems by means of reductions. The appropriate notion of reduction here is a symmetric version of AC^0 reductions, i.e. reductions given by families of *constant-depth*, *polynomial-size* Boolean circuits. In order to formally define such reductions, we have to consider circuits which do not have a single output but can, instead, be used to define a relation on $[n]$ when presented with an n-vertex graph, and we need to extend the definition of symmetry to such circuits. This is done formally in [1]. To be precise, we have a circuit C along with an injective function $\Omega : [n] \to C$. The requirement for symmetry now says any permutation $\pi \in S_n$ extends to an automorphism of C so that $\pi(\Omega(x)) = \Omega(\pi(x))$.

Reductions defined by constant-depth symmetric circuits are closely related to reductions given by formulas of first-order logic (FO-reductions). In particular, it is easy to show that if a problem P is recucible to Q by means of an FO-reduction, it is reducible by means of a symmetric AC^0-reduction. It is known from the work of Lovász and Gács [14] that a version of the Boolean satisfiability problem SAT (suitably represented as a class of relational structures) is NP-complete under FO-reductions. It follows immediately that this problem cannot be solved by polynomial-size symmetric threshold circuits. If it could, all problems in NP would be solved by such circuit families and we have already established that this is not the case. A similar result holds for the class of Hamiltonian graphs, since this is also known to be NP-complete under FO-reductions by a construction due to Dahlhaus [6].

On the other hand, there are natural graph problems, such as 3-colourability, which are NP-complete under the usual sense of polynomial-time reductions but which are provably not NP-complete under FO-reductions (the latter follows from results in [8]). It remains an open question whether 3-SAT (with a natural representation as a class of relational structures) is NP-complete under FO-reductions. Nonetheless, it is possible to use FO-reductions from established lower bounds to show that neither 3-SAT nor 3-colourability is solvable by polynomial-size symmetric threshold circuits (these results follow from constructions in [3]). In the case of 3-colourability, there is also a construction that deploys bijection games (as in [7]) directly to show that this problem is not decidable by such families. These lower bounds should be contrasted with the fact that the existence of

perfect matchings in graphs is definable in the logic FPC [2] and therefore is decidable by polynomial-size symmetric threshold circuits.

The following table gives a list of problems that are known to be solvable by polynomial-size symmetric threshold circuits (under *upper bounds*) and a list of closely related problems that are provably not solvable by such families (under *lower bounds*). It should be noted that the lower bound results are *unconditional* in that they do not rely on unproved complexity-theoretic assumptions such as $\mathsf{P} \neq \mathsf{NP}$.

Upper Bounds	*Lower Bounds*
Circuit Value Problem	SAT
2-Colourability	3-Colourability
2-SAT	3-SAT
Perfect Matching	Hamiltonian Cycle
Linear Programming	XOR-SAT
Isomorphism on planar graphs	Isomorphism on bounded-degree graphs

Choiceless Computation

Another way of describing algorithms that work directly with relational structures rather than strings that encode them are the *abstract state machines* of Gurevich (see [4] and references therein). Here, the ability of a classical machine to make arbitrary choices is eschewed in favour of a high degree of parallelism. While such machines are universal, and can describe any algorithm at a high level of abstraction, it remains an open question whether every polynomial-time decidable class of structures is decidable in polynomial-time by such a machine. Specifically, Blass et al. [4] define the class CPTC of problems decidable in *choiceless polynomial time with counting* and ask the question whether this includes all of P. They conjecture that it does not, but the question remains unresolved. An interesting angle to this question is to examine it through the lens of circuits.

References

1. M. Anderson and A. Dawar. On symmetric circuits and fixed-point logics. In *31st Intl. Symp. Theoretical Aspects of Computer Science (STACS 2014)*, pages 41–52, 2014.
2. M. Anderson, A. Dawar, and B. Holm. Maximum matching and linear programming in fixed-point logic with counting. In *28th Annual ACM/IEEE Symp. Logic in Computer Science*, pages 173–182, 2013.
3. A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666–1683, 2009.
4. A. Blass, Y. Gurevich, and S. Shelah. Choiceless polynomial time. *Annals of Pure and Applied Logic*, 100:141–187, 1999.
5. J-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

6. E. Dahlhaus. Reduction to NP–complete problems by interpretation. In *LNCS 171*, pages 357–365. Springer-Verlag, 1984.
7. A. Dawar. A restricted second order logic for finite structures. *Information and Computation*, 143:154–174, 1998.
8. A. Dawar and E. Grädel. Properties of almost all graphs and generalized quantifiers. *Fundam. Inform.*, 98(4):351–372, 2010.
9. A. Dawar and D. Richerby. The power of counting logics on restricted classes of finite structures. In *CSL 2007:Computer Science Logic*, volume 4646 of *LNCS*, pages 84–98. Springer, 2007.
10. H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 1999.
11. E. Grädel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications*. Springer, 2007.
12. L. Hella. Logical hierarchies in PTIME. In *Proc. 7th IEEE Symp. on Logic in Computer Science*, pages 360–368, 1992.
13. N. Immerman. *Descriptive Complexity*. Springer, 1999.
14. L. Lovász and P. Gács. Some remarks on generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 23:27–144, 1977.