

Chapter 2

Complexity in Linguistics

Abstract The topic of language complexity has surfaced in many different contexts and can be measured in many different ways. In this chapter, I discuss notions relevant to the computational and descriptive complexity of language. I introduce the notion of ‘complexity class’ (e.g. P and NP), the corresponding logical distinctions (e.g. definability), and the Cobham-Edmonds thesis identifying the class of practically computable problems with P. Then, I survey how the complexity notions have been applied in the study of syntax and semantics of natural language. This discussion culminates in putting forward Ristad’s Thesis, claiming that our everyday language is semantically bounded by the properties expressible in the existential fragment of second-order logic (belongs to NP). Finally, I discuss, very common in formal semantics, restriction to finite interpretations. This chapter gives, therefore, an additional argument for studying the computational complexity of natural language expressions.

Keywords Computational complexity · Tractability · Chomsky’s hierarchy · Grammar · Semantics · Anaphora · Ristad’s thesis · Semantic bounds · Reasoning · Finiteness

The topic of language complexity has surfaced in many different contexts and can be measured in many different ways.¹ In this chapter we discuss notions relevant to the computational and descriptive complexity of language: computational and descriptive complexity, complexity of syntax and semantics, and finally the finiteness restriction.

¹The book focuses on computational and descriptive complexity of language, but there are many other aspects of complexity, like lexical, information-theoretic (Kolmogorov complexity), structural, or functional complexity.

2.1 Computational Complexity

With the development of programming practice it has become apparent that for some computable problems finding effective algorithms is an art in itself. Some problems need too much computational resources, e.g., time or memory, to be practically computable. Computational complexity theory is concerned with the amount of resources required for the execution of algorithms and, hence, the inherent difficulty of computational problems.² This means that the theory does not deal directly with concrete algorithmic procedures, but instead studies the abstract computational properties of problems. It aims at explaining why for some computable questions we cannot come up with useful algorithms.

An important goal of computational complexity theory is to categorize computational problems via complexity classes, and in particular, to identify efficiently solvable problems and draw a line between tractability and intractability. From our perspective the most important distinction is that between the problems which can be computed in polynomial time with respect to the size of the problem, i.e., relatively quickly, and those which are believed to have only exponential time algorithmic solutions. The class of problems of the first type is called PTIME (P, for short). Problems belonging to the second class are referred to as NP-hard problems (see Appendix A.2.3 for mathematical details). Intuitively, a problem is NP-hard if there is no ‘snappy’ algorithm for solving it, and the only way to deal with it is by using brute-force methods: searching through all possible combinations of elements over a universe. In other words, NP-hard problems lead to combinatorial explosion.

Notice that all complexity claims attempting to explain empirical reality make sense only under the assumption that the complexity classes defined in the theory are essentially different. These inequalities are sometimes extremely difficult to prove.³ One famous complexity question deserves mentioning here. NP-hard problems are by definition at least as difficult as those in the NPTIME (NP) class. The problems in this latter class can be computed by nondeterministic Turing machines in polynomial time. NP-complete problems are NP-hard problems belonging to NPTIME, hence they are intuitively the most difficult problems among the NPTIME problems. In particular, it is known that $P = NP$ if any NP-complete problem is PTIME computable. Unfortunately, we do not know whether $P = NP$, i.e., whether there is any NP-complete problem that is computable in polynomial time. This famous question is worth at least the prize of \$1,000,000 offered by the Clay Institute of Mathematics for solving one of the seven greatest open mathematical problems of our time. However, even without a mathematical answer at hand, the experience and practice of computational complexity theory suggest that these two classes are different.

²See Appendix A.2 for more details.

³We discuss these, mainly technical, issues in Appendix A.2.3.

Complexity Assumption $P \neq NP$.

The satisfiability problem for propositional formulae SAT is to decide whether a given propositional formula is not a contradiction. Let φ be a propositional formula with p_1, \dots, p_n distinct variables. Let us use the well-known algorithm based on truth-tables to decide whether φ has a satisfying valuation. How big is the truth-table for φ ? The formula has n distinct variables occurring in it, and therefore the truth-table has 2^n rows. One can get a sense of the growth rate by comparing the number of rows for $n = 10$ (1,024) and for $n = 20$ (1,048,576). In the worst case, to decide whether φ is satisfiable we have to check all rows. Hence, in such a case, the time needed to find a solution is exponential with respect to the number of different propositional letters of the formula. A seminal result of computational complexity theory states that this is not a property of the truth-table method but of the inherent complexity of the satisfiability problem.

Theorem 2.1 (Cook 1971) *SAT is NP-complete.*

The following thesis, formulated by independently by Cobham (1965) and Edmonds (1965), captures the intuitive notion of tractability:

Cobham-Edmonds Thesis *The class of practically computable problems is identical to the PTIME class, that is the class of problems which can be computed by a deterministic Turing machine in a number of steps bounded by a polynomial function of the length of a query.*

The thesis is accepted by most computer scientists. For example, Garey and Johnson (1990) claim:

Most exponential time algorithms are merely variations on exhaustive search, whereas polynomial time algorithms generally are made possible only through the gain of some deeper insight into the nature of the problem. There is wide agreement that a problem has not been ‘well-solved’ until a polynomial time algorithm is known for it. Hence, we shall refer to a problem as intractable, if it is so hard that no polynomial time algorithm can possibly solve it. (Garey and Johnson 1990, p. 8)

2.2 Syntax

Noam Chomsky has led the way to viewing language from the computational perspective (see, e.g., Chomsky 1957, 1965). In the early 1950s he captured language’s recursive nature, or, in Wilhelm von Humboldt’s famous words, the linguistic ability to make ‘infinite use of finite means,’ inventing formal language theory. Chomsky’s complexity hierarchy of finite-state, context-free, and context-sensitive languages associated with their automata counterparts (see Appendix A.2.1) changed linguistics and prepared it for interactions with computer science and cognitive science.

Chomsky’s insight was to abstractly measure the computational complexity of a language. First of all, even though the sentences we encounter in natural language

are all of bounded length—certainly less than 100 words long—Chomsky assumed that they might be arbitrarily long. This assumption directly led to the discovery of the computational model of language generation and the complexity hierarchy.

Next, using his complexity hierarchy Chomsky asked in which complexity class natural language lies. He demonstrated early the inadequacy of finite-state description of natural languages and claimed that natural languages are context-free (see Chomsky 1957). This was a groundbreaking way of thinking. First of all, it has shown how one can mathematically measure certain types of complexity in natural language. In addition, it has placed a methodological constraint on linguistic theories of syntax; they should be able to account at least for the context-free linguistic structures. Actually, it has also started a long-standing debate about whether natural language is context-free or not.⁴

Chomsky noticed very early that studying whether a grammar can generate all strings of a given language, so-called weak generative capacity, can serve only as a kind of ‘stress test’ that does not tell us much unless a grammar fails the test. He wrote:

The study of generative capacity is of rather marginal linguistic interest. It is important only in those cases where some proposed theory fails even in weak generative capacity — that is, where there is some natural language even the *sentences* of which cannot be enumerated by any grammar permitted by this theory. [...] It is important to note, however, that the fundamental defect of many systems is not their limitation in weak generative capacity but rather their many inadequacies in strong generative capacity.⁵ [...] Presumably, discussion of weak generative capacity marks only a very early and primitive stage of the study of generative grammar.

(Chomsky 1957, p. 60f)

In the subsequent years this belief has led to deeper study of generative formalisms. In particular, using computational complexity one has been able to answer the question whether some proposed generative formalism is plausible in the sense of being ‘easy enough to process.’ To be more precise, computational complexity of parsing and recognition has become a major topic along with the development of computational linguistics.⁶ In general, the results show that even for relatively simple grammatical frameworks some problems might be intractable. For example, regular and context-free languages have tractable parsing and recognition problems. However, Lambek Grammars, Tree-Adjoining Grammars, Head-Driven Phrase Structure Grammar, and context-sensitive grammars give rise to intractable problems.

⁴See, e.g., Pullum and Gazdar (1982); Shieber (1985); Culy (1985), and Manaster-Ramer (1987).

⁵Chomsky uses the term ‘strong generative capacity’ to refer to the set of structures (trees) that can be generated by a grammar.

⁶The early results achieved are summarized in a book by Barton et al. (1987). A more recent survey is found in Pratt-Hartmann (2008).

2.3 Descriptive Syntax

One can specify the complexity of grammars not only in terms of generative mechanisms, but also via a set of general constraints to which sentences generated by these grammars have to conform. On this view, a string (or a tree) is grammatical if it satisfies these constraints. How can we define complexity from this perspective? We can identify the complexity of a grammar with the logical complexity of the formulae which express the constraints. In other words, we ask here about descriptive complexity of grammars in a similar way to how we will be asking about descriptive complexity of quantifiers in the following chapters.

We measure the complexity of the sentences that define constraints by discovering how strong a logic we need to formulate them. Particularly, we refer to fragments of second-order logic (see, e.g., Rogers 1983) or various extensions of modal logic (see, e.g., Kracht 2003). For illustration, the two best known results of this approach are as follows. In his seminal paper Büchi (1960) showed that a language is definable by the so-called monadic fragment of second-order logic if and only if it is regular. McNaughton and Papert (1971) proved that a set of strings is first-order definable if and only if it is star-free. These two results have their counterpart in modal logic: the temporal logic of strings captures star-free languages and propositional dynamic logic captures regular languages (see, e.g., Moss and Tiede 2006).

It is often possible to draw conclusions about computational complexity from such descriptive results. It is enough to know the complexity of the corresponding logics. For instance, it is known that monadic second-order logic on trees is decidable but intractable, and therefore, many linguistic questions about parse trees which are formulated in this logic might be ‘difficult’ to answer. There is also a strong correspondence between existential second-order logic and NP (see Sect. 7.1 for more details):

Theorem 2.2 (Fagin 1974) Σ_1^1 captures NP.

2.4 Semantics

Some of the earlier work combining computational complexity with semantics can be found in Sven Ristad’s book ‘The Language Complexity Game’ (1993). The author carefully analyzes the comprehension of anaphoric dependencies in discourse. He considers a few approaches to describing the meaning of anaphora and he proves their complexity. Finally, he concludes that the problem is inherently NP-complete and that all good formalisms accounting for it should be exactly as strong. The book contains not only a technical contribution to the semantic theory of anaphora, but also some methodological claims on the role of computational complexity in linguistics.

First of all, Ristad claims that natural language contains constructions whose semantics is essentially NP-complete:

The second major result of this monograph is the thesis that language computations are NP-complete. This complexity thesis is a substantive, falsifiable claim about human language that is directly supported by the quiescent state of the language complexity game.

(Ristad 1993, p. 14)

Secondly, Ristad suggests that a good linguistic theory cannot be too strong:

The central consequence of this complexity thesis for human language is that empirically adequate models (and theories) of language will give rise to NP-completeness, under an appropriate idealization to unbounded inputs. If a language model is more complex than NP, say PSPACE-hard, then our complexity thesis predicts that the system is unnaturally powerful, perhaps because it overgeneralizes from the empirical evidence or misanalyses some linguistic phenomena.

(Ristad 1993, p. 15)

In other words, Ristad claims that every semantic model has to be at least NP-hard to be able to capture complex linguistic phenomena. On the other hand, in his opinion, every computationally stronger formalism must overgeneralize linguistic capacities. This second methodological claim can be seen as a kind of semantic Ockham's razor. Summing up, Ristad proposes NP-completeness as a methodological test for the plausibility of linguistic theories.

We share Ristad's intuitions, however, we prefer to restrict claims of this sort to some part of language as the whole of natural language contains expressions whose semantic complexity go beyond practical computability. Nevertheless, we can see a natural intuition supporting the use of a concept of *natural language* that excludes 'technical' expressions. However, in this narrow sense we prefer to use the term *everyday language*. In a sense we understand 'everyday language' here as a pretheoretic part of natural language.

In this book we study the complexity of natural language quantifiers via their definability properties. In other words, we use descriptive complexity theory, i.e., we draw complexity conclusions on the basis of logical descriptions of semantics. Fagin's theorem (Theorem 7.2) shows that existential second-order properties correspond to NPTIME. From this perspective, we state the following:

Ristad's Thesis: *Our everyday language is semantically bounded by the properties expressible in the existential fragment of second-order logic.* (Mostowski and Szymanik 2012)

In other words, we claim that everyday language can be described in the existential fragment of second-order logic (see Definition 7.1). If some property is not definable by any Σ_1^1 -formula, then it falls outside the scope of everyday language. For example, the quantifiers 'there exists', 'all', 'exactly two', 'at least four', 'every other', and 'most' belong to everyday language. The counterexample is the notion 'there exist at most countably many', which is not definable by any Σ_1^1 -formula.

Let us give one argument in favor of accepting this methodological statement.⁷ The core sentences of everyday language are sentences which can be more or less effectively verifiable. In the case of small finite interpretations this means that their

⁷The other one, which is more domain specific, will be formulated in Sect. 9.3.4.

truth-value can be practically computed, directly or indirectly. Direct practical computability means that there is an algorithm which for a given finite interpretation computes the truth-value in a reasonable time. However, we frequently understand sentences indirectly, evoking their inferential dependencies with other sentences. Let us consider the following three sentences:

- (1) There were more boys than girls at the party.
- (2) At the party every girl was paired with a boy.
- (3) Peter came alone to the party.

We know that sentence (1) can be inferred from sentences (2) and (3). Then we can establish the truth-value of sentence (1) indirectly, knowing that sentences (2) and (3) are true.

In the case of NPTIME problems the nondeterministic behavior of an algorithm can be described as follows:

Firstly, choose a certificate of a size polynomially depending on the size of the input. Then apply a PTIME algorithm for finding the answer. The nondeterministic algorithm answers YES exactly when there is a certificate for which we get a positive answer.

(Garey and Johnson 1990, p. 28)

Such certificates can be viewed as proofs. When we have a proof or an argument for a statement, then we can easily check whether it is correct and therefore whether the sentence is true. In this sense one may say that NPTIME sentences (Σ_1^1 -properties) are practically justifiable. Suppose that we know that the following are true statements:

- (4) Most villagers are communists.
- (5) Most townsmen are capitalists.
- (6) All communists and all capitalists hate each other.

From these sentences we can easily infer the NP-complete branching interpretation of the following sentence⁸:

- (7) Most villagers and most townsmen hate each other.

Sentences (4), (5), and (6) have to be given or guessed. They are in a sense certificates or proofs of the truth of sentence (7).

In this sense sentences with NPTIME meaning—or, by Fagin's theorem (see Theorem 7.2) Σ_1^1 -expressible sentences—are indirectly verifiable. Moreover, NPTIME seems to exactly capture indirect verifiability.

Of course, to make the story complete one needs to define tractable inferences, that is, restrict the class of computationally plausible inferences. We will not pursue this topic in this book, but again computational complexity could be used to this end. Intuitively speaking, we would like to understand why, for instance, the syllogistic reasoning in (*) is much simpler than the reasoning in (**).

⁸For a discussion on the computational complexity of sentences like this, see Sect. 7.3 and Chap. 9.

This puts our position very close to Ristad's perspective. Additionally, it provides a way to connect our investigations with dynamic formalisms describing other aspects of meaning. Such an integration is needed to account for the big picture of natural language meaning and understanding.⁹

2.5 Finite Universes

The above computational perspective leads us to another methodological aspect of our approach. Most of the authors considering the semantics of natural language are interested only in finite universes. This is also common in works devoted to natural language quantifiers:

In general these cardinals can be infinite. However, we now lay down the following constraint:

(FIN) Only finite universes are considered.

This is a drastic restriction, no doubt. It is partly motivated by the fact that a great deal of the interest of the present theory of determiners comes from applications to natural language, where this restriction is reasonable.

(Westerståhl 1984, p. 154)

In typical communication situations we indeed refer to finite sets of objects. For example, the intended interpretations of the following sentences are finite sets:

- (8) Exactly five of my children went to the cinema.
- (9) Everyone from my family has read *Alice's Adventures in Wonderland*.

In many cases the restriction to finite interpretations essentially simplifies our theoretic considerations.

First of all, there is a conceptual problem with computational complexity theory for an infinite universe. Even though from the mathematical point of view we can work with infinite computations, the classic study of resource bounds in such a setting does not make much sense as we would need infinite time and infinite memory resources. In addition, even in such cases one may need only finitely many states or loops. Hence, in the end it may be a matter of proposing reasonable definitions and finding interesting applications of infinite computations in cognitive modeling. For example, they can be useful as a framework for some cognitive processes which at least in theory are supposed to continue working indefinitely, like equilibration (see Kugel 1986, for more examples).

From the semantic perspective there is, however, an additional problem with infinite universes. Namely, defining an intuitive semantics for natural language in an arbitrary universe is a very difficult task. As an example of potential troublemakers

⁹One interesting question is how Ristad's Thesis fits within cognitive science, where complexity bounds are considered as methodological guidelines for computational models (see, e.g., Rooij 2008; Isaac et al. 2014 and Sect. 8.5).

we can give quantifiers, like ‘more’ or ‘most’. We usually reduce the problem of their meaning to a question about the relations between sets of elements (see Chap. 3). In finite universes there is an easy and commonly accepted solution, which is to compare the sizes of these sets. But extending this solution to the infinite case seems to be very counterintuitive. Let us have a look at the following sentence.

(10) There are more nonprimes than primes among integers.

The sentence is false if we interpret ‘more’ in terms of cardinalities. Intuitively, we agree that the sentence is meaningful and even true. One possible solution to this problem is to treat such quantifiers as measure quantifiers. In infinite universes we can compare quantities by proper measure functions, which are nonlogical and context-dependent concepts (see Krynicki and Mostowski 1999, for more details). A related approach would be to consider various sampling algorithms associated with each quantifier—this would give rise to a probabilistic semantics of quantifiers. One can imagine such a semantics producing a standard meaning of quantifiers on finite models and working intuitively for infinite universes. However, in this book we work with finite universes as we are mainly interested in the computational aspects of the semantics of natural language.

In this part of the book we have discussed some motivations and philosophical assumptions behind the research presented in the next chapters. These assumptions do not have a direct influence on our technical results. However, we believe that they give an additional argument for studying the computational complexity of natural language expressions. The rest of the book provides an overview of the most common quantificational fragments of natural language from the logical, computational, and cognitive perspectives. We hope that our technical insights will give additional arguments in favor of the methodological approaches discussed above.

References

- Barton, E. G., Berwick, R., & Ristad, E. S. (1987). *Computational Complexity and Natural Language*. The MIT Press: Bradford Books.
- Büchi, J. (1960). Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6, 66–92.
- Chomsky, N. (1957). *Syntactic Structures* (2nd ed.). Walter de Gruyter.
- Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press.
- Cobham, A. (1965). The intrinsic computational difficulty of functions. In Y. Bar-Hillel (Ed.), *Proceedings of the 1964 Congress for Logic, Methodology, and the Philosophy of Science* (pp. 24–30). Jerusalem: North-Holland Publishing.
- Cook, S. A. (1971). The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the Third Annual ACM Symposium on Theory of Computing* (pp. 151–158). New York: ACM Press.
- Culy, C. (1985). The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, 8(3), 345–351.
- Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17, 449–467.

- Fagin, R. (1974). Generalized first-order spectra and polynomial time recognizable sets. In R. Karp (Ed.), *Complexity of Computation*, SIAM–AMS Proceedings (Vol. 7, pp. 43–73). American Mathematical Society.
- Garey, M. R., & Johnson, D. S. (1990). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: W. H. Freeman.
- Isaac, A., Szymanik, J., & Verbrugge, R. (2014). Logic and complexity in cognitive science. In A. Baltag, & S. Smets (Eds.), *Johan van Benthem on Logic and Information Dynamics*. Outstanding Contributions to Logic (Vol. 5, pp. 787–824). Springer.
- Kracht, M. (2003). *The Mathematics of Language*. Studies in Generative Grammar (Vol. 63). Walter de Gruyter.
- Krynicky, M., & Mostowski, M. (1999). Ambiguous quantifiers. In E. Orłowska (Ed.), *Logic at Work* (pp. 548–565). Heidelberg: Springer.
- Kugel, P. (1986). Thinking may be more than computing. *Cognition*, 22(2), 137–198.
- Manaster-Ramer, A. (1987). Dutch as a formal language. *Linguistics and Philosophy*, 10(2), 221–246.
- McNaughton, R., & Papert, S. A. (1971). *Counter-Free Automata*. M.I.T. Research Monograph no. 65. The MIT Press.
- Moss, L., & Tiede, H. J. (2006). Applications of modal logic in linguistics. In P. Blackburn, J. van Benthem, & F. Wolter (Eds.), *Handbook of Modal Logic*. Studies in Logic and Practical Reasoning (pp. 1031–1077). Elsevier.
- Mostowski, M., & Szymanik, J. (2012). Semantic bounds for everyday language. *Semiotica*, 188 (1–4), 363–372.
- Pratt-Hartmann, I. (2004). Fragments of language. *Journal of Logic, Language and Information*, 13(2), 207–223.
- Pratt-Hartmann, I. (2008). Computational complexity in natural language. In A. Clark, C. Fox, & S. Lappin (Eds.), *Computational Linguistics and Natural Language Processing Handbook*. Blackwell.
- Pratt-Hartmann, I., & Moss, L. S. (2009). Logics for the relational syllogistics. *The Review of Symbolic Logic*, 2(04), 647–683.
- Pullum, G. K., & Gazdar, G. (1982). Natural languages and context-free languages. *Linguistics and Philosophy*, 4(4), 471–504.
- Ristad, E. S. (1993). *The Language Complexity Game*. Artificial Intelligence. The MIT Press.
- Rogers, J. (1983). *A Descriptive Approach to Language-Theoretic Complexity*. Studies in Logic, Language, and Information. Stanford: CSLI Publications.
- van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science: A Multidisciplinary Journal*, 32(6), 939–984.
- Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3), 333–343.
- Thorne, C. (2012). Studying the distribution of fragments of English using deep semantic annotation. In H. Bunt (Ed.), *Proceedings of the ISA8 Workshop, SIGSEM*.
- Westerståhl, D. (1984). Some results on quantifiers. *Notre Dame Journal of Formal Logic*, 25, 152–169.

Quantifiers and Cognition: Logical and Computational
Perspectives

Szymanik, J.

2016, XIV, 211 p. 49 illus., Hardcover

ISBN: 978-3-319-28747-8