Succint circuit representations and leaf languages are basically the same concept

Bernd Borchert
Antoni Lozano

Report LSI–95–52–R

Succinct Circuit Representations and Leaf Languages are Basically the same Concept*

Bernd Borchert†
Universität Heidelberg

Antoni Lozano‡
Universitat Politècnica de Catalunya

Abstract

This paper connects two topics of Complexity Theory: The topic of *succinct circuit representations* initiated by Galperin and Wigderson [GW83], and the topic of *leaf languages* initiated by Bovet *et al.* [BCS92]. A Boolean circuit $c(x_1, \ldots, x_n)$ describes in a natural way a word of length 2^n, namely the result column in the truth table representation. This way, each language A determines its *succinct version* $S(A)$. In Bovet *et al.* [BCS92] it is shown how any language A (the *leaf language*) determines a complexity class called $\mathcal{C}(A, \overline{A})$. It will be shown for any language A that its succinct version $S(A)$ is polynomial-time many-one complete for $\mathcal{C}(A, \overline{A})$. Also it will be shown that if one uses for the succinct version branching programs instead of circuits then one will get complete problems for logspace classes.

1 Introduction

The topic of *succinct circuit representations* was started by Galperin and Wigderson [GW83]. The idea is the following. Instead of representing a graph having 2^n nodes by its adjacency matrix, it is represented by a circuit with two length-n vectors of variables: An assignment to a vector encodes a node, and for each assignment to the two vectors the circuit determines whether there is an edge from one node to the other. Graphs with a number of nodes which is not a power of 2 are encoded with the help of an extra output bit of the circuit which indicates whether the assignment encodes two nodes. If the graph is kind of regular, then

*This result was observed when the first author visited the site of the second author, supported by the EC Human Capital and Mobility project CoLoReT CHRX-CT93-0415.

†Im Neuenheimer Feld 294, 69120 Heidelberg, Germany, email: bb@math.uni-heidelberg.de

‡Departament L.S.I., Pau Gargallo 5, E08028 Barcelona, Catalonia, email: lozano@lsi.upc.es (correspondence author)

there may be a circuit which is logarithmically smaller than the adjacency matrix. Therefore it is no surprise that the computational complexity of several graph problems increases significantly if the succinct circuit representation is used, see [GW83, Wa86, PY86, LB89, BLT92, Ba95]. Balcázar *et al.* [BLT92] showed that the concept can easily be generalized from graph problems to general word problems. This idea will be adopted in this paper, with the modification that the indicator for the length of the word will be shifted from the output of the circuit to the input of the problem. This slight change of the definition of succinct circuit representation is necessary for the formulation of the main result, but also it will be argued that this definition is more natural.

The topic of *leaf languages* as a way to unify the definition of complexity classes was started by Bovet, Crescenzi, and Silvestri [BCS91, BCS92], and later by Vereshchagin [Ve94]. They show that many well-known complexity classes in the polynomial-time setting can be determined by just one language. The typical example is the class NP which is determined by the language consisting of the words which contain at least one letter 1. Several following investigations refined this approach, see [HLSVW93, Bo94, He94, HVW94, JMT94, Bo95].

In this paper, it will be shown that both topics from above are closely related: For any given language A, its succinct circuit representation is polynomial-time many-one complete ($\leq^{\mathrm{P}}_{\mathrm{m}}$-complete from now) for the class determined by A as a leaf language. For a slightly different definition of leaf language classes this result will be strengthened to polylogarithmic-time completeness.

Branching programs are an alternate way to describe Boolean functions. In Section 5 it will be shown that the succinct branching program version of a language is log-space many-one complete for a log-space analogue of leaf-languages defined by Bovet *et al.*

We noticed that H. Veith in his paper [Ve95] independently found our main result and some of the corollaries, formulated in the language of Finite Model Theory.

2 Succinct circuit representations

For a standard definition of *circuits* see for example [BDG88]. It is assumed that there is a linear order $<$ on the variables occuring in circuits. Let a circuit $c(x_1, \ldots, x_n)$ with n occuring variables $x_1 < \ldots < x_n$ be given. The circuit $c = c(x_1, \ldots, x_n)$ describes in a natural way a word $\mathrm{w}(c)$ of length 2^n: Its ith letter is the value of the ith assignment, where assignments are ordered lexicographically. In other words, $\mathrm{w}(c)$ is the result column of the truth table representation of c.

Examples. For $c = c(x_1, x_2) = x_1 \wedge x_2$ the word $\mathrm{w}(c)$ equals 0001, for $d = d(x_1, x_2, x_3) = \neg x_1 \vee (x_2 \wedge \neg x_3)$ the word $\mathrm{w}(d)$ equals 11110010.

In order to let circuits also describe words with a length not a power of 2, take any coded pair $\langle c, m \rangle$ of a circuit $c = c(x_1, \ldots, x_n)$ and a number $m \in \mathbb{N}$ in binary.

Let the *word described by c and m*, formally $w(c, m)$, be the length-m prefix of $w(c)$. Note that this implies $w(c) = w(c, 2^n)$. If a word x does not encode a circuit then $w(x, m)$ is defined to be the empty word.

Example. For the circuit $c = c(x_1, x_2) = x_1 \wedge x_2$ from above, $w(c, 0) = \epsilon$, $w(c, 1) = 0$, $w(c, 2) = 00$, $w(c, 3) = 000$, and $w(c, m) = 0001$ for any $m \geq 4$.

Definition 1 *The* succinct version $S(A)$ *of a language A is the set of coded pairs $\langle c, m \rangle$ such that $w(c, m) \in A$.*

Example. Let A be the language consisting of the words which contain at least one letter 1. Then $S(A)$ equals the set of coded pairs $\langle c, m \rangle$ such that the circuit c has a satisfying assignment among its first m assignments, a well-known NP-complete problem.

This is nearly the same concept as the definition of the succinct version sA of A in Balcázar *et al.* [BLT92], whereas here the length of the described string belongs to the problem input and is not indicated by some additional output bit of the circuit. In their paper [BLT92] and also in the original paper [GW83] it is implicitly assumed that the circuit is consistent in its length-indicating output bit, i. e. has the following property: If any assignment i evaluates the length-indicating output bit to 0, then also every lexicographically larger assignment evaluates it to 0.

Proposition 2 *The problem if a given circuit is consistent is* co-NP*-complete.*

Proof. The problem is obviously in co-NP. The co-NP-complete tautology problem can be reduced to it by the reduction which checks for a circuit c if $c(0, \ldots, 0) = 1$ and maps c to $\neg c$ in that case, and to some fixed non-consistent circuit (for example $x_1 \wedge x_2$) otherwise. □

Therefore, in the sense of [GW83, BLT92], given a circuit c with two outputs, there is no efficient way of checking if the circuit describes a word *at all*, unless P = NP.

3 Leaf languages

The following definition is from Bovet *et al.* [BCS92], besides that what is called here $C(A)$ is called $C(A, \overline{A})$ there.

Definition 3 (Bovet et al. **[BCS92])** *For a language A let $C(A)$ be the class consisting of the languages B such that there exist two polynomial-time computable functions $R : \Sigma^* \times \mathbb{N} \longrightarrow \{0, 1\}$ and $l : \Sigma^* \longrightarrow \mathbb{N}$ (numbers are represented in binary) such that*

$$x \in B \iff R(x, 0)R(x, 1)\ldots R(x, l(x)) \in A.$$

Example. Let A be the language consisting of all words which contain at least one letter 1. Then it is easy to verify that $\mathcal{C}(A) = \mathrm{NP}$.

Theorem 4 (Bovet *et al.* [BCS92], Borchert [Bo94]) *The set of classes $\mathcal{C}(A)$ is equal to the set of complexity classes which, with respect to $\leq^{\mathrm{p}}_{\mathrm{m}}$-reducibility, have a complete language and are closed downward.*

4 The main result

The theorem connecting succinct circuit representations and leaf languages will be stated, it may justify the titel of this paper.

Theorem 5 *For any language A it holds: The succinct version $S(A)$ of A is $\leq^{\mathrm{p}}_{\mathrm{m}}$-complete for the leaf language class $\mathcal{C}(A)$.*

Proof. First it is proven that for a fixed language A the language $S(A)$ belongs to the class $\mathcal{C}(A)$. Define $R : \Sigma^* \times \mathbb{N} \longrightarrow \{0,1\}$ to be the following function: It reads an input consisting of a pair (x,i); if x is not of the form $\langle c,m\rangle$ for a circuit c and a binary number m then R maps the input to (arbitrary) 0; and if $x = \langle c,m\rangle$ for a circuit c then R maps the input to the value of $c = c(x_1,\ldots,x_n)$ on the ith assignment for c if i is not larger than 2^n and to (arbitrary) 0 otherwise. Let l be the function $\Sigma^* \longrightarrow \mathbb{N}$ which maps an input $\langle c,m\rangle$ to the number 0 if c is not a circuit, and to m otherwise. R and l are computable in polynomial-time and it is easy to see that

$$\mathrm{w}(c,m) = R(\langle c,m\rangle,0)R(\langle c,m\rangle,1)\ldots R(\langle c,m\rangle,l(\langle c,m\rangle)),$$

both for the case that c is a circuit and also for the other case. Thus it holds $x \in S(A) \iff R(x,0)R(x,1)\ldots R(x,l(x)) \in A$. Therefore $S(A)$ is in $\mathcal{C}(A)$.

For the other direction let for a language B two polynomial-time computable functions $R : \Sigma^* \times \mathbb{N} \longrightarrow \{0,1\}$ and $l : \Sigma^* \longrightarrow \mathbb{N}$ be given such that for all x it holds:

$$x \in B \iff R(x,0)R(x,1)\ldots R(x,l(x)) \in A.$$

It has to be shown that B is $\leq^{\mathrm{p}}_{\mathrm{m}}$-reducible to $S(A)$. Let $f(x,y)$ be the polynomial-time computable function which first computes internally the number i representing which place in the lexicographical order y has among all words of length $|y|$, and then computes the value $R(x,i)$. Let $d_x(v_1,\ldots,v_{|y|})$ be the polynomial-size circuit which can be constructed from a program for f (by the methods of Savage [Sa72], see also [BDG88], Section 5.4.) such that for every $y = y_1\ldots y_{|y|}$ the value of $d_x(y_1,\ldots,y_{|y|})$ equals the value $f(x,y)$. The $\leq^{\mathrm{p}}_{\mathrm{m}}$-reduction from B to $S(A)$ is the following: Given an input x, construct the coded pair $\langle d_x(v_1,\ldots,v_{|l(x)|}),l(x)\rangle$. Note that $2^{|l(x)|}$ is an upper bound for $l(x)$. By the construction it is easy to verify that

$$R(x,0)R(x,1)\ldots R(x,l(x)) \in A \iff \langle d_x(v_1,\ldots,v_{|l(x)|}),l(x)\rangle \in S(A).$$

Therefore, B is $\leq_m^p$-reducible to $S(A)$ by the reduction function $x \longrightarrow \langle d_x(v_1, \ldots, v_{|l(x)|}), l(x)\rangle$. Note that the constructions in both parts of the proof are independent of A. □

Together with Theorem 4 the above theorem implies the following corollary.

Corollary 6 *Each polynomial-time many-one degree contains a succinct version.*

In Papadimitriou & Yannakakis [PY86], it was implicit in the main theorem that projection reducibility among languages implies $\leq_m^p$-reducibility among their succinct versions. Balcázar *et al.* [BLT92] generalized this from projection reducibility to logarithmic-time reducibility $\leq^{LT}$. Eiter *et al.* [EGM94] observed a generalization to polylogarithmic-time reducibility, which also follows as Corollary 8 from the above Theorem 5 and Theorem 3.4 in Bovet *et al.* [BCS92]. First the definition of $\leq^{PLT}$ is given, note that in [HLSVW93] and [BCS92] the symbols $\leq_m^{plt,bit}$ and $\leq_m^{pl}$, respectively, are used instead of $\leq^{PLT}$.

Definition 7 *A language A is* polylogarithmic-time reducible *to a language B ($A \leq^{PLT} B$) if there are two functions $R : \Sigma^* \times \mathbb{N} \longrightarrow \{0,1\}$ and $l : \Sigma^* \longrightarrow \mathbb{N}$ (numbers are represented in binary) computable in polylogarithmic time with random access to their input such that*

$$x \in A \iff R(x,0)R(x,1)\ldots R(x,l(x)) \in B.$$

It is easy to see that $\leq^{PLT}$-reducibility is reflexive and transitive and implies polynomial-time many-one reducibility. The function $x \longrightarrow R(x,0)R(x,1)\ldots R(x,l(x))$ is called a *polylogarithmic-time reduction function.* Note that a polylogarithmic-time reduction function is generally not a function computable in polylogarithmic-time (like l above) for which the output can only be of polylogarithmic size.

Corollary 8 (Eiter *et al.* [EGM94]) *If $A \leq^{PLT} B$ then $S(A)$ is $\leq_m^p$-reducible to $S(B)$.*

It should be stated that in [BCS92] it is shown that in a way the $\leq^{PLT}$-reducibility is an exact bound in the premise of Corollary 8: If $A \not\leq^{PLT} B$ then there is an oracle X such that $\mathcal{C}^X(A)$ is not a subset of $\mathcal{C}^X(B)$.

Observe that the connection between succinct representations and leaf languages provides an alternative way to express polynomial-time bit reductions ($\leq_m^{p,bit}$) studied in [HLSVW93].

Corollary 9 *$A \leq_m^{p,bit} B$ if and only if $A \leq_m^p S(B)$.*

In the proof of the main Theorem 5 it can be seen that the reduction of a language in $\mathcal{C}(A)$ to $S(A)$ is in fact very simple, besides that the function l really needs the full power of polynomial-time. The definition of leaf language classes

will be modified slightly so that it is possible to get completeness with respect to a stronger reducibility, namely polylogarithmic-time reducibility $\leq^{\mathrm{PLT}}$: Let the class $C'(A)$ be defined like the class $C(A)$ in Definition 3 besides that the function l is restricted to be a polylogarithmic-time reduction function. In the following theorem it is assumed that the pairing function and circuits are encoded in a finer way so that they are recognizable by polylogarithmic-time computations (which only have random access to the input).

Theorem 10 *For any language A it holds: The succinct version $S(A)$ of A is $\leq^{\mathrm{PLT}}$-complete for the leaf language class $C'(A)$.*

This implies that Corollary 8 has its following stronger version (this could of course also be proven directly):

Corollary 11 *If $A \leq^{\mathrm{PLT}} B$ then $S(A) \leq^{\mathrm{PLT}} S(B)$.*

Corollary 11 can be considered as an improved *Conversion Lemma* [BLT92, LB89] which allows us to sharpen Theorem 5 in [LB89].

Corollary 12 *Let $f(n) \geq \log n$ be a nondecreasing bound. Then, if A is $\leq^{\mathrm{PLT}}$-hard for $DTIME(f(n))$, $NTIME(f(n))$, $DSPACE(f(n))$, or $NSPACE(f(n))$, then $S(A)$ is $\leq^{\mathrm{PLT}}$-hard for $DTIME(f(2^n))$, $NTIME(f(2^n))$, $DSPACE(f(2^n))$, or $NSPACE(f(2^n))$ respectively.*

This way, any $\leq^{\mathrm{PLT}}$-complete language A for a class defined by a time or space bound generates an infinite sequence of $\leq^{\mathrm{PLT}}$-complete versions $S(A)$, $S(S(A))$, ... for exponentially larger classes.

5 Succinct branching program representations

Branching programs are a different way of representing Boolean functions, see the books of Meinel [Me89] and Wegener [We87]. The word $\mathrm{w}(b,m)$ for a branching program b and a number m is defined the same way as for circuits: take the length-m prefix of the result column of the truth table of the Boolean function represented by b.

Definition 13 *The* succinct branching program version $S^{\mathrm{BP}}(A)$ *of a language A is the set of coded pairs $\langle b,m\rangle$, where b is a coded branching program and m is a binary number, such that $\mathrm{w}(b,m) \in A$.*

The next definition is the log-space analogue of Definition 3.

Definition 14 *For a language A let $C^L(A)$ be the class consisting of the languages B such that there exist two log-space computable functions $R : \Sigma^* \times \mathbb{N} \longrightarrow \{0,1\}$ and $l : \Sigma^* \longrightarrow \mathbb{N}$ (numbers are represented in binary) such that*

$$x \in B \iff R(x,0)R(x,1)\ldots R(x,l(x)) \in A.$$

Theorem 15 *For any language A it holds: $S^{\mathrm{BP}}(A)$ is $\leq_m^{log}$-complete for $C^L(A)$.*

The proof is by the methods of Meinel [Me89], Theorem 1.1, who refers to unpublished papers by Cobham (1966) and Pudlak & Zak (1983).

Remark. A *formula* is a circuit in which each gate has fanout 1. The computational power of branching programs is between the one of formulas and circuits in the sense that formulas can be turned into branching programs of polynomial size, and branching programs can be turned into circuits of polynomial size (but the two opposite directions are open problems), see the book of Wegener [We87]. Because the concept of formulas is the least powerful among these three concepts one should expect an even finer distribution of computational problems if one looks at the succinct *formula* versions of languages.

6 Concluding remarks

For any language A it was shown that its succinct circuit version is $\leq_m^{\mathrm{P}}$-complete for the class determined by A as a leaf language in the sense of Bovet, Crescenzi, and Silvestri [BCS91,BCS92]. This result was strengthened to polylogarithmic-time completeness by modifying the definition of leaf language classes.

We also proved that branching programs are the right model to get a similar result for the leaf language classes obtained with log-space computable functions. This result raises the question whether it is possible to show a similar fact for the log-space leaf languages $\mathbf{Leaf}^L(\cdot)$ and *Balanced*-$\mathbf{Leaf}^L(\cdot)$ defined by Jenner *et al.* in [JMT94], where an NL-machine is involved.

Note that the main Theorem 5 could in some sense be considered as a generalization of the classical result in [Co71, Ka72] that SAT is NP-complete, see the examples after Definitions 1 and 3. More generally, the theorem may be considered to be a formalization of the folklore knowledge that, given a class determined by some leaf language, the corresponding circuit problem is complete for the class, like the set of circuits c such that a majority (an odd number) of the assignments for c evaluates c to 1 is complete for PP (for $\oplus$P).

The notion $\mathrm{w}(c, m)$ may give another kind of resource-bounded Kolmogorov complexity, namely *circuit Kolmogorov complexity*: For a string x let its *circuit Kolmogorov complexity* be the size of the shortest string $\langle c, m\rangle$ such that $\mathrm{w}(c, m) = x$. Several notions from Kolmogorov Complexity Theory can be transferred to this modified concept, for example, a *circuit incompressible string* is a string whose circuit Komogorov complexity is not smaller than its length.

Acknowledgements

The authors are grateful to José Balcázar for helpful comments.

References

[Ba95] J. L. Balcázar. The complexity of Searching Implicit Graphs, *Proc. 22nd ICALP, Springer LNCS* 944, 1995, pp. 208–219.

[BCS91] D. P. Bovet, P. Crescenzi, R. Silvestri. Complexity classes and sparse oracles, *Proc. 6th Structure in Complexity Theory Conference*, 1991, pp. 102–108.

[BCS92] D. P. Bovet, P. Crescenzi, R. Silvestri. A uniform approach to define complexity classes, *Theoretical Computer Science* **104**, 1992, pp. 263–283.

[BDG88] J. L. Balcázar, J. Díaz, J. Gabarró. Structural Complexity I, *Springer Verlag*, 1988.

[BLT92] J. L. Balcázar, A. Lozano, J. Torán. The complexity of algorithmic problems on succinct instances, *Computer Science*, edited by R. Baeza-Yates and U. Manber, Plenum Press, N.Y., 1992, pp. 351–377.

[Bo94] B. Borchert. Predicate classes and promise classes, *Proc. 9th Structure in Complexity Theory Conference*, 1994, pp. 235–241.

[Bo95] B. Borchert. On the acceptance power of regular languages, *Theoretical Computer Science* **148**, 1995, pp. 207–225.

[Co71] S. A. Cook. The complexity of theorem proving procedures, *Proc. 3rd Annual ACM Symposium on the Theory of Computing (STOC)*, 1971, pp. 151–158.

[EGM94] T. Eiter, G. Gottlob, H. Maninila. Adding Disjunction to Datalog, *Proc. 12th ACM SIGACT SIGMOD-SIGART Symposium on Principles of Databases Systems*, 1994.

[GW83] H. Galperin, A. Wigderson. Succinct representations of graphs, *Information and Control* **56**, 1983, pp. 183–198.

[He94] U. Hertrampf. Über Komplexitätsklassen, die mit Hilfe von k-wertigen Funktionen definiert werden, Habilitationsschrift, Universität Würzburg, 1994.

[HLSVW93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, K. Wagner. On the power of polynomial-time bit-computations, *Proc. 8th Structure in Complexity Theory Conference*, 1993, pp. 200–207.

[HVW94] U. Hertrampf, H. Vollmer, K. W. Wagner. On balanced vs. unbalanced computation trees, Technical Report No. 82, Institut für Informatik, Universität Würzburg, 1994.

[JMT94] B. Jenner, P. McKenzie, D. Thérien. Logspace and logtime leaf languages, *Proc. 9th Structure in Complexity Theory Conference*, 1994, pp. 242–254.

[Ka72] R. Karp. Reducibility among combinatorial problems, *Complexity of Computer Computations*, edited by R. E. Miller and J. W. Thatcher, Plenum Press, N. Y., 1972, pp. 85–103.

[LB89] A. Lozano, J. L. Balcázar. The complexity of graph problems for succinctly represented graphs, *Proc. 15th Graph-Theoretic Concepts in Computer Science, Springer LNCS* 411, 1989, pp. 277–286.

[Me89] C. Meinel. Modified Branching Programs and their Computational Power, *Springer LNCS* 370, 1989.

[PY86] C. H. Papadimitriou, M. Yannakakis. A note on succinct representations of graphs, *Information and Control* **71**, 1986, pp. 181–185.

[Sa72] J. E. Savage. Computational work and time of finite machines, *Journal of the ACM* **19**, 1972, pp. 660–674.

[Ve94] N. K. Vereshchagin. Relativizable and nonrelativizable theorems in the polynomial theory of algorithms, *Russian Acad. Sci. Izv. Math.* **42**, 1994, pp. 261–298.

[Ve95] H. Veith. Succinct Representation and Leaf Languages, Technical Report CD-TR 95/81, TU Wien, 1995 (this paper is announced as ECCC report TR95-048)

[Wa86] K. W. Wagner. The complexity of combinatorial problems with succinct input representation, *Acta Informatica* **23**, 1986, pp. 325–356.

[We87] I. Wegener. *The Complexity of Boolean Functions*, Teubner Verlag, Stuttgart, 1987.