On Universally Easy Classes for NP-complete Problems

Erik D. Demaine* Alejandro López-Ortiz† J. Ian Munro*

Abstract

We explore the natural question of whether all **NP**-complete problems have a common restriction under which they are polynomially solvable. More precisely, we study what languages are *universally easy* in that their intersection with any **NP**-complete problem is in **P**. In particular, we give a polynomial-time algorithm to determine whether a regular language is universally easy. While our approach is language-theoretic, the results bear directly on finding polynomial-time solutions to very broad and useful classes of problems.

1 Introduction and Overview

Empirically, it has been observed that some classes of instances result in polynomial-time algorithms for what are otherwise **NP**-complete problems. For example, COLOURING, CLIQUE and INDEPENDENT SET are well-known **NP**-complete problems that have polynomial-time solutions when restricted to interval graphs [7]. But this property is not universal: list coloring in graphs and determining the existence of k vertex-disjoint paths (where k is part of the input) remain **NP**-complete for interval graphs [1, 6].

This leads to a natural question about the existence of universally easy classes for **NP**-complete problems. It turns out that such languages exist, and it seems difficult to give a complete characterization. Thus we focus on two natural classes of languages: regular languages and context-free languages. In particular, we characterize precisely which regular languages are universally easy in the sense defined in Section 2.

Various particular restrictions have been studied before; see for example Brandstadt, Le, and Spinrad [8] for a detailed survey of graph classes.

2 Definitions

For simplicity of exposition, assume that the alphabet $\Sigma = \{0, 1\}$. We use interchangably the notions of a language, a decision problem, and a class of instances.

DEFINITION 2.1. *The* restriction *of a problem P to a class of instances C is the intersection $P \cap C$.*

DEFINITION 2.2. *Given an **NP**-complete problem P, a class C is a* simplifying restriction *if the restriction of P to C is not **NP**-complete, and C is a* polynomial restriction *if there is a polynomial-time Turing machine that recognizes the restriction of P to C.*

Of course this definition is vacuous if $\mathbf{P} = \mathbf{NP}$

DEFINITION 2.3. *A language $C \in \mathbf{NP}$ is* universally simplifying *if it is a simplifying restriction of all **NP**-complete problems.*

DEFINITION 2.4. *A language $C \in \mathbf{P}$ is* universally polynomial *if it is a polynomial restriction of all **NP**-complete problems.*

3 Easy Languages

A natural question is whether there exist universally simplifying languages if $\mathbf{P} \neq \mathbf{NP}$. This can be readily answered in the affirmative by noticing that all finite languages are universally polynomially, which is not very enlightening. A more general class to consider is regular languages, which can be characterized according to their simplicity.

DEFINITION 3.1. *The* growth function *of a language L is the function $\gamma_L(n) = |\{x \in L : |x| \leq n\}|$. A language is* sparse *if its growth function is bounded from above by a polynomial, and is* exponentially dense *if the growth function is bounded from below by $2^{\Omega(n)}$.*

THEOREM 3.1. *A sparse language L is either universally simplifying or universally polynomial.*

Proof. Consider a sparse language L. If it is universally simple, there is nothing to show. If it is not universally simple, there is a problem $P \subseteq \Sigma^*$ such that the restriction $P \cap L$ is **NP**-complete. Because $P \cap L \subseteq L$, this restriction is also a sparse set, and it is **NP**-complete. Mahaney [5] proved that if a language is sparse and **NP**-complete, then $\mathbf{P} = \mathbf{NP}$. Therefore $\mathbf{P} = \mathbf{NP}$ and consequently $P \cap L \in \mathbf{P}$ for all **NP**-complete languages L. □

*Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, email: {eddemaine, imunro}@uwaterloo.ca

†Faculty of Computer Science, University of New Brunswick, P. O. Box 4400, Fredericton, N. B. E3B 5A3, Canada, email: alopezo@unb.ca

Definition 3.2. *A loop in a DFA A is a directed cycle in the state graph of A.*

Definition 3.3. *Let C_1 and C_2 be two DFA loops such that neither is a subgraph of the other. We say that C_1 and C_2* interlace *if there is an accepting computation path in the DFA containing the sequence $C_1 \cdots C_2 \cdots C_1$ or the sequence $C_2 \cdots C_1 \cdots C_2$.*

The following theorem was proved by Flajolet [2]. Our proof uses a constructive argument needed for Theorem 3.3.

Theorem 3.2. *Every regular language is either sparse or exponentially dense.*

Proof. Consider $L \subseteq \Sigma^*$ recognized by a DFA A. If L is finite, then it is trivially sparse; otherwise, and contains strings of arbitrary length. The pumping lemma states that any DFA accepting a sufficiently large string has at least one loop in its state graph, which can be traversed (pumped) zero or more times.

If A has no interlacing loops, then each accepting computation T_k can be written as $T_k = (s_1, t_1, s_2, t_2, \ldots, C_1^*, s_i, t_i, \ldots, C_j^*, \ldots, q_f)$, where the s_i's are states, t_i's are transition symbols, C_i's are disjoint loops, q_f is a final state of A, and $s_i \neq s_j$ for all $i \neq j$. Notice that, apart from the actual value represented by the Kleene star, there are only finitely many such orderings of states and loops, and thus the language L can be written as the finite union of T_k's. Let j_k denote the number of loops and r_k the number of states in T_k. Then the total number of strings of length n generated by T_k is at most $\binom{n-r_k}{j_k} = O(n^{j_k})$. A union of finitely many such sets, each with a polynomially bounded number of strings of length n, is itself polynomially bounded and therefore sparse.

We now proceed to show that a DFA A with interlacing loops accepts an exponentially dense language. Consider an accepting computation path T_k of A with interlacing loops, that is, $T_k = (s_1, t_1, \ldots, C_1, \ldots, C_2, \ldots, C_1, \ldots, q_f)$. Now we pump a subsequence, obtaining $T_k = (s_1, t_1, \ldots, [C_1^*, \ldots, C_2^*, \ldots]^*, C_1, \ldots, q_f)$. We replace with a special character w_1 the sequence of transitions taken in the $(C_1, \ldots)$ portion of T_k above, and with w_2 the transitions in $(C_2, \ldots)$. Then T_k can be rewritten as the regular expression $t_1 \cdots \{w_1, w_2\}^* w_1 \cdots t_f$. From this it follows that there are at least 2^{n-r_k} strings of length n in $(\Sigma \cup \{w_1, w_2\})^*$. Thus $\gamma_L(n) \geq 2^{(n-r_k)/m}$, where $m = \max\{|w_1|, |w_2|\}$, which implies $\gamma_L(n) = 2^{\Omega(n)}$ □

Theorem 3.3. *No exponentially dense regular language L is universally simplifying.*

Proof. From the proof of Theorem 3.2 we know that a DFA accepting L necessarily contains interlacing loops. We define an injective polynomial-time transformation $F : \Sigma^* \to L$ as follows. Let T_k be a computation path with interlacing loops, i.e., $T_k = (t_1 \cdots \{w_1, w_2\}^* \cdots t_f)$. Now we map 0 to w_1, and 1 to w_2. So a string $x_1 x_2 \cdots x_j \in \Sigma^*$ is mapped to $w_{x_1+1} w_{x_2+1} \cdots w_{x_j+1}$. Note that F and its inverse can be computed in polynomial time.

Given any **NP**-complete language P, we define $\hat{P} = \{x \in L : x = F(y) \text{ for some } y \in P\}$. $\hat{P}$ is **NP**-complete, because the y's together with polynomial length certificates from P serve as certificates for $\hat{P}$, and F is a reduction from P to $\hat{P}$. Because $\hat{P} \subseteq L$, we have $\hat{P} \cap L = \hat{P}$, which is **NP**-complete. Thus L is not universally simplifying. □

Corollary 3.1. *If an exponentially dense regular language is universally polynomial, then* $\mathbf{P} = \mathbf{NP}$.

Note that the property of interlacing loops for regular languages, and hence "easiness," can be tested in polynomial time.

4 Open Problems

Recently the sparse/exponential-density property in Theorem 3.2 has been generalized to context-free languages [3, 4]. We conjecture that our results also generalize to CFLs; the main obstruction is in finding a polynomially constructive proof.

References

[1] E. M. Arkin, E. B. Silverberg. Scheduling jobs with fixed start and end times. *DAM*, 18(1):1–8, 1987.

[2] P. Flajolet. Analytic models and ambiguity of context-free languages. *TCS*, 49:283–309, 1987.

[3] L. Ilie, G. Rozenberg, and A. Salomaa. A characterization of poly-slender context-free languages. *Theoret. Informatics Appl.*, 34(1):77–86, 2000.

[4] R. Incitti. The growth function of context-free languages. To appear in *TCS*, 2000.

[5] S. R. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *JCSS*, 25(2):130–143, 1982.

[6] S. Natarajan and A. P. Sprague. Disjoint Paths in Circular Arc Graphs. *Nordic J. Comput.*, 3(3):256–270, Fall 1996.

[7] C. H. Papadimitrou. *Computational Complexity.* Addison-Wesley, 1994.

[8] A. Brandstadt, V. B. Le and J. P. Spinrad. *Graph Classes: A Survey.* SIAM Monographs on Discrete Mathematics and Applications, 1999.