Quantifiers in TIME and SPACE

Computational Complexity of Generalized Quantifiers
in Natural Language

Jakub Szymanik

Quantifiers in TIME and SPACE

Computational Complexity of Generalized Quantifiers
in Natural Language

ILLC Dissertation Series DS-2009-01

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: `illc@science.uva.nl`
homepage: `http://www.illc.uva.nl/`

Quantifiers in TIME and SPACE

Computational Complexity of Generalized Quantifiers
in Natural Language

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Agnietenkapel
op vrijdag 6 maart 2009, te 14.00 uur

door

Jakub Krzysztof Szymanik

geboren te Warschau, Polen.

The investigations were supported by a Marie Curie Early Stage Research fellowship in the project GloRiClass (MEST-CT-2005-020841).

Cover design by Nina Gierasimczuk.
Printed and bound by Ipskamp Drukkers.

ISBN: 978–90–5776–187–4

Rodzicom

It is the striving for truth that drives us always to advance from the sense to the reference. (Frege, 1892, p. 63)

Contents

Acknowledgments

It was quite a long climb. It would have never happened without the support and inspiration from many people. If I had given a full account of everyone's help I would have never succeed in finishing on time. Therefore, some of you will have to forgive me for the following, only partial notes.

First of all, my parents have been a continuous source of motivation. They were always interested in my research and helped so much in building my confidence, optimism and energy. I am also very grateful to the rest of my family for providing me with many expressions of support.

I got for the first time interested in language, logic and cognition over 10 years ago thanks to Anna Wojdanowicz. She was the first to teach me philosophy and to encourage my independent thinking. She was also the very first critic of my very first essay on language and cognition which I wrote for the Philosophical Olympiad. This dissertation would not come to existence without her and I would be probably doing something completely different, presumably less fun.

I am grateful to all my friends with whom I had opportunity to discuss various philosophical issues in many wonderful places: climbing mountains, sailing lakes and seas, kayaking along the rivers, and bivouacking in the wildness. Let me mention Tomasz Adamski, Nina Gierasimczuk, Rafał Gryko, Konrad Grzegorzewicz, Andrzej Koroluk, Maciek Malicki, Paweł Mleczko, Maciek Mostowski, Mateusz Orliński, Jakub Pilarek, Tomasz Puczyłowski, Paweł Szewczyk, and Marcin Zajenkowski.

It was a lucky choice for me to start studying in the inter-departmental programme at the University of Warsaw (Collegium MISH), where I had a lot of freedom to develop my interests across different scientific disciplines. I am in debt to my tutors in the Collegium: Jerzy Pelc and Barbara Stanosz. Their supervision encouraged me to use the freedom for rigorous studies in the philosophy of language and related fields (logic, linguistics, mathematics, and psychology). They had a great impact on my scientific taste. I also have grown a lot through endless interdisciplinary discussions with my fellow students. For that, I especially thank: Tadeusz Ciecierski, Nina, Justyna Grudzińska, Witold Kieraś, Maria

Spychalska, and Marcin. The MISH Philosophical Circle was the first playground for organizing things in academia: edit books, prepare conferences, co-ordinate projects, apply for grants, and co-operate with other people. Now, after many years, I know that those were all very precious experiences. I am very grateful to Zbigniew Kloch who was supporting all those student initiatives.

In the second year of my studies I met the person who had the biggest impact on this thesis and my research in general. Marcin Mostowski not only taught me logic and suggested research topic, which first led me to MA thesis and then directly culminated in this dissertation, but it was also thanks to him that I started to be surrounded by logicians. I was lucky to learn from, study, and discuss logic with: Zofia Adamowicz, Mikołaj Bojańczyk, Cezary Cieśliński, Marek Czarnecki, Nina, Joanna Golińska-Pilarek (she was the first to show me Hintikka's sentence), Witek, Leszek Aleksander Kołodziejczyk, Eryk Kopczyński, Henryk Kotlarski, Michał Krynicki, Filip Murlak, Damian Niwiński, Paula Quinon, Jerzy Tyszkiewicz, Dominika Wojtyniak, Łukasz Wojtyniak, and Konrad Zdanowski. Especially, Leszek and Konrad explained to me uncountably many things about logic and inspired me with their attitude to work. I think that it is very likely that I would have never become a scientist (although I could have easily become a parody of a scientist) if I had not met Marcin Mostowski and his group.

Then, at the beginning of 2006, I took a great decision and came to Amsterdam to join ILLC as a PhD student in the GLoRiClass project. I was brave enough to jump in, partially because I had met some ILLC members before at Szklarska Poręba Workshop on the Roots of Pragmasemantics. In connection to this I want to thank Reinchard Blutner and Henk Zeevat for organizing the conference and giving us a chance to meet on the top of Szrenica every year. I had very soft landing in Amsterdam thanks to many people. First of all, the ILLC staff deserve many thanks for their support and incredible ability of solving all problems without letting me feel importunate: Marjan Veldhuisen, Ingrid van Loon, Tanja Kassenaar, Jessica Pogorzelski, Peter van Ormondt and Karin Gigengack. GloRiClass coordinators: Krzysztof Apt, Paul Dekker, Benedikt Löwe, Ulle Endriss and my friends from the first GloRiClass generation: Andreas Witzel and Daisuke Ikegami created extremely friendly atmosphere from the beginning, even though we were struggling with some problems in the project at that time. Moreover, I was introduced to ILLC and Amsterdam by many colleagues, among others: Tikitu de Jager (he took me on my first tour around Amsterdam and borrowed a map which I have lost, sorry for that!), and Merlijn Sevenster (we had discussed branching quantifiers long before I arrived to Amsterdam), Reut Tsarfaty, and Jelle Zuidema who were "forcing" me to have lunches (although it was usually just after I had breakfast) with other ILLC members integrating me with the Institute. Soon Amsterdam became my home when Nina, against some difficulties, decided to move to the Netherlands.

Obviously, the thesis profited a lot from my Amsterdam supervisors: Johan van Benthem and Theo Janssen. I enjoyed working with them and benefited a lot from the ways they complemented each other.

Johan has been a great source of vision and inspiration. He suggested to me plenty of problems and turned around my perspective on logic, language and computation. I am sure that his ideas will inspire my research long after graduation and maybe one day I will be able to answer some of the questions he suggested during our conversations in his office. I have been very much motivated by Johan's never-ceasing energy for taking new research directions and creating fruitful academic life.

I am very gratefull to Theo for agreeing to be my daily-supervisor and taking the most of the responsibility for my PhD research, even though my interests and plans were not exactly in line with his. My ideas were very much shaped by our regular Monday conversations and in the process of endless rewriting to meet his standards and account for criticism. I learned a lot about how to write scientific papers and prepare presentations from him. I am also very grateful for Theo's open-mindness in supporting my vision on complexity and cognition.

ILLC is a great place for research and Amsterdam is a beautiful place to live in. I was indeed very lucky to do my PhD here. There were many people who contributed to the great atmosphere, on both scientific ans social levels, and all deserve my most gratefull thanks.

Many members of the ILLC faculty have helped me in shaping my ideas: Paul Dekker, Peter van Emde Boas, Dick de Jongh, Michiel van Lambalgen, Robert van Rooij, Remko Scha, Leen Torenvliet, Jouko Väänänen, Yde Venema, and Jelle Zuidema. Through the Institute I had also a great opportunity to meet and collaborate with researchers all over the world: Denis Bonnay, Robin Clark, Jan van Eijck, Bart Geurts, Juha Kontinen, Allen Mann, Rick Nouwen, Eric Pacuit, Ramaswamy Ramanujam, Iris van Rooij, Ingmar Visser, Dag Westerståhl, Yoad Winter. I especially thank Robin Clark, Paul Dekker, Jouko Väänänen, and Dag Westerståhl for comments on the manuscript and being on my thesis committee.

Finally, I learned a lot through discussions and collaboration with my fellow students and friends. What is important I learned not only logic but also a bunch of more practical stuff, like, how to play Texas Hold'em and win, how to fly on a redeemer or how to say "coffee filter" in Finish. I want to thank: Marian Counihan, Jakub Dotlacil, Pablo Cubides Kovacsics (I admit that I have stolen a few climbing moves from him), Cédric Dégremont, Michael Franke, Amélie Gheerbrant, Nina, Patrick Girard, Olga Grigoriadou, Daisuke, Tikitu (thanks to him "the" and "a" are mostly in their right places except perhaps for this chapter), Sophia Katrenko, Lauri Keskinen, Jarmo Kontinen (for the joint struggle to understand computational complexity theory and a lot of other kinds of fun), Lena Kurzen, Olivia Ladinig, Raul Leal Rodriguez, Michał Łukasiewicz, Stefan Minica, Petter Remen, Floris Roelofsen, Federico Sangati, Marieke Schouwstra, Jonathan Shaheen, Leigh Smith, Marc Staudacher, Reut, Joel and Sara Uckelman, Levan Uridia, Fernando Velazquez-Quesada, Aurel Wannig, Andi, Yun Qi Xue, and Jonathan Zvesper.

I am particularly in debt to Andi. First of all, since the very beginning he has been a great office-mate. Without him I would never written the thesis, simply because of all the computer problems I had not been able to solve by myself. But more importantly, I am grateful for many interesting discussions about life and logic and all the fun we had together through the last 3 years.

I would like to express appreciations for my co-authors: Nina Gierasimczuk, Witold Kieraś, Juha Kontinen, Marcin Mostowski, and Marcin Zajenkowski whose ideas directly contributed to the thesis.

Special thanks go to Nina for her incredible support. She has not only contributed directly to this research in almost all its stages (from being the co-author of one of the chapters to designing the cover of this book), but first of of all: she is filling every day of my life with joy and love giving reasons for work. My appreciation for her support is beyond any words that I could possibly find.

Introduction

Everyone who has ever tried to understand, explain, and describe natural language knows how complicated it is. This dissertation is concerned with some aspects of complexity in natural language. In particular, we try to shed some light on the interplay between complexity and expressibility.

Complexity is a very broad, although quite intuitive, notion. There are at least two levels of language complexity — syntactic level and semantic level. The latter is what we will consider in the thesis. Moreover, we will focus on **complexity of meaning (semantics) of natural language quantifiers**. Especially, we are interested in complexity of finding the truth-values of natural language quantified sentences in finite situations. The general question we aim to answer is why the meanings of some sentences are more difficult than the meanings of others. For instance, why we will probably all agree that it is easier to evaluate sentence (1) than sentence (2) and why sentence (3) seems hard while sentence (4) sounds odd.

(1) Every book on the shelf is yellow.

(2) Most of the books on the shelf are yellow.

(3) Less than half of the members of parliament refer to each other.

(4) Some book by every author is referred to in some essay by every critic.

To discuss such differences in a precise manner, in the dissertation we use tools of computability theory to measure complexity. By doing it we commit ourselves to the idea that part of a "meaning" can be identified with an algorithm that checks the truth-value of an expression. Having this in mind we formalize linguistic descriptions in logic. Then we use **descriptive computational complexity** to study properties of formalisms we have defined. Among other things we try to draw a systematic line between easy (tractable) and difficult (intractable) quantifiers with respect to model-checking complexity. Moreover, as we are interested in the empirical value of complexity claims we provide some **linguistic case studies** as well as **psycholinguistic empirical experiments**. We apply complexity analysis in the domain of reciprocal expressions to account for some pragmasemantic phenomena, study various combinations of quantifiers in natural language, and empirically investigate the comprehension of simple quantifier sentences. Our results show that computational complexity can be very useful in investigating natural language semantics.

As far as our research topics and methods are concerned this work is **highly interdisciplinary**. The study can be placed within logic, natural language semantics, philosophy of mind and language, theoretical computer science, and cognitive science. It is an example of applying computational complexity theory to linguistics and cognitive science. Therefore, the intended audience consists mostly of linguists, philosophers and cognitive scientists interested in formal approaches to complexity in natural language. However, we hope that also logicians and computer scientists can find this study a source of inspiration, not only for possible applications, but also for developments in their research fields. The dissertation is intended to be self-contained, but some general background in logic and natural language semantics is assumed. Below we briefly overview every chapter.

Chapter 1, "Algorithmic Semantics", is a proper introduction to the thesis, where we place our study on the map of research themes. We discuss our methodological assumptions there; in particular, we outline the idea of treating **referential meaning as an algorithm**. We also argue that computational complexity can be taken as a reasonable measure for the difficulty of natural language expressions. Moreover, we propose some hypotheses summing up our perspective on the role of computational complexity in linguistics and cognitive science.

Chapter 2, "Mathematical Prerequisites", shows the background of the technical work of the thesis. We introduce here our main logical tools: the notions of generalized quantifier theory, computability theory, and descriptive complexity theory. The chapter may be skipped by readers with background knowledge on generalized quantifiers and computability. For the rest of the readers it can serve as a basic reference to the mathematics used elsewhere in the dissertation.

Chapter 3, "Complexity of Polyadic Quantifiers", is devoted to a logical study of the computational complexity of **polyadic generalized quantifiers**. We focus here on constructions that are interesting from the semantic point of view. We prove that quantifier iteration, cumulation and resumption do not carry us outside polynomial time computability. Other polyadic quantifiers often used in linguistics, like branching and Ramsey quantifiers, lead to NP-complete natural language sentences. This chapter prepares the ground for the more linguistic discussion in the next parts of the thesis, particularly in Chapters 4 and 6.

Chapter 4, "Complexity of Quantified Reciprocals", is concerned with the linguistic case study. We investigate the computational complexity of different interpretations of **reciprocal expressions**, like "each other", in English. Especially, we show that Ramsey quantifiers express the semantics of reciprocal sentences with a quantifier in the antecedent. As a result we find a computational dichotomy between different interpretations of reciprocals: some of them stay in PTIME when others are NP-complete. This dichotomy is consistent with well-known semantic distinctions among different interpretations of “each other”. We discuss the impact of this dichotomy on the so-called Strong Meaning Hypothesis proposed as a pragmatic explanation for shifts occurring among different reciprocal readings.

Chapter 5 “Complexity of Collective Quantification” discusses yet another form of quantification in natural language. We investigate logic of **collective quantifiers** and propose to analyze them in terms of second-order generalized quantifiers. In particular, our research shows that the widely accepted type-shifting approach to modelling collectivity in natural language is probably not expressive enough to cover all instances. Additionally, it is also extremely complex and, as such, implausible. As a result we suggest to study algebraic (many-sorted) formalisms. Another interpretation of our results is that computational complexity restricts expressibility of everyday language and as a result collective readings of some quantifiers, e.g., proportional determiners, is not realized in that fragment of natural language.

Chapter 6, “Hintikka’s Thesis Revisited”, is the first empirical fragment of our work. We study Hintikka’s well-known claim concerning the necessity of a **branching interpretation** for some natural language sentences, e.g., “Some relatives of each villager and some relatives of each townsman hate each other” or “Most boys and most girls from my class dated each other”. We argue against Hintikka’s Thesis and propose our own reading of these sentences, which we call the conjunctional reading. Next, we show how to investigate such a problem empirically using insights from linguistics, logic and computability. The results of our tests confirm that the conjunctional reading is a widely accepted interpretation for these sentences.

Chapter 7, “Processing Simple Quantifiers”, is about a computational semantics for simple (monadic) quantifiers in natural language. In this case we can model the meaning of quantifiers using finite and push-down automata. Our aim is to show the empirical relevance of computational descriptions. We start by presenting the neurological research studying an automata-theoretic model of **simple quantifier comprehension**. Next we criticize the methodology of this research and propose a better set of hypotheses. We use these to conduct our own empirical study comparing the reaction times needed for recognizing the truth-value of sentences with simple quantifiers. The results strongly confirm the complexity predictions of the computational model for simple quantifiers in natural language. Hence, this chapter directly links computational predictions to linguistic processing.

Chapter 8, “Conclusions and Perspectives”, closes the dissertation with a short summary of results achieved from the perspective of the grand issues recognized in the first chapter. This is followed by a discussion of related open questions and directions for further research. This final discussion focuses on general loose ends, as particular research questions directly implied by our technical work are discussed at the end of the corresponding chapters.

Chapter 1

Algorithmic Semantics

1.1 Meaning and Comprehension

Meaning and Information

Natural language is arguably the most important mean we have for exchanging information. It influences our everyday interactions with the external world and those with other people. It allows information to be conveyed between speakers and hearers about the world, their beliefs, intentions and so on. To make the communication possible many cognitive processes have to be engaged. First of all, a natural language expression carrying information has to be produced by a speaker. This generation process requires the ability of producing grammatical sequences meaning whatever a speaker intends to share with a hearer. Secondly, a hearer has to decode the information contained in an expression, i.e., a hearer has to understand the meaning of an expression (see e.g. Kamp and Stokhof, 2008, for elaboration on information and language).

Different Aspects of Meaning

The comprehension process can work in many different ways depending on the many ways in which natural language can be used. For example, consider the following sentences.

(1) Mt. Everest is the highest mountain in the world.

(2) K2 might be the most difficult mountain to climb.

If we consider the declarative use of language — a speaker stating facts, like in sentence (1), or raising possibilities as in (2) — then understanding may be described as the ability to imagine the world satisfying the sentence and adjust one's beliefs accordingly.

However, there are many more possibilities. Let us consider the following sentences.

(3) Is K2 higher than Mt. Everest?

(4) Climb K2 in winter!

To understand question (3) one has to realize what are the possible answers. On the other hand, to comprehend directive (4) we have to recognize the speaker's intentions, i.e., realize what he wants us to do.

Moreover, there are many indirect uses of language such that their comprehension depends very much on cooperation among speakers. For example consider the following:

(5) K2 has either been climbed in winter or not.

(6) I met Alice and Bob in the Everest base camp. She climbed it in one day and he failed.

(7) I am here right now.

When I am stating sentence (5) I do not only claim directly this obvious alternative. Rather I want to say that I actually do not know whether anyone has in fact done it. In some other situations meaning is crucially connected to a context. For instance, in understanding discourse and dealing with anaphora one has to recognize the references of pronouns to the entities mentioned previously in a conversation. Simply consider example (6). In other cases, a hearer has to know the context to recognize the denotation of indexical expressions, like in sentence (7).

Obviously it is impossible to exhaust all the functions of language in a short list like the one above, and this was not our aim. We only wanted to illustrate some varieties of linguistic meaning; next we will try to identify some common features of understanding.

Basic Elements of Comprehension

What are the building-blocks of natural language comprehension? We believe that there are at least two basic processes:

- Confronting meaning with the actual world.

- Confronting meaning with knowledge and beliefs.

The first process relates new information conveyed by the meaning of a natural language expression to the actual state of affairs. In the case of declarative sentences it allows us to decide whether the new information is true, or what the world should look like to make it true. In the case of questions we have to check the possible answers against the real world in order to be able to give a true answer. We even need to relate to the world to realize that some sentences bring indirect information. For example, to draw the intended implicatures from (5) I first have to realize that the truth-value of the statement is independent from the facts.

The second process, confronting meaning with knowledge, crucially determines our belief changes triggered by the new information carried by an utterance. When we hear a declarative sentence we check whether it is consistent with our knowledge. Sometimes we even make one step more, namely we check whether the implications of a sentence agree with our knowledge. Moreover, we search through our knowledge to answer a question. Also processing discourse can be seen in this perspective. For example, when we deal with a sequence of two sentences, like (6), we may add the first sentence to our knowledge and then simply look for referents for the pronouns among the most recently added beliefs.

1.2 Geography of Perspectives

1.2.1 Traditional Semantics and Pragmatics

Formal semantics describes functions which assign some set-theoretic object in the universe of discourse (possible worlds, models, possible interpretations of language) to every well-formed expression. We say that the object is an extension (denotation) of this expression. Such an approach — motivated by the work of Alfred Tarski (1944) and successfully applied to natural language by his student Richard Montague (1970) — is often called set-theoretic (or model-theoretic) semantics. In other words, formal semantics establishes a potential link between a well-formed natural language expression and an abstraction of the real world in the form of a model.

Some non-linguistic aspects of the reference relation, like the influence of the context or of knowledge, have been generating a lot of interest. These problematic cases were traditionally falling under the scope of pragmatics, however the distinction between semantics and pragmatics has never been clear. They were early investigated in the philosophy of language. Let us mention only the analysis given by John Langshaw Austin (1975) for performative utterances, like the following:

(8) I name this mountain K2.

Additionally, recall Paul Grice's (1991) cooperative principle and conversational maxims explaining indirect communication, e.g., sentence (5).

In the model-theoretic semantics there were, from the beginning, serious attempts to incorporate context-dependence on the agenda, e.g., by studying indexicals expressions (see e.g. Bar-Hillel, 1954; Montague, 1970; Kamp, 1971). One of the first systematic accounts of context dependence within this tradition were provided by David Kaplan and Robert Stalnaker. Kaplan (1979) pointed out that circumstances intervene into the process of determining an utterance's truth-value twice rather than only once as possible-worlds semantics appeared to suggest. He has argued that the utterance must be first confronted with its context to yield an intension, which only then can be confronted with a possible world to yield the truth-value of the utterance with respect to the actual world. Thus, to understand an utterance such as (7) we must first exploit the context to unfold the indexicals "I", "here", and "now", to reach a proposition. Then this proposition can be evaluated in the possible worlds semantics.

Stalnaker has been one of the most influential theorists exploring the philosophical aspects of possible worlds semantics. According to his view of possible worlds, they stand for ways this world could have been, and are the maximal properties that this world could have had. Moreover, he has used the apparatus of possible worlds semantics to explore counterfactuals, conditionals, and presupposition. For example, central to his account of intentionality is the view of propositions as sets of possible worlds (see e.g. Stalnaker, 2003). Finally, his view of assertion as narrowing the conversational common ground to exclude situations in which the asserted content is false was a major impetus for the recent development of belief revision — a very rich theory which accounts for belief changes triggered by new information.

1.2.2 The Dynamic Turn in Semantics

The traditional model-theoretic approach has problems with combining the two basics aspects of meaning: relation to the external world and correspondence to knowledge. Simply put, model-theoretic semantics focuses only on the static relation between expressions and their denotations. This problem was recognized very early. For example, recall Carnap's notion of a "meaning postulate", trying to extend the inferential power of traditional semantic analysis by adding bits of lexical awareness (see Carnap, 2007). However, in the extensional tradition the problem has always been treated very partially and statically. Not much was done about changes in knowledge caused by new linguistic information before the so-called Dynamic Turn in the seventies. Around that time the real change in perspective can be observed. Since then a lot of work has been put into grasping dynamic aspects of knowledge change triggered by linguistic information (see e.g. Peregrin, 2003, for an overview). Below we briefly recall three important research axes.

Discourse Representation

Pioneering work in the dynamic framework was Discourse Representation Theory, introduced by Hans Kamp. This is a theoretical framework for dealing with issues in the semantics and pragmatics of anaphora, tense and presuppositions. Its distinctive features are that it is a mentalist and representationalist theory of interpretation, not only of individual sentences but of discourse as well. In these respects it made a clear break with classical formal semantics, but in other respects it continues the tradition, e.g., in its use of model-theoretic tools (see Kamp and Reyle, 1993).

Belief Revision

In the logic of belief revision, a belief state is represented by a set of sentences. The major operations of change are those consisting in the introduction or removal of a belief-representing sentence. In both cases, changes affecting other sentences may be needed, for instance in order to retain consistency. Rationality postulates for such operations have been proposed, and representation theorems have been obtained that characterize specific types of operations in terms of these postulates. There are many technical methods of representing these processes. The logical perspective is mostly formed around Dynamic Epistemic Logic and the algebraic perspective exploits mostly the AGM framework (see e.g. Gärdenfors, 2003; van Ditmarsch et al., 2007). Belief revision perspective can be treated as an in-depth study of the second fundamental of comprehension: “knowledge-checking”. However, its relationship with natural language semantics is still waiting for a development.

Games and Pragmatics

The work of Stalnaker and Kaplan put pragmatic considerations into the agenda of dynamic semantics. In parallel, David Lewis also worked on these ideas, for example formulating the analysis of counterfactual conditionals in terms of the theory of possible worlds. However, arguably his most important input into linguistics has given rise to game-theoretic considerations in that area. Lewis (2002) claimed that social conventions are solutions to game-theoretic “coordination problems”, but, what is more important from our perspective, he has pointed out that the use of a language in a population relies on conventions of truthfulness and trust among the members of the population. He has recast in this framework notions such as truth and analyticity, claiming that they are better understood as relations between sentences and a language, rather than as properties of sentences. This has led directly to the development of highly formal models of communication explaining Grice’s principles in terms of signaling games or evolutionary equilibria (see e.g. Benz et al., 2005). Recently language is seen as an interaction between speaker and hearer, as kind of a game in which by mutual interpretation players come to an interpretation. This approach underlines Optimality Theory (see e.g. Kager, 1999) and it spans across semantics and pragmatics.

1.2.3 Meaning as Algorithm

Traditionally, in logic finite models were considered as a pathological case while infinite universes were in the center of attention. In parallel with the Dynamic Turn, researchers have started to be interested in finite interpretations. With this change in the perspective some new philosophical dimensions have arisen. Particularly, in natural language semantics the idea of treating meanings in terms of truth-conditions has naturally started to evolve towards algorithmic explanations. Below we briefly present this evolution and in Chapter 1.6 we take a closer look at finite interpretations in natural language semantics. This approach is particularly important for our thesis.

Sinn und Bedeutung

This is a tradition, going back to Gottlob Frege (1892) (see also Church, 1973, 1974; Dummett, 1978), of thinking about the meaning of a sentence as *the mode of presenting* its truth-value. In modern terms we can try to explicate the Fregean *Art des Gegebenseins* of a referent (the way the referent is given) by saying that the meaning of an expression is a procedure for finding its extension in a model. Accordingly, a sentence meaning is a procedure for finding the truth-value.[1] Quoting Frege:

> It is the striving for truth that drives us always to advance from the sense to the reference. (Frege, 1892, p. 63)

Similar ideas can be found around the same time also in the writings of other philosophers.

Let us quote Ludwig Wittgenstein:

> To understand a proposition means to know what is the case if it is true.[2] (Wittgenstein, 1922, 4.024)

Also Kazimierz Ajdukiewicz has claimed:

[1]This way of interpreting Frege's "Sinn" is not a standard view in the philosophy of language. Although, it could help to solve some notorious puzzles of the Fregean theory of meaning, e.g., those related to the meaning of indirect speech and discussed in the recent paper of Saul Kripke (2008).

[2]An even more procedural account is proposed in "Philosophical Investigations" where Wittgenstein asks the reader to think of language and its uses as a multiplicity of language-games. The parts of language have meaning only within these games. (Wittgenstein, 1953, §23).

> For two different persons an expression brings the same meaning, whenever it gives them the same method of deciding whether this expression can be applied to a given object or not.[3]
>
> (Ajdukiewicz, 1931)

In model theory such procedures are often called model-checking algorithms. This approach has been adopted by many theoreticians, to different degrees of explicitness, and we can trace it back to Fregean ideas. For obvious reasons, Frege himself could not speak about procedures or algorithms directly.

The First Explicit Formulation

Historically — as far as we are aware — the Fregean idea was for the first time explicitly formulated in procedural terms by the Czech logician, philosopher and mathematician Pavel Tichý (1969). In the paper, which can be best summarized by its title "Intension in terms of Turing machines", he identified the meaning of an expression with a Turing machine computing its denotation. The main technical objective of the paper is to account for the distinction between analytic and logical truths by treating concepts as procedures.[4] However, the author also recognized the broader application of the algorithmic idea. He noticed that:

> [. . .] the fundamental relationship between sentence and procedure is obviously of a semantic nature; namely, the purpose of sentences is to record the outcome of various procedures. Thus e.g. the sentence "The liquid X is an acid" serves to record that the outcome of a definite chemical testing procedure applied to X is positive. The present paper is an attempt to make this simple and straightforward idea the basis for an exact semantic theory. (Tichý, 1969, p. 7)

Moreover, he directly argues for identifying meaning with an algorithm, a dynamic procedure of searching for the denotation instead of a static model-theoretic entity:

> For what does it mean to understand, i.e. to know the sense of an expression? It does not mean actually to know its denotation but to know how the denotation can be found, how to pinpoint the denotation of the expression among all the objects of the same type. E.g. to know the sense of "taller" does not mean actually to know who is

[3]"Dwaj ludzie rozumieją pewne wyrażenie w tym samym znaczeniu, gdy rozumienie to uzbraja ich obu w tę samą metodę rozstrzygania, czy wyrażenie to zastosować do jakiegoś przedmiotu, czy też nie."

[4]The idea was for the first time presented at the Third International Congress for Logic, Methodology and Philosophy of Science in Amsterdam (1967) under the title "Analyticity in terms of Turing Machines".

> taller than who, but rather to know what to do whenever you want to decide whether a given individual is taller than another one. In other words, it does not mean to know which of the binary relations on the universe is the one conceived by the sense of "taller", but to know a method or procedure by means of which the relation can be identified. Thus it seems natural to conceive of concepts as procedures. (Tichý, 1969, p. 9)

Later Tichý developed a system of intensional logic with an extensive philosophical justification of the Fregean idea (see Tichý, 1988).

Other Appearances of the Same Idea

Let us trace the idea of a meaning as an algorithm a little bit more as it has been gradually formulated in more recent terminology. All approaches that we briefly outline below try to account for the very same idea. However, none of them refers to the work of Tichý.

Patrick Suppes (1982) has investigated an algebraic semantics for natural language and finally came to the conclusion that the meaning of a sentence is a procedure or a collection of procedures. His motivation seems to be mainly psychological; let us quote the author:

> The basic and fundamental psychological point is that, with rare exceptions, in applying a predicate to an object or judging that a relation holds between two or more objects, we do not consider properties or relations as sets. We do not even consider them as somehow simply intensional properties, but we have procedures that compute their values for the object in question. Thus, if someone tells me that an object in the distance is a cow, I have a perceptual and conceptual procedure for making computations on the input data that reach my peripheral sensory system [...] Fregean and other accounts scarcely touch this psychological aspect of actually determining application of a specific algorithmic procedure. (Suppes, 1982, p. 29)

He also has made a point that meaning can be treated not only in terms of single procedures but as collections of those:

> I have defended the thesis that the meaning of a sentence is a procedure or a collection of procedures and that this meaning in its most concrete representation is wholly private and idiosyncratic to each individual. (Suppes, 1982, p. 33)

Another approach — similar to Tichý's way of thinking in its direct use of automata — was formulated by Johan van Benthem (1986) to describe the meanings of quantifiers in natural language. Van Benthem's semantic automata recognize the truth-value of a generalized quantifier expression on a structure. We will study this approach in more detail in Chapter 7, where we even show that it is psychologically plausible. In particular, it correctly predicts some aspects of the processing of natural language quantifiers, both on a cognitive and a neurological level. Here let us only note that the intuition behind introducing the automata-theoretic perspective were not only technical:

> An attractive, but never very central idea in modern semantics has been to regard linguistic expressions as denoting certain "procedures" performed within models for the language.
>
> (van Benthem, 1986, p. 151)

The algorithmic theory of meaning found its strongest linguistic formulation in the works of Yiannis Moschovakis (1990). He has analyzed the Fregean notions of sense and denotation as algorithm and value, and then developed a rigorous logical calculus of meaning and synonymy (Moschovakis, 2006). He also succeeded in popularizing this idea. Reinhard Muskens (2005) has provided a similar theory built on a considerably lighter formalization. All these works are mainly of a linguistic character and try to present the Fregean distinction between meaning and denotation in a strict mathematical framework, throwing some light on the classical problems studied in the philosophy of language.

This line of research is also followed by Michiel van Lambalgen and Fritz Hamm (2004). They have proposed to study the meaning-as-algorithm idea in the paradigm of logic programming. The idea was taken further in the book "The Proper Treatment of Events" (Lambalgen and Hamm, 2005). There the combination of the event calculus (as developed in Artificial Intelligence) with type free theory and logic programming techniques is used to formulate an axiomatized semantic theory for a broad range of linguistic applications. The authors argue that the proposed architecture, which sees the heart of the semantics of tense and aspect in the notion of planning, has cognitive plausibility. The argument proceeds via an examination of the role of time in cognitive processes.

Theories Similar in Spirit

Other advances in natural language semantics might also be viewed as incorporating some procedural ideas. First of all, Montague (1970) committed himself to the idea of intensional semantics, where the meaning of an expression can be identified with a function choosing its denotation in every possible world. This function can be interpreted as corresponding to some model-checking procedure. Even if Montague's approach cannot be directly labeled "procedural", to some extent he has motivated most of the works exploring algorithmic ideas, notably that of Moschovakis and his followers. Moreover, the procedural spirit can be found in the works creating the dynamic turn in linguistics, as all of them were strongly influenced by Montague. E.g., they adopted possible world semantics, compositionality and other tools developed by Montague.

Two other examples of procedural theories are dynamic semantics and game-theoretic semantics. Dynamic semantics (see Groenendijk and Stokhof, 1991; van den Berg, 1996) formalizes meaning in terms of transformations between states. The idea is very simple; quoting Paul Dekker's (2008) guide to dynamic semantics:

> People use language, they have cognitive states, and what language does is change these states. 'Natural languages are programming languages for mind', it has been said. (Dekker, 2008, p. 1)

Game-theoretic semantics (see Lorentzen, 1955; Hintikka and Sandu, 1997) sees the meaning of a sentence as a winning strategy in a game leading to its verification. The game-theoretic semantics have initiated many advances in logic and linguistics. The game-theoretic metaphor was extended even further by Merlijn Sevenster (2006) who has proposed a strategic framework for evaluating some fragments of language. However, it is not completely clear how these relate to more traditional model-checking approaches.

Considering language in a strategic paradigm as a goal-directed process rather than a recursive, rule-governed system (see Hintikka, 1997) may help to understand aspects of meaning different than model-checking, like inferential properties, confronting new information with knowledge and so on. For example, so-called model construction games which build models for a set of formulas (see van Benthem, 2003; Hodges, 1997) can help to understand how language users integrate new information into a consistent information state. And, at least from a logical point of view, model-building games are related to Lorenzen dialogical games for proofs.

Synonymy

One of the most famous philosophical problems with meaning is concerned with the synonymy of linguistic expressions. It is widely claimed that we do not understand the meaning of "meaning" as long as we cannot define synonymy (see Quine, 1964). According to the algorithmic proposal the problem is equivalent to the question about the identity relation between algorithms. In other words, having a set A of of algorithms we search for an equivalence relation $\approx$ on A such that for every $f, g \in A$:

$$f \approx g \iff f \text{ and } g \text{ realize the same algorithm.}$$

We will see some consequences of this reformulation of the synonymy problem when discussing computability issues later in this chapter.

1.3 Our Procedural Perspective on Meaning

1.3.1 Questions

In this thesis we are mainly concerned with the first-mentioned logical element of natural language comprehension. Namely, we study how meaning relates to the external world. However, as the model-theoretic perspective cannot be torn apart from knowledge representation we also sometimes (possibly between the lines) refer to the problem of the relation between meaning and knowledge, e.g., we will speak later about referential and inferential meaning. To understand the processes lying behind these aspects of comprehension we need to consider, among others, the following questions:

- What is the meaning of a given expression?
- How do people recognize the denotations of linguistic expressions?
- Why do some sentences seem to be more difficult to understand than others?

There are many connections between these questions and this is why in trying to answer one of them we have to take a position about the rest.

1.3.2 Methodology

Above we have discussed a certain computational approach to these problems which we adopt throughout the whole thesis. Therefore, we align with the dynamic way of thinking by treating comprehension not from the static extensional perspective but as a dynamic procedure.

The algorithmic model-checking approach to meaning seems reasonable for a big fragment of natural language in many different contexts. In the dissertation we follow this philosophical line and identify the meaning of an expression with an algorithm that recognizes its extension in a given finite model. We sometimes refer to the meaning understood in such a way as *referential meaning* to stress that this is just one of the possible aspects of meaning — as discussed earlier in this chapter. Even though we see this line of thinking as another manifestation of the dynamic turn we have to restrict ourselves to studying a basic element of comprehension, model-checking, and not pursue its more context-dependent aspects.

In other words, we say that the referential meaning of a sentence φ is given by a method of establishing the truth-value of φ in possible situations. Such procedures can be described by investigating how language users look for the truth-value of a sentence in various situations.[5]

[5]See Chapter 6 for an example of research which tries to describe referential meaning for some class of natural language sentences.

In the thesis we do not study the algorithmic theory of meaning in itself. However, our technical work is motivated mostly by these ideas. Particularly, we are in debt to the formulations of procedural semantics which appear in the research of Tichý (1969) and van Benthem (1986) as we will also work mainly with machines recognizing quantifiers. Moreover, we follow Suppes (1982) and claim that for one semantic construction[6] there are many essentially different algorithms. Their usefulness depends on the situation. To understand this idea better let us consider the following sentence:

(9) The majority of people at the party were women.

The referential meaning of sentence (9) can be expressed by a simple counting procedure. However, in the case when there are over five hundred people at the party the counting procedure is not very efficient. But it may happen that guests perform some traditional dance in pairs. Then we could apply a different algorithm, for instance simply check if some woman remains without a man. In this case the second method would be much more efficient than the first one. Actually, in Chapter 7 we present empirical research indicating that the same linguistic construction can be understood by people using different procedures which are triggered by various contexts.

1.3.1. Example. Let us give one more related example capturing some of the intuitions. Imagine two people: John, who is a mathematician, and Tom, who is a geologist. They both speak English, using such expressions as: "being higher than" and "prime number". Both of them understand these expressions. However, only John knows the simple ancient algorithm, the Sieve of Eratosthenes, for finding all prime numbers up to a specified integer. On the other hand, only Tom understands the principles of using engineering instruments for measuring levels. Therefore, there are expressions such that their truth-value in certain situations can be decided only by John or only by Tom. For example, they can both easily decide whether the following sentences are true:

(10) 17 is a prime number.

(11) John is taller than Tom.

However, if we pick a big number or two objects whose lengths cannot be compared directly or via simple measurements, then it is very likely that only one of them can decide the truth-value of the sentence. For instance, John but not Tom can decide the truth-value of the following sentence:

[6] We talk about "constructions" and not "expressions" to avoid misunderstandings as the same expression can be used for several different semantic constructions. For example, the expression "and" can serve as a propositional connective or as an operation between noun phrases.

(12) 123456789 is a prime number.

Analogously, only Tom knows how to decide the truth-value of the following statement:

(13) Gasherbrum I is 50 meters higher than Gasherbrum II.

The point of this example is to stress that when dealing with a natural language sentence we very often want to know whether it is true or false and we may need to use different meanings of it in various situations to find out. This linguistic ability is based on the fact that we are continuously using various techniques for recognizing the extensions of natural language constructions. These techniques can be identified with meanings. Moreover, learning natural language constructions consists essentially of collecting procedures for finding denotations (see Gierasimczuk, 2007). This way of thinking is in line with the algorithmic view of meaning.[7]

Summing up, what follows is the main methodological assumption of the thesis:

Main Assumption *The referential meaning of an expression χ is a collection of algorithms computing the extension of χ in a given finite model.*

However, having model-checking algorithms for all sentences is not the only possible way to understand language. As we discussed, the other basic component of language comprehension is the ability to relate it to knowledge. For example, we can also recognize some inferential relations between sentences establishing so-called *inferential meaning*. For instance, knowing that a sentence ψ is true and φ follows from ψ we know that φ is true. Already Jean Piaget (2001) has noticed that this is a mechanism used very often to evaluate sentences. We also study such situations in the thesis. One of the interesting problems — which we leave open — is how inferential meaning relates to belief revision.

1.3.3 Psychological Motivations

Our main aim in applying a dynamic, procedural framework is motivated by our interest in psycholinguistics. We share this motivation with Suppes (1982) and Lambalgen and Hamm (2005). We believe that a good linguistic theory should give direct input for empirical studies on language processing. Unfortunately, most of the dynamic frameworks do not make a single step in this direction.

[7]Notice that this way of thinking can also contribute to the philosophical idea of the division of linguistic labor (see Putnam, 1985). Simply, experts know more relevant meaning-algorithms and hence understand the meaning of an expression belonging to the domain of their expertise in a more sophisticated way.

We study the computational properties of natural language to shed some light on the cognitive processes behind language comprehension. However, as at this stage of research we are far from formulating a satisfactory, detailed model of comprehension, we rather focus on abstract computational properties which do not depend on any specific implementation. Therefore, we are not studying concrete procedures formalizing meaning but properties common to all possible procedures.

In the rest of this chapter we give an intuitive description of computations (mathematical details are presented in Section 2.3). Then we discuss the links between computability theory and cognitive science as we treat comprehension as a cognitive task. Our aim is to put computational semantics in a broader context of cognitive science.

1.4 Algorithms and Computations

1.4.1 What is an "Algorithm"?

We have decided to view some semantic issues from the computational perspective. In this section we will then try to shed some light on what these procedures or algorithms we are talking about really are. Actually, the issue is highly nontrivial. Even though the intuitions behind the notion of "algorithm" are probably more precise than those standing behind "meaning" they are still far from being completely clear. On the other hand, this is not a serious obstacle because we are interested rather in the inherent properties of computational problems than in any concrete algorithmic implementations.

Euclid's method for finding the greatest common divisor for any two positive integers (developed around 4 BC) is commonly believed to be the first non-trivial algorithm. As far as etymology is concerned the word "algorithm" is connected to the name of the Persian astronomer and mathematician Al-Khwarizmi. He wrote a treatise in Arabic in 825 AD "On Calculation with Hindu Numerals". It was translated into Latin in the 12th century as "Algoritmi de numero Indorum", which title was likely intended to mean "Algoritmi on the numbers of the Indians", where "Algoritmi" was the translator's rendition of the author's name. However, some people misunderstood the title, and treated Algoritmi as a Latin plural. This led to the word "algorithm" coming to mean "calculation method". Coincidentally, in his treatise he introduced the decimal positional number system to the Western world and many simple algorithms for dealing with addition, substraction, multiplication and division of decimal numbers. We all learn these algorithms in the beginning of our education.

There is no generally accepted formal definition of "algorithm" yet. An informal definition could be "an algorithm is a mechanical procedure that calculates something". It is an interesting problem in the philosophical foundations of computer science to come up with a reasonable and widely acceptable formal definition. Actually, there is some research going in this direction and the proposed methods for dealing with this problem are very similar to those used for solving foundational issues of mathematics in the beginnings of the 20th century. For example, people have suggested formalizing algorithms in set theory, rejecting their existence (there are no algorithms just programs) or axiomatizing their theories (see e.g. Moschovakis, 2001, for a discussion).

1.4.2 Turing Computability

Following tradition, we have decided to view computability in terms of Turing machines (see Section 2.3.2). This decision may be justified by the following widely believed philosophical claim:

Church-Turing Thesis *A problem has an algorithmic solution if and only if it can be computed by a Turing machine.*

The Church-Turing Thesis (Turing, 1936; Church, 1936) states that everything that ever might be mechanically calculated can be computed by a Turing machine.

Let us briefly try to justify the Church-Turing Thesis here.[8] Obviously, if a problem is computable by some Turing machine then we can easily calculate it by following the steps of this machine. But why should we believe that there are no computable problems beyond Turing machine computability? First of all, we do not know of any counterexample. Every function which we know how to compute can be computed by a Turing machine. Moreover, all the known methods of constructing computable functions from computable functions lead from Turing computable functions to other Turing computable functions. Additionally, the class of all Turing computable functions is very natural in the following sense. All attempts so far to explicate computability (many of them entirely independent) have turned out to define exactly the same class of problems. For instance, definitions via abstract machines (Random Access Machines, quantum computers, cellular automata and genetic algorithms), formal systems (the lambda calculus, Post rewriting systems) and particular classes of function (recursive functions) are all equivalent to the definition of Turing machine computability.

Moreover, the Church-Turing Thesis is interesting because it allows us to conclude that a problem is not decidable from the proof that it is not Turing computable. Take the class of all functions from natural numbers to natural numbers. There are uncountably many such functions, but there are only countably many Turing machines (this follows easily from the definition, see Section 2.3.2). Hence, some natural number functions must not be computable. The

[8]In a recent paper Mostowski (2008) gives a proof of the thesis based on the assumption that a finite but potentially infinite world is a good mathematical model of our reality.

most famous non-computable problems are the Halting Problem (decide whether Turing machine M will halt on input x), the decision problem for first-order logic (i.e., the question whether a given formula φ is a theorem), and the Tenth Hilbert Problem (the algorithmic solvability in integers of Diophantine equations).

Identity of meanings-as-algorithms

Let us relate the above arguments directly to the semantic considerations. We have noticed, discussing procedural approaches to meaning, that the synonymy problem can be stated in terms of an identity criterion for algorithms.

The minimal demand on the identity relation is to identify notational variants of the same algorithm. The widest reasonable definition will identify algorithms computing the same partial functions, i.e., terminating on the same inputs and giving the same output on identical input.

Let us observe that this criterion of identity is not computable as otherwise we could easily compute the Halting Problem. In this case, we can not find an algorithm deciding whether two expressions have the same meaning or not.[9] It is possible only in some simple cases, for instance, when meanings can be identified with finite automata as in the case of Van Benthem's approach studied in Chapter 7. This fact nicely matches the status of the long-standing and unsolved synonymy problem in the philosophy of language.[10]

Moreover, notice that two in the above sense identical algorithms do not have to be equally good in every sense. For instance, if we take a sentence containing n words it is possible that one algorithm will need n^2 steps to compute its referential meaning, when the other needs 2^{2^n} steps. Thus, the identity definition should probably be enriched by a condition saying that in order for two algorithms to be identical they not only have to compute the same partial function but also must be comparable with respect to their efficiency. But how can we say that two different algorithms are similarly effective?

Computational Complexity

Inherent Complexity With the development of programming practice it has been observed that there are computable problems for which we do not know any effective algorithms. Some problems need too much of our computational resources, like time or memory, to get an answer. Computational complexity theory, described briefly in Section 2.3, investigates the amount of resources required for the execution of algorithms and the inherent difficulty of computational problems. This means that the theory does not deal directly with concrete algorithmic

[9]Notice that this can cause some serious real life problems, e.g., for copyright legislation in the domain of software.

[10]However, see Moschovakis (1990) for a sophisticated algorithmic theory of meaning with a computable synonymy problem.

procedures, but instead studies the abstract computational properties of queries. These properties determine in a precise mathematical sense some properties of all possible algorithms which can be used to solve problems. As a result, the theory explains why for some computable questions we cannot come up with useful algorithms.

Tractability and Intractability An important aspect of computational complexity theory is to categorize computational problems via complexity classes. In particular, we want to identify efficiently solvable problems and draw a line between tractability and intractability. From our perspective the most important distinction is that between 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 are referred to as NP-hard problems (see Section 2.3.3 for mathematical details). Intuitively, a problem is NP-hard if there is no "clever" algorithm for solving it. The only way to deal with it is by using brute-force methods: searching throughout all possible combinations of elements over a universe. In other words, NP-hard problems lead to combinatorial explosion.

Notice that all complexity claims reaching out to 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. We will discuss these, mainly technical, issues in Section 2.3.3, where we give formal definitions. Now, let us only mention the most famous complexity problem. As we said above PTIME is the class of problems which can be computed by deterministic Turing machines in polynomial time. Moreover, speaking precisely, NP-hard problems are problems being at least as difficult as problems belonging to the NPTIME (NP) class. This is the class of problems which 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 NPTIME-complete problem is PTIME computable. Unfortunately, we do not know whether P=NP. It is the famous question 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, the experience and practice of computational complexity theory allows us to safely assume that these two classes are different. This is what almost all computer scientists believe and we also take it for granted.

Complexity Assumption $P \neq NP$.

SAT-problem Before we move to more general considerations let us give some examples. Many natural problems are computable in polynomial time, for instance calculating the greatest common divisor of two numbers or looking something up in a dictionary. However, we will focus here on a very important NP-complete problem, the satisfiability problem for propositional formulae.

The problem is to decide whether a given propositional formula is not a contradiction. Let φ be a propositional formula with $p_1, \ldots, p_n$ 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. If $n = 10$ there are 1,024 rows, for $n = 20$ there are already 1,048,576 rows and so on. 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 letter 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. We have the following:

1.4.1. THEOREM (COOK 1971). SAT *is NP-complete.*

What is Tractable? To answer this question the following thesis was formulated, for the first time by Jack Edmonds (1965):

Edmonds' Thesis *The class of practically computable problems is identical to 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 (1979) 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, 1979, p. 8)

This also justifies the following definition of identity between algorithms (synonymy), where we do not distinguish between algorithms which differ in complexity up to some polynomial. According to it, we say that algorithms f and g

are identical if and only if they compute the same partial functions and moreover their working time on the same inputs differ at most by a polynomial function of that input. Notice that with such a definition we get the whole hierarchy of algorithms for a given expression generated by possible model-checking algorithms. This will be important for our considerations in Section 1.8.

Invariance As we said before, computational complexity theory is interested in the inherent complexity of problems independent of particular algorithmic solutions and their implementations. The most common model of computation used in the theory is the Turing machine. However, to justify computational complexity distinctions, e.g., between tractable and intractable problems, one has to give some argument that those distinctions are in fact independent of the particular implementation. The situation here is very similar to assuming the Church-Turing Thesis and the analogous arguments suggest another commonly believed assumption, the so-called Invariance Thesis (see e.g. Garey and Johnson, 1979):

Invariance Thesis *Given a "reasonable encoding" of the input and two "reasonable machines", the complexity of computation of these machines on that input will differ by at most a polynomial amount.*

By "reasonable machine" any type of deterministic Turing machine or any other realistic computing machine (including neural networks, often suggested as a plausible model of brain computational architecture) are meant. Notice, however, that non-deterministic Turing machines (as well as quantum computers) are not realistic in this sense (see the argument for Theorem 2.3.13).

Assuming the Invariance Thesis we get that a task is difficult if it corresponds to a function of a high computational complexity, independent of the computational devices we are working with, at least as long as they are reasonable.

Are PTIME Algorithms Always Tractable? The common belief in Edmonds' Thesis stems from the practice of programmers. NP-hard problems often lead to algorithms which are not practically implementable even for inputs of not very large size. Assuming the Church-Turing Thesis, $P \neq NP$, and the Invariance Thesis one comes to the conclusion that this has to be due to some internal properties of these problems and not to the restrictions of the current computing technology. However, even with these assumptions there are still some doubts. They lead to a better understanding of the nature of computational complexity claims and that is why we briefly discuss some of them below.

First of all, in some cases the polynomial algorithm is essentially better just for very large data. For example, not so long ago the best deterministic procedure checking whether a given number n is prime was bounded by $n^{\frac{1}{5}\log\log n}$. This algorithm was practically used even though it is not polynomial.[11] The reason is very simple: $n^{\frac{1}{5}\log\log n} > n^2$ for only $n > e^{e^{e^{10}}}$, where $e \approx 2.718281$ (Blass and Gurevich, 2003). In other words, the polynomial algorithm is essentially better only for very big input.

Secondly, a polynomial algorithm might be also practically intractable.

$n^{98466506514687}$ is a polynomial but even for small n an algorithm of that working time would not be practical. However, many theorists believe that if we come up with polynomial algorithms of a high degree then it is only a matter of time before they will be simplified. In practice, programmers implement mainly algorithms with time complexity not grater than n^3. On the other hand, exponential procedures are sometimes used if they are supposed to work on small input data. Related additional difficulty comes from the fact that the computational complexity measures do not take constants into considerations (see Definition 2.3.14) as when n increases their influence on the complexity decreases. Then, an algorithm working in time $n^3 + 2^{630}$ is still taken as a reasonable polynomial bound of degree 3.

Finally, let us consider an even more drastic example (see Gurevich, 1995). Let g be an uncomputable function and define f as follows:

$$f(x) = \begin{cases} 1 & \text{if length of } x < 2^{9999999999999999999999999999999} \\ g(x) & \text{otherwise.} \end{cases}$$

Is f computable? For all inputs of reasonable size f is computable and has value 1. It is uncomputable only for extremely big inputs. Therefore, can we intuitively claim that f is computable?

We will discuss again the pros and coins of setting the tractability border along PTIME computability limits in the following chapter. However, now we turn to the cognitive perspective which is more relevant for understanding the background of our technical work.

1.5 Computability and Cognition

Accepting Edmonds' Thesis we have to agree that problems beyond PTIME are computationally intractable. Our practical experience suggests that this is a reasonable assumption even though we raised some theoretic doubts in the previous section. The question appears how this belief influences cognitive science and, particularly, psycholinguistics. Below we briefly review this issue.

[11]In 2002, Indian scientists at IIT Kanpur discovered a new deterministic algorithm known as the AKS algorithm. The amount of time that this algorithm takes to check whether a number n is prime depends on a polynomial function of the logarithm of n (Agrawal et al., 2004).

1.5.1 Computational Explanation of Cognitive Tasks

Cognitive Tasks What is a cognitive task? Taking a very abstract perspective we can say that a cognitive task is an information-processing or computational task. Namely, the aim of a cognitive task is to transform the initial given state of the world into some desired final state. Therefore, cognitive tasks can be identified with functions from possible initial states of the world into possible final states of the world. Notice that this understanding of cognitive tasks is very closely related to psychological practice (see e.g. Sternberg, 2008). First of all, experimental psychology is naturally task oriented, because subjects are typically studied in the context of specific experimental tasks. Furthermore, the dominant approach in cognitive psychology is to view human cognition as a form of information processing (see e.g. van Rooij, 2004).

Marr's Levels One of the primary objectives of behavioral psychology is to explain human cognitive tasks understood in the very abstract way outlined above. David Marr (1983) was the first to propose a commonly accepted general framework for analyzing levels of explanation in cognitive sciences. In order to focus on the understanding of specific problems, he identified three levels (ordered according to decreasing abstraction):

- computational level (problems that a cognitive ability has to overcome);
- algorithmic level (the algorithms that may be used to achieve a solution);
- implementation level (how this is actually done in neural activity).

Marr argues that the best way to achieve progress in cognitive science is by studying descriptions at the computational level in psychological theories. He claims:

> An algorithm is likely to be understood more readily by understanding the nature of the problem being solved than by examining the mechanism (and the hardware) in which it is embodied.
> (Marr, 1983, p. 27)

Marr's ideas in the context of computational complexity were nicely summarized by Frixione (2001):

> The aim of a computational theory is to single out a function that models the cognitive phenomenon to be studied. Within the framework of a computational approach, such a function must be effectively computable. However, at the level of the computational theory, no assumption is made about the nature of the algorithms and their implementation. (Frixione, 2001, p. 381)

1.5.2 Computational Bounds on Cognition

Computable Cognition

How can we apply all the above considerations to cognitive science? First of all, we want to talk about some very general properties of cognitive tasks which can be studied from the perspective of Marr's computational levels. As we have noticed before, one of the abstract computational properties of functions, independent from any particular implementation, is their complexity. Viewing cognitive tasks as functions we can therefore pose questions about their computational complexity. Notice that as we do not know the details of our cognitive hardware or the precise algorithms implemented in the brain, the inherent perspective of computational complexity theory is very well-suited for the purposes of cognitive science.

The most common computational claim about cognition is a so-called psychological version of the Church-Turing thesis.

Psychological Version of the Church-Turing Thesis *The human mind can only deal with computable problems.*

In other words, cognitive tasks consist of computable functions.

This claim is commonly believed. However, despite its wide acceptance, the psychological version of the Church-Turing Thesis has its critics. The first opposition comes from researchers who believe that cognitive systems can do more than Turing machines. In fact, we agree that there are some uncomputable problems in the research scope of cognitive science as well. For example, the general framework of learning (identifiability in the limit) is not computable (see Gold, 1967). Presumably, there are more cognitive tasks lying beyond Turing computability (see Kugel, 1986, for an extensive discussion). Researchers from that camp often argue for the possibility of hyper-computations, i.e., the realizability of super-Turing machines, like Zeno-machines (Accelerated Turing machines) that allow a countably infinite number of algorithmic steps to be performed in finite time (see e.g. Syropoulos, 2008).

Some of them even misuse Gödel's theorems to claim that the human mind cannot have an algorithmic nature. J.R. Lucas (1961) has claimed:

> Goedel's theorem seems to me to prove that Mechanism is false, that is, that minds cannot be explained as machines. So also has it seemed to many other people: almost every mathematical logician I have put the matter to has confessed to similar thoughts, but has felt reluctant to commit himself definitely until he could see the whole argument set out, with all objections fully stated and properly met. This I attempt to do. (Lucas, 1961, p. 112)

Then he gives the following argument. A computer behaves according to a program, hence we can view it as a formal system. Applying Gödel's theorem to this system we get a true sentence which is unprovable in the system. Thus, the machine does not know that the sentence is true while we can see that it is true. Hence, we cannot be a machine.

Lucas' argument was revived by Roger Penrose (1996) who additionally supported it by claiming that the human mind can solve uncomputable problems thanks to a specific quantum feature of the brain's neural system. Let us only mention that Lucas' argument has been strongly criticized by logicians and philosophers (see e.g. Benacerraf, 1967; Pudlak, 1999).

Another stream of opposition is related to practical computability. It is believed that cognitive systems, being physical systems, perform their tasks under computational resource constraints. Therefore, the functions computed by cognitive systems need to be computable in realistic time and with the use of a realistic amount of memory. We agree with this objection, and will consider it in more detail in the next section.

Tractable Cognition

What is tractable cognition? We do not know what kind of a device the human cognitive system is. Hence, it would be particularly useful to define tractable cognition in terms of computational complexity constraints on cognitive functions, taking into account the Invariance Thesis.

As far as we are aware the version of Edmonds' Thesis for cognitive science was for the first time formulated explicitly in print[12] by Frixione (2001) (see e.g. van Rooij, 2008, for a discussion):

P-cognition Thesis *Human cognitive (linguistic) capacities are constrained by polynomial time computability.*

In other words, the P-cognition Thesis states that a cognitive task is (hard) easy if it corresponds to a(n) (in)tractable problem. In our psycholinguistic setting this means that the intractable natural language constructions are those for which the model-checking computational complexity goes beyond polynomial time computability.

Let us give an argument in favor of the P-cognition Thesis. We know that a brain contains roughly 10^{15} synapses operating at about 10 impulses per second, giving more or less 10^{16} synapse operations per second (see e.g. Kandel et al., 2000). Obviously, not every cognitive capacity can claim all of the processing power of the entire brain as there are many parallel cognitive tasks going on

[12]Independently, exactly the same idea was formulated in the talk of Marcin Mostowski at Alfred Tarski Centenary Conference in Warsaw on 1 June 2001 and appeared in print 3 years later (see Mostowski and Wojtyniak, 2004).

every moment. Now, assume that we are dealing with two cognitive problems: the polynomial problem Φ and the exponential problem Ψ. For illustrative purposes assume that for Φ the brain needs to make n^2 steps if the instance of the problem is of size n, while the problem Ψ requires 2^n steps. Table 1.1 shows how much time a brain would need to compute these problems for different sizes of input (if it can work with the speed of $10,000$ steps per second on each of them).

n	Time for Φ	Time for Ψ
10	0.01 sec	0.10 sec
20	0.04 sec	1.75 min
30	0.09 sec	1.2 days
50	0.2 sec	8.4 centuries
00	1 sec	9×10^{17} years

Table 1.1: The table compares the running time of a brain working with power 10,000 steps per second to solve two cognitive tasks Φ and Ψ. Φ is computable in n^2 and Ψ requires 2^n steps.

It is clearly visible from the table that there is an essential difference between polynomial and exponential problems. Even for not very big inputs exponential problems can lie beyond brain power. The P-cognition Thesis emphasizes this difference in the context of cognitive science. In the dissertation we accept the P-cognition Thesis.

Psychological Practice

Complexity claims are used in cognitive science, among other functions, to evaluate the feasibility of computational theories of cognition. For example, Levesque (1988) has recognized the computational complexity of general logic problems (like SAT) and has concluded that we need to adjust logic in order to obtain psychologically realistic models of human reasoning. Similarly, Tsotsos (1990) has noticed that visual search in its general (bottom-up) form is NP-complete. As a result he has suggested adjusting the visual model by assuming that top-down information helps to constrain the visual search space. Moreover, in the study of categorization and subset choice computational complexity serves as a good evaluation of psychological models (see e.g. van Rooij et al., 2005). Additionally, complexity restrictions on cognitive tasks have been noted in philosophy of language and of mind. For instance, Cherniak (1981), Chalmers (1994) and Hofstadter (2007) have argued that a philosophical theory of mind should take computational restrictions seriously. Iris van Rooij (2004) in her dissertation gives many examples of the influence computational restrictions have on cognitive science.

1.5.3 Common Doubts

Our experience shows that many researchers are skeptic about applicability of computational complexity in cognitive science. Below we answer some common doubts.

Limit Behavior

Computational complexity is defined in terms of limit behavior (see Section 2.3.3). In other words, a typical question of computational complexity theory is of the form:

> As the size of the input increases, how do the running time and memory requirements of the algorithm change?

Therefore, computational complexity theory, among other things, investigates the scalability of computational problems and algorithms, i.e., it measures the rate of increase in computational resources required as a problem grows. The implicit assumption here is that the size of the problem is unbounded. For example, models can be of any finite size, formulae can contain any number of distinct variables and so on.

It is often claimed that as complexity theory deals only with relative computational difficulty of problems it has not much to offer for cognitive considerations. Simply put, the inputs we are dealing with in our everyday life are not of arbitrary size. In typical situations they are even relatively small. In fact, critics claim, computational complexity does not say how difficult it is to solve a given problem for a fixed size of the input.

In general, even though computational complexity is formally defined in terms of limit behavior it can still be reasonably interpreted as saying something about problem difficulty on a fixed model. Namely, if the computational complexity of the problem is high, then it means that there are no "clever" algorithms for solving it, e.g., we have to do a complete search throughout the whole universe. Therefore, it is very likely that on a given fixed model we also have to use a brute-force method and this will be again difficult (even for not so big n). For example, checking SAT for a formula containing 5 variables is already quite challenging.

Moreover, even if in typical cognitive situations we are dealing with reasonably small inputs we have no bounds on their size. Potentially, the inputs can grow without limit. Therefore, computational complexity makes perfect sense as a difficulty measure.

Additionally, in experimental practice subjects usually do not realize the size of the universe in which they are supposed to solve some given problem. Therefore, it seems that they develop strategies (algorithms) for all possible universes (sizes).

However, if the size of a problem is too big subjects often start to approximate their solution, using heuristic methods instead of "precise" algorithms. But even then we can still measure the rate of increase of the difficulty (e.g. by changes in reaction time) on the interval where subjects realize their algorithmic procedures. For example, it is known that reaction time increases linearly when subjects are asked to count objects between 3 and 15. Up to 3 or 4 objects the answer is immediate, so-called subitizing. For tasks containing more than 15 elements subjects start to approximate: reaction time is constant and the number of incorrect answers increases dramatically (see e.g. Sternberg, 2008). Analogously, we can ask how reaction time increases when subjects are solving PTIME problems compared to a situation when they are dealing with an NP-hard task.

There are also other experimental possibilities to observe computational complexity restrictions. For example, one can manipulate the universe to get a drop in the complexity and see what happens. In Chapter 7 we study the push-down recognizable quantifier "most" over an appropriately ordered universe, where it could be recognized by finite automata, and we show a decrease in reaction time to the level fitting finite-automata problems. Similar methods can be used in case of NP-hard problems by giving various certificates simplifying tasks and observing changes in subjects' behavior.

Finally, theoretically speaking, it may be more beneficial to assume that tasks are finite but that we do not know upper bounds than to try to develop a complexity measure up to fixed size of a problem. Such a general approach works very well for syntax, where the assumption that language sentences can be of any length has led to to the development of generative grammar and mathematical linguistics (see Section 1.7.1 for a short discussion).

Summing up, various computational complexity measures seem to be a very attractive tool for task-oriented cognitive science. At that point they are the best established proposition for measuring the difficulty of problems in precise mathematical terms. Moreover, this perspective seems promising as it was successfully applied in a few domains of psychological study which we mentioned above. However, there is still not enough work in this direction to draw any strong conclusions.

Last but not least, the biggest quest in applying computational complexity to cognitive science might be in discovering the right complexity measures. For sure, such measures we are using now just give a rough approximation of upper-bound difficulty.

Why Worst-case Complexity?

In our opinion the most serious criticism about the status of the P-cognition Thesis is that attacking worst-case computational complexity as a measure of difficulty for cognitive processes. The point here is whether we really need to consider the worst-case performance of cognitive algorithms on arbitrary inputs. It might be the case that inputs generated by "nature" have some special properties which make problems tractable on those inputs even though in principle they might be NP-hard. It is also possible that our cognitive (linguistic) perception is simplifying things, pre-computing possible inputs. For example, it might be the case that intractability of some problems comes from a parameter which is usually very small no matter how big is the whole input. This way of thinking leads to the consideration of so-called parametrized complexity theory (see Downey and Fellows, 1998; Flum and Grohe, 2006) as a measure for the complexity of cognitive processes. Iris van Rooij (2008) dwells on this subject proposing the Fixed-parameter Tractability Thesis as a refinement of the P-cognition Thesis. Another natural way is to turn toward so-called average-case complexity theory (see e.g. Impagliazzo, 1995; Bogdanov and Trevisan, 2006). It studies the computational complexity of problems on randomly generated inputs. The theory is motivated by the fact that many NP-hard problems are in fact easily computable on "most" of the inputs. One potential problem with applications of this measure to cognitive science is that we are unable to come up with a probability distribution on the space of all possible inputs generated by the "nature".

In principle, we agree that worst-case computational complexity might not be the measure best tailored for describing the hardness of cognitive tasks. However we also think that an asymptotic analysis of cognitive capacities is the necessary first step in the quest for understanding their complexity. In the book we have tried to initiate this first step in the domain of natural language semantics and hope that it will be the beginning of a journey finally leading to a better understanding of natural language processing.

1.5.4 Explaining the Comprehension Task

Let us summarize what we have said so far by answering to the following question:

> How does computational cognition relate to our psycholinguistic perspective?

We want to explain one of the basic components of comprehension. Using the algorithmic theory of meaning we can think about that component in terms of model-checking procedures. Model-checking takes a situation and a natural language expression as an input and then transforms them to the output, i.e., a denotation of the expression in the world. In the paradigmatic case of sentences we get the truth-value. Therefore, model-checking might be treated as an example of a cognitive task.

Now, from the most abstract, computational level of the explanation, we are interested in a mathematical specification of the properties of model-checking. In particular, we will study the computational complexity with respect to logical definability of a given expression. That is why we are interested in the combination of complexity and definability. The theory which pursues the relation between them is called descriptive complexity theory (see Section 2.4). Using it we will investigate the logical properties of natural language quantifiers postulated by some linguistic formalisms to track down the distinction between quantifiers with tractable (PTIME) and intractable (NP-hard) referential meaning (the model-checking problem).

Another aspect which should be taken into account is the algorithmic level of explanation. Even though it is not the perspective we have explicitly taken in the thesis, algorithmic considerations are a part of our study. We give explicit algorithms which can be used to solve model-checking problems for example as parts of our proofs in Chapter 3. We consider concrete procedures for simple quantifiers in Chapter 7 and investigate what kind of model-checking subjects are realizing for some special sorts of sentences in Chapter 6. However, it might be the case that humans are actually using completely different algorithms. In other words, we only claim that complexity restricts agents. Nevertheless, we do not think that humans are always using the simplest possible procedures.

As far as the implementation level is concerned we do not study neurological activity when natural language is processed. However, our work was to some extent motivated by the neurological research discussed in Chapter 7. Therefore, even though we formally restrict ourselves to classical cognitive science we see a possible direct connection with neurocognitive reality.

Below we discuss more specific impacts of computational complexity on the domain of our interest.

1.6 Finite Interpretations

The above discussion leads us to another restriction of our approach which is partially motivated by the computational nature of our considerations.

Most of the authors considering the semantics of natural language are interested only in finite universes. This is also common in papers devoted to natural language quantifiers; let us cite Dag Westerståhl (1984):

> 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:

(14) Exactly five of my children went to the cinema.

(15) 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 classical study of resource bounds in such a setting does not make a lot of sense as we need infinite time and infinite memory resources. On the other hand, maybe we need only finitely many states or loops. Hence, after all it is 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 equilibrioception.

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 trouble-makers 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 Section 2.2). In finite universes there is an easy and commonly accepted solution, which is to compare the cardinal numbers 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.

(16) There are more non-primes than primes among integers.

The sentence is false if we interpret "more" in terms of cardinal numbers of sets. However, 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 a proper measure functions, which are non-logical and context dependent concepts (see Krynicki and Mostowski, 1999, for more details).

Another approach to the problem can be formulated in terms of the so-called weak semantics for second-order definable quantifiers. The main idea of the weak semantics for second-order logic is to consider structures of the form $(\mathbb{M}, \mathcal{K})$, where $\mathbb{M}$ is a model and $\mathcal{K}$ is a class of relations over the universe closed under definability with parameters in a given language. The class $\mathcal{K}$ is used to interpret second-order quantification. Phrases like: "for every relation R", "there is a relation R" are interpreted in $(\mathbb{M}, \mathcal{K})$ as "for every relation R belonging to $\mathcal{K}$" and "there is a relation R belonging to $\mathcal{K}$". This way of interpreting second-order quantification essentially modifies the standard semantics for second-order logic and gives an intuitive semantics for second-order definable quantifiers in arbitrary universes from the natural language point of view (see e.g. Mostowski, 1995).

In this dissertation we work with finite universes as we are mainly interested in the algorithmic aspects of the semantics. However, we find a development toward covering arbitrary universes by adopting the methods outlined above and a proper complexity measure a fascinating challenge for later work.

1.7 Complexity in Linguistics

The topic of language complexity has surfaced in many different contexts and can be measured in many different ways. We are talking only about the computational and descriptive complexity of language, but there are many other aspects of complexity, like lexical, information-theoretic (Kolmogorov complexity), structural or functional. These are usually studied in less logical manner (see e.g. Miestamo et al., 2008) and we do not compare these approaches directly.

1.7.1 Syntax

Noam Chomsky has opened the door to a view of language from the computational perspective (see e.g. Chomsky, 2002, 1969). In the early 1950s he captured language's recursive nature, or ability to make "infinite use of finite means", in Wilhelm von Humboldt's famous words, inventing formal language theory. Chomsky's complexity hierarchy of finite-state, context-free, and context-sensitive languages associated with their automata counterparts (see Section 2.3.1) changed linguistics and opened it to many interactions with computer science and cognitive science.

Chomsky's insight was really of the nature of computational complexity theory. First of all, even though the sentences we encounter are all of bounded length — certainly less than 100 words long — Chomsky has assumed that they might be arbitrarily long. Notice that this is an assumption of exactly the same sort as discussed in Section 1.5.3 on the limit behavior of computational claims, i.e., considering inputs of arbitrary size. This assumption directly lead 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 has early demonstrated the inadequacy of finite-state description of natural languages and has claimed that natural languages are context-free (see Chomsky, 2002). This was a pathbreaking way of thinking. First of all, it has shown how one can mathematically measure some complexity aspects of natural language. Then, it has given a methodological constraint on linguistic theories of syntax; they have to be able to account at least for the context-free languages. Actually, it has also started a long standing debate whether natural language is context-free or not (see e.g. Pullum and Gazdar, 1982; Shieber, 1985; Culy, 1985; Manaster-Ramer, 1987).

Chomsky himself noticed very quickly that studying whether a grammar can generate all strings from a given language, so-called weak generative capacity, can serve only as a kind of "stress test" that doesn't tell us much unless a grammar fails the test. He has claimed:

> 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.[13] [. . .] Presumably, discussion of weak generative capacity marks only a very early and primitive stage of the study of generative grammar. (Chomsky, 2002, :60f)

In subsequent years this belief has led to deeper study of generative formalisms. In particular, using computational complexity one could 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. The early results achieved are summarized in a book by Barton et al. (1987). A more recent survey is due to Pratt-Hartmann (2008). 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 raise to intractable problems.

1.7.2 Between Syntax and Semantics

Model-theoretic 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? The idea is to identify the complexity of a grammar with the logical complexity of the formulae which express the constraints. In other words, we ask

[13]Chomsky uses the term "strong generative capacity" referring to the set of structures that can be generated by a grammar.

here about descriptive complexity of grammars in a similar way to how we will be asking about descriptive complexity of model-checking problems in the following chapters.

We measure the complexity of the sentences that define constraints by saying 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) have proven 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).

Notice that 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, we know that monadic second order logic on trees is decidable although intractable and therefore many linguistic questions about parse trees which are formulated in this logic might be "difficult" to answer. Our approach to studying the complexity of semantics goes along the same descriptive line.

Discourse Complexity

Some of the earliest research trying to combine 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 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.

Pratt-Hartmann (2004) has shown that different fragments of natural language capture various complexity classes. More precisely, he has studied the computational complexity of satisfiability problems for various fragments of natural language. He has proven that the satisfiability problem of the syllogistic fragment is in PTIME as opposed to the fragment containing relative clauses which is NP-complete. He has also described fragments of language such that their computational complexity is even harder (with non-copula verbs or restricted anaphora) and finally he provides the reader with an undecidable fragment containing unrestricted anaphora.

It is also worth to mention very recent paper of Pagin (2009). The author tries to explain compositionality principle in terms of computational complexity, cognitive difficulty as experienced by language users during communication, and language learnability. He argues that compositionality simplifies complexity of language communication.

Unfortunately — as far as we are aware — not much is known about the computational complexity of dynamic formalisms. This is suprising because complexity questions match the dynamic spirit perfectly and moreover can give deeper insights into the plausibility of proposed formalisms. Some complexity results are known for belief revision — mostly showing that the logic lying behind revision mechanisms is highly intractable (see e.g. van Benthem and Pacuit, 2006) — and for game-theoretic semantics (see e.g. Sevenster, 2006). But there are some very interesting questions untouched, for example: What is the complexity of a belief revision (game) when an intractable sentence is the input? Does model-checking complexity relate to the complexity of belief revision or the corresponding game in any way? But first of all, there has not been much done about the complexity of this model from the agent's perspective. For example, how difficult is belief revision for a single agent?

The worst news is that complexity seems to be a totally open problem for formal pragmatics, even though the question of how difficult it is to solve coordination problems in linguistic games seems to be very basic. Probably, some results might be achieved via translation from complexity considerations in algorithmic game theory (see e.g. Nisan et al., 2007). We believe that computational complexity issues should be incorporated in these models to better understand their plausibility.

In this book we study the computational complexity of model-checking for a quantifier fragment of natural language. We often evoke the descriptive perspective to achieve our aims. In other words, we try to understand some aspects of meaning from the computational perspective. Below we say a few words on the underlying assumptions in this enterprise, summing up the previous considerations on semantics, cognitive science and complexity.

1.8 Classifying Meaning by Intractability

We have argued that meaning is an algorithm. This allows us to apply computability considerations to linguistics. Moreover, according to what we said above linguistic capacities, like all cognitive processes, are bounded by computational restrictions. In particular, comprehension of natural language constructions is linked to their computational complexity. We provide some more arguments in favor of this connection throughout the dissertation. We give an example for this dependence in Section 4.5 where we discuss the interpretation of NP-complete reciprocal sentences and suggest that their meaning might be in most cases only comprehended by recognizing inferential properties. Then in Chapter 5 we have a look at collective quantification in natural language, where among others we can observe how computational complexity restricts expressibility of everyday language. In Chapters 6 and 7 we present empirical evidence supporting the statement that computational complexity influences the comprehension of natural language. In the first of these chapters we show that people tend to choose computationally simpler interpretations of so-called Hintikka-like sentences. In the following chapter we give empirical evidence directly linking the reaction time needed to recognize the meaning of sentences with simple quantifiers to the complexity of automata computing this meaning. Below we try to capture our experience with studying the complexity of natural language into the form of a methodological statement about the descriptive power of the linguistic formalism which we needed.

1.8.1 Ristad's Theses

The book of Ristad (1993) 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 which 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 is overgeneralizing linguistic capacities. This second methodological claim can be treated as a kind of semantic Ockham's razor. Summing up, Ristad proposes NP-completeness as a kind of methodological test for plausibility of linguistic theories.

1.8.2 Descriptive Bounds on Everyday Language

We do not completely agree with Ristad but we share his intuitions. However, we prefer to restrict claims of that sort to some fragment of language as the whole of natural language contains expressions whose complexity go beyond practical computability. Nevertheless, we can see a natural intuition supporting the use of the concept of *natural language* excluding "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 pre-theoretic part of natural language.

In this dissertation 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. From this perspective, our experience allows us to state the following claim which resembles Ristad's thesis and was proposed by Mostowski and Szymanik (2005):

Σ_1^1-thesis *Our everyday language is semantically bounded by the properties expressible in the existential fragment of second-order logic.*

In other words, we claim that everyday language can be described in the existential fragment of second-order logic (see Definition 2.1.1). If some property is not definable by any Σ_1^1-formula, then it falls outside the scope of everyday language. For example, quantifiers "there exists", "all", "exactly two", "at least four", "every other" and "most" belongs to everyday language. The counterexample is the notion "there exist at most countably many" which is not definable by any Σ_1^1-formula. Moreover, we know from Fagin's theorem (2.4.4) that Σ_1^1-properties correspond to NPTIME. Hence, our thesis basically restricts the methodological claim of Ristad to the realm of everyday language.

Let us give one argument in favor of accepting this methodological statement. The other one, more domain specific, will be formulated in Section 6.3.4. The intuition is that the core sentences of everyday language are sentences which can be more or less effectively verifiable. We argued that in the case of small finite interpretations this means that their 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. From the P-cognition Thesis it follows that the referential meaning of an expression which can be directly computed is in PTIME. However, as we mentioned in Section 1.3 we frequently understand sentences indirectly, evoking their inferential dependencies with other sentences. Let us consider the following three sentences:

(17) There were more boys than girls at the party.

(18) At the party every girl was paired with a boy.

(19) Peter came alone to the party.

We know that sentence (17) can be inferred from sentences (18) and (19). Then we can establish the truth-value of sentence (17) indirectly, knowing that sentences (18) and (19) are true.

The question arises: What do we mean by a tractable inferential meaning?

First notice that in the case of NPTIME problems the non-deterministic behavior of an algorithm can be described as follows:

> Firstly, choose a certificate of a size polynomially depending on the size of 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, 1979, p. 28)

Let us observe that in a sense such certificates are proofs. When we have a proof of a statement then we can easily check whether the sentence is true. Therefore NPTIME problems (Σ_1^1-properties) are practically justifiable. Let us consider an example.

Suppose that we know that the following are true statements:

(20) Most villagers are communists.

(21) Most townsmen are capitalists.

(22) All communists and all capitalists hate each other.

From these sentences we can easily infer the NP-complete branching interpretation of the following sentence (for a discussion on the computational complexity of sentences like this see Section 3.2 and Chapter 6):

(23) Most villagers and most townsmen hate each other.

Sentences (20), (21), and (22) have to be given or guessed. They are in a sense certificates or proofs of the truth of sentence (23).

In this sense sentences with NPTIME meaning — or by Fagin's theorem (see Theorem 2.4.4) Σ_1^1-expressible sentences — are indirectly verifiable. Moreover, NPTIME seems to capture exactly indirect verifiability. This observation gives an argument for our claim — everyday sentences have practically computable inferential meaning.

We assume the Σ_1^1-thesis — as a methodological principle — throughout the dissertation. We are using it to argue in favor of or against some linguistic formalizations. For instance, in Chapter 5 we study a common way of modeling collective quantification in natural language, the so-called type-shifting approach. By using the formalism of second-order generalized quantifiers, we observe that a uniform treatment of collectivity via type-shifting has excessive computational complexity. Therefore, we claim that either such a model cannot be plausible, and propose turning to computationally better behaved many-sorted models or that some instances of collective quantification do not appear in everyday language. On the other hand, the positive use of the Σ_1^1-thesis can be found in Chapter 4, where we analyze the computational complexity of different interpretations of English reciprocal sentences with quantifiers in the antecedents. We work with a model which is expressible in the existential fragment of second-order logic and show that it predicts some reciprocal sentences to be NP-complete. Our thesis allows us to believe that the model is plausible. Moreover, in Chapter 6 we consider Barwise's test of negation normality and observe that it makes sense only when assuming that everyday sentences are expressible in the existential fragment of second order-logic (see Remark 6.3.2).

Moreover, the Σ_1^1-thesis partially bridges the two basic comprehension mechanisms formulated in Section 1.1. It explains in some cases the subtle dependence between the complexity of a sentence, its model-checking difficulty, and its relation to knowledge. It also relaxes our assumption about tractability. As a matter of fact, we found NP-complete semantic constructions in natural language. We claimed that such expressions are intractable for human agents but now we add one proviso: their direct model-checking can be intractable but we can still evoke some indirect mechanism making the task perfectly tractable. This puts our position very close to Ristad's perspective. Additionally, it provides one idea how 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 comprehension.

1.9 Summary

In this chapter we have discussed some motivations and philosophical assumptions lying in the background of 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.

Below we summarize our assumptions:

- We mostly study a basic element of comprehension: model-checking, to which we sometimes refer as "referential meaning".

- We identify (referential) meanings (algorithms) with Turing machines (computable functions).

- We assume that problems are tractable if they are computable in polynomial time. Otherwise, they are intractable.

- We assume that human cognitive (linguistic) capacities are constrained by polynomial time computability. Therefore, if a meaning corresponds to intractable functions we believe it is too hard for direct comprehension.
- We claim that the semantics of everyday language can be adequately described in the existential fragment of second order-logic (Σ_1^1-thesis).
- We restrict ourselves to finite universes.

We hope that our technical insights will give additional arguments in favor of our assumptions.

Chapter 2

Mathematical Prerequisites

This chapter describes the notation and basic terminology from generalized quantifier theory and computation theory. We focus only on concepts which will be explored in the following parts of the thesis. We assume some familiarity with the basics of first-order and second-order logic (see e.g. Ebbinghaus et al., 1996).

2.1 Basic Notation

Let A, B be sets. We will write $\emptyset$ for the *empty set*, $\mathrm{card}(A)$ to denote the *cardinality of the set* A and $\mathcal{P}(B)$ for the *power set* of B. The operations $\cup$, $\cap$, $-$, and the relation $\subseteq$ on sets are defined as usual. The set $A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$ denotes the *Cartesian product* of A and B, where (a,b) is the ordered pair. If R is a relation then by R^{-1} we denote its inverse. *The set of natural numbers* is denoted by ω. If $k \in \omega$ then A^k denotes

$$\underbrace{A \times \ldots \times A}_{k}.$$

We will write $\mathbb{Q}$ for *the set of rational numbers.* If $q \in \mathbb{Q}$ we write $\lceil q \rceil$ for the ceiling function of q, i.e. the smallest integer greater than q.

Let φ and ψ be formulae. We write $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \implies \psi$, $\varphi \iff \psi$, $\exists x \varphi(x)$, and $\forall x \varphi(x)$ with the usual meaning. We will denote first-order logic (elementary logic) by FO and second-order logic by SO.

Now, let us recall the definition of the *hierarchy of second-order formulae.*

2.1.1. Definition. The class Σ^1_0 is identical to the class Π^1_0 and both contain (all and only) the first-order formulae. The class Σ^1_{n+1} is the set of formulae of the following form: $\exists P_1 \ldots \exists P_k \psi$, where $\psi \in \Pi^1_n$. The class Π^1_{n+1} consists of formulae of the form: $\forall P_1 \ldots \forall P_k \psi$, where $\psi \in \Sigma^1_n$. We additionally assume that all formulae equivalent to some Σ^1_n (or Π^1_n) formula are also in Σ^1_n (respectively Π^1_n).

■

In the thesis a *vocabulary* is a finite set consisting of relation symbols (predicates) with assigned arities. Let $\tau = \{R_1, \ldots, R_k\}$ be a vocabulary, where for each i, n_i is the arity of R_i. Then a τ-*model* will be a structure of the following form: $\mathbb{M} = (M, R_1, \ldots, R_k)$, where M is the universe of model $\mathbb{M}$ and $R_i \subseteq M$ is an n_i-ary relation over M, for $1 \leq i \leq k$. If φ is a τ-sentence (a sentence over the vocabulary τ) then the class of τ-models of φ is denoted by $\mathrm{Mod}(\varphi)$. We will sometimes write R_i^M for relations to differentiate them from the corresponding predicates R_i.

2.2 Generalized Quantifier Theory

Generalized quantifiers are one of the basic tools of today's linguistics and their mathematical properties have been extensively studied since the fifties (see Peters and Westerståhl, 2006, for a recent overview). In its simplest form generalized quantifier theory assigns meanings to statements by defining the semantics of the quantifiers occurring in them. For instance, the quantifiers "every", "some", "at least 7", "an even number of", and "most" build the following sentences.

(1) Every poet has low self-esteem.

(2) Some dean danced nude on the table.

(3) At least 7 grad students prepared presentations.

(4) An even number of the students saw a ghost.

(5) Most of the students think they are smart.

(6) Less than half of the students received good marks.

What is the semantics assigned to these quantifiers? Formally they are treated as relations between subsets of the universe. For instance, in sentence (1) "every" is a binary relation between the set of poets and the set of people having low self-esteem. Following the natural linguistic intuition we will say that sentence (1) is true if and only if the set of poets is included in the set of people having low self-esteem. Hence, the quantifier "every" corresponds in this sense to the inclusion relation.

Let us now have a look at sentence (4). It is true if and only if the intersection of the set of all students with the set of people who saw a ghost is of even cardinality. That is, this quantifier says something about the parity of the intersection of two sets.

Finally, let us consider example (5). Let us assume that the quantifier "most" simply means "more than half".[1] Hence, sentence (5) is true if and only if the

[1]We would say that the stronger, but vague, meaning that seems more natural is a result of pragmatics.

cardinality of the set of students who think they are smart is greater than the cardinality of the set of students who do not think they are smart. That is, the quantifier "most" expresses that these two kinds of student exist in a specific proportion.

2.2.1 Two Equivalent Concepts of Generalized Quantifiers

Frege was one of the major figures in forming the modern concept of quantification. In his Begriffsschrift (1879) he made a distinction between bound and free variables and treated quantifiers as well-defined, denoting entities. He thought of quantifiers as third-order objects — relations between subsets of a given fixed universe. This way of thinking about quantifiers is still current, particularly in linguistics. However, historically speaking the notion of a generalized quantifier was formulated for the first time in a different, although equivalent, way: generalized quantifiers were defined as classes of models closed under isomorphisms. Firstly, in a seminal paper of Andrzej Mostowski (1957) the notions of existential and universal quantification were extended to the concept of a monadic generalized quantifier binding one variable in one formula, and later this was generalized to arbitrary types by Per Lindström (1966). Below we give the formal definition.

2.2.1. Definition. Let $t = (n_1, \ldots, n_k)$ be a k-tuple of positive integers. *A Lindström quantifier* of type t is a class Q of models of a vocabulary $\tau_t = \{R_1, \ldots, R_k\}$, such that R_i is n_i-ary for $1 \leq i \leq k$, and Q is closed under isomorphisms, i.e. if $\mathbb{M}$ and $\mathbb{M}'$ are isomorphic, then

$$(\mathbb{M} \in \mathsf{Q} \iff \mathbb{M}' \in \mathsf{Q}).$$

■

2.2.2. Definition. If in the above definition for all i: $n_i \leq 1$, then we say that a quantifier is *monadic*, otherwise we call it *polyadic*.

■

2.2.3. Example. Let us explain this definition further by giving a few examples. Sentence (1) is of the form Every A *is* B, where A stands for poets and B for people having low self-esteem. As we explained above the sentence is true if and only if $A \subseteq B$. Therefore, according to the definition, the quantifier "every" is of type (1, 1) and corresponds to the class of models (M, A, B) in which $A \subseteq B$. For the same reasons the quantifier "an even number of" corresponds to the class of models in which the cardinality of $A \cap B$ is an even number. Finally, let us consider the quantifier "most" of type (1, 1). As we mentioned before the sentence Most A*s are* B is true if and only if $\mathrm{card}(A \cap B) > \mathrm{card}(A - B)$ and therefore the quantifier corresponds to the class of models where this inequality holds.

Therefore, formally speaking:

$$\begin{aligned}
\forall &= \{(M, A) \mid A = M\}.\\
\exists &= \{(M, A) \mid A \subseteq M \text{ and } A \neq \emptyset\}.\\
\mathsf{Every} &= \{(M, A, B) \mid A, B \subseteq M \text{ and } A \subseteq B\}.\\
\mathsf{Even} &= \{(M, A, B) \mid A, B \subseteq M \text{ and } \mathrm{card}(A \cap B) \text{ is even}\}.\\
\mathsf{Most} &= \{(M, A, B) \mid A, B \subseteq M \text{ and } \mathrm{card}(A \cap B) > \mathrm{card}(A - B)\}.
\end{aligned}$$

The first two examples are the standard first-order universal and existential quantifiers, both of type (1). They are classes of models with one unary predicate such that the extension of the predicate is equal to the whole universe in case of the universal quantifier and is nonempty in case of the existential quantifier.

Why do we assume that these classes are closed under isomorphisms? Simply put, this guarantees that the quantifiers are topic neutral. The quantifier "most" means exactly the same when applied to people as when applied to natural numbers.

Let us now give the definition of a generalized quantifier. This definition is commonly used in linguistics as opposed to the previous one which finds its applications rather in logic.

2.2.4. DEFINITION. A *generalized quantifier* Q of type $t = (n_1, \ldots, n_k)$ is a functor assigning to every set M a k-ary relation Q_M between relations on M such that if $(R_1, \ldots, R_k) \in \mathsf{Q}_M$ then R_i is an n_i-ary relation on M, for $i = 1, \ldots, k$. Additionally, Q is preserved by bijections, i. e., if $f : M \longrightarrow M'$ is a bijection then $(R_1, \ldots, R_k) \in \mathsf{Q}_M$ if and only if $(fR_1, \ldots, fR_k) \in \mathsf{Q}_{M'}$, for every relation $R_1, \ldots, R_k$ on M, where $fR = \{(f(x_1), \ldots, f(x_i)) \mid (x_1, \ldots, x_i) \in R\}$, for $R \subseteq M^i$.

■

In other words, a generalized quantifier Q is a functional relation associating with each model $\mathbb{M}$ a relation between relations on its universe, M. Hence, if we fix a model $\mathbb{M}$, then we can treat a generalized quantifier as a relation between relations over the universe, and this is the familiar notion from natural language semantics.

Notice that we have the following equivalence between a Lindström quantifier and a generalized quantifier:

$$(M, R_1, \ldots, R_k) \in \mathsf{Q} \iff \mathsf{Q}_M(R_1, \ldots, R_k), \text{where } R_i \subseteq M^{n_i}.$$

For instance, in a given model $\mathbb{M}$ the statement $\mathsf{Most}_M(A, B)$ says that $\mathrm{card}(A^M \cap B^M) > \mathrm{card}(A^M - B^M)$, where $A^M, B^M \subseteq M$.

2.2.5. COROLLARY. *The definitions of a Lindström quantifier (2.2.1) and a generalized quantifier (2.2.4) are equivalent.*

Studying the properties of quantifiers in most cases we invoke Definition 2.2.1 but for descriptive purposes over some fixed universe we mostly make use of Definition 2.2.4, treating quantifiers as third-order concepts (relations between relations).

2.2.2 Branching Quantifiers

As a matter of chronology, the idea of generalizing Frege's quantifiers arose much earlier than the work of Lindström (1966). The idea was to analyze possible dependencies between quantifiers — dependencies which are not allowed in the standard linear (Fregean) interpretation of logic. *Branching quantification* (also called partially ordered quantification, or Henkin quantification) was proposed by Leon Henkin (1961) (for a survey see Krynicki and Mostowski (1995)). Branching quantification significantly extends the expressibility of first-order logic; for example the so-called Ehrenfeucht sentence, which uses branching quantification, expresses infinity:

$$\exists t \begin{pmatrix} \forall x \exists x' \\ \forall y \exists y' \end{pmatrix} \big[(x = y \iff x' = y') \wedge x' \neq t\big].$$

Informally speaking, the idea of such a construction is that for different rows the values of the quantified variables are chosen independently. The semantics of branching quantifiers can be formulated mathematically in terms of Skolem functions. For instance, the Ehrenfeucht sentence after Skolemization has the following form:

$$\exists t \exists f \exists g \big[\forall x \forall y (x = y \iff f(x) = g(y)) \wedge f(x) \neq t\big].$$

Via simple transformations this sentence is equivalent to the following:

$$\exists f \forall x \forall y \big[(x \neq y \implies f(x) \neq f(y)) \wedge (\exists t \forall x (f(x) \neq t))\big],$$

and therefore, it expresses Dedekind's infinity: there exists an injective function from the universe to itself which is not surjective.

The idea of the independent (branching) interpretation of quantifiers has given rise to many advances in logic. Let us mention here only the logical study of (in)dependence by investigating Independence Friendly Logic (see Hintikka, 1996) and Dependence Logic (see Väänänen, 2007). It is also worth noting that Game-Theoretic Semantics (see Hintikka and Sandu, 1997) was originally designed as an alternative semantics for branching quantification (Independence Friendly Logic). Now it is considered as a useful tool for studying different variants of independence, like imperfect information games (see Sevenster, 2006). We present computational complexity results for branching quantifiers in Section 3.2 and discuss their linguistic applications in Chapter 6.

2.2.3 Logic Enriched by Generalized Quantifiers

Generalized quantifiers enable us to enrich the expressive power of a logic in a very controlled and minimal way. Below we define the extension of an arbitrary logic $\mathcal{L}$ by a generalized quantifier Q.

2.2.6. Definition. We define the extension, $\mathcal{L}(\mathsf{Q})$, of logic $\mathcal{L}$ by a quantifier Q of type $t = (n_1, \ldots, n_k)$ in the following way:

- The formula formation rules of $\mathcal{L}$-language are extended by the rule:

  if for $1 \leq i \leq k$, $\varphi_i(\overline{x}_i)$ is a formula and $\overline{x}_i$ is an n_i-tuple of pairwise distinct variables, then $\mathsf{Q}\, \overline{x}_1, \ldots, \overline{x}_k\, [\varphi_1(\overline{x}_1), \ldots, \varphi_k(\overline{x}_k)]$ is a formula.

■

2.2.7. Convention. Let us observe that this definition can be modified according to common notational habits as follows. Q is treated as binding $n = \max(n_1, \ldots, n_k)$ variables in k formulae. For example, the quantifier Every of type $(1, 1)$ which expresses the property $\forall x[P_1(x) \implies P_2(x)]$ can be written according to the first convention as:

$$\mathsf{Every}\ xy\ [P_1(x), P_2(y)]$$

and according to the modified one as:

$$\mathsf{Every}\ x\ [P_1(x), P_2(x)].$$

2.2.8. Definition. The satisfaction relation of $\mathcal{L}$ is extended by the rule:

$$\mathbb{M} \models \mathsf{Q}\, \overline{x}_1, \ldots, \overline{x}_k\, [\varphi_1(\overline{x}_1), \ldots, \varphi_k(\overline{x}_k)] \text{ iff } (M, \varphi_1^{\mathbb{M}}, \ldots, \varphi_k^{\mathbb{M}}) \in \mathsf{Q},$$

where $\varphi_i^{\mathbb{M}} = \{\overline{a} \in M^{n_i} \mid \mathbb{M} \models \varphi_i(\overline{a})\}$.

■

In this dissertation we will mainly be concerned with extensions of first-order logic, FO, by different generalized quantifiers Q. Following Definition 2.2.6 such an extension will be denoted by $\mathrm{FO}(\mathsf{Q})$.

2.2.4 Definability of Generalized Quantifiers

Some generalized quantifiers, like $\exists^{\leq 3}$, $\exists^{=3}$, and $\exists^{\geq 3}$, are easily expressible in elementary logic. This is also true for many natural language determiners. For example, we can express the type (1, 1) quantifier Some by the type (1) first-order existential quantifier in the following way:

$$\mathsf{Some}\ x\ [A(x), B(x)] \iff \exists x[A(x) \wedge B(x)].$$

However, it is well-known that many generalized quantifiers are not definable in first-order logic. Standard application of the compactness theorem shows that the quantifiers “there exist (in)finitely many“ are not FO-definable. Moreover, using Ehrenfeucht-Fraïssé games it is routine to prove the following:

2.2.9. Proposition. *The quantifiers* Most *and* Even *are not first-order definable.*[2]

Dealing with quantifiers not definable in first-order logic we will consider their definitions in higher-order logics, for instance in fragments of second-order logic. To give one example here, in a model $\mathbb{M} = (M, A^M, B^M)$ the sentence $\mathsf{Most}\ x\ [A(x), B(x)]$ is true if and only if the following condition holds:

$$\exists f : (A^M - B^M) \longrightarrow (A^M \cap B^M) \text{ such that f is injective but not surjective.}$$

In general, definability theory investigates the expressibility of quantifiers in various logics. Informally, definability of a quantifier Q in a logic $\mathcal{L}$ means that there is a uniform way to express every formula of the form $\mathsf{Q}\, x\, \varphi$ in $\mathcal{L}$. The following is a precise definition.

2.2.10. Definition. Let Q be a generalized quantifier of type t and $\mathcal{L}$ a logic. We say that the quantifier Q is *definable* in $\mathcal{L}$ if there is a sentence $\varphi \in \mathcal{L}$ of vocabulary τ_t such that for any τ_t-structure $\mathbb{M}$:

$$\mathbb{M} \models \varphi \text{ iff } \mathbb{M} \in \mathsf{Q}\,.$$

■

2.2.11. Definition. Let $\mathcal{L}$ and $\mathcal{L}'$ be logics. The logic $\mathcal{L}'$ is *at least as strong as* the logic $\mathcal{L}$ ($\mathcal{L} \leq \mathcal{L}'$) if for every sentence $\varphi \in \mathcal{L}$ over any vocabulary there exists a sentence $\psi \in \mathcal{L}'$ over the same vocabulary such that:

$$\models \varphi \iff \psi.$$

The logics $\mathcal{L}$ and $\mathcal{L}'$ are *equivalent* ($\mathcal{L} \equiv \mathcal{L}'$) iff $\mathcal{L} \leq \mathcal{L}'$ and $\mathcal{L}' \leq \mathcal{L}$.

■

Below we assume that a logic $\mathcal{L}$ has the so-called*substitution property*, i.e., that the logic $\mathcal{L}$ is closed under substituting predicates by formulas. The following fact is well-known for Lindström quantifiers.

[2]For more undefinability results together with mathematical details and an introduction into Ehrenfeucht-Fraïssé techniques we suggest consulting the literature (e.g. Chapter 4 in Peters and Westerståhl, 2006).

2.2.12. PROPOSITION. *Let* Q *be a generalized quantifier and* $\mathcal{L}$ *a logic. The quantifier* Q *is definable in* $\mathcal{L}$ *iff*

$$\mathcal{L}(\mathsf{Q}) \equiv \mathcal{L}.$$

Proof The direction $\mathcal{L} \leq \mathcal{L}(\mathsf{Q})$ is obvious since every formula of $\mathcal{L}$ is also a formula of $\mathcal{L}(\mathsf{Q})$ (see Definition 2.2.6). To show that $\mathcal{L}(\mathsf{Q}) \leq \mathcal{L}$ we use inductively the fact that if φ is the formula which defines Q and $\psi_1(\overline{x}_1), \ldots, \psi_k(\overline{x}_k)$ are formulae of $\mathcal{L}$, then

$$\models \mathsf{Q}\, \overline{x}_1, \ldots, \overline{x}_k \, [\psi_1(\overline{x}_1), \ldots, \psi_k(\overline{x}_k)] \iff \varphi(R_1/\psi_1, \ldots, R_k/\psi_k),$$

where the formula on the right is obtained by substituting every occurrence of $R_i(\overline{x}_i)$ in φ by $\psi_i(\overline{x}_i)$. □

2.2.5 Linguistic Properties of Generalized Quantifiers

It was realized by Montague (1970) that the notion of a generalized quantifier — as a relation between sets — can be used to model the denotations of noun phrases in natural language. Barwise and Cooper (1981) introduced the apparatus of generalized quantifiers as a standard semantic toolbox and started the rigorous study of their properties from the linguistic perspective. Below we define some properties of quantifiers important in linguistics which we use in our further research (see Peters and Westerståhl, 2006, for a general overview).

Boolean Combinations of Quantifiers

To account for complex noun phrases, like those occurring in sentences (7)–(10), we define disjunction, conjunction, outer negation (complement) and inner negation (post-complement) of generalized quantifiers.

(7) At least 5 or at most 10 departments can win EU grants. (disjunction)

(8) Between 100 and 200 students started in the marathon. (conjunction)

(9) Not all students passed. (outer negation)

(10) All students did not pass. (inner negation)

2.2.13. DEFINITION. Let Q, Q' be generalized quantifiers, both of type $(n_1, \ldots, n_k)$. We define:

$$(\mathsf{Q} \wedge \mathsf{Q}')_M[R_1, \ldots, R_k] \iff \mathsf{Q}_M[R_1, \ldots, R_k] \text{ and } \mathsf{Q}'_M[R_1, \ldots, R_k] \text{ (conjunction)}$$

$$(\mathsf{Q} \vee \mathsf{Q}')_M[R_1, \ldots, R_k] \iff \mathsf{Q}_M[R_1, \ldots, R_k] \text{ or } \mathsf{Q}'_M[R_1, \ldots, R_k] \text{ (disjunction).}$$

$$(\neg\, \mathsf{Q})_M[R_1, \ldots, R_k] \iff \text{ not } \mathsf{Q}_M[R_1, \ldots, R_k] \text{ (complement)}$$

$$(\mathsf{Q}\, \neg)_M[R_1, \ldots, R_k] \iff Q_M[R_1, \ldots, R_{k-1}, M - R_k] \text{ (post-complement)}$$

■

Relativization of Quantifiers

Every statement involving a quantifier Q is about some universe M. Sometimes it is useful to define a new quantifier saying that Q restricted to some subset of M behaves exactly as it behaves on the whole universe M. Below we give the formal definition.

2.2.14. Definition. Let Q be of type $(n_1, \ldots, n_k)$; then the *relativization* of Q, Q^{rel}, has the type $(1, n_1, \ldots, n_k)$ and is defined for $A \subseteq M, R_i \subseteq M^{n_i}, 1 \leq i \leq k$ as follows:

$$\mathsf{Q}_M^{rel}[A, R_1, \ldots, R_k] \iff \mathsf{Q}_A[R_1 \cap A^{n_1}, \ldots, R_k \cap A^{n_k}].$$

■

In particular, for Q of type (1) we have:

$$\mathsf{Q}_M^{rel}[A, B] \iff \mathsf{Q}_A[A \cap B].$$

2.2.15. Example. This already shows that many natural language determiners of type (1, 1) are just relativizations of some familiar logical quantifiers, e.g.:

$$\mathsf{Some} = \exists^{rel};$$

$$\mathsf{Every} = \forall^{rel}.$$

Domain Independence

Domain independence is a property characteristic of natural language quantifiers. It says that the behavior of a quantifier does not change when you extend the universe. The formal definition follows.

2.2.16. Definition. A quantifier of type $(n_1, \ldots, n_k)$ satisfies *domain independence* (EXT) iff the following holds:

$$\text{If } R_i \subseteq M^{n_i}, 1 \leq i \leq k, M \subseteq M', \text{ then } \mathsf{Q}_M[R_1, \ldots, R_k] \iff \mathsf{Q}_{M'}[R_1, \ldots R_k].$$

■

Conservativity

The property which in a sense extends EXT is conservativity:

2.2.17. Definition. A type (1, 1) quantifier Q is *conservative* (CONS) iff for all M and all $A, B \subseteq M$:

$$\mathsf{Q}_M[A, B] \iff \mathsf{Q}_M[A, A \cap B].$$

■

CE-quantifiers

Quantifiers closed under isomorphisms and satisfying CONS and EXT are known in the literature as CE-quantifiers. It has been hypothesized that all natural language determiners correspond to CE-quantifiers (see e.g. Barwise and Cooper, 1981). From this perspective expressions like "John", $\forall$, and "only" are excluded from the realm of natural language determiners by not being CE-quantifiers. The proper name "John" does not satisfy isomorphism closure, $\forall$ is not EXT and "only" violates conservativity.

Notice that CE-quantifiers of type (1, 1) over finite universes may be identified with pairs of natural numbers.

2.2.18. Definition. Let Q be a CE-quantifier of type (1, 1). We define a binary relation also denoted by Q:

$$\mathsf{Q}(k,m) \iff \text{there is } M, \text{ and } A, B \subseteq M \text{ such that}$$
$$\text{card}(A-B)=k, \text{card}(A\cap B)=m, \text{ and } \mathsf{Q}_M[A,B].$$

■

2.2.19. Theorem. *If Q is a CE-quantifier of type (1, 1), then for all M and all $A, B \subseteq M$ we have:*

$$\mathsf{Q}_M[A,B] \iff \mathsf{Q}(\text{card}(A-B), \text{card}(A\cap B)).$$

Proof It is enough to show that whenever Q is CE, $A, B \subseteq M$, $A', B' \subseteq M'$ are such that: $\text{card}(A-B) = \text{card}(A'-B')$ and $\text{card}(A\cap B) = \text{card}(A'\cap B')$, $Q_M[A,B] \iff Q_{M'}[A',B']$.

□

Monotonicity

One of the most striking intuitions about quantifiers is that they say that some sets are "large enough". Therefore, we would expect that quantifiers are closed on some operations changing universe. The simplest among such operations is taking subsets and supersets. Monotonicity properties state whether a quantifier is closed under these operations.

2.2.20. Definition. A quantifier Q_M of type $(n_1,\ldots,n_k)$ is *monotone increasing in the i'th argument* (upward monotone) iff the following holds:

$$\text{If } \mathsf{Q}_M[R_1,\ldots,R_k] \text{ and } R_i \subseteq R_i' \subseteq M^{n_i}, \text{ then}$$
$$\mathsf{Q}_M[R_1,\ldots,R_{i-1},R_i',R_{i+1},\ldots,R_k], \text{ where } 1 \leq i \leq k.$$

■

2.2.21. Definition. A quantifier Q_M of type $(n_1, \ldots, n_k)$ is *monotone decreasing in the i'th argument* (downward monotone) iff the following holds:

$$\text{If } \mathsf{Q}_M[R_1, \ldots, R_k] \text{ and } R_i' \subseteq R_i \subseteq M^{n_i}, \text{ then } \mathsf{Q}_M[R_1, \ldots, R_{i-1}, R_i', R_{i+1}, \ldots, R_k], \text{ where } 1 \leq i \leq k.$$

■

In particular, for a quantifier Q of type (1, 1) we can define the following basic types of monotonicity:

↑MON $\mathsf{Q}_M[A, B]$ and $A \subseteq A' \subseteq M$ then $\mathsf{Q}_M[A', B]$.

↓MON $\mathsf{Q}_M[A, B]$ and $A' \subseteq A \subseteq M$ then $\mathsf{Q}_M[A', B]$.

MON↑ $\mathsf{Q}_M[A, B]$ and $B \subseteq B' \subseteq M$ then $\mathsf{Q}_M[A, B']$.

MON↓ $\mathsf{Q}_M[A, B]$ and $B' \subseteq B \subseteq M$ then $\mathsf{Q}_M[A, B']$.

We also consider combinations of these basic types, for example we will write ↑MON↓ for a quantifier that is monotone increasing in its left argument and decreasing in its right argument.

Upward monotonicity in the left argument is sometimes called persistence. It is an important property for the study of noun phrases in natural language (see e.g. Peters and Westerståhl, 2006).

2.2.22. Definition. We say that a quantifier is *monotone* if it is monotone decreasing or increasing in any of its arguments. Otherwise, we call it *non-monotone.* ■

Obviously, monotonicity interacts in a subtle way with outer and inner negations.

2.2.23. Proposition. *Let* Q *be any type (1, 1) quantifier.* Q *is MON↑*

(1) iff $\neg\mathsf{Q}$ *is MON↓.*

(2) iff $\mathsf{Q}\neg$ *is MON↓.*

Q *is ↑MON*

(1) iff $\neg\mathsf{Q}$ *is ↓MON.*

(2) iff $\mathsf{Q}\neg$ *is ↑MON.*

Similarly (with reversed arrows) for the downward monotone case.

Proof Obviously, outer negation reverses monotonicity, in any argument. Inner negation reverses monotonicity only in the second argument, since there we are looking at complements, but not in the first argument.

□

2.2.24. Example. Consider the Aristotelian square of opposition. It consists of the following four quantifiers: Some, ¬ Some = No, Some ¬ = Not all, ¬ Some ¬ = All . Some is ↑MON↑. Therefore, No is ↓MON↓, Not all is ↑MON↓, and All is ↓MON↑.

Moreover, Most is an example of a quantifier which is not persistent but is upward monotone in its right argument (i.e., ∼MON↑). Even is non-monotone (∼MON∼).

It is not surprising that monotonicity is one of the key properties of quantifiers, both in logic and linguistics. In model theory it contributes to definability (see e.g. Väänänen and Westerståhl, 2002), in linguistics it is used — among other applications — to explain the phenomenon of negative polarity items (see e.g. Ladusaw, 1979). Moreover, there are good reasons to believe that it is a crucial feature for processing natural language quantifiers, as has already been suggested by Barwise and Cooper (1981) and empirically supported by Geurts (2003) as well as our research presented in Chapter 7. There are also strong links between monotonicity and the learnability of quantifiers (see e.g. Tiede, 1999; Gierasimczuk, 2009).

2.3 Computability

Throughout the thesis we use the general notions of computability theory (see e.g. Hopcroft et al., 2000; Cooper, 2003). In particular, we refer to the basic methods and notions of complexity theory (see e.g. Papadimitriou, 1993; Kozen, 2006). Below we review some of them briefly to keep the thesis self-contained.

2.3.1 Languages and Automata

Formal language theory — which we briefly survey below — is an important part of logic, computer science and linguistics (see e.g. Hopcroft et al., 2000, for a complete treatment). Historically speaking, formal language theory forms the foundation of modern (mathematical) linguistics and its connection with psycholinguistics (see e.g. Partee et al., 1990).

Languages

By an *alphabet* we mean any non-empty finite set of symbols. For example, $A = \{a, b\}$ and $B = \{0, 1\}$ are two different binary alphabets.

A *word (string)* is a finite sequence of symbols from a given alphabet, e.g., "1110001110" is a word over the alphabet B.

The *empty word* is a sequence without symbols. It is needed mainly for technical reasons and written ε.

The *length of a word* is the number of symbols occurring in it. We write $lh()$ for length, e.g., $lh(111) = 3$ and $lh(\varepsilon) = 0$.

If Γ is an alphabet, then by Γ^k we mean the *set of all words of length k over* Γ. For instance, $A^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$. For every alphabet Γ we have $\Gamma^0 = \{\varepsilon\}$.

For any letter a and a natural number n by a^n we denote a string of length n consisting only from the letter a.

The set of all words over alphabet Γ is denoted by Γ^*, e.g., $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$. In other words, $\Gamma^* = \bigcup_{n\in\omega} \Gamma^n$. Almost always Γ^* is infinite, except for two cases: for $\Gamma = \emptyset$ and $\Gamma = \{\varepsilon\}$.

By xy we mean the *concatenation* of the word x with the word y, i.e., the new word xy is built from x followed by y. If $x = a_1 \ldots a_i$ and $y = b_1 \ldots b_n$, then xy is of length $i + n$ and $xy = a_1 \ldots a_i b_1 \ldots b_n$. For instance, if $x = 101$ and $y = 00$, then $xy = 10100$. For any string α the following holds: $\varepsilon\alpha = \alpha\varepsilon = \alpha$. Hence, ε is the neutral element for concatenation.

Any set of words, a subset of Γ^*, will be called a *language*. If Γ is an alphabet and $L \subseteq \Gamma^*$, then we say that L is a language over Γ. For instance, the set $L \subseteq A^*$ such that $L = \{\alpha \mid \text{the number of occurrences of } b \text{ in } \alpha \text{ is even}\}$ is a language over the alphabet A.

Finite Automata

A finite state automaton is a model of computation consisting of a finite number of states and transitions between those states. We give a formal definition below.

2.3.1. DEFINITION. A *non-deterministic finite automaton* (FA) is a tuple (A, Q, q_s, F, δ), where:

- A is an input alphabet;
- Q is a finite set of states;
- $q_s \in Q$ is an initial state;
- $F \subseteq Q$ is a set of accepting states;
- $\delta : Q \times A \longrightarrow \mathcal{P}(Q)$ is a transition function.

■

If $H = (A, Q, q_s, \delta, F)$ is a FA such that for every $a \in A$ and $q \in Q$ we have $\mathrm{card}(\delta(q, a)) \leq 1$, then H is a *deterministic* automaton. In that case we can describe a transition function as a partial function: $\delta : Q \times A \longrightarrow Q$.

Finite automata are often presented as graphs, where vertices (circles) symbolize internal states, the initial state is marked by an arrow, an accepting state is double circled, and arrows between nodes describe a transition function on letters given by the labels of these arrows. We will give a few examples in what follows.

2.3.2. Definition. Let us first define the *generalized transition function* $\bar{\delta}$ which describes the behavior of an automata reading a string w from the initial state q:

$$\bar{\delta} : Q \times A^* \longrightarrow \mathcal{P}(Q), \text{ where:}$$

$$\bar{\delta}(q, \varepsilon) = \{q\}$$

$$\text{and for each } w \in A^* \text{and } a \in A,\ \bar{\delta}(q, wa) = \bigcup_{q' \in \bar{\delta}(q,w)} \delta(q', a).$$

■

2.3.3. Definition. *The language accepted (recognized) by some FA H* is the set of all words over the alphabet A which are accepted by H, that is:

$$L(H) = \{w \in A^* : \bar{\delta}(q_s, w) \cap F \neq \emptyset\}.$$

■

2.3.4. Definition. We say that a *language $L \subseteq A^*$ is regular* if and only if there exists some FA H such that $L = L(H)$.

■

The following equivalence is a well known fact.

2.3.5. Theorem. *Deterministic and non-deterministic finite automata recognize the same class of languages, i.e. regular languages.*

Proof First of all notice that every deterministic FA is a non-deterministic FA. Then we have to only show that every non-deterministic FA can be simulated by some deterministic FA. The proof goes through the so-called subset construction. It involves constructing all subsets of the set of states of the non-deterministic FA and using them as states of a new, deterministic FA. The new transition function is defined naturally (see Hopcroft et al., 2000, for details). Notice that in the worst case the new deterministic automaton can have 2^n states, where n is the number of states of the corresponding non-deterministic automaton.

□

2.3.6. Example. Let us give a few simple examples of regular languages together with the corresponding accepting automata.

Let $A = \{a, b\}$ and consider the language $L_1 = A^*$. $L_1 = L(H_1)$, where $H_1 = (Q_1, q_1, F_1, \delta_1)$ such that: $Q_1 = \{q_1\}$, $F_1 = \{q_1\}$, $\delta_1(q_1, a) = q_1$ and $\delta_1(q_1, b) = q_1$. The automaton is shown in Figure 2.1.

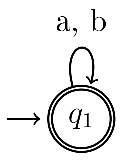


Figure 2.1: Finite automaton recognizing language $L_1 = A^*$.

Now let $L_2 = \emptyset$; then $L_2 = L(H_2)$, where $H_2 = (Q_2, q_2, F_2, \delta_2)$ such that: $Q_2 = \{q_2\}$, $F_2 = \emptyset$, $\delta_2(q_2, a) = q_2$ and $\delta_2(q_2, b) = q_2$. The automaton is depicted in Figure 2.2

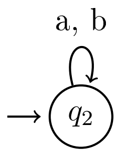


Figure 2.2: Finite automaton recognizing language $L_2 = \emptyset$.

Finally, let $L_3 = \{\varepsilon\}$. $L_3 = L(H_3)$, where $H_3 = (Q_3, q_0, F_3, \delta_3)$ such that: $Q_3 = \{q_0, q_1\}$, $F_3 = \{q_0\}$, $\delta_3(q_0, i) = q_1$ and $\delta_3(q_1, i) = q_1$, for $i = a, b$.

The finite automaton accepting this language is presented in Figure 2.3.

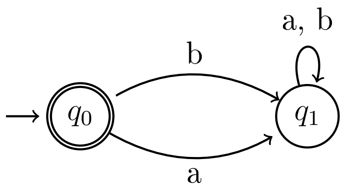


Figure 2.3: Finite automaton recognizing language $L_3 = \{\varepsilon\}$.

Beyond Finite Automata

It is well-known that not every formal language is regular, i.e., recognized by a finite automata. For example, the language $L_{ab} = \{a^n b^n : n \geq 1\}$ cannot be recognized by any finite automaton. Why is that? Strings from that language can be arbitrary long and to recognize them a machine needs to count whether the number of letters "a" is equal to the number of letters "b". A string from L_{ab} can start with any number of letters "a" so the corresponding machine needs to be able to memorize an arbitrarily large natural number. To do this a machine has to be equipped with an unbounded internal memory. However, a finite automaton with k states can remember only numbers smaller than k. This claim is precisely formulated in the following lemma which implies that the language L_{ab} is not regular.

2.3.7. Theorem (Pumping Lemma for Regular Languages). *For any infinite regular language $L \subseteq A^*$ there exists a natural number n such that for every word $\alpha \in L$, if $lh(\alpha) \geq n$, then there are x, y, $z \in A^*$ such that:*

(1) $\alpha = xyz$;

(2) $y \neq \varepsilon$;

(3) $lh(xz) \leq n$;

(4) For every $k \geq 0$ *the string* xy^kz *is in* L.

Push-down Automata

To account for languages which are not regular we need to extend the concept of a finite automaton. A push-down automaton (PDA) is a finite automaton that can make use of a stack (internal memory). The definition follows.

2.3.8. Definition. A non-deterministic push-down automaton (PDA) is a tuple $(A, \Gamma, \#, Q, q_s, F, \delta)$, where:

- A is an input alphabet;
- Γ is a stack alphabet;
- $\# \notin \Gamma$ is a stack initial symbol, empty stack consists only from it;
- Q is a finite set of states;
- $q_s \in Q$ is an initial state;
- $F \subseteq Q$ is a set of accepting states;
- $\delta : Q \times (A \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma^*)$ is a transition function. We denote a single transition by: $(q, a, n) \xrightarrow{H} (p, \gamma)$, if $(p, \gamma) \in \delta(q, a, n)$, where $q, p \in Q, a \in A, n \in \Gamma, \gamma \in \Gamma^*$.

■

If $H = (A, \Gamma, \#, Q, q_s, q_a, \delta)$ is a PDA and for every $a \in A$, $q \in Q$ and $\gamma \in \Gamma$ $\mathrm{card}(\delta(q, a, \gamma)) \leq 1$ and $\delta(q, \varepsilon, \gamma) = \emptyset$, then H is a *deterministic push-down automaton (DPDA).*

The language recognized by a PDA H is the set of strings accepted by H. A string w is accepted by H if and only if starting in the initial state q_0 with the empty stack and reading the string w, the automaton H terminates in an accepting state $p \in F$.

2.3.9. Definition. We say that a language $L \subseteq A^*$ is context-free if and only if there is a PDA H such that $L = L(H)$.

■

Observe that (non-deterministic) PDAs accept a larger class of languages than DPDAs. For instance, the language consisting of palindromes is context-free but cannot be recognized by any DPDA as a machine needs to "guess" which is the middle letter of every string.

2.3.10. Example. Obviously, the class of all context-free languages is larger than the class of all regular languages. For instance, the language $L_{ab} = \{a^n b^n : n \geq 1\}$, which we argued to be non-regular, is context-free. To show this we will construct a PDA H such that $L_{ab} = L(H)$. H recognizes L_{ab} reading every string from left to right and pushes every occurrences of the letter "a" to the top of the stack. After finding the first occurrence of the letter "b" the automaton H pops an "a" off the stack when reading each "b". H accepts a string if after processing all of it the stack is empty.

Formally, let $H = (A, \Gamma, \#, Q, q_s, F, \delta)$, where $A = \{a, b\} = \Gamma$, $Q = \{q_s, q_1, q_2, q_a\}$, $F = \{q_a\}$ and the transition function is specified in the following way:

- $(q_s, a, \#) \xrightarrow{H} (q_s, \#a)$;
- $(q_s, a, a) \xrightarrow{H} (q_s, aa)$;
- $(q_s, b, a) \xrightarrow{H} (q_1, \varepsilon)$;
- $(q_s, b, \#) \xrightarrow{H} (q_2, \varepsilon)$;
- $(q_1, \varepsilon, \#) \xrightarrow{H} (q_a, \varepsilon)$;
- $(q_1, b, a) \xrightarrow{H} (q_1, \varepsilon)$;
- $(q_1, b, b) \xrightarrow{H} (q_2, \varepsilon)$;
- $(q_1, a, b) \xrightarrow{H} (q_2, \varepsilon)$;

- $(q_1, b, \#) \xrightarrow{H} (q_2, \varepsilon)$;
- $(q_1, a, \#) \xrightarrow{H} (q_2, \varepsilon)$.

Context-free languages also have restricted description power. For example, the language $L_{abc} = \{a^k b^k c^k : k \geq 1\}$ is not context-free. This fact follows from the extended version of the pumping lemma.

2.3.11. Theorem (Pumping Lemma for Context-free Languages). *For every context-free language $L \subseteq A^*$ there is a natural number k such that for each $w \in L$, if $lh(w) \geq k$, then there are $\beta_1, \beta_2, \gamma_1, \gamma_2, \eta$ such that:*

- *$\gamma_1 \neq \varepsilon$ or $\gamma_2 \neq \varepsilon$;*
- *$w = \beta_1 \gamma_1 \eta \gamma_2 \beta_2$;*
- *for every $m \in \omega$: $\beta_1 \gamma_1^m \eta \gamma_2^m \beta_2 \in L$.*

Extending push-down automata with more memory (e.g., one additional stack) we reach the realm of Turing machines.

2.3.2 Turing Machines

The basic device of computation in this thesis is a multi-tape Turing (1936) machine. Most of the particulars of Turing machines are not of direct interest to us. Nevertheless, we recall the basic idea. A *multi-tape Turing machine* consists of a read-only *input tape*, a read and write *working tape*, and a write-only *output tape*. Every tape is divided into cells scanned by the *read-write head* of the machine. Each cell contains a symbol from some finite alphabet. The tapes are assumed to be arbitrarily extendable to the right. At any time the machine is in one of a finite number of *states*. The actions of a Turing machine are determined by a finite *programme* which determines, according to the current *configuration* (i.e., the state of the machine and the symbols in the cells being scanned) which action should be executed next. A *computation* of a Turing machine consists thus of a series of successive configurations. A Turing machine is *deterministic* if its state transitions are uniquely defined, otherwise it is *non-deterministic*. Therefore, a deterministic Turing machine has a single computation path (for any particular input) and a non-deterministic Turing machine has a computation tree. A Turing machine *accepts an input* if its computation on that inputs halts after finite time in an accepting state. It *rejects an input* if it halts in a rejecting state.

2.3.12. Definition. Let Γ be some finite *alphabet* and $L \subseteq \Gamma^*$ a language. We say that a *deterministic Turing machine, M, decides* L if for every $x \in \Gamma^*$ M halts in the accepting state on x whenever $x \in L$ and in the rejecting state, otherwise.

A *non-deterministic Turing machine, M, recognizes L* if for every $x \in L$ there is a computation of M which halts in the accepting state and there is no such computation for any $x \notin L$.

■

It is important to notice that non-deterministic Turing machines recognize the same class of languages as deterministic ones. This means that for every problem which can be recognized by a non-deterministic Turing machine there exists a deterministic Turing machine deciding it.

2.3.13. Theorem. *If there is a non-deterministic Turing machine N recognizing a language L, then there exists a deterministic Turing machine M for language L.*

Proof The basic idea for simulating N is as follows. Machine M considers all computation paths of N and simulates N on each of them. If N would halt on a given computation path in an accepting state then M also accepts. Otherwise, M moves to consider the next computation path of N. M rejects the input if machine N would not halt in an accepting state at any computation path.

□

However, the length of an accepting computation of the deterministic Turing machine is, in general, exponential in the length of the shortest accepting computation of the non-deterministic Turing machines as a deterministic machine has to simulate all possible computation paths of a non-deterministic machine.[3] The question whether this simulation can be done without exponential growth in computation time leads us to computational complexity theory.

2.3.3 Complexity Classes

Let us start our complexity considerations with the notation used for comparing the growth rates of functions.

2.3.14. Definition. Let $f, g : \omega \longrightarrow \omega$ be any functions. We say that $f = O(g)$ if there exists a constant $c > 0$ such that $f(n) \leq cg(n)$ for almost all (i.e., all but finitely many) n. ■

Let $f : \omega \longrightarrow \omega$ be a natural number function. TIME(f) is the class of languages (problems) which can be recognized by a deterministic Turing machine in time bounded by f with respect to the length of the input. In other words,

[3]In general, the simulation outlined above leads to a deterministic Turing machine working in time $O(c^{f(n)})$, where $f(n)$ is the time used by a non-deterministic Turing machine solving the problem and $c > 1$ is a constant depending on that machine (see e.g. Papadimitriou, 1993, page 49 for details).

$L \in$ TIME(f) if there exists a deterministic Turing machine such that for every $x \in L$, the computation path of M on x is shorter than $f(n)$, where n is the length of x. TIME(f) is called a *deterministic computational complexity class.* A *non-deterministic complexity class*, NTIME(f), is the class of languages L for which there exists a non-deterministic Turing machine M such that for every $x \in L$ all branches in the computation tree of M on x are bounded by $f(n)$ and moreover M decides L. One way of thinking about a non-deterministic Turing machine bounded by f is that it first guesses the right answer and then deterministically in a time bounded by f checks if the guess is correct.

SPACE(f) is the class of languages which can be recognized by a deterministic machine using at most $f(n)$ cells of the working-tape. NSPACE(f) is defined analogously.

Below we define the most important and well-known complexity classes, i.e., the sets of languages of related complexity. In other words, we can say that a complexity class is the set of problems that can be solved by a Turing machine using $O(f(n))$ of time or space resource, where n is the size of the input. To estimate these resources mathematically natural functions have been chosen, like logarithmic, polynomial and exponential functions. It is well known that polynomial functions grow faster than any logarithmic functions and exponential functions dominate polynomial functions. Therefore, it is commonly believed that problems belonging to logarithmic classes need essentially less resources to be solved than problems from the polynomial classes and likewise that polynomial problems are easier than exponential problems.

2.3.15. Definition.

- LOGSPACE $= \bigcup_{k \in \omega}$ SPACE$(k \log n)$
- NLOGSPACE $= \bigcup_{k \in \omega}$ NSPACE$(k \log n)$
- PTIME $= \bigcup_{k \in \omega}$ TIME(n^k)
- NP $= \bigcup_{k \in \omega}$ NTIME(n^k)
- PSPACE $= \bigcup_{k \in \omega}$ SPACE(n^k)
- NPSPACE $= \bigcup_{k \in \omega}$ NSPACE(n^k)
- EXPTIME $= \bigcup_{k \in \omega}$ TIME(k^n)
- NEXPTIME $= \bigcup_{k \in \omega}$ NTIME(k^n)

■

If $L \in$ NP, then we say that *L is decidable (computable, solvable) in nondeterministic polynomial time* and likewise for other complexity classes.

It is obvious that for any pair of the complexity classes presented above, the lower one includes the upper one. However, when it comes to the strictness of these inclusions not much is known. One instance that has been proven is for LOGSPACE and PSPACE (see e.g. Papadimitriou, 1993, for so-called Hierarchy Theorems).

The complexity class of all regular languages, i.e., languages recognized by finite automata, is sometimes referred to as REG and equals SPACE(O(1)), the decision problems that can be solved in constant space (the space used is independent of the input size). The complexity class of all languages recognized by push-down automata (i.e. context-free languages) is contained in LOGSPACE.

The question whether PTIME is strictly contained in NP is the famous Millennium Problem — one of the most fundamental problems in theoretical computer science, and in mathematics in general. The importance of this problem reaches well outside the theoretical sciences as the problems in NP are usually taken to be *intractable* or *not efficiently computable* as opposed to the problems in P which are conceived of as *efficiently solvable*. In the thesis we take this distinction for granted and investigate semantic constructions in natural language from that perspective (see Chapter 1 for a discussion of this claim).

Moreover, it was shown by Walter Savitch (1970) that if a nondeterministic Turing machine can solve a problem using $f(n)$ space, an ordinary deterministic Turing machine can solve the same problem in the square of the space. Although it seems that nondeterminism may produce exponential gains in time, this theorem shows that it has a markedly more limited effect on space requirements.

2.3.16. Theorem (Savitch (1970)). *For any function $f(n) \geq \log(n)$:*

$$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{SPACE}(f(n)^2).$$

2.3.17. Corollary. PSPACE = NPSPACE

2.3.18. Definition. For any computation class $\mathcal{C}$ we will denote by co-$\mathcal{C}$ the class of complements of languages in $\mathcal{C}$. ■

Every deterministic complexity class coincides with its complement. It is enough to change accepting states into rejecting states to get a machine computing the complement L from a deterministic machine deciding L itself. However, it is unknown whether NP = co-NP. This is very important questions, as P = NP would imply that NP = co-NP.

2.3.4 Oracle Machines

An *oracle machine* can be described as a Turing machine with a black box, called an oracle, which is able to decide certain decision problems in a single step. More precisely, an oracle machine has a separate write-only oracle tape for writing down queries for the oracle. In a single step, the oracle computes the query, erases its input, and writes its output to the tape.

2.3.19. Definition. If $\mathcal{B}$ and $\mathcal{C}$ are complexity classes, then $\mathcal{B}$ *relativized to* $\mathcal{C}$, $\mathcal{B}^{\mathcal{C}}$, is the class of languages recognized by oracle machines which obey the bounds defining $\mathcal{B}$ and use an oracle for problems belonging to $\mathcal{C}$. ■

2.3.5 The Polynomial Hierarchy

The Polynomial Hierarchy, PH, is a very well-known hierarchy of classes above NP. It is usually defined inductively using oracle machines and relativization (see e.g. Papadimitriou, 1993) as below.

2.3.20. Definition.

(1) $\Sigma_1^P = \mathrm{NP}$;

(2) $\Sigma_{n+1}^P = \mathrm{NP}^{\Sigma_n^P}$;

(3) $\Pi_n^P = \text{co-}\Sigma_n^P$;

(4) $\mathrm{PH} = \bigcup_{i \geq 1} \Sigma_i^P$.

■

It is known that $\mathrm{PH} \subseteq \mathrm{PSPACE}$ (see e.g. Papadimitriou, 1993).

2.3.6 The Counting Hierarchy

The polynomial hierarchy defined above is an oracle hierarchy with NP as the building block. If we replace NP by probabilistic polynomial time, PP, in the definition of PH, then we arrive at a class called the counting hierarchy, CH. The class PP consists of languages L for which there is a polynomial time bounded nondeterministic Turing machine M such that, for all inputs x, $x \in L$ iff more than half of the computations of M on input x end up accepting. In other words, a language L belongs to the class PP iff it is accepted with a probability more than one-half by some nondeterministic Turing machine in polynomial time.

The counting hierarchy can be defined now as follows, in terms of oracle Turing machines:

2.3.21. Definition.

(1) $C_0P = \mathrm{PTIME}$;

(2) $C_{\mathrm{k}+1}P = \mathrm{PP}^{C_{\mathrm{k}}P}$;

(3) $\mathrm{CH} = \bigcup_{k \in \mathbb{N}} C_k P$.

■

It is known that the polynomial hierarchy, PH, is contained in the second level C_2P of the counting hierarchy CH (see Toda, 1991). The question whether $\mathrm{CH} \subseteq \mathrm{PH}$ is still open. However, it is widely believed that this is not the case. Under this assumption we will deliver in Chapter 5 an argument for restrictions on the type-shifting strategy in modeling the meanings of collective determiners in natural language.

2.3.7 Reductions and Complete Problems

The intuition that some problems are more difficult than others is formalized in complexity theory by the notion of a *reduction*. We will use only polynomial time many-one (Karp (1972)) reductions.

2.3.22. DEFINITION. We say that a function $f : A \longrightarrow A$ is a *polynomial time computable function* iff there exits a deterministic Turing machine computing $f(w)$ for every $w \in A$ in polynomial time. ■

2.3.23. DEFINITION. A *problem* $L \subseteq \Gamma^*$ *is polynomial reducible to a problem* $L' \subseteq \Gamma^*$ if there is a polynomial time computable function $f : \Gamma^* \longrightarrow \Gamma^*$ from strings to strings, such that

$$w \in L \iff f(w) \in L'.$$

We will call such function f a *polynomial time reduction* of L to L'. ■

2.3.24. DEFINITION. A language L is complete for a complexity class $\mathcal{C}$ if $L \in \mathcal{C}$ and every language in $\mathcal{C}$ is reducible to L. ■

Intuitively, if L is complete for a complexity class $\mathcal{C}$ then it is among the hardest problems in this class. The theory of complete problems was initiated with a seminal result of Cook (1971), who proved that the satisfiability problem for propositional formulae, SAT , is complete for NP. Many other now famous problems were then proven NP-complete by Karp (1972) — including some versions of satisfiability, like 3SAT (the restriction of SAT to formulae in conjunctional normal form such that every clause contains 3 literals), as well as some graph problems, e.g. CLIQUE, which we define below. The book of Garey and Johnson (1979) contains a list of NP-complete languages.

2.3.25. EXAMPLE. Let us give an example of a polynomial reduction. We will prove that the problem CLIQUE is NP-complete by reducing 3SAT to it. We will define other versions of the CLIQUE problem and use them to prove some complexity results for quantifiers in Chapter 3.

2.3.26. Definition. Let $G = (V, E)$ be a graph and take a set $Cl \subseteq V$. We say that Cl is a *clique* if there is $(i, j) \in E$ for every $i, j \in Cl$. ■

2.3.27. Definition. The *problem* CLIQUE can be formulated now as follows. Given a graph $G = (V, E)$ and a natural number k, determine whether there is a clique in G of cardinality at least k. ■

2.3.28. Theorem. CLIQUE *is NP-complete.*

Proof First we have to show that CLIQUE belongs to NP. Once we have located k or more vertices which form a clique, it is trivial to verify that they do, this is why the clique problem is in NP.

To show NP-hardness we will reduce 3SAT to CLIQUE. Assume that our input is a set of clauses in the form of 3SAT:
$Z = \{(\ell_1^1 \vee \ell_2^1 \vee \ell_3^1), \ldots, (\ell_1^m \vee \ell_2^m \vee \ell_3^m)\}$, where ℓ_i^j is a literal. We construct (G, k) such that:

- $k = m$;
- $G = (V, E)$, where:
  - $V = \{v_{ij} \mid i = 1, \ldots, m; j = 1, 2, 3\}$;
  - $E = \{(v_{ij}, v_{\ell k}) \mid i \neq \ell; \ell_i^j \neq \neg\ell_\ell^k\}$.

To complete the proof it suffices to observe that in graph G there is a clique of cardinality k if and only if the set Z is satisfiable (see e.g. Papadimitriou, 1993). □

2.4 Descriptive Complexity Theory

Classical descriptive complexity deals with the relationship between logical definability and computational complexity. The main idea is to treat classes of finite models over a fixed vocabulary as computational problems. In such a setting rather than the computational complexity of a given class of models we are dealing with its *descriptive complexity*, i.e., the question how difficult it is to describe the class using some logic. This section very briefly explains the fundamentals of descriptive complexity as a subfield of finite model theory. More details may be found in the books of Immerman (1998), Ebbinghaus and Flum (2005), Libkin (2004), and Grädel et al. (2007).

2.4.1 Encoding Finite Models

$\mathbb{M}$ *is a finite model* when its universe, M, is finite. A widely studied class of finite models are *graphs*, i.e., structures of the form $\mathbb{G} = (V, E)$, where the finite universe V is called the set of *vertices (nodes)* of the graph and a binary relation $E \subseteq V^2$ is the set of *edges* of the graph.

Notice that in a linguistic context it makes sense to restrict ourselves to finite models as in a typical communication situation we refer to relatively small finite sets of objects. We have discussed this assumption further in Section 1.6.

Let $\mathcal{K}$ be a class of finite models over some fixed vocabulary τ. We want to treat $\mathcal{K}$ as a problem (language) over the vocabulary τ. To do this we need to code τ-models as finite strings. We can assume that the universe of a model $\mathbb{M} \in \mathcal{K}$ consists of natural numbers: $M = \{1, \ldots, n\}$. A natural way of encoding a model $\mathbb{M}$ (up to isomorphism) is by listing its universe, M, and storing the interpretation of the symbols in τ by writing down their truth-values on all tuples of objects from M.

2.4.1. Definition. Let $\tau = \{R_1, \ldots, R_k\}$ be a relational vocabulary and $\mathbb{M}$ a τ-model of the following form: $\mathbb{M} = (M, R_1, \ldots, R_k)$, where $M = \{1, \ldots, n\}$ is the universe of model $\mathbb{M}$ and $R_i \subseteq M$ is an n_i-ary relation over M, for $1 \leq i \leq k$. We define a *binary encoding for τ-models*. The code for $\mathbb{M}$ is a word over $\{0, 1, \#\}$ of length $O((\text{card}(M))^c)$, where c is the maximal arity of the predicates in τ (or $c = 1$ if there are no predicates).

The code has the followning form:

$$\tilde{n}\#\tilde{R}_1\#\ldots\#\tilde{R}_n, \text{ where:}$$

- $\tilde{n}$ is the part coding the universe of the model and consists of n 1s.
- $\tilde{R}_i$ — the code for the n_i-ary relation R_i — is an n^{n_i}-bit string whose j-th bit is 1 iff the j-th tuple in M^{n_i} (ordered lexicographically) is in R_i.
- $\#$ is a separating symbol.[4]

■

2.4.2. Example. Let us give an example of a binary code corresponding to a model. Consider vocabulary $\sigma = \{P, R\}$, where P is a unary predicate and R a binary relation. Take the σ-model $\mathbb{M} = (M, P^M, R^M)$, where the universe $M = \{1, 2, 3\}$, the unary relation $P^M \subseteq M$ is equal to $\{2\}$ and the binary relation $R^M \subseteq M^2$ consists of the pairs $(2, 2)$ and $(3, 2)$. Notice, that we can think about such models as graphs in which some nodes are "colored" by P.

Let us construct the code step by step:

[4] See also Definition 2.1 of (Immerman, 1998) for a binary coding without separating symbol.

- $\tilde{n}$ consists of three 1s as there are three elements in M.
- $\tilde{P^M}$ is the string of length three with 1s in places corresponding to the elements from M belonging to P^M. Hence $\tilde{P^M} = 010$ as $P^M = \{2\}$.
- $\tilde{R^M}$ is obtained by writing down all $3^2 = 9$ binary strings of elements from M in lexicographical order and substituting 1 in places corresponding to the pairs belonging to R^M and 0 in all other places. As a result $\tilde{R^M} = 000010010$.

Adding all together the code for $\mathbb{M}$ is 111#010#000010010.

2.4.2 Logics Capturing Complexity Classes

Now we can formulate the central definition of descriptive complexity theory.

2.4.3. DEFINITION. Let $\mathcal{L}$ be a logic and $\mathcal{C}$ a complexity class. We say that $\mathcal{L}$ *captures* $\mathcal{C}$, if for any vocabulary τ and any class of τ-models the following holds:

$$\mathcal{K} \text{ is in } \mathcal{C} \text{ if and only if } \mathcal{K} \text{ is definable by an } \mathcal{L}\text{-sentence.}$$

■

The following are two classical results of descriptive complexity theory:

2.4.4. THEOREM (FAGIN (1974)). *Σ^1_1 captures* NP.

2.4.5. THEOREM (STOCKMEYER (1976)). *For any m, Σ^1_m captures Σ^P_m.*

Fagin's theorem establishes a correspondence between existential second order logic and NP. Stockmeyer's extends it for the hierarchy of second–order formulae and the polynomial hierarchy. There are many other logical characterizations of complexity classes known (see e.g. Immerman, 1998), for instance that first-order logic is contained in LOGSPACE (see Immerman, 1998, Theorem 3.1). One of the famous results is the characterization of PTIME over ordered graph structures in terms of fixed-point logic, due to Immerman (1982) and Vardi (1982). Namely, in the presence of a linear ordering of the universe it is possible to use tuples of nodes to build a model of a Turing machine inside the graph and imitate the polynomial time property by a suitable fixed point sentence (e.g. see Immerman, 1998). One of the most important open problems is the question what logic $\mathcal{L}$ captures PTIME on graphs if we do not have an ordering of the vertices. Knowing $\mathcal{L}$ one could try to show that $\mathcal{L} \neq \Sigma^1_1$, from which it would follow P $\neq$ NP.

Kontinen and Niemistö (2006) showed that the extension of first-order logic by second-order majority quantifiers of all arities (see Section 5.3 for a definition) describes exactly problems in the counting hierarchy. We will investigate the linguistic consequences of that claim in Chapter 5.

Let us now define one of the crucial concepts of descriptive complexity which we use throughout the thesis.

2.4.6. Definition. If every $\mathcal{L}$-definable class $\mathcal{K}$ is in $\mathcal{C}$ then we say that *model checking (data complexity)* for $\mathcal{L}$ is in $\mathcal{C}$. ■

2.4.7. Remark. In the computational literature many other complexity measures besides model-checking are considered, most notably the so-called *expression complexity* and *combined complexity* introduced by Vardi (1982). The main difference between them and model-checking is as follows. In the latter our input is a model and we measure complexity with respect to the size of its universe. For expression complexity a formula from some set is fixed as an input and we measure its complexity — given as a function of the length of the expression — in different models. Expression complexity is sometimes referred to as a measure for succinctness of a language. There is a great difference between those two measures, for example Σ_1^1 expression complexity is NEXPTIME, but its model-checking is NP-complete (see e.g. Gottlob et al., 1999, for a systematic comparison). The third possibility is given by combined complexity: both a formula and a structure are given as an input and the complexity is defined with respect to their combined size. In this thesis we investigate model-checking complexity for quantifiers.

2.5 Quantifiers in Finite Models

Below we review some recent work in the field of generalized quantifiers on finite models. For more detailed discussion of this subject we refer to the survey of Makowsky and Pnueli (1995) and the paper of Väänänen (1997a).

Recall Definition 2.2.1, which says that generalized quantifiers are simply classes of models. Finite models can be encoded as finite strings over some vocabulary (recall Definition 2.4.1). Therefore, generalized quantifiers can be treated as classes of such finite strings, i.e., languages. Now we can easily fit the notions into the descriptive complexity paradigm.

2.5.1. Definition. By the *complexity of a quantifier* Q we mean the computational complexity of the corresponding class of finite models. ■

2.5.2. Example. Consider a quantifier of type (1, 2): a class of finite colored graphs of the form $\mathbb{M} = (M, A^M, R^M)$. Let us take a model of this form, $\mathbb{M}$, and a quantifier Q. Our computational problem is to decide whether $\mathbb{M} \in \mathsf{Q}$; or equivalently, to solve the query whether $\mathbb{M} \models \mathsf{Q}[A, R]$.

This can simply be viewed as the model-checking problem for quantifiers. These notions can easily be generalized to quantifiers of arbitrary types $(n_1, \ldots, n_k)$ by considering classes of models of the form $\mathbb{M} = (M, R_1, \ldots, R_k)$, where $R_i \subseteq M^{n_i}$, for $i = 1, \ldots, k$.

Generalized quantifiers in finite models were considered from the point of view of computational complexity for the first time by Blass and Gurevich (1986). They investigated various forms of branching (Henkin) quantifiers (see Section 2.2.2) defining NP or NLOGSPACE complete graph problems. They introduced the following terminology.

2.5.3. Definition. We say that *quantifier* Q *is NP-hard* if the corresponding class of finite models is NP-hard. Q *is mighty* (NP-complete) if the corresponding class belongs to NP and is NP-hard. ■

In the previous chapter we mentioned that one of the fundamental problems of descriptive complexity theory is to find a logic which expresses exactly the polynomial time queries on unordered structures. Studying the computational complexity of quantifiers can contribute to this question. For instance, Hella et al. (1996) have proven that there is a representation of PTIME queries in terms of fixed-point logic enriched by the quantifier Even, which holds on a randomly chosen finite structure with a probability approaching one as the size of the structure increases. However, Hella (1996) has shown that on unordered finite models, PTIME is not the extension of fixed-point logic by finitely many generalized quantifiers.

Recently there has been some interest in studying the computational complexity of generalized quantifiers in natural language. For example, Mostowski and Wojtyniak (2004) have observed that some natural language sentences, like (4), when assuming their branching interpretation, are mighty.

(4) Some relative of each villager and some relative of each townsman hate each other.

Sevenster (2006) has extended this results and proved that proportional branching quantifiers, like those possibly occurring in sentence (5), are also NP-complete.

(5) Most villagers and most townsmen hate each other.

We will overview these results in Chapter 3.

In the thesis we pursue the subject of the computational complexity of natural language quantifier constructions further. We prove some new results and study the role of descriptive computational complexity in natural language semantics.

Chapter 3

Complexity of Polyadic Quantifiers

In this chapter we focus on the computational complexity of some polyadic quantifiers. In what follows we offer rather mathematical considerations, which will be applied to linguistics in the following parts of the dissertation. Particularly, Chapter 4 on quantified reciprocal sentences and Chapter 6 on combinations of quantifiers in natural language make use of the facts discussed below.

Firstly, we will study iteration, cumulation and resumption — lifts turning monadic quantifiers into polyadic ones. These lifts are widely used in linguistics to model the meanings of complex noun phrases. We observe that they do not increase computational complexity when applied to simple determiners. More precisely, PTIME quantifiers are closed under application of these lifts. Most of the natural language determiners correspond to monadic quantifiers computable in polynomial time. Thus this observation suggests that typically polyadic quantifiers in natural language are tractable.

Next, we move to a short discussion of the branching operation. This polyadic lift can produce intractable semantic constructions from simple determiners. In particular, when applied to proportional determiners it gives NP-complete polyadic quantifiers.

There has been a lot of discussion between linguists and philosophers whether certain natural language sentences combining a few quantifiers can in fact be interpreted as branching sentences. We will come back to this issue in detail in Chapter 6, which is devoted to Hintikka's Thesis. Now it is enough to say that this claim is controversial and such sentences are at least ambiguous between a branching reading and other interpretations.

Therefore, in the last section of this chapter — motivated by the search for non-controversial NP-complete semantic constructions in natural language — we investigate the so-called Ramsey quantifiers. We outline some links between them and branching quantifiers. Then we prove that some Ramsey quantifiers, e.g. the proportional, define NP-complete classes of finite models. Moreover, we observe that so-called bounded Ramsey quantifiers are PTIME computable.

After this we make the claim that Ramsey quantifiers have a natural application in linguistics. They are the interpretations of natural language expressions such as “each other” and “one another”. We discuss the details of this approach in the following chapter, which is devoted to reciprocal expressions in English.

Some of the results presented in this chapter were published in the Proceedings of the Amsterdam Colloquium (see Szymanik, 2007b) and Lecture Notes in Computer Science (Szymanik, 2008).

3.1 Standard Polyadic Lifts

Monadic generalized quantifiers provide the most straightforward way to give the semantics for noun phrases in natural language. For example, consider the following sentence:

(1) Some logicians smoke.

It consists of a noun phrase “Some logicians” followed by the intransitive verb “smoke”. The noun phrase is built from the determiner “Some” and the noun “logicians”. In a given model the noun and the verb denote subsets of the universe. Hence, the determiner stands for a quantifier denoting a binary relation between the subsets. In other words, with varying universes, the determiner “some” is a type $(1, 1)$ generalized quantifier.

Most research in generalized quantifier theory has been directed towards monadic quantification in natural language. The recent monograph on the subject by Peters and Westerståhl (2006) bears witness to this tendency, leaving more than 90% of its volume to discussion of monadic quantifiers. Some researchers, e.g., Landman (2000), claim even that polyadic generalized quantifiers do not occur in natural language, at all. However, it is indisputable that sentences can combine several noun phrases with verbs denoting not only sets but also binary or ternary relations. In such cases the meanings can be given by *polyadic quantifiers.*

This perspective on quantifiers is captured by the definition of generalized quantifiers, Definition 2.2.4 from the Mathematical Prerequisites chapter. Recall that we say that a *generalized quantifier* Q of type $t = (n_1, \ldots, n_k)$ is a functor assigning to every set M a k-ary relation $\mathsf{Q_M}$ between relations on M such that if $(R_1, \ldots, R_k) \in \mathsf{Q_M}$ then R_i is an n_i-ary relation on M, for $i = 1, \ldots, k$. Additionally, Q is preserved by bijections. If for all i the relation R_i is unary, i.e. it denotes a subset of the universe, then we say that the quantifier is *monadic*. Otherwise, it is *polyadic*.

One way to deal with polyadic quantification in natural language is to define it in terms of monadic quantifiers using Boolean combinations (see Section 2.2.5) and so-called *polyadic lifts*. Below we introduce some well-known lifts: iteration, cumulation, resumption and branching (see e.g. van Benthem, 1989). We observe that the first three do not increase the computational complexity of quantifiers, as opposed to branching which does.

3.1.1 Iteration

The Fregean nesting of first-order quantifiers, e.g., $\forall\exists$, can be applied to any generalized quantifier by means of iteration.

3.1.1. Example. Iteration may be used to express the meaning of the following sentence in terms of its constituents.

(2) Most logicians criticized some papers.

The sentence is true (under one interpretation) iff there is a set containing most logicians such that every logician from that set criticized at least one paper, or equivalently:

$$\mathsf{It}(\mathsf{Most}, \mathsf{Some})[\text{Logicians, Papers, Criticized}].$$

However, similar sentences sometimes correspond to lifts other than iteration. We will introduce another possibility in Section 3.1.2. But first we define iteration precisely.

3.1.2. Definition. Let Q and Q' be generalized quantifiers of type (1, 1). Let A, B be subsets of the universe and R a binary relation over the universe. Suppressing the universe, we will define the *iteration* operator as follows:

$$\mathsf{It}(\mathsf{Q}, \mathsf{Q}')[A, B, R] \iff \mathsf{Q}[A, \{a \mid \mathsf{Q}'(B, R_{(a)})\}],$$

where $R_{(a)} = \{b \mid R(a, b)\}$. ■

Therefore, the iteration operator produces polyadic quantifiers of type (1, 1, 2) from two monadic quantifiers of type (1, 1). The definition can be extended to cover iteration of monadic quantifiers with an arbitrary number of arguments (see e.g. Peters and Westerståhl, 2006, page 347).

Notice that the iteration operator is not symmetric, i.e., it is not the case that for any two quantifiers Q and Q' we have $\mathsf{It}(\mathsf{Q}, \mathsf{Q}')[A, B, R] \iff \mathsf{It}(\mathsf{Q}', \mathsf{Q})[B, A, R^{-1}]$. (For example, consider the unary quantifiers $\mathsf{Q} = \forall$ and $\mathsf{Q}' = \exists$.) The interesting open problem is to find a complete characterization of those quantifiers which are order independent or, in other words, for which the equivalence is true. Partial solutions to this problem are discussed in (Peters and Westerståhl, 2006, pages 348–350).

The observation that quantifiers are order dependent will play a crucial role when we discuss possible readings of determiner combinations and scope dominance between them in Chapter 6.

3.1.2 Cumulation

Consider the following sentence:

(3) Eighty professors taught sixty courses at ESSLLI'08.

The analysis of this sentence by iteration of the quantifiers "eighty" and "sixty" implies that there were $80 \times 60 = 4800$ courses at ESSLLI. Therefore, obviously this is not the meaning we would like to account for. This sentence presumably means neither that each professor taught 60 courses ($\mathsf{It}(80, 60)$) nor that each course was taught by 80 professors ($\mathsf{It}(60, 80)$). In fact, this sentence is an example of so-called cumulative quantification, saying that each of the professors taught at least one course and each of the courses was taught by at least one professor. Cumulation is easily definable in terms of iteration and the existential quantifier as follows.

3.1.3. Definition. Let Q and Q' be generalized quantifiers of type (1, 1). A, B are subsets of the universe and R is a binary relation over the universe. Suppressing the universe we will define the *cumulation* operator as follows:

$$\mathsf{Cum}(\mathsf{Q}, \mathsf{Q}')[A, B, R] \iff \mathsf{It}(\mathsf{Q}, \mathsf{Some})[A, B, R] \wedge \mathsf{It}(\mathsf{Q}', \mathsf{Some})[B, A, R^{-1}].$$

■

3.1.3 Resumption

The next lift we are about to introduce — resumption (vectorization) — has found many applications in theoretical computer science (see e.g. Makowsky and Pnueli, 1995; Ebbinghaus and Flum, 2005). The idea here is to lift a monadic quantifier in such a way as to allow quantification over tuples. This is linguistically motivated when ordinary natural language quantifiers are applied to pairs of objects rather than individuals. For example, this is useful in certain cases of adverbial quantification (see e.g. Peters and Westerståhl, 2006, Chapter 10.2).

Below we give a formal definition of the *resumption* operator.

3.1.4. Definition. Let Q be any monadic quantifier with n arguments, U a universe, and $R_1, \ldots, R_n \subseteq U^k$ for $k \geq 1$. We define the *resumption* operator as follows:

$$\mathsf{Res}^k(\mathsf{Q})_U[R_1, \ldots, R_n] \iff (\mathsf{Q})_{U^k}[R_1, \ldots, R_n].$$

■

That is, $\mathsf{Res}^k(\mathsf{Q})$ is just Q applied to a universe, U^k, containing k-tuples. In particular, $\mathsf{Res}^1(\mathsf{Q}) = \mathsf{Q}$. Clearly, one can use $\mathsf{Res}^2(\mathsf{Most})$ to express the meaning of sentence (4).

(4) Most twins never separate.

3.1.4 PTIME GQs are Closed under It, Cum, and Res

When studying the computational complexity of quantifiers a natural problem arises in the context of these lifts. Do they increase complexity? Is it the case that together with the growth of the universe the complexity of deciding whether quantifier sentences holds increases dramatically? For example, is it possible that two tractable determiners can be turned into an intractable quantifier?

Sevenster (2006) claims that if quantifiers are definable in the Presburger arithmetic of addition, then the computational complexity of their Boolean combinations, iteration, cumulation, and resumption stay in LOGSPACE.[1] We do not know whether all natural language determiners are definable in the arithmetic of addition.

Therefore, we do not want to restrict ourselves to this class of quantifiers. Hence, we show that PTIME computable quantifiers are closed under Boolean combinations and the three lifts defined above. As in the dissertation we are interested in the strategies people may use to comprehend quantifiers we show a direct construction of the relevant procedures. In other words, we show how to construct a polynomial model-checker for our polyadic quantifiers from PTIME Turing machines computing monadic determiners.

3.1.5. Proposition. *Let* Q *and* Q' *be monadic quantifiers computable in polynomial time with respect to the size of a universe. Then the quantifiers: (1)* $\neg\,\mathsf{Q}$*; (2)* $\mathsf{Q}\,\neg$*; (3)* $\mathsf{Q} \wedge \mathsf{Q}'$*; (4)* $\mathsf{It}(\mathsf{Q}, \mathsf{Q}')$*; (5)* $\mathsf{Cum}(\mathsf{Q}, \mathsf{Q}')$*; (6)* $\mathsf{Res}(\mathsf{Q})$ *are PTIME computable.*

Proof Let us assume that there are Turing machines M and M' computing quantifiers Q and Q', respectively. Moreover M and M' work in polynomial time with respect to any finite universe U.

(1) A Turing machine computing $\neg\,\mathsf{Q}$ is like M. The only difference is that we change accepting states into rejecting states and *vice versa*. In other words, we accept $\neg\,\mathsf{Q}$ whenever M rejects Q and reject whenever M accepts. The working time of a so-defined new Turing machine is exactly the same as the working time of machine M. Hence, the outer negation of PTIME quantifiers can be recognized in polynomial time.

(2) Recall that on a given universe U we have the following equivalence: $(\mathsf{Q}\,\neg)_U[R_1, \ldots, R_k] \iff Q_U[R_1, \ldots, R_{k-1}, U - R_k]$. Therefore, for the inner negation of a quantifier it suffices to compute $U - R_k$ and then use the polynomial Turing machine M on the input $Q_U[R_1, \ldots, R_{k-1}, U - R_k]$.

(3) To compute $\mathsf{Q} \wedge \mathsf{Q}'$ we have to first compute Q using M and then Q' using M'. If both machines halt in an accepting state then we accept. Otherwise,

[1]Moreover, he conjectures that the circuit complexity class ThC_0 (see e.g. Chapter 5.4 Immerman, 1998, for definition) is also closed under taking these operations on quantifiers.

we reject. This procedure is polynomial, because the sum of the polynomial bounds on working time of M and M' is also polynomial.

(4) Recall that $\mathsf{It}(\mathsf{Q},\mathsf{Q}')[A,B,R] \iff \mathsf{Q}[A,A']$, where $A' = \{a|\,\mathsf{Q}'(B,R_{(a)})\}$, for $R_{(a)} = \{b|R(a,b)\}$. Notice that for every a from the universe, $R_{(a)}$ is a monadic predicate. Having this in mind we construct in polynomial time A'. To do this we execute the following procedure for every element from the universe. We initialize $A' = \emptyset$. Then we repeat for each a from the universe the following: Firstly we compute $R_{(a)}$. Then using the polynomial machine M' we compute $\mathsf{Q}'[B,R_{(a)}]$. If the machine accepts, then we add a to A'. Having constructed A' in polynomial time we just use the polynomial machine M to compute $\mathsf{Q}[A,A']$.

(5) Notice that cumulation is defined in terms of iteration and existential quantifier (see Definition 3.1.3). Therefore, this point follows from the previous one.

(6) To compute $\mathsf{Res}^k(\mathsf{Q})$ over the model $\mathbb{M} = \{\{1,\ldots,n\},R_1,\ldots,R_n\}$ for a fixed k, we just use the machine M with the following input $\tilde{n^k}\#\tilde{R_1}\#\ldots\#\tilde{R_n}$ instead of $\tilde{n}\#\tilde{R_1}\#\ldots\#\tilde{R_n}$. Recall Definition 2.4.1.

$\square$

Let us give an informal argument that the above proposition holds for all generalized quantifiers not only for monadic ones. Notice that the Boolean operations as well as iteration and cumulation are definable in first-order logic. Recall that the model-checking problem for first-order sentences is in LOGSPACE $\subseteq$ PTIME (see Section 2.4). Let A be a set of generalized quantifiers of any type from a given complexity class $\mathcal{C}$. Then the complexity of model-checking for sentences from FO(A) is in LOGSPACE$^{\mathcal{C}}$ (deterministic logarithmic space with an oracle from $\mathcal{C}$, see Section 2.3.4). One simply uses a LOGSPACE Turing machine to decide the first-order sentences, evoking the oracle when a quantifier from A appears. Therefore, the complexity of Boolean combinations, iteration and cumulation of PTIME generalized quantifiers has to be in LOGSPACE$^{\text{PTIME}}$ = PTIME.

The case of the resumption operation is slightly more complicated. Resumption is not definable in first-order logic for all generalized quantifiers (see Hella et al., 1997; Luosto, 1999). However, notice that our arguments given in point (6) of the proof do not make use of any assumption about the arity of R_i. Therefore, the same proof works for resumption of polyadic quantifiers. The above considerations allow us to formulate the following theorem which is the generalization of the previous proposition.

3.1.6. THEOREM. *Let Q and Q' be generalized quantifiers computable in polynomial time with respect to the size of a universe. Then the quantifiers: (1)*

$\neg\,\mathsf{Q}$*; (2)* $\mathsf{Q}\,\neg$*; (3)* $\mathsf{Q} \wedge \mathsf{Q}'$*; (4)* $\mathsf{It}(\mathsf{Q}, \mathsf{Q}')$*; (5)* $\mathsf{Cum}(\mathsf{Q}, \mathsf{Q}')$*; (6)* $\mathsf{Res}(\mathsf{Q})$ *are PTIME computable.*

We have argued that PTIME quantifiers are closed under Boolean operations as well as under the polyadic lifts occurring frequently in natural language. In other words, these operations do not increase the complexity of quantifier semantics. As we can safely assume that most of the simple determiners in natural language are PTIME computable then the semantics of the polyadic quantifiers studied above is tractable. This seems to be good news for the theory of natural language processing. Unfortunately, not all natural language lifts behave so nicely from a computational perspective. In the next section we show that branching can produce NP-complete quantifier constructions from simple determiners. Speaking in the terminology of Blass and Gurevich (1986), introduced in Definition 2.5.3, some branching quantifiers are mighty.

3.2 Branching Quantifiers

Branching quantifiers are a very well-known example of polyadic generalized quantifiers. We introduced them in Section 2.2.2 and below we study their computational complexity.

3.2.1 Henkin's Quantifiers are Mighty

The famous linguistic application of branching quantifiers is for the study of sentences like:

(7) Some relative of each villager and some relative of each townsman hate each other.

(8) Some book by every author is referred to in some essay by every critic.

(9) Every writer likes a book of his almost as much as every critic dislikes some book he has reviewed.

According to Jaakko Hintikka (1973), to express the meaning of such sentences we need branching quantifiers. In particular the interpretation of sentence (7) is expressed as follows:

$$(10)\quad \begin{pmatrix} \forall x \exists y \\ \forall z \exists w \end{pmatrix} [(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))],$$

where unary predicates V and T denote the set of villagers and the set of townsmen, respectively. The binary predicate symbol $R(x,y)$ denotes the relation "x and y are relatives" and $H(x,y)$ the relation "x and y hate each other".

The polyadic generalized quantifier Z of type (2, 2), called Hintikka's form, can be used to express the prefix "some relative of each ... and some relative of each ...". A formula $\mathsf{Z}xy\ [\varphi(x,y), \psi(x,y)]$ can be interpreted in a second-order language as:

$$\exists A \exists B[\forall x \exists y(A(y) \wedge \varphi(x,y)) \wedge \forall x \exists y(B(y) \wedge \varphi(x,y)) \\ \wedge \forall x \forall y(A(x) \wedge B(y) \implies \psi(x,y))].$$

We will discuss Hintikka's claim and the role of branching quantifiers in Chapter 6. Now we only state that the problem of recognizing the truth-value of formula (10) in a finite model is NP-complete (Mostowski and Wojtyniak, 2004). In other words:

3.2.1. THEOREM. *The quantifier* Z *is mighty.*

Therefore, branching — as opposed to iteration, cumulation, and resumption — substantially effects computational complexity.

3.2.2 Proportional Branching Quantifiers are Mighty

Not only the universal and existential quantifiers can be branched. The procedure of branching works in a very similar way for other quantifiers. Below we define the branching operation for arbitrary monotone increasing generalized quantifiers.

3.2.2. DEFINITION. Let Q and Q' be both MON↑ quantifiers of type $(1,1)$. Define the *branching* of quantifier symbols Q and Q' as the type (1, 1, 2) quantifier symbol $\mathsf{Br}(\mathsf{Q}, \mathsf{Q}')$. A structure $\mathbb{M} = (M, A, B, R) \in \mathsf{Br}(\mathsf{Q}, \mathsf{Q}')$ if the following holds:

$$\exists X \subseteq A\ \exists Y \subseteq B[(X, A) \in \mathsf{Q} \wedge (Y, B) \in \mathsf{Q}' \wedge X \times Y \subseteq R].$$

■

The branching operation can also be defined for monotone decreasing quantifiers as well as for pairs of non-monotone quantifiers (see e.g. Sher, 1990).

The branching lift can be used to account for some interpretations of proportional sentences like the following:

(11) Most villagers and most townsmen hate each other.

(12) One third of all villagers and half of all townsmen hate each other.

We will discuss these examples in Section 6 of the thesis. Now we will only consider their computational complexity.

It has been shown by Merlijn Sevenster (2006) that the problem of recognizing the truth-value of formula (11) in finite models is NP-complete. Actually, it can also be proven that all proportional branching sentences, like (12), define an NP-complete class of finite models. In other words the following holds.

3.2.3. Theorem. *Let* Q *and* Q' *be proportional quantifiers, then the quantifier* $\mathsf{Br}(\mathsf{Q}, \mathsf{Q}')$ *is mighty.*

By proportional branching sentences (e.g. (12)), we mean the branching interpretations of sentences containing proportional quantifiers, i.e., quantifiers saying that some fraction of a universe has a given property (see also Definition 4.3.3), for example "most", "less than half", and "many" (although only under some interpretations). Therefore, the above result gives another example of a polyadic quantifier construction in natural language which has an intractable reading.

3.2.3 Branching Counting Quantifiers are Mighty

Below we will briefly discuss the complexity of branching readings of sentences, which plays an important role in our empirical studies described in Chapter 6. Consider sentences like:

(13) More than 5 villagers and more than 3 townsmen hate each other.

Their branching readings have the following form:

(14) $\begin{pmatrix} \text{More than k } x : V(x) \\ \text{More than m } y : T(y) \end{pmatrix} H(x,y),$

where k, m are any integers. Notice that for fixed k and m the above sentence is equivalent to the following first-order formula and hence PTIME computable.

$$\exists x_1 \ldots \exists x_{k+1} \exists y_1 \ldots \exists x_{m+1} \Big[\bigwedge_{1 \leq i < j \leq k+1} x_i \neq x_j \ \wedge \bigwedge_{1 \leq i < j \leq m+1} y_i \neq y_j \\ \wedge \bigwedge_{1 \leq i \leq k+1} V(x_i) \ \wedge \bigwedge_{1 \leq j \leq m+1} T(y_j) \ \wedge \bigwedge_{\substack{1 \leq i \leq k+1 \\ 1 \leq j \leq m+1}} H(x_i, y_j) \Big].$$

However, the general schema, for unbounded k and m, defines an NP-complete problem. Let us formulate the idea precisely. We start by defining the counting quantifier $\mathsf{C}^{\geq \mathsf{A}}$ of type (1) which says that the number of elements satisfying a given formula in a model $\mathbb{M}$ is greater than the cardinality of a set $A \subseteq M$. Alternatively we could introduce a two-sorted variant of finite structures, augmented by an infinite number sort. Then we can define counting quantifiers in such a way that the numeric constants in a quantifier refer to the number domain (see e.g. Otto, 1997; Grädel and Gurevich, 1998).

3.2.4. Definition. Let $\mathbb{M} = (M, A, \ldots)$. We define the counting quantifier of type (1) as follows:

$$\mathbb{M} \models \mathsf{C}^{\geq \mathsf{A}} x \ \varphi(x) \iff \mathrm{card}(\varphi^{\mathbb{M},x}) \geq \mathrm{card}(A).$$

■

Now, we consider the computational complexity of the branching counting quantifier: $\mathsf{Br}(\mathsf{C}^{\geq \mathsf{A}}, \mathsf{C}^{\geq \mathsf{B}})$.

We identify models of the form $\mathbb{M} = (M, A, B, V, T, H)$ with colored undirected graphs. $\mathbb{M} \in \mathsf{Br}(\mathsf{C}^{\geq \mathsf{A}}, \mathsf{C}^{\geq \mathsf{B}})$ if and only if there exists two sets of vertices $V' \subseteq V$ and $T' \subseteq T$ such that $\mathrm{card}(V') \geq \mathrm{card}(A)$, $\mathrm{card}(T') \geq \mathrm{card}(B)$ and $V' \times T' \subseteq H$. Then we show that a generalized version of the BALANCED COMPLETE BIPARTITE GRAPH problem (BCBG) is equivalent to our problem. We need the following notions.

3.2.5. DEFINITION. A graph $G = (V, E)$ is *bipartite* if there exists a partition V_1, V_2 of its vertices (i.e., $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$) such that $E \subseteq V_1 \times V_2$. ■

3.2.6. DEFINITION. BCBG is the following problem. Given a bipartite graph $G = (V, E)$ and integer k we must determine whether there exist sets W_1, W_2 both of size at least k such that $W_1 \times W_2 \subseteq E$. ■

BCBG is an NP-complete problem, as was noticed by Garey and Johnson (1979, p. 196, problem GT24). We need a slightly different version of BCBG with two parameters k_1 and k_2 constraining the size of sets W_1 and W_2, respectively. Also this variant is clearly NP-complete as it has $k_1 = k_2 = k$ as a special case. Now we can state the following.

3.2.7. PROPOSITION. *The quantifier* $\mathsf{Br}(\mathsf{C}^{\geq \mathsf{A}}, \mathsf{C}^{\geq \mathsf{B}})$ *is mighty.*

Proof Let us take a colored bipartite graph model $\mathbb{G} = (V, A, B, E)$, such that $V = V_1 \cup V_2$ and $E \subseteq V_1 \times V_2$. Notice that $\mathbb{G} \in \mathsf{Br}(\mathsf{C}^{\geq \mathsf{A}}, \mathsf{C}^{\geq \mathsf{B}})$ if and only if graph $\mathbb{G}$ and integers $\mathrm{card}(A)$ and $\mathrm{card}(B)$ are in BCBG. □

This constitutes another class of branching mighty quantifiers.

3.2.4 Linguistic Remark and Branching of Dissertation

There is one linguistic proviso concerning all these examples. Namely, they are ambiguous. Moreover, such sentences can hardly be found in a linguistic corpus (see Sevenster, 2006, footnote 8 p. 140). In Chapter 6 we continue that topic and we show that their readings vary between easy (PTIME) and difficult (branching) interpretations. Additionally, we argue that the non-branching reading is the dominant one.

On the other hand, this proviso motivates us to look for mighty natural language quantifiers which not only occur frequently in everyday English but are also one of the sources of its complexity. Chapter 4 of the dissertation is devoted to presenting the so-called reciprocal expressions, which are a common element of everyday English. They can be interpreted by so-called Ramsey quantifiers and as a result they give rise to examples of uncontroversial NP-complete natural language constructions. The rest of this chapter is devoted to studying the computational complexity of Ramsey quantifiers.

3.3 Ramsey Quantifiers

3.3.1 Ramsey Theory and Quantifiers

Essentially all of the proofs of NP-completeness for branching quantifiers are based on a kind of Ramsey property which is expressible by means of branching (see the following Section 3.3.2 for an example). Some Ramsey properties have been considered in the literature as generalized quantifiers since the seventies (see e.g. Hella et al., 1997; Luosto, 1999; Magidor and Malitz, 1977; Paris and Harrington, 1977; Schmerl and Simpson, 1982). Comparisons of Henkin quantifiers with Ramsey quantifiers can be found in (Mostowski, 1991; Krynicki and Mostowski, 1995).

Informally speaking Ramsey (1929) Theorems state the following:[2]

The Infinite Ramsey Theorem — general schema *For each coloring of the set U^k — for a large infinite set U — there is a large set $A \subseteq U$ such that A^k are of the same colour.*

The Finite Ramsey Theorem — general schema *When coloring a sufficiently large complete finite graph, one will find a large homogeneous subset, i.e., a complete subgraph with all edges of the same colour, of arbitrary large finite cardinality.*

For suitable explications of what “large set” means we obtain various Ramsey properties. In the case $U = \omega$ the countable Ramsey Theorem takes “large set” as meaning an “infinite set”. When dealing with models for Peano Arithmetic it is sometimes interpreted as “co-final set” (see e.g. Macintyre, 1980) or “set of cardinality greater than the minimal element of this set” (see e.g. Paris and Harrington, 1977). Other known explications for “large set” are “set of cardinality at least κ” (see e.g. Magidor and Malitz, 1977), and “set of cardinality at least $f(n)$”, where f is a function from natural numbers to natural numbers on a universe with n elements (see e.g. Hella et al., 1997). We will adopt this last interpretation in our work.

A related concept is that of the Ramsey number, $R(r, s)$, i.e., the size of the smallest complete graph for which when the edges are colored, e.g., red or blue, there exists either a complete subgraph on r vertices which is entirely blue, or a complete subgraph on s vertices which is entirely red. Here $R(r, s)$ signifies an integer that depends on both r and s. The existence of such number is guaranteed by the Finite Ramsey Theorem.

The high complexity of Ramsey type questions was observed very early. The following quotation from Paul Erdös has become a part of mathematical folklore:

[2]More details may be found in the monograph on Ramsey Theory by Graham et al. (1990).

Imagine an alien force, vastly more powerful than us landing on Earth and demanding the value of $R(5,5)$ or they will destroy our planet. In that case, we should marshal all our computers and all our mathematicians and attempt to find the value. But suppose, instead, that they asked for $R(6,6)$, we should attempt to destroy the aliens.[3]

(see e.g. Spencer, 1987)

The question of how hard it is to compute $R(r,s)$ is not very interesting for complexity theory, since $R(r,s)$ might be exponentially large compared to r and s. However, consider a similar problem, called ARROWING. Let $F \longrightarrow (G,H)$ mean that for every way of coloring the edges of F red and blue, F will contain either a red G or a blue H. The ARROWING problem is to decide whether $F \longrightarrow (G,H)$, for given finite graphs F, G, and H. ARROWING was proven to be complete for the second level of the polynomial hierarchy by Marcus Schaefer (2001).

In the next section we study the computational complexity of quantifiers expressing some Ramsey properties on finite models. In the following chapter of the dissertation we argue that such quantifiers are one of the sources of complexity in natural language.

We identify models of the form $\mathbb{M} = (M,R)$, where $R \subseteq M^2$, with graphs. If R is symmetric then we are obviously dealing with undirected graphs. Otherwise, our models become directed graphs. In what follows we will restrict ourselves to undirected graphs.

Let us start with the general definition of Ramsey quantifiers.

3.3.1. DEFINITION. A *Ramsey quantifier* R is a generalized quantifier of type (2), binding two variables, such that $\mathbb{M} \models \mathsf{R}xy\, \varphi(x,y)$ exactly when there is $A \subseteq M$ (large relative to the size of M) such that for each $a,b \in A$, $\mathbb{M} \models \varphi(a,b)$. ■

We study the computational complexity of various Ramsey quantifiers determined by suitable explications of the phrase "large relative to the universe".

3.3.2 The Branching Reading of Hintikka's Sentence

We start by giving some connections between Ramsey quantifiers and branching quantifiers. One of the possible explications of the phrase "large relative to the universe" can be extracted from the meaning of the branching interpretation of Hintikka's sentence, (7) (see Mostowski and Wojtyniak, 2004). Let us consider models of the form $\mathbb{M} = (M,E,\ldots)$, where E is an equivalence relation. Being a "large set" in this case means having nonempty intersection with each E-equivalence class (compare with the quantifier Z from Section 3.2). We define the corresponding Ramsey quantifier, $\mathsf{R_e}$.

[3]Currently the number $R(19,19)$ is known (Luo et al., 2002). It is equal to 178,859,075,135,299.

3.3.2. DEFINITION. $\mathbb{M} \models \mathsf{R_e}xy\ \varphi(x, y)$ means that there is a set $A \subseteq M$ such that $\forall a \in M\ \exists b \in A\ E(a, b)$ and for each $a, b \in A$, $\mathbb{M} \models \varphi(a, b)$. ■

It is argued by Mostowski and Wojtyniak (2004) that the computational complexity of the branching reading of Hintikka's sentence can be reduced to that of the quantifier $\mathsf{R_e}$ and then the following is proven:

3.3.3. THEOREM. *The quantifier* $\mathsf{R_e}$ *is mighty.*

This gives one example of a mighty Ramsey quantifier which arises when studying natural language semantics. Below we give more such Ramsey quantifiers.

3.3.3 Clique Quantifiers

Let us start with simple Ramsey quantifiers expressing the CLIQUE problem.

3.3.4. DEFINITION. Define for every $k \in \omega$ the Ramsey quantifier $\mathsf{R_k}$ in the following way. $\mathbb{M} \models \mathsf{R_k}xy\ \varphi(x, y)$ iff there is $A \subseteq M$ such that $\text{card}(A) \geq k$ and for all $a, b \in A$, $\mathbb{M} \models \varphi(a, b)$. ■

Notice that for a fixed k the sentence $\mathsf{R_k}xy\ \varphi(x, y)$ is equivalent to the following first-order formula:

$$\exists x_1 \ldots \exists x_k \Big[\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \ \wedge \bigwedge_{\substack{1 \leq i \leq k \\ 1 \leq j \leq k}} \varphi(x_i, x_j) \Big].$$

Therefore, we can decide whether some model $\mathbb{M}$ belongs to the class corresponding to a Ramsey quantifier $\mathsf{R_k}$ in LOGSPACE. In other words, model checking for $\mathsf{R_k}$ is like solving the CLIQUE problem for $\mathbb{M}$ and k. A brute force algorithm to find a clique in a graph is to examine each subgraph with at least k vertices and check if it forms a clique. This means that for every fixed k the computational complexity of $\mathsf{R_k}$ is in PTIME. However, in general — for unbounded k — this is a well-known NP-complete problem (see Garey and Johnson, 1979, problem GT19) (see also Theorem 2.3.28 for the proof). Below we define the Ramsey counting quantifier corresponding to the general CLIQUE problem.

3.3.5. DEFINITION. Let us consider models of the form $\mathbb{M} = (M, A, \ldots)$. We define the Ramsey counting quantifier, $\mathsf{R_A}$, as follows: $\mathbb{M} \models \mathsf{R_A}xy\ \varphi(x, y)$ iff there is $X \subseteq M$ such that $\text{card}(X) \geq \text{card}(A)$ and for all $a, b \in X$, $\mathbb{M} \models \varphi(a, b)$.[4] ■

The Ramsey quantifier $\mathsf{R_A}$ expresses the general CLIQUE problem and as a result it inherits its complexity.

[4]Compare this definition with Definition 3.2.4.

3.3.6. PROPOSITION. *The Ramsey quantifier* R_A *is mighty.*

Proof Let as take any model $\mathbb{M} = (M, A, \ldots)$. We have to decide whether $\mathbb{M} \models \mathsf{R}_\mathsf{A} xy\ \varphi(x, y)$. This is equivalent to the CLIQUE problem for $\mathbb{M}$ and card(A). Therefore, the Ramsey quantifier R_A defines an NP-complete class of finite models. □

Below we extend this observation to cover cases where the size of a clique is supposed to be relative to the size of the universe.

3.3.4 Proportional Ramsey Quantifiers

Let us start with a precise definition of "large relative to the universe".

3.3.7. DEFINITION. For any rational number q between 0 and 1 we say that *the set* $A \subseteq U$ *is* q*-large relative to* U if and only if

$$\frac{card(A)}{card(U)} \geq q.$$

■

In this sense q determines the *proportional Ramsey quantifier* R_q.

3.3.8. DEFINITION. $\mathbb{M} \models \mathsf{R}_\mathsf{q} xy\ \varphi(x, y)$ iff there is a q-large (relative to M) $A \subseteq M$ such that for all $a, b \in A$, $\mathbb{M} \models \varphi(a, b)$. ■

We will prove that for every rational number $0 < q < 1$ the corresponding Ramsey quantifier R_q defines an NP-complete class of finite models.[5]

3.3.9. THEOREM. *For every rational number* q*, such that* $0 < q < 1$*, the corresponding Ramsey quantifier* R_q *is mighty.*

To prove this theorem we will define the corresponding CLIQUE problem and show its NP-completeness.

3.3.10. DEFINITION. Let q be a rational number, such that $0 < q < 1$, and $G = (V, E)$ an undirected graph. We define the *problem* CLIQUE$_{\geq q}$ as a decision problem whether in graph G at least a fraction q of the vertices form a complete subgraph. ■

[5]The following result was obtained in co-operation with Marcin Mostowski (see Mostowski and Szymanik, 2007).

Now we can state the following lemma.

3.3.11. LEMMA. *For any rational number q between 0 and 1 the problem* CLIQUE$_{\geq q}$ *is NP-complete.*

Proof The problem CLIQUE$_{\geq q}$ is obviously in NP as it might be easily verified in polynomial time by a nondeterministic Turing machine. The machine simply guesses a set $A \subseteq V$ and then it can easily check in polynomial time whether A satisfies $\frac{\text{card}(A)}{\text{card}(V)} \geq q$ and that the graph restricted to A is complete. Therefore, it suffices to prove hardness.

To prove this we will polynomially reduce the problem CLIQUE to the problem CLIQUE$_{\geq q}$. Recall that the standard CLIQUE problem is to decide for a graph G and an integer $k > 0$, if G contains a complete subgraph of size at least k (see Example 2.3.25).

Let $G = (V, E)$ and $k \in \omega$ be an instance of CLIQUE. Assume that $\text{card}(V) = n$. Now we construct from G in polynomial time a graph $G' = (V', E')$ belonging to CLIQUE$_{\geq q}$.

Let $m = \lceil \frac{qn-k}{1-q} \rceil$, where $\lceil p \rceil$ is the ceiling function of p. Then we take G' consisting of G and a complete graph of m vertices, K_m. Every vertex from the copy of G is connected to all nodes in K_m and there are no other extra edges. Hence, $\text{card}(V') = n + m$ and $\text{card}(Cl') = \text{card}(Cl) + m$, where Cl and Cl' are the largest clique in G and G', respectively. We claim that the graph G has a clique of size k iff graph G' has a q-large clique.

For proving our claim we need the following:

$$k + m = \lceil q(n + m) \rceil$$

Proof:

$$(\geq) : \; m = \left\lceil \frac{qn-k}{1-q} \right\rceil . \text{ Hence, } m \geq \left\lceil \frac{qn-k}{1-q} \right\rceil .$$

$$\text{Now, } m \geq \frac{qn-k}{1-q} \text{ ,then } (1-q)m \geq qn - k.$$

$$\text{Therefore, } k + m \geq \lceil q(n + m) \rceil.$$

$$(\leq) : \text{Notice that } \; m(1-q) = (1-q) \left\lceil \frac{qn-k}{1-q} \right\rceil \leq (1-q) \left(\frac{qn-k}{1-q} + 1 \right).$$

$$(1-q) \left(\frac{qn-k}{1-q} + 1 \right) = qn - k + 1 - q < qn - k + 1.$$

$$\text{So } m(1-q) < qn - k + 1 \text{ and } m(1-q) + k - 1 < qn.$$

$$\text{Hence, } k + m - 1 < q(n + m) \leq \lceil q(n + m) \rceil \text{ and } k + m - 1 < \lceil q(n + m) \rceil.$$

$$\text{Therefore, } k + m \leq \lceil q(n + m) \rceil.$$

Therefore, the following are equivalent:

(1) In G there is a clique of size at least k;

(2) $\text{card}(Cl) \geq k$;

(3) $\text{card}(Cl') \geq k + m$;

(4) $\text{card}(Cl') \geq \lceil q(n+m) \rceil$;

(5) $\text{card}(Cl') \geq q(n+m)$;

(6) $\frac{\text{card}(Cl')}{\text{card}(V')} \geq \frac{q(n+m)}{n+m}$

(7) The clique Cl' is q-large in G'.

Hence, we have shown that the problem $\text{CLIQUE}_{\geq q}$ is NP-complete. $\square$

Theorem 3.3.9 follows directly from the lemma. It suffices to notice that for any rational number q between 0 and 1: $\mathbb{M} \models \mathsf{R_q}xy\ \varphi(x,y)$ iff there is a q-large $A \subseteq M$ such that for all $a, b \in A$, $\mathbb{M} \models \varphi(a,b)$. Therefore, given a model $\mathbb{M}$ the model checking procedure for the query $\mathbb{M} \models \mathsf{R_q}xy\ \varphi(x,y)$ is equivalent to deciding whether there is a q-large $A \subseteq M$ complete with respect to the relation being defined by the formula φ. From our lemma this problem is NP-complete for φ being of the form $R(x,y)$.

3.3.5 Tractable Ramsey Quantifiers

We have shown some examples of NP-complete Ramsey quantifiers. In this section we will describe a class of Ramsey quantifiers computable in polynomial time.

Let us start with considering an arbitrary function $f : \omega \longrightarrow \omega$.

3.3.12. Definition. We say that a *set $A \subseteq U$ is f-large relatively to U* iff

$$card(A) \geq f(card(U)).$$

■

Then we define Ramsey quantifiers corresponding to the notion of "f-large".

3.3.13. Definition. We define $\mathsf{R_f}$ as follows $\mathbb{M} \models \mathsf{R_f}xy\ \varphi(x,y)$ iff there is an f-large set $A \subseteq M$ such that for each $a, b \in A$, $\mathbb{M} \models \varphi(a,b)$. ■

Notice that the above definition is very general and covers all previously defined Ramsey quantifiers. For example, we can reformulate Theorem 3.3.9 in the following way:

3.3.14. COROLLARY. *Let $f(n) = \lceil rn \rceil$, for some rational number r such that $0 < r < 1$. Then the quantifier $\mathsf{R_f}$ is mighty.*

Let us put some further restrictions on the class of functions we are interested in. First of all, as we will consider f-large subsets of the universe we can assume that for all $n \in \omega$, $f(n) \leq n+1$. In that setting the quantifier $\mathsf{R_f}$ says about a set A that it has at least $f(n)$ elements, where n is the cardinality of the universe. We allow the function to be equal to $n+1$ just for technical reasons as in this case the corresponding quantifier has to be always false.

Our crucial notion goes back to a paper of Väänänen (1997b).

3.3.15. DEFINITION. We say that a function f *is bounded* if

$$\exists m \forall n [f(n) < m \vee n - m < f(n)].$$

Otherwise, f is *unbounded.* ■

Typical bounded functions are: $f(n) = 1$ and $f(n) = n$. The first one is bounded from above by 2 as for every n we have $f(n) = 1 < 2$. The second one is bounded below by 1, for every n, $n - 1 < n$. Unbounded functions are for example: $\lceil \frac{n}{2} \rceil$, $\lceil \sqrt{n} \rceil$, $\lceil \log n \rceil$. We illustrate the situation in Figure 3.1.

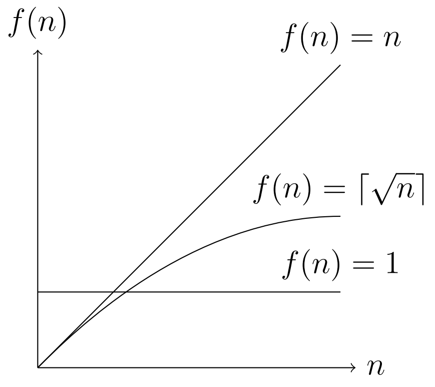


Figure 3.1: The functions $f(n) = 1$ and $f(n) = n$ are bounded. The function $\lceil \sqrt{n} \rceil$ is unbounded.

In what follows we will show that Ramsey quantifiers corresponding to the bounded polynomial time computable functions are in PTIME.

3.3.16. THEOREM. *If f is PTIME computable and bounded, then the Ramsey quantifier $\mathsf{R_f}$ is PTIME computable.*

Proof Assume that f is PTIME computable and bounded. Then there exists a number m such that for every n the following disjunction holds $[f(n) < m \text{ or } n - m < f(n)]$.

Let us fix a graph model $\mathbb{G} = (V, E)$, where $\mathrm{card}(V) = n$.

In the first case assume that $f(n) < m$. First observe that if there exists a clique of size greater than $f(n)$ then there has to be also a clique of size exactly $f(n)$. Thus to decide whether $\mathbb{G} \in \mathsf{R_f}$ it is enough to check if there is a clique of size $f(n)$ in $\mathbb{G}$. We know that $f(n) < m$. Hence we only need to examine all subgraphs up to m vertices. For each of them we can check in polynomial time whether it forms a clique. Hence, it is enough to observe that the number of all subgraphs of size between 1 up to m is bounded by a polynomial. In fact this is the case as the number of k-combinations from a set is smaller than the number of permutations with repetitions of length k from that set. Therefore, we have:

$$\binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{m} \leq n^1 + n^2 + \ldots + n^m \leq m(n^m).$$

Let us consider the second case; assume that $n - m < f(n)$. This time we have to only check large subgraphs; to be precise, we need to examine all subgraphs containing from n down to $n - m$ vertices. Again, the number of such subgraphs is bounded by a polynomial for fixed m. We use the following well known equality $\binom{n}{n-k} = \binom{n}{k}$ to show that we have to inspect only a polynomial number of subsets:

$$\binom{n}{n} + \binom{n}{n-1} + \ldots + \binom{n}{n-m} = \binom{n}{n} + \binom{n}{1} + \ldots + \binom{n}{m}$$
$$\leq 1 + n^1 + n^2 + \ldots + n^m \leq m(n^m).$$

Therefore, in every case when f is bounded and computable in a polynomial time we simply run the two algorithms given above. This model-checking procedure for $\mathsf{R_f}$ simply tests the clique property on all subgraphs up to m elements and from n to $n - m$ elements, where m is fixed and independent of the size of a universe. Therefore, it is bounded by a polynomial. $\square$

The property of boundedness plays also a crucial role in the definability of polyadic lifts. Hella et al. (1997) showed that the Ramseyfication of Q is definable in $\mathrm{FO}(\mathsf{Q})$ if and only if Q is bounded. They also obtained similar results for branching and resumption (see Hella et al., 1997, for details).

Moreover, in a similar way, defining "joint boundness" for pairs of quantifiers Q_f and Q_g (see Hella et al., 1997, page 321), one can notice that $\mathsf{Br}(\mathsf{Q}_f, \mathsf{Q}_g)$ is definable in $\mathrm{FO}(\mathsf{Q}_f, \mathsf{Q}_g)$ (see Hella et al., 1997, Theorem 3.12) and therefore PTIME computable for polynomial functions f and g.

Actually, the above theorems follow from a more general observation. Let us consider a property Q (corresponding to boundness) such that $Q(X)$ iff there exists m such that X differs from the universe or empty set on at most m elements. Now observe that second-order quantification restricted to Q is definable in first-order logic with $m+1$ parameters. We simply have the following equivalence:

$$\exists X Q(X) \iff \forall t_1 \ldots \forall t_m \forall t_{m+1} \Big[\big(\bigwedge_{1\leq i<j\leq m+1} X(t_i) \implies \bigvee_{1\leq i<j\leq m+1} t_i = t_j \big) \vee \big(\bigwedge_{1\leq i<j\leq m+1} \neg X(t_i) \implies \bigvee_{1\leq i<j\leq m+1} t_i = t_j \big) \Big].$$

This formula says that X has a property Q if and only if X consists of at most m elements or X differs from the universe on at most m elements. Notice, that this argument works also for infinite sets.

3.4 Summary

In this chapter we have investigated the computational complexity of polyadic quantifiers, preparing the ground for the linguistic discussion in the following parts of the thesis. We have shown that some polyadic constructions do not increase computational complexity, while others — such as branching quantifiers and Ramsey quantifiers — might be NP-complete. In particular we have observed the following:

- PTIME quantifiers are closed under Boolean operations, iteration, cumulation, and resumption.
- When branching PTIME quantifiers we may arrive at NP-complete polyadic quantifiers, e.g. branching proportional quantifiers are mighty.
- Ramsey counting quantifiers are mighty.
- Proportional Ramsey quantifiers are mighty.
- Bounded Ramsey quantifiers (and branching quantifiers) are PTIME computable.

In the next chapter we apply Ramsey quantifiers to the study of reciprocal expressions in English. Namely, we define so-called reciprocal lifts which turn monadic quantifiers into Ramsey quantifiers.

As far as future work is concerned the following seem to be the most intriguing questions.

We have shown that proportional Ramsey quantifiers define NP-complete classes of finite models. On the other hand, we also observed that bounded Ramsey quantifiers are in PTIME. It is an open problem where the precise border lies between tractable and mighty Ramsey quantifiers.

3.4.1. Question. Can we prove under some complexity assumptions that the PTIME Ramsey quantifiers are exactly the bounded Ramsey quantifiers?

3.4.2. Question. Is it the case that for every function f from some class we have a duality theorem, i.e., $\mathsf{R_f}$ is either PTIME computable or NP-complete?

The proper class of functions can be most likely obtained by a combination of unboundness together with some conditions on growth-rate.

Last, but not least, there is the question of possible applications.

3.4.3. Question. Do differences in computational complexity of polyadic quantifiers play any role in natural language interpretation?

In the next Chapter we will argue that they do.

Chapter 4

Complexity of Quantified Reciprocals

The reciprocal expressions *each other* and *one another* are common elements of everyday English. Therefore, it is not surprising that they have been extensively studied in the formal semantics of natural language. There are two main approaches to reciprocals in the literature. The long trend of analyzing reciprocals as anaphoric noun phrases with the addition of plural semantics culminates in a paper of Beck (2000). A different tendency — recently represented by Sabato and Winter (2005) — is to analyze reciprocals as polyadic quantifiers.

In this chapter we study the computational complexity of quantified reciprocal sentences. We ally ourselves to the second tradition and treat reciprocal sentences as examples of a natural language semantic construction that can be analyzed in terms of so-called polyadic lifts of simple generalized quantifiers (see Chapter 3 of the thesis).

First, we propose new polyadic lifts expressing various possible meanings of reciprocal sentences with quantified antecedents, i.e., sentences where "each other" refers in a co-reference to the quantified noun phrase (see Dalrymple et al., 1998, Chapter 7). In other words, we will consider quantified reciprocal sentences, like "Five professors discuss with each other", where reciprocal phrase "each other" refers to quantified noun phrase, in this case "five professors". All these lifts are definable in the existential fragment of second-order logic. Therefore, according to the Σ_1^1-thesis formulated in Section 1.8 the model we investigate seems plausible.

Then we study the computational complexity of reciprocal lifts with respect to the quantifiers in the antecedents. Using results from the previous chapter (on the computational complexity of Ramsey quantifiers) we observe a computational dichotomy between different interpretations of reciprocity. Namely, we treat reciprocal expressions as polyadic lifts turning monadic quantifiers into Ramsey-like quantifiers. Differences in computational complexity beetween various interpretations of reciprocal expressions give an additional argument for the robustness of the semantic distinctions established between reciprocal meanings (see Dalrymple et al., 1998).

In particular, we give a sufficient condition for a generalized quantifier to make its strong reciprocal interpretation PTIME computable. Moreover, we present NP-complete natural language quantifier constructions which occur frequently in everyday English. For instance, strong interpretations of reciprocal sentences with counting and proportional quantifiers in the antecedents are intractable. As far as we are aware, all other known NP-complete quantifier constructions are based on ambiguous and artificial branching operations (see Section 3.2 and Chapter 6 for more discussion).

Finally, we investigate the cognitive status of the so-called Strong Meaning Hypothesis proposed by Dalrymple et al. (1998). We argue that if one assumes some kind of algorithmic theory of meaning as we do in Chapter 1, then the shifts between different interpretations of reciprocal sentences, predicted by the Strong Meaning Hypothesis, have to be extended by accommodating the possible influence of differences in computational complexity between various readings of reciprocity.

The considerations of this chapter are based on papers from the Amsterdam Colloquium 2007 (see Szymanik, 2007b), Lecture Notes in Computer Science (Szymanik, 2008), and results proven in Chapter 3 of the thesis.

4.1 Reciprocal Expressions

We start by recalling examples of reciprocal sentences, versions of which can be found in ordinary (spoken and written) English (see footnote 1 in Dalrymple et al., 1998). Let us first consider sentences (1)–(3).

(1) At least 4 members of parliament refer to each other indirectly.

(2) Most Boston pitchers sat alongside each other.

(3) Some Pirates were staring at each other in surprise.

The possible interpretations of reciprocity exhibit a wide range of variation. For example, sentence (1) implies that there is a subset of parliament members of cardinality at least 4 such that each parliament member in that subset refers to each of the other parliament members in that subset. However, the reciprocals in sentences (2) and (3) have different meanings. Sentence (2) states that each pitcher from a set containing most of the pitchers is directly or indirectly in the relation of sitting alongside with each of the other pitchers from that set. Sentence (3) says that there was a group of pirates such that every pirate belonging to the group stared at some other pirate from the group. Typical models satisfying (1)–(3) are illustrated in Figure 4.1. Following Dalrymple et al. (1998) we will call the illustrated reciprocal meanings *strong*, *intermediate*, and *weak*, respectively.

In general, according to Dalrymple et al. (1998) there are 2 parameters characterizing variations of reciprocity. The first one relates to how the scope relation,

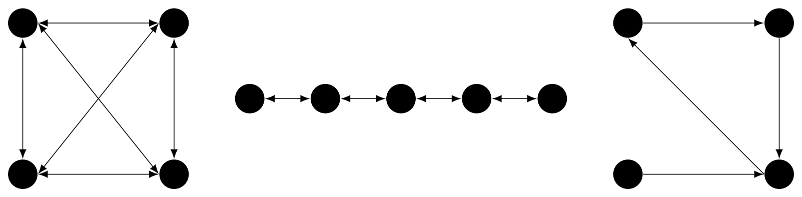


Figure 4.1: On the left is a typical model satisfying sentence (1) under the so-called *strong reciprocal* interpretation. Each element is related to each of the other elements. In the middle is an example of a model satisfying sentence (2) in a context with at most 9 pitchers. This is the *intermediate reciprocal* interpretation. Each element in the witness set of the quantifier Most is related to each other element in that set by a chain of relations. On the right, a model satisfying sentence (3), assuming the so-called *weak reciprocal* interpretation. For each element there exists a different related element.

R, should cover the domain, A, (in our case restricted by a quantifier in the antecedent). We have 3 possibilities:

FUL Each pair of elements from A participates in R directly.

LIN Each pair of elements from A participates in R directly or indirectly.

TOT Each element in A participates directly with at least one element in R.

The second parameter determines whether the relation R between individuals in A is the extension of the reciprocal's scope (R), or is obtained from the extension by ignoring the direction in which the scope relation holds ($R^{\vee} = R \cup R^{-1}$).

By combining these two parameters Dalrymple et al. (1998) gets six possible meanings for reciprocals. We have already encountered three of them: strong reciprocity, FUL(R); intermediate reciprocity, LIN(R); and weak reciprocity, TOT(R). There are three new logical possibilities: strong alternative reciprocity, FUL($R^{\vee}$); intermediate alternative reciprocity, LIN($R^{\vee}$); and weak alternative reciprocity, TOT($R^{\vee}$). Among these, two interpretations are linguistically attested: intermediate alternative reciprocity is exhibited by sentence (4) and weak alternative reciprocity occurs in sentence (5) (see Figure 4.2 for typical models).

(4) Most stones are arranged on top of each other.

(5) All planks were stacked on top of each other.

If we do not put any restrictions on the scope of the relation R, then stronger reciprocal interpretations imply weaker ones — as it is depicted in the left part of Figure 4.3. However, assuming certain properties of the relation some of the possible definitions become equivalent. For example, if the relation in question is symmetric, then obviously alternative versions reduce to their "normal" counterparts

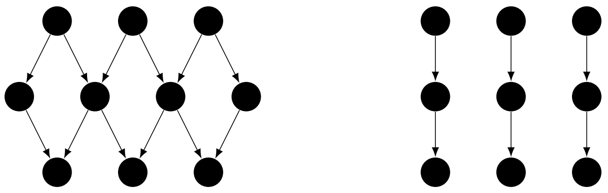


Figure 4.2: On the left is a typical model satisfying sentence (4) under the so-called *intermediate alternative reciprocal* interpretation. Ignoring the direction of arrows, every element in the witness set of the quantifier Most is connected directly or indirectly. On the right is an example of a model satisfying sentence (5) under the so-called *weak alternative reciprocal* reading. Each element participates with some other element in the relation as the first or as the second argument, but not necessarily in both roles.

and we have only three different reciprocal interpretations: $FUL(R) = FUL(R^\vee)$, $LIN(R) = LIN(R^\vee)$, and $TOT(R) = TOT(R^\vee)$. If the relation R is transitive, then $FUL(R) = LIN(R)$ and the classification of different reciprocal meanings collapses to the one depicted in Figure 4.3 on the right.

$$\begin{tikzcd}
 & \mathrm{FUL}(R) \arrow[dl] \arrow[dr] & \\
\mathrm{LIN}(R) \arrow[d] \arrow[drr] & & \mathrm{FUL}(R^\vee) \arrow[d] \\
\mathrm{TOT}(R) \arrow[dr] & & \mathrm{LIN}(R^\vee) \arrow[dl] \\
 & \mathrm{TOT}(R^\vee) &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathrm{FUL}(R)=\mathrm{LIN}(R) \arrow[dl] \arrow[dr] & \\
\mathrm{TOT}(R) \arrow[ddr] & & \mathrm{FUL}(R^\vee) \arrow[d] \\
 & & \mathrm{LIN}(R) \arrow[dl] \\
 & \mathrm{TOT}(R^\vee) &
\end{tikzcd}$$

Figure 4.3: On the left, inferential dependencies between the six interpretations of reciprocity. On the right, the situation when the reciprocal relation is transitive. In these diagrams implications are represented by arrows.

4.1.1 Strong Meaning Hypothesis

In an attempt to explain variations in the literal meaning of the reciprocal expressions Dalrymple et al. (1998) proposed the *Strong Meaning Hypothesis* (SMH).

According to this principle, the reading associated with the reciprocal in a given sentence is the strongest available reading which is consistent with the properties of reciprocal relation and with relevant information supplied by the context. Sabato and Winter (2005) proposed a considerably simpler system in which reciprocal meanings are derived directly from semantic restrictions using the SMH.

4.1.1. Example. Let us give one of the examples described by Dalrymple et al. (1998) of using SMH to derive proper interpretation of reciprocal statements. Consider the following sentence:

(6) The children followed each other.

This sentence can be interpreted in many ways depending on what is permitted by the context. First, consider:

(7) The children followed each other into the church.

The relation "following into the church" is asymmetric and intransitive disallowing strong (alternative) reciprocal interpretation. Moreover, the intermediate interpretation is impossible since children who go into the church first cannot even indirectly be said to follow children who go into the church later. Additionally, if the group of children is finite then the weak reading is excluded as it is not possible for each child to be a follower; simply put, someone must be the first to go into the church. This analyzes leaves 2 possibilities: the alternative intermediate reading and the weak alternative interpretation. The first suggested that children entered in one group while the later allows more than one group. As the alternative intermediate reading implies the alternative weak reading then SMH predicts that the sentence has the first meaning, assuming that the context does not supply additional information that the children enter the church in multiple groups. However, when you consider the similar sentence:

(8) The children followed each other around the Maypole.

Then unlike in the context described above, the path traversed by the children is circular. Hence, the intermediate reading appears as one of the possible interpretations. This is logically strongest possibility and according to SMH it properly describes the meaning of that sentence.

Our results show that the various meanings assigned to reciprocals with quantified antecedents differ drastically in their computational complexity. This fact can be treated as a suggestion to improve the SMH by taking into an account complexity constraints. We elaborate on this in the last section of this chapter, before we draw some conclusions.

4.2 Reciprocals as Polyadic Quantifiers

Monadic generalized quantifiers provide the most straightforward way to define the semantics of noun phrases in natural language (see Peters and Westerståhl, 2006, for a recent overview; also consult Section 2.2.). Sentences with reciprocal expressions transform such monadic quantifiers into polyadic ones. We will analyze reciprocal expressions in that spirit by defining appropriate lifts on monadic quantifiers. These lifts are definable in existential second-order logic.

For the sake of simplicity we will restrict ourselves to reciprocal sentences with right monotone increasing quantifiers in their antecedents. Recall from Section 2.2.5 that a quantifier Q of type $(1, 1)$ is monotone increasing in its right argument whenever: if $Q_M[A, B]$ and $B \subseteq B' \subseteq M$, then $Q_M[A, B']$. The lifts defined below can be extended to cover also sentences with decreasing and non-monotone quantifiers, for example by following the strategy of bounded composition suggested by Dalrymple et al. (1998) or the determiner fitting operator proposed by Ben-Avi and Winter (2003). The situation is here analogous to problems with collective lifts for non-increasing quantifiers, discussed in Section 5.2.3 of this dissertation.

4.2.1 Strong Reciprocal Lift

In order to define the meaning of strong reciprocity we make use of the well-known operation on quantifiers called *Ramseyfication* (see e.g. Hella et al., 1997, and Section 3.3 of this thesis).

4.2.1. Definition. Let Q be a right monotone increasing quantifier of type $(1, 1)$. We define:

$$\mathsf{Ram}_\mathsf{S}(\mathsf{Q})[A, R] \iff \exists X \subseteq A[\mathsf{Q}(A, X) \wedge \forall x, y \in X(x \neq y \implies R(x, y))].$$

■

We will call the result of such lifting a *Ramsey quantifier*.[1] It says that there exists a subset X of the domain A, restricted by a quantifier Q, such that every two elements from X are directly related via the reciprocal relation R.

In the same way we can also easily account for alternative strong reciprocity:

4.2.2. Definition.

$$\mathsf{Ram}_\mathsf{S}{}^{\vee}(\mathsf{Q})[A, R] \iff \exists X \subseteq A[\mathsf{Q}(A, X) \wedge \forall x, y \in X(x \neq y \implies (R(x, y) \vee R(y, x)))].$$

■

This expresses an analogous condition to the one before, but this time it is enough for the elements of X to be related either by R or by R^{-1}.

[1]Notice that in the previous chapter we defined Ramsey quantifiers to be of type (2) (see Definition 3.3.1). Hence, to be precise the result of the Ramseyfication gives the relativized (see Section 2.2.5) Ramsey quantifier.

4.2.2 Intermediate Reciprocal Lift

In a similar way we define more lifts to express intermediate reciprocity and its alternative version.

4.2.3. Definition.

$$\begin{aligned}\mathsf{Ram_I}(\mathsf{Q})[A,R] \iff \exists X \subseteq A[\mathsf{Q}(A,X) \wedge \forall x, y \in X \\ (x \neq y \implies \exists \text{ sequence } z_1,\ldots,z_\ell \in X \text{ such that} \\ (z_1 = x \wedge R(z_1,z_2) \wedge \ldots \wedge R(z_{\ell-1},z_\ell) \wedge z_\ell = y)].\end{aligned}$$

■

This condition guarantees that there exists a subset X of domain A which is connected with respect to R, i.e. any two elements from X are in the relation directly or indirectly.

4.2.4. Definition.

$$\begin{aligned}\mathsf{Ram_I}^{\vee}(\mathsf{Q})[A,R] \iff \exists X \subseteq A[\mathsf{Q}(A,X) \wedge \forall x, y \in X \\ (x \neq y \implies \exists \text{ sequence } z_1,\ldots,z_\ell \in X \text{ such that} \\ (z_1 = x \wedge (R(z_1,z_2) \vee R(z_2,z_1)) \wedge \ldots \\ \wedge (R(z_{\ell-1},z_\ell) \vee R(z_\ell,z_{\ell-1})) \wedge z_\ell = y)].\end{aligned}$$

■

In other words, $\mathsf{Ram_I}^{\vee}$ says that any two elements from X are in the relation $R^{\vee}$ directly or indirectly. The property of graph connectedness is not elementary expressible; we need a universal monadic second-order formula. Hence from the definability point of view $\mathsf{Ram_I}$ ($\mathsf{Ram_I}^{\vee}$) seems more complicated than $\mathsf{Ram_S}$ ($\mathsf{Ram_S}^{\vee}$). However, as we will see, this is not the case from the computational complexity point of view.

4.2.3 Weak Reciprocal Lift

For weak reciprocity we take the following lifts.

4.2.5. Definition.

$$\mathsf{Ram_W}(\mathsf{Q})[A,R] \iff \exists X \subseteq A[\mathsf{Q}(A,X) \wedge \forall x \in X\ \exists y \in X(x \neq y \wedge R(x,y))].$$

■

4.2.6. Definition.

$$\mathsf{Ram_W}^{\vee}(\mathsf{Q})[A,R] \iff \exists X \subseteq A[\mathsf{Q}(A,X) \wedge \forall x \in X \exists y \in X \\ (x \neq y \wedge (R(x,y) \vee R(y,x))].$$

■

The weak lifts say that there exists a subset X of the domain A such that for every element from this subset there exists another element in the subset related by R (or $R^{\vee}$ in the case of the alternative lift).

4.2.4 The Reciprocal Lifts in Action

All reciprocal lifts produce polyadic quantifiers of type (1, 2). We will call the values of these lifts (alternative) strong, (alternative) intermediate and (alternative) weak reciprocity, respectively.

4.2.7. Remark. Before we continue with an example, notice that all these lifts can be defined analogously for unary quantifiers, just as for type (1, 1). Simply replace condition $\mathsf{Q}(A,X)$ by $\mathsf{Q}(X)$ in the definitions.

4.2.8. Example. The linguistic application of reciprocal lifts is straightforward. For example, using them we can account for the meanings of the reciprocal sentences (1)–(5) discussed in Section 4.1. Below we recall these sentences one by one. Each sentence is associated with a meaning representation expressed in terms of reciprocal lifts and quantifiers corresponding to the simple determiners occurring in these sentences.

(1) At least 4 parliament members refer to each other indirectly.

(9) $\mathsf{Ram_S}(\mathsf{At\ least}\ 4)[\text{MP},\ \text{Refer-indirectly}]$.

(2) Most Boston pitchers sat alongside each other.

(10) $\mathsf{Ram_I}(\mathsf{Most})[\text{Pitcher},\ \text{Sit-next-to}]$.

(3) Some pirates were staring at each other in surprise.

(11) $\mathsf{Ram_W}(\mathsf{Some})[\text{Pirate},\ \text{Staring-at}]$.

(4) Most stones are arranged on top of each other.

(12) $\mathsf{Ram_I}^{\vee}(\mathsf{Most})[\text{Stones},\ \text{Arranged-on-top-of}]$.

(5) All planks were stacked on top of each other.

(13) $\mathsf{Ram_W}^{\vee}(\mathsf{All})[\text{Planks},\ \text{Stack-on-top-of}]$.

It is easy to see that our formulae express the appropriate reciprocal meanings of these sentences, i.e. (alternative) strong, (alternative) intermediate and (alternative) weak reciprocity, respectively. They are true in the corresponding models depicted in Figures 4.1 and 4.2.

4.3 Complexity of Strong Reciprocity

In this section we investigate the computational complexity of quantified strong reciprocal sentences. In other words, we are interested in how difficult it is to evaluate the truth-value of such sentences in finite models. Studying this problem we make direct use of facts proven in Section 3.3 and we refer to the methods of descriptive complexity theory introduced in Section 2.4 of the Mathematical Prerequisites chapter.

Recall that we identify models of the form $\mathbb{M} = (M, A, R)$, where $A \subseteq U$ and $R \subseteq U^2$, with colored graphs and that we consider only monotone increasing quantifiers. Hence, in graph-theoretical terms we can say that $\mathbb{M} \models \mathsf{Ram_S}(\mathsf{Q})[A, R]$ if and only if there is a subgraph in A complete with respect to R, of a size bounded below by the quantifier Q. R is the extension of a reciprocal relation. If R is symmetric then we are dealing with undirected graphs. In such cases $\mathsf{Ram_S}$ and $\mathsf{Ram_S}^{\vee}$ are equivalent. Otherwise, if the reciprocal relation R is not symmetric, our models become directed graphs.

In what follows we will restrict ourselves to undirected graphs. We show that certain strong reciprocal quantified sentences interpreted in such graphs are NP-complete. Notice that undirected graphs are a special case of directed graphs; then our NP-complete sentences are also intractable over directed graphs.

4.3.1 Counting Quantifiers in the Antecedent

To decide whether in some model $\mathbb{M}$ sentence $\mathsf{Ram_S}(\mathsf{At\ least\ k})[A, R]$ is true we have to solve the CLIQUE problem for M and k. Recall from Section 3.3.3 that a brute force algorithm to find a clique in a graph is to examine each subgraph with at least k vertices and check if it forms a clique. This means that for every fixed k the computational complexity of $\mathsf{Ram_S}(\mathsf{At\ least\ k})$ is in PTIME. For instance, $\mathsf{Ram_S}(\mathsf{At\ least\ 5})$ is computable in polynomial time. In general, notice that the strong reciprocal sentence $\mathsf{Ram_S}(\exists^{\geq k})[A, R]$ is equivalent to the following first-order formula:

$$\exists x_1 \ldots \exists x_k \Big[\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \quad \wedge \bigwedge_{1 \leq i \leq k} A(x_i) \quad \wedge \bigwedge_{\substack{1 \leq i \leq k \\ 1 \leq j \leq k}} R(x_i, x_j) \Big].$$

However, when we consider natural language semantics from a procedural point of view it is natural to assume that people have one quantifier concept

$\mathsf{At\ least}\ k$, for every natural number k, rather than the infinite set of concepts $\mathsf{At\ least}\ 1$, $\mathsf{At\ least}\ 2, \ldots$. It seems reasonable to suppose that we learn one mental algorithm to understand each of the counting quantifiers $\mathsf{At\ least}\ k$, $\mathsf{At\ most}\ k$, and $\mathsf{Exactly}\ k$, no matter which natural number k actually is. Mathematically, we can account for this idea by introducing counting quantifiers. Recall from Definition 3.2.4 that the counting quantifier $\mathsf{C}^{\geq \mathsf{A}}$ says that the number of elements satisfying some property is greater than or equal to the cardinality of the set A. In other words, the idea here is that determiners like $\mathsf{At\ least}\ k$ express a relation between the number of elements satisfying a certain property and the cardinality of some prototypical set A. For instance, the determiner $\mathsf{At\ least}\ k$ corresponds to the quantifier $\mathsf{C}^{\geq \mathsf{A}}$ such that $\mathrm{card}(A) = k$. Therefore, the determiners $\mathsf{At\ least}\ 1$, $\mathsf{At\ least}\ 2$, $\mathsf{At\ least}\ 3, \ldots$ are interpreted by one counting quantifier $\mathsf{C}^{\geq \mathsf{A}}$ — the set A just has to be chosen differently in every case.

The quantifier $\mathsf{Ram}_{\mathsf{S}}(\mathsf{C}^{\geq \mathsf{A}})$ expresses the general schema for a reciprocal sentence with a counting quantifier in the antecedent. Such a general pattern defines an NP-complete problem.

4.3.1. PROPOSITION. *The quantifier* $\mathsf{Ram}_{\mathsf{S}}(\mathsf{C}^{\geq \mathsf{A}})$ *is mighty.*

Proof This fact is equivalent to proposition 3.3.6 from the previous chapter, where we observed that the so-called Ramsey counting quantifier, R_{A}, is mighty. □

4.3.2. COROLLARY. *The quantifier* $\mathsf{Ram}_{\mathsf{S}}^{\vee}(\mathsf{C}^{\geq \mathsf{A}})$ *is mighty.*

These results indicate that even though in a given situation checking the truth-value of a sentence with a fixed number, such as (1), is tractable, the general schema characterizing strong reciprocal sentences with counting quantifiers is NP-complete.

4.3.2 Proportional Quantifiers in the Antecedent

We can give another example of a family of strong reciprocal sentences which are intractable. Let us consider the following sentences:

(14) Most members of parliament refer to each other.

(15) At least one third of the members of parliament refer to each other.

(16) At least $q \times 100\%$ of the members of parliament refer to each other.

We will call these sentences *strong reciprocal sentences with proportional quantifiers*. Their general form is given by the sentence schema (16), where q can be interpreted as any rational number between 0 and 1. These sentences say that with respect to the reciprocal relation, R, there is a complete subset $Cl \subseteq A$, where A is the set of all parliament members, such that $\mathrm{card}(Cl) \geq q \times \mathrm{card}(A)$.

Recall that for any rational number $0 < q < 1$ we say that a set $A \subseteq U$ is q-large relative to U if and only if $\frac{card(A)}{card(U)} \geq q$ (see Definition 3.3.7). In this sense q determines a proportional quantifier $\mathsf{Q_q}$ of type (1, 1) as follows.

4.3.3. Definition.

$$\mathbb{M} \models \mathsf{Q_q}[A, B] \text{ iff } \frac{\mathrm{card}(A \cap B)}{\mathrm{card}(A)} \geq q.$$

■

4.3.4. Example. Let us give two examples of proportional quantifiers.

$$\mathbb{M} \models \mathsf{Most}[A, B] \text{ iff } \frac{\mathrm{card}(A \cap B)}{\mathrm{card}(A)} > \frac{1}{2}.$$

$$\mathbb{M} \models \mathsf{At\ least\ one\ third}\ [A, B] \text{ iff } \frac{\mathrm{card}(A \cap B)}{\mathrm{card}(A)} \geq \frac{1}{3}.$$

The strong reciprocal lift of a proportional quantifier, $\mathsf{Ram_S(Q_q)}$, is of type (1, 2) and obviously might be used to express the meaning of sentences like (14)–(16). We will call quantifiers of the form $\mathsf{Ram_S(Q_q)}$ *proportional Ramsey quantifiers*. Notice that a quantifier $\mathsf{Ram_S(Q_q)}$ is simply a relativization (see 3.3.9 for a definition) of the mighty proportional Ramsey quantifier $\mathsf{R_q}$ defined in Chapter 3.3.4. Therefore, it inherits the computational complexity of $\mathsf{R_q}$.

4.3.5. Proposition. *If q is a rational number and $0 < q < 1$, then the quantifier* $\mathsf{Ram_S(Q_q)}$ *is mighty.*

Proof Notice that $\mathsf{Ram_S(Q_q)} = \mathsf{R_q^{rel}}$ and see the proof of Theorem 3.3.9 in the previous chapter. □

4.3.6. Corollary. *If q is a rational number and $0 < q < 1$, then the quantifier* $\mathsf{Ram_S{}^{\vee}(Q_q)}$ *is mighty.*

Therefore, strong reciprocal sentences with proportional quantifiers in the antecedent, like (14) or (15), are intractable (NP-complete).

4.3.3 Tractable Strong Reciprocity

Our examples show that the strong interpretation of some reciprocal sentences is intractable. In this section we will describe a class of unary monadic quantifiers for which the strong reciprocal interpretation is tractable (PTIME computable).

Following Väänänen (1997b) we will identify monotone simple unary quantifiers with number-theoretic functions, $f : \omega \longrightarrow \omega$, such that for all $n \in \omega$, $f(n) \leq n + 1$. In that setting the quantifier Q_f (corresponding to f) says of a set A that it has at least $f(n)$ elements, where n is the cardinality of the universe.

4.3.7. Definition. Given $f : \omega \longrightarrow \omega$, we define:

$$(\mathsf{Q}_f)_M[A] \iff card(A) \geq f(\mathrm{card}(M)).$$

■

4.3.8. Example.

- $\exists = (\mathsf{Q}_f)_M$, where $f(\mathrm{card}(M)) \geq 1$.
- $\forall = (\mathsf{Q}_g)_M$, where $g(\mathrm{card}(M)) = \mathrm{card}(M)$.
- $\mathsf{Most} = (\mathsf{Q}_h)_M$, where $h(\mathrm{card}(M)) > \frac{\mathrm{card}(M)}{2}$.

Notice that having a monotone increasing quantifier we can easily find the function corresponding to it.

4.3.9. Definition. Let Q be a monotone increasing quantifier of type (1). Define:

$$f(n) = \begin{cases} \text{least } k \text{ such that:} & \\ \exists U \exists A \subseteq U[\mathrm{card}(U) = n \wedge \mathrm{card}(A) = k \wedge \mathsf{Q}_U(A)] & \text{if such a } k \text{ exists} \\ n+1 & \text{otherwise.} \end{cases}$$

■

4.3.10. Proposition. *If Q is a monotone increasing quantifier of type (1) and function f is defined according to Definition 4.3.9 then*

$$\mathsf{Q} = \mathsf{Q}_f.$$

Proof The equality follows directly from the definitions. □

In the previous Chapter we have shown that for every PTIME computable and bounded (see Definition 3.3.15) function, f, the Ramsey quantifier R_f is also PTIME computable (see Theorem 3.3.16). The strong reciprocal lift — as we mentioned — produces Ramsey quantifiers from simple determiners. Hence, $\mathsf{Ram}_\mathsf{S}(\mathsf{Q}_f)$ corresponds to R_f. Therefore, we can claim that polynomial computable bounded quantifiers are closed under the strong reciprocal lift.

4.3.11. PROPOSITION. *If a monotone increasing quantifier Q_f is PTIME computable and bounded, then the reciprocal quantifier $\mathsf{Ram}_\mathsf{S}(\mathsf{Q}_f)$ is PTIME computable.*

Proof See the proof of Theorem 3.3.16 from the previous chapter. □

4.3.12. REMARK. Notice that it does not matter whether we consider undirected or directed graphs, as in both cases checking whether a given subgraph is complete can be done in polynomial time. Therefore, the result holds for $\mathsf{Ram}_\mathsf{S}{}^\vee(\mathsf{Q}_f)$ as well.

4.3.13. COROLLARY. *If a monotone increasing quantifier Q_f is PTIME computable and bounded, then the quantifier $\mathsf{Ram}_\mathsf{S}{}^\vee(\mathsf{Q}_f)$ is PTIME computable.*

4.3.14. REMARK. Moreover, notice, that the relativization, Q_f^{rel}, of Q_f is the right monotone type (1, 1) quantifier:

$$(Q_f^{rel})_M\ [A, B] \iff \text{card}(A \cap B) \geq f(\text{card}(A)).$$

Thus, the restriction to unary quantifiers is not essential and the result may be easily translated for type (1, 1) determiners.

What are the possible conclusions from Proposition 4.3.11? We have shown that not all strong reciprocal sentences are intractable. As long as a quantifier in the antecedent is bounded the procedure of checking the logical value of the sentence is practically computable. For example, the quantifiers Some and All are relativizations of the PTIME computable bounded quantifiers $\exists$ and $\forall$. Therefore, the following strong reciprocal sentences are tractable:

(17) Some members of parliament refer to each other indirectly.

(18) All members of parliament refer to each other indirectly.

4.4 Intermediate and Weak Lifts

Below we show that intermediate and weak reciprocal sentences — as opposed to strong reciprocal sentences — are tractable, if the determiners occurring in their antecedents are practically computable.

Analogous to the case of strong reciprocity, we can also express the meanings of intermediate and weak reciprocal lifts in graph-theoretical terms. We say that $\mathbb{M} \models \mathsf{Ram}_\mathsf{I}(\mathsf{Q})[A, R]$ if and only if there is a connected subgraph in A of a size bounded from below by the quantifier Q. $\mathbb{M} \models \mathsf{Ram}_\mathsf{W}(\mathsf{Q})[A, R]$ if and only if there is a subgraph in A of the proper size without isolated vertices. All three are with respect to the reciprocal relation R, either symmetric or asymmetric.

We prove that the class of PTIME quantifiers is closed under the (alternative) intermediate lift and the (alternative) weak lift.

4.4.1. Proposition. *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram_I}(\mathsf{Q})$ *is PTIME computable.*

Proof Let $\mathbb{G} = (V, A, E)$ be a directed colored graph-model. To check whether $\mathbb{G} \in \mathsf{Ram_I}(\mathsf{Q})$ compute all connected components of the subgraph determined by A. For example, you can use a breadth-first search algorithm that begins at some node and explores all the connected neighboring vertices. Then for each of those nearest nodes, it explores their unexplored connected neighbor vertices, and so on, until it finds the full connected subgraph. Next, it chooses a node which does not belong to this subgraph and starts searching for the connected subgraph containing it. Since in the worst case this breadth-first search has to go through all paths to all possible vertices, the time complexity of the breadth-first search on the whole $\mathbb{G}$ is $O(\text{card}(V) + \text{card}(E))$. Moreover, the number of the components in A is bounded by $\text{card}(A)$. Having all connected components it is enough to check whether there is a component C of the proper size, i.e., does $\mathsf{Q}[A, C]$ hold for some connected component C? This can be checked in polynomial time as Q is a PTIME computable quantifier. Hence, $\mathsf{Ram_I}(\mathsf{Q})$ is in PTIME. $\square$

4.4.2. Corollary. *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram_I}^{\vee}(\mathsf{Q})$ *is PTIME computable.*

The next proposition follows immediately.

4.4.3. Proposition. *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram_W}(\mathsf{Q})$ *is PTIME computable.*

Proof To check whether a given graph-model $\mathbb{G} = (V, A, E)$ is in $\mathsf{Ram_W}(\mathsf{Q})$, compute all connected components $C_1, \ldots, C_t$ of the A-subgraph. Take $X = C_1 \cup \ldots \cup C_t$ and check whether $\mathsf{Q}[A, X]$. From the assumption this can be done in polynomial time. Therefore, $\mathsf{Ram_W}(\mathsf{Q})$ is in PTIME. $\square$

4.4.4. Corollary. *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram_W}^{\vee}(\mathsf{Q})$ *is PTIME computable.*

These results show that the intermediate and weak reciprocal lifts do not increase the computational complexity of quantifier sentences in such a drastic way as may happen in the case of strong reciprocal lifts. In other words, in many contexts the intermediate and weak interpretations are relatively easy, as opposed to the strong reciprocal reading. For instance, the sentences (2), (3), (4), and (5) we discussed in the introduction are tractable. Hence from a computational complexity perspective the intermediate and reciprocal lifts behave similar to iteration, cumulation and resumption (discussed in Chapter 3).

In the next section we discuss the potential influence of computational complexity on the shifts in meaning of reciprocal sentences predicted by the Strong Meaning Hypothesis.

4.5 A Complexity Perspective on the SMH

Dalrymple et al. (1998) proposed a pragmatic principle, the Strong Meaning Hypothesis, to predict the proper reading of sentences containing reciprocal expressions. According to the SMH the reciprocal expression is interpreted as having the logically strongest truth conditions that are consistent with the given context. Therefore, if it is only consistent with the specified facts, a statement containing *each other* will be interpreted as a strong reciprocal sentence. Otherwise, the interpretation will shift toward the logically weaker intermediate or weak readings, depending on context (see Section 4.1.1).

The SMH is quite an effective pragmatic principle (see Dalrymple et al., 1998). We will discuss the shifts the SMH predicts from a computational complexity point of view, referring to the results provided in the previous sections.

Let us first think about the meaning of a sentence in the intensional way, identifying the meaning of an expression with an algorithm recognizing its denotation in a finite model.[2] Such algorithms can be described by investigating how language users evaluate the truth-value of sentences in various situations. On the cognitive level this means that subjects have to be equipped with mental devices to deal with the meanings of expressions. Moreover, it is cognitively plausible to assume that we have a single mental device to deal with most instances of the same semantic construction. For example, we believe that there is one mental algorithm to deal with the counting quantifier, At least k, in most possible contexts, no matter what natural number k is. Thus, in the case of logical expressions like quantifiers, the analogy between meanings and algorithms seems uncontroversial.

However, notice that some sentences, being intractable, are too complex to identify their truth-value directly by investigating a model. The experience of programming suggests that we can claim a sentence to be difficult when it cannot be computed in polynomial time. Despite the fact that some sentences are sometimes[3] too hard for comprehension, we can find their inferential relations

[2]We have argued for this approach in Chapter 1. Now we only recall that it goes back to Frege (1892) and exists in the linguistic literature at different levels of transparency (see e.g. Moschovakis, 2006).

[3]The fact that the general problem is hard does not show that all instances normally encountered are hard. It is the matter for empirical study to provide us with data about the influence of computational complexity on our everyday linguistic experience. However, we believe that it is reasonable to expect that this happens at least in some situations. We refer the reader to Section 1.5.3 for a more substantial discussion.

with relatively easier sentences. See Section 1.8 for more discussion on indirect verification.

According to the SMH any reciprocal sentence, if it is only possible, should be interpreted as a strong reciprocal sentence. We have shown that the strong interpretation of sentences with quantified antecedents is sometimes intractable but the intermediate and weak reading are always easy to comprehend. In other words, it is reasonable to suspect that in some linguistic situations the strong reciprocal interpretation is cognitively much more difficult than the intermediate or the weak interpretation. This prediction makes sense under the assumption that $P \neq NP$ and that the human mind is bounded by computational restrictions. We omit a discussion here but see Chapter 1. We only recall that computational restrictions for cognitive abilities are widely treated in the literature (see e.g. Cherniak, 1981; Chalmers, 1994; Mostowski and Wojtyniak, 2004; Levesque, 1988; Mostowski and Szymanik, 2005). Frixione (2001) explicitly formulates the so-called P-cognition Thesis:

P-cognition Thesis *Human cognitive (linguistic) capacities are constrained by polynomial time computability.*

What happens if a subject is supposed to deal with a sentence too hard for direct comprehension? One possibility — suggested in Section 1.8 — is that the subject will try to establish the truth-value of a sentence indirectly, by shifting to an accessible inferential meaning. That will be, depending on the context, the intermediate or the weak interpretation, both being entailed by the strong interpretation.

Summing up, our descriptive complexity perspective on reciprocity shows that it might not always be possible to interpret a reciprocal sentence in the strong way, as the SMH suggests. If the sentence in question would be intractable under the strong reciprocal interpretation then people will turn to tractable readings, like intermediate and weak reciprocity. Our observations give a cognitively reasonable argument for some shifts to occur, even though they are not predicted by the SMH. For example, the SMH assumes that the following sentence should be interpreted as a strong reciprocal statement.

(19) Most members of parliament refer to each other indirectly.

However, we know that this sentence is intractable. Therefore, if the set of parliament members is large enough then the statement is intractable under the strong interpretation. This gives a perfect reason to switch to weaker interpretations.

4.6 Summary

By investigating reciprocal expressions in a computational paradigm we found differences in computational complexity between various interpretations of reciprocal sentences with quantified antecedents. In particular, we have shown that:

- There exist non-branching natural language constructions whose semantics is intractable. For instance, strong reciprocal sentences with proportional quantifiers in the antecedent, e.g. sentence (14), are NP-complete.
- For PTIME computable quantifiers the intermediate and weak reciprocal interpretations (see e.g. sentences (2) and (3)) are PTIME computable.
- If we additionally assume that a quantifier is bounded, like Some and All, then also the strong reciprocal interpretation stays in PTIME, e.g. sentences (17) and (18).

Therefore, we argue that:

- The semantic distinctions of Dalrymple et al. (1998) seem solid from a computational complexity perspective.
- The Strong Meaning Hypothesis should be improved to account for shifts in meaning triggered by the computational complexity of sentences.

Many questions arise which are to be answered in future work. Here we will mention only a few of them:

4.6.1. Question. Among the reciprocal sentences we found NP-complete constructions. For example, we have shown that the strong reciprocal interpretations of proportional quantifiers are NP-complete. On the other hand, we also proved that the strong reciprocal interpretations of bounded quantifiers are PTIME computable. It is an open problem where the precise border is between those natural language quantifiers for which Ramseyfication is in PTIME and those for which it is NP-complete. Is it the case that for every quantifier, Q, $\mathsf{Ram}_{\mathsf{S}}(\mathsf{Q})$ is either PTIME computable or NP-complete? We stated this question already in the previous chapter.

4.6.2. Question. There is a vast literature on the definability of polyadic lifts of generalized quantifiers (e.g. Väänänen, 1997b; Hella et al., 1997). We introduced some new linguistically relevant lifts, the weak and intermediate reciprocal lifts. The next step is to study their definability. For example, we would like to know how the definability questions for $\mathsf{Ram}_{\mathsf{S}}(\mathsf{Q}_f)$, $\mathsf{Ram}_{\mathsf{I}}(\mathsf{Q}_f)$, and $\mathsf{Ram}_{\mathsf{W}}(\mathsf{Q}_f)$ depend on the properties of f. Another interesting point is to link our operators with other polyadic lifts, like branching.

4.6.3. Question. We could empirically compare the differences in shifts from the strong interpretation of reciprocal sentences with bounded and proportional quantifiers in antecedents. Our approach predicts that subjects will shift to easier interpretations more frequently in the case of sentences with proportional quantifiers. Can we prove it empirically?

Chapter 5

Complexity of Collective Quantification

Most of the efforts in generalized quantifier theory focus on distributive readings of natural language determiners. In contrast — as properties of plural objects are becoming more and more important in many areas (e.g. in game-theoretical investigations, where groups of agents act in concert) — this chapter is devoted to collective readings of quantifiers. We focus mainly on definability issues, but we also discuss some connections with computational complexity.

For many years the common strategy in formalizing collective quantification has been to define the meanings of collective determiners, quantifying over collections, using certain type-shifting operations. These type-shifting operations, i.e., lifts, define the collective interpretations of determiners systematically from the standard meanings of quantifiers. We discuss the existential modifier, neutral modifier, and determiner fitting operators as examples of collective lifts considered in the literature. Then we show that all these lifts turn out to be definable in second-order logic.

Next, we turn to a discussion of so-called second order generalized quantifiers — an extension of Lindström quantifiers to second-order structures. We show possible applications of these quantifiers in capturing the semantics of collective determiners in natural language. We also observe that using second-order generalized quantifiers is an alternative to the type-shifting approach.

Then we study the collective reading of the proportional quantifier "most". We define a second-order generalized quantifier corresponding to this reading and show that it is probably not definable in second-order logic. If it were definable then the polynomial hierarchy in computational complexity theory would collapse; this is very unlikely and commonly believed to be false, although no proof is known.

Therefore, probably there is no second-order definable lift expressing the collective meaning of the quantifier "most". This is clearly a restriction of the type-shifting approach. One possible alternative would be to use second-order generalized quantifiers in the study of collective semantics. However, we notice that the computational complexity of such approach is excessive and hence it is not a plausible model (according to the Σ^1_1-thesis formulated in Section 1.8) of collective quantification in natural language. Hence, we suggest to turn in the direction of another well-known way of studying collective quantification, the many-sorted (algebraic) tradition. This tradition seems to overcome the weak points of the higher-order approach.

Another interpretation of our results might be that computational complexity restricts the expressive power of everyday language (see discussion in Section 1.8). Namely, even though natural language can in principle realize collective proportional quantifiers its everyday fragment does not contain such constructions due to their high complexity.

The main observations of this chapter are the result of joint work with Juha Kontinen (see Kontinen and Szymanik, 2008).

5.1 Collective Quantifiers

5.1.1 Collective Readings in Natural Language

Already Bertrand Russell (1903) noticed that natural language contains quantification not only over objects, but also over collections of objects. The notion of a collective reading is a semantic one — as opposed to the grammatical notion of plurality — and it applies to the meanings of certain occurrences of plural noun phrases. The phenomenon is illustrated by the following sentences, discussed in letters between Frege and Russel in 1902 (see Frege, 1980):

(1) Bunsen and Kirchoff laid the foundations of spectral theory.

(2) The Romans conquered Gaul.

Sentence (1) does not say that Bunsen laid the foundations of spectral theory and that Kirchoff did also, even though they both contributed. It rather says that they did it together, that spectral theory was a result of a group (team) activity by Kirchoff and Bunsen. Similarly, sentence (2) claims that the Romans conquered Gaul together. For a contrast in meaning compare sentence (1) to (3) and sentence (2) to (4).

(3) Armstrong and Aldrin walked on the moon.

(4) Six hundred tourists visited the Colosseum.

Sentence (3) says that Armstrong walked on the moon and Aldrin walked on the moon. Similarly, sentence (4) is still true if six hundred tourists visited the Colosseum separately.

Notice that not all plural noun phrases are read collectively. Let us consider the following sentences:

(5) Some boys like to sing.

(6) All boys like to sing.

(7) $\exists^{\geq 2}x[\mathrm{Boy}(x) \wedge \mathrm{Sing}(x)]$.

(8) $\forall x[\mathrm{Boy}(x) \implies \mathrm{Sing}(x)]$.

The interpretations of sentences (5)–(6) can be expressed using standard first-order distributive quantifiers, by formulae (7) and (8) respectively.

Therefore, linguistic theory should determine what triggers collective readings. In fact this is to a large degree determined by a context. Noun phrases which can be read collectively may also be read distributively. Compare sentence (1) with proposition (9):

(1) Bunsen and Kirchoff laid the foundations of spectral theory.

(9) Cocke, Younger and Kasami discovered (independently) the algorithm which determines whether a string can be generated by a given context-free grammar.

However, there are so-called collective properties which entail some sort of collective reading; for example the phrases emphasized in the following sentences usually trigger a collective reading:

(10) All the Knights but King Arthur *met in secret.*

(11) Most climbers *are friends.*

(12) John and Mary *love each other.*

(13) The samurai *were twelve in number.*

(14) Many girls *gathered.*

(15) Soldiers *surrounded* the Alamo.

(16) Tikitu and Samson *lifted* the table.

5.1.2 Modelling Collectivity

The Algebraic Approach

When we have decided that a noun phrase has a collective reading the question arises how we should model collective quantification in formal semantics. Many authors have proposed different mathematical accounts of collectivity in language (see Lønning, 1997, for an overview and references). According to many, a good semantic theory for collectives should obey the following intuitive principles:

Atomicity Each collection is constituted by all the individuals it contains.

Completeness Collections may be combined into new collections.

Atoms Individuals are collections consisting of only a single member.

Among structures satisfying these requirements are many-sorted (algebraic) models, e.g. complete atomic join semilattices (see Link, 1983). The idea — often attributed to Frege and Leśniewski (see e.g. Lønning, 1997, pages 1028–1029) — is roughly to replace the domain of discourse, which consists of entities, with a structure containing also collections of entities. The intuition lying behind this way of thinking is as follows.

Consider the following question and two possible answers to it:

(17) Who played the game?

(18) John did.

(19) The girls did.

Both answers, (18) and (19), are possible. It suggests that a plural noun phrase like "the girls" should denote an object of the same type as "John", and a verb phrase like "played the game" should have one denotation which includes both individuals and collections. The algebraic approach to collectives satisfies this intuition. One of the advantage of this algebraic perspective is that it unifies the view on collective predication and predication involving mass nouns (see e.g. Lønning, 1997, Chapter 4.6).

Algebraic models come with formal languages, like many sorted first-order logic, i.e. a first-order language for plurals. The first sort corresponds to entities and the second one to collections. Such logics usually contain a pluralization operator turning individuals into collections (see e.g. Lønning, 1997, for more details). A similar approach is also adopted in Discourse Representation Theory to account not only for the meaning of collective quantification but also for anaphoric links (see Kamp and Reyle, 1993, Chapter 4).

The Higher-order Approach

From the extensional perspective all that can be modeled within algebraic models can be done in type theory as well (see e.g. van der Does, 1992). This tradition, starting with the works of Bartsch (1973) and Bennett (1974), uses extensional type-theory with two basic types: e (entities) and t (truth values), and compound types: $\alpha\beta$ (functions mapping type α objects onto type β objects). Together with the idea of type-shifting, introduced independently by van Benthem (1983) (see also van Benthem, 1995) and Partee and Rooth (1983), this gives a way to model collectivity in natural language. The strategy — introduced by Scha (1981) and later advocated and developed by van der Does (1992, 1993) and Winter (2001) — is to lift first-order generalized quantifiers to a second-order setting. In type-theoretical terms the trick is to shift determiners of type $((et)((et)t))$ (corresponding to relations between sets), related to the distributive readings of quantifiers, into determiners of type $((et)(((et)t)t))$ (relations between sets and collections of sets) which can be used to formalize the collective readings of quantifiers.

In the next section we describe the type-shifting approach to collectivity in a more systematic and detailed way. Then we introduce second-order generalized quantifiers, and show that the type theoretic approach can be redefined in terms of second-order generalized quantifiers. The idea of type-shifting turns out to be very closely related to the notion of definability which is central in generalized quantifier theory.

5.2 Lifting First-order Determiners

5.2.1 Existential Modifier

Let us consider the following sentences involving collective quantification:

(20) At least five people lifted the table.

(21) Some students played poker together.

(22) All combinations of cards are losing in some situations.

The distributive reading of sentence (20) claims that the total number of students who lifted the table on their own is at least five. This statement can be formalized in elementary logic by formula (23):

(23) $\exists^{\geq 5}x[\text{People}(x) \wedge \text{Lift-the-table}(x)]$.

The collective interpretation of sentence (20) claims that there was a collection of at least five students who jointly lifted the table. This can be formalized by shifting formula (23) to the second-order formula (24), where the predicate "Lift" has been shifted from individuals to sets:

(24) $\exists X[\text{Card}(X) = 5 \wedge X \subseteq \text{People} \wedge \text{Lift-the-table}(X)]$.

In a similar way, by lifting the corresponding first-order determiners, we can express the collective readings of sentences (21)–(22) as follows:

(25) $\exists X[X \subseteq \text{Students} \wedge \text{Play-poker}(X)]$.

(26) $\forall X[X \subseteq \text{Cards} \implies \text{Lose-in-some-situation}(X)]$.

All the above examples can be described in terms of a uniform procedure of turning a determiner of type $((et)((et)t))$ into a determiner of type $((et)(((et)t)t))$ by means of a type-shifting operator introduced by van der Does (1992) and called the existential modifier, $(\cdot)^{EM}$.

5.2.1. Definition. Let us fix a universe of discourse U and take any $X \subseteq U$ and $Y \subseteq \mathcal{P}(U)$. Define the *existential lift*, Q^{EM}, of a type (1, 1) quantifier Q in the following way:

$$\mathsf{Q}^{EM}[X, Y] \text{ is true } \iff \exists Z \subseteq X[\mathsf{Q}(X, Z) \wedge Z \in Y].$$

■

One can observe that the collective readings of sentences (20)–(22) discussed above agree with the interpretation predicted by the existential modifier.

We can now ask about the monotonicity of collective quantification defined via the existential lift. First of all, notice that the existential lift works properly only for right monotone increasing quantifiers. For instance, the sentence:

(27) No students met yesterday at the coffee shop.

with the ↓MON↓ quantifier No gets a strange interpretation under the existential modifier.

The existential modifier predicts that this sentence is true if and only if the empty set of students met yesterday at the coffee shop, which is clearly not what sentence (27) claims. This is because No^{EM} is ↑MON↑ but the collective interpretation of No should remain ↓MON↓, as sentence (27) entails both (28) and (29):

(28) No left-wing students met yesterday at the coffee shop.

(29) No students met yesterday at the "Che Guevara" coffee shop.

This is the so-called van Benthem problem for plural quantification (see van Benthem, 1986, pages 52–53): any general existential lift, like $(\cdot)^{EM}$, will be problematic as it turns any $((et)((et)t))$ determiner into a $((et)(((et)t)t))$ determiner that is upward monotone in the right argument. Obviously, this is problematic with non-upward monotone determiners.

5.2.2. Example. Consider the following sentence with a non-monotone quantifier and the reading obtained by applying the existential lift:

(30) Exactly 5 students drank a whole keg of beer together.

(31) $(\exists^{=5})^{EM}[\text{Student}, \text{Drink-a-whole-keg-of-beer}]$.

Formula (31) is true if and only if the following holds:

$$\exists A \subseteq \text{Student}[\text{card}(A) = 5 \wedge \text{Drink-a-whole-keg-of-beer}(A)].$$

This would yield truth as the logical value of sentence (30) even if there were actually six students drinking a keg of beer together. Therefore, it fails to take into account the total number of students who drank a keg of beer.

5.2.2 The Neutral Modifier

Aiming to solve problems with the collective reading of downward monotone quantifiers, like in sentence (27), van der Does (1992) proposed the so-called neutral modifier, $(\cdot)^N$.

5.2.3. Definition. Let U be a universe, $X \subseteq U$, $Y \subseteq \mathcal{P}(U)$, and Q a type (1, 1) quantifier. We define the *neutral modifier*:

$$\mathsf{Q}^N[X, Y] \text{ is true} \iff \mathsf{Q}\big[X, \bigcup(Y \cap \mathcal{P}(X))\big].$$

■

The neutral modifier can easily account for sentences with downward monotone quantifiers, like proposition (27). But what about non-monotone quantifiers, for example sentence (30)? Now we can express its meaning in the following way:

(32) $(\exists^{=5})^N[\text{Student}, \text{Drink-a-whole-keg-of-beer}]$.

This analysis requires that the total number of students in sets of students that drank a keg of beer together is five. Formula (32) is true whenever:

$$\text{card}\big(\{x | \exists A \subseteq \text{Student}[x \in A \wedge \text{Drink-a-whole-keg-of-beer}(A)]\}\big) = 5.$$

However, it does not require that there was one set of five students who drank a whole keg of beer together: in a situation where there were two groups of students, containing three and two members, sentence (30) would be true according to formula (32). Again, this is not something we would expect, because the collective reading triggered by "together" suggests that we are talking about one group of students.

In general, the following is true about monotonicity preservation under the neutral lift (see Ben-Avi and Winter, 2003, Fact 7):

5.2.4. Fact. Let Q be a distributive determiner. If Q belongs to one of the classes $\uparrow$MON$\uparrow$, $\downarrow$MON$\downarrow$, MON$\uparrow$, MON$\downarrow$, then the collective determiner Q^N belongs to the same class. Moreover, if Q is conservative and $\sim$MON (MON$\sim$), then Q^N is also $\sim$MON (MON$\sim$).

5.2.3 The Determiner Fitting Operator

To overcome problems with non-monotone quantifiers Winter (2001) combined the existential and the neutral modifiers into one type-shifting operator called dfit, abbreviating determiner fitting. The $(\cdot)^{dfit}$ operator turns a determiner of type $((et)((et)t))$ into a determiner of type $(((et)t)(((et)t)t))$.

5.2.5. Definition. For all $X, Y \subseteq \mathcal{P}(U)$ and a type (1, 1) quantifier Q we define the *determiner fitting operator*:

$$\mathsf{Q}^{dfit}[X, Y] \text{ is true}$$

$$\Longleftrightarrow$$

$$\mathsf{Q}\Big[\bigcup X, \bigcup (X \cap Y)\Big] \wedge \Big[X \cap Y = \emptyset \vee \exists W \in (X \cap Y)\, \mathsf{Q}\Big(\bigcup X, W\Big)\Big].$$

■

Using dfit we get the following interpretation of sentence (30):

(33) $(\exists^{=5})^{dfit}[\text{Student}, \text{Drink-a-whole-keg-of-beer}]$.

Formula (33) assigns a satisfactory meaning to sentence (30). It says that exactly five students participated in sets of students drinking a whole keg of beer together and moreover that there was a set of 5 students who drank a keg of beer together. It is true if and only if:

$$\begin{aligned}\text{card}\big(\{x \in A | A \subseteq \text{Student} \wedge \text{Drink-a-whole-keg-of-beer}(A)\}\big) = 5 \\ \wedge\, \exists W \subseteq \text{Student}[\text{Drink-a-whole-keg-of-beer}(W) \wedge \text{card}(W) = 5].\end{aligned}$$

Moreover, notice that the determiner fitting operator will assign the proper meaning also to downward and upward monotone sentences, like:

(27) No students met yesterday at the coffee shop.

(34) Less than 5 students ate pizza together.

(35) More than 5 students ate pizza together.

For sentence (27) the determiner fitting operator does not predict that the empty set of students met at the coffee shop as the existential modifier does. It simply does not demand existence of a witness set in cases when the intersection of arguments is empty. For the downward monotone sentence (34) the first conjunct of the determiner fitting lift counts the number of students in the appropriate collections and guarantees that they contain not more than five students. In the case of the upward monotone sentence (35) the second conjunct of dfit claims existence of the witness set. Table 5.1 adapted from (Ben-Avi and Winter, 2004) sums up the monotonicity behavior of determiner fitting.

Monotonicity of Q	Monotonicity of Q^{dfit}	Example
$\uparrow$MON$\uparrow$	$\uparrow$MON$\uparrow$	Some
$\downarrow$MON$\downarrow$	$\downarrow$MON$\downarrow$	Less than five
$\downarrow$MON$\uparrow$	$\sim$MON$\uparrow$	All
$\uparrow$MON$\downarrow$	$\sim$MON$\downarrow$	Not all
$\sim$MON$\sim$	$\sim$MON$\sim$	Exactly five
$\sim$MON$\downarrow$	$\sim$MON$\downarrow$	Not all and less than five
$\sim$MON$\uparrow$	$\sim$MON$\uparrow$	Most
$\downarrow$MON$\sim$	$\sim$MON$\sim$	All or less than five
$\uparrow$MON$\sim$	$\sim$MON$\sim$	Some but not all

Table 5.1: Monotonicity under the determiner fitting operator.

5.2.4 A Note on Collective Invariance Properties

We have briefly discussed the monotonicity properties of every lift (for more details see Ben-Avi and Winter, 2003, 2004). What about other invariance properties (see Section 2.2.5) in the collective setting?

Let us for example consider conservativity. Recall from Section 2.2.5 that a distributive determiner of type (1, 1) is conservative if and only if the following holds for all M and all $A, B \subseteq M$:

$$\mathsf{Q}_M[A, B] \iff Q_M[A, A \cap B].$$

In that sense every collective quantifier Q^{EM} trivially does not satisfy conservativity, as for every X, Y the intersection $X \cap \mathcal{P}(Y) = \emptyset$. Therefore, for every Z $Z \notin X \cap \mathcal{P}(Y)$.

In the case of the neutral lift and determiner fitting we can conclude the same because of a similar difficulty.

5.2.6. Fact. For every Q the collective quantifiers Q^{EM}, Q^{N}, and Q^{dfit} are not conservative.

The failure of this classical invariance is for technical reasons but still we feel that in the intuitive sense the collective quantifiers defined by these lifts satisfy conservativity. To account for this intuition let us simply reformulate the conservativity property in the collective setting (see also Chapter 5 in van der Does, 1992).

5.2.7. Definition. We say that a collective determiner Q of type $((et)(((et)t)t))$ satisfies *collective conservativity* iff the following holds for all M and all $A, B \subseteq M$:

$$\mathsf{Q}_M[A, B] \iff Q_M[A, \mathcal{P}(A) \cap B].$$

■

Do lifted quantifiers satisfy collective conservativity? The following fact gives a positive answer to this question.

5.2.8. FACT. For every Q the collective quantifiers Q^{EM}, Q^{N}, and Q^{dfit} satisfy collective conservativity.

Notice that collective conservativity is satisfied by our lifts no matter whether the distributed quantifier itself satisfies it. Therefore, it is fair to say that conservativity is incorporated into these lifts. We personally doubt that this is a desirable property of collective modelling.

Below we introduce an alternative method of grasping collectivity by means of extending Lindström quantifiers to second-order structures. Among other things our approach does not arbitrarily decide the invariance properties of collective determiners.

5.3 Second-order Generalized Quantifiers

Second-order generalized quantifiers (SOGQs) were first defined and applied in the context of descriptive complexity theory by Burtschick and Vollmer (1998). The general notion of a second-order generalized quantifier was later formulated by Andersson (2002). The following definition of second-order generalized quantifiers is a straightforward generalization from the first-order case (see Definition 2.2.1). However, note that the types of second-order generalized quantifiers are more complicated than the types of first-order generalized quantifiers, since predicate variables can have different arities. Let $t = (s_1, \ldots, s_w)$, where $s_i = (\ell_1^i, \ldots, \ell_{r_i}^i)$, be a tuple of tuples of positive integers. A *second order structure* of type t is a structure of the form $(M, P_1, \ldots, P_w)$, where $P_i \subseteq \mathcal{P}(M^{\ell_1^i}) \times \cdots \times \mathcal{P}(M^{\ell_{r_i}^i})$. Below, we write $f[A]$ for the image of A under the function f.

5.3.1. DEFINITION. A *second-order generalized quantifier* Q of type t is a class of structures of type t such that Q is closed under isomorphisms: If $(M, P_1, \ldots, P_w) \in$ Q and $f\colon M \to N$ is a bijection such that $S_i = \{(f[A_1], \ldots, f[A_{r_i}]) \mid (A_1, \ldots, A_{r_i}) \in P_i\}$, for $1 \leq i \leq w$, then $(N, S_1, \ldots, S_w) \in$ Q. ■

5.3.2. CONVENTION. In what follows, second-order quantifiers are denoted Q, whereas first-order quantifiers are denoted Q.

5.3.3. EXAMPLE. The following examples show that second-order generalized quantifiers are a natural extension of the first-order case. While Lindström quantifiers are classes of first-order structures (a universe and its subsets), second-order generalized quantifiers are classes of second-order structures consisting not only of a universe and its subsets, but also of collections of these subsets.

$$\begin{array}{rcl}
\exists^2 & = & \{(M, P) \mid P \subseteq \mathcal{P}(M) \text{ and } P \neq \emptyset\}. \\
\textsc{Even} & = & \{(M, P) \mid P \subseteq \mathcal{P}(M) \text{ and } \operatorname{card}(P) \text{ is even}\}. \\
\textsc{Even}' & = & \{(M, P) \mid P \subseteq \mathcal{P}(M) \text{ and } \forall X \in P(\operatorname{card}(X) \text{ is even})\}. \\
\textsc{Most} & = & \{(M, P, S) \mid P, S \subseteq \mathcal{P}(M) \text{ and } \operatorname{card}(P \cap S) > \operatorname{card}(P - S)\}.
\end{array}$$

The first quantifier is the unary second-order existential quantifier. The type of $\exists^2$ is $((1))$, i.e., it applies to one formula binding one unary second-order variable. The type of the quantifier Even is also $((1))$ and it expresses that a formula holds for an even number of subsets of the universe. On the other hand, the quantifier Even′ expresses that all the subsets satisfying a formula have an even number of elements. The type of the quantifier Most is $((1), (1))$ and it is the second-order analogue of the quantifier Most.

Examples of linguistic applications of second-order generalized quantifiers are given in the next section. However, already now we can notice that one can think about second-order generalized quantifiers as relations between collections of subsets of some fixed universe. Therefore, also from the descriptive perspective taken in linguistics the notion of second-order generalized quantifiers is a straightforward generalization of Lindström quantifiers.

5.3.4. Convention. Throughout the text we will write Most, Even, Some for Lindström quantifiers and Most, Even, Some for the corresponding second-order generalized quantifiers.

5.3.1 Definability for SOGQs

5.3.5. Definition. As in the first-order case, we define the extension, FO(Q), of FO by a second-order generalized quantifier Q of type $t = (s_1, \ldots, s_w)$, where $s_i = (\ell^i_1, \ldots, \ell^i_{r_i})$, in the following way:

- Second order variables are introduced to the FO language.
- The formula formation rules of FO-language are extended by the rule:

  if for $1 \leq i \leq w$, $\varphi_i(\overline{X_i})$ is a formula and $\overline{X_i} = (X_{1,i}, \ldots, X_{r_i,i})$ is a tuple of pairwise distinct predicate variables, such that $\operatorname{arity}(X_{j,i}) = \ell^i_j$, for $1 \leq j \leq r_i$, then

  $$\mathrm{Q}\overline{X}_1, \ldots, \overline{X}_w \left[\varphi_1(\overline{X}_1), \ldots, \varphi_w(\overline{X}_w)\right]$$

  is a formula.

- The satisfaction relation of FO is extended by the rule:

$$\mathbb{M} \models \mathrm{Q}\overline{X}_1, \ldots, \overline{X}_w \left[\varphi_1, \ldots, \varphi_w\right] \text{ iff } (M, \varphi_1^{\mathbb{M}}, \ldots, \varphi_w^{\mathbb{M}}) \in \mathrm{Q},$$

where $\varphi_i^{\mathbb{M}} = \left\{\overline{R} \in \mathcal{P}(M^{\ell_1^i}) \times \cdots \times \mathcal{P}(M^{\ell_{r_i}^i}) \mid \mathbb{M} \models \varphi_i(\overline{R})\right\}$.

■

The notion of definability for second-order generalized quantifiers can be formulated as in the case of Lindström quantifiers (see Definition 2.2.10; see also Kontinen (2004) for technical details). However, things are not completely analogous to the first-order case. With second-order generalized quantifiers the equivalence of two logics $\mathcal{L}(\mathrm{Q}) \equiv \mathcal{L}$ does not imply that the quantifier Q is definable in the logic $\mathcal{L}$ (see the next paragraph for an example). The converse implication is still valid.

5.3.6. Proposition (Kontinen (2004)). *Let* Q *be a second-order generalized quantifier and* $\mathcal{L}$ *a logic. If the quantifier* Q *is definable in* $\mathcal{L}$ *then*

$$\mathcal{L}(\mathrm{Q}) \equiv \mathcal{L}.$$

Proof The idea and the proof is analogous to the first-order case (see proof of the Proposition 2.2.12). Here we substitute second-order predicates by formulae having free second-order variables. □

Kontinen (2002) has shown that the extension $\mathcal{L}^*$ of first-order logic by all Lindström quantifiers cannot define the monadic second-order existential quantifier, $\exists^2$. In other words, the logic $\mathcal{L}^*$, in which all properties of first-order structures can be defined, cannot express in a uniform way that a collection of subsets of the universe is non-empty. This result also explains why we cannot add the second implication to the previous proposition. Namely, even though $\mathcal{L}^* \equiv \mathcal{L}^*(\exists^2)$ the quantifier $\exists^2$ is not definable in $\mathcal{L}^*$.

Moreover, this observation can be used to argue for the fact that first-order generalized quantifiers alone are not adequate for formalizing all natural language quantification. For example, as the quantifier $\exists^2$ is not definable in $\mathcal{L}^*$, the logic $\mathcal{L}^*$ cannot express the collective reading of sentences like:

(36) Some students gathered to play poker.

Therefore, we need to extend logic beyond $\mathcal{L}^*$ to capture the semantics of collective quantifiers in natural language.

Last but not least, let us notice that natural invariance properties for SOGQs have not been investigated. Studying invariance properties in the context of classical generalized quantifier theory led to many definability results of mathematical and linguistic value. In the case of SOGQs these questions are still waiting for systematic research.

In the next section we will show how to model collectivity by adding second-order generalized quantifiers to elementary logic.

5.4 Defining Collective Determiners by SOGQs

In this section we show that collective determiners can be easily identified with certain second-order generalized quantifiers. Thereby, we explain the notion of second-order generalized quantifiers a little bit more — this time with linguistic examples. We also observe that the second-order generalized quantifiers corresponding to lifted first-order determiners are definable in second-order logic.

Recall that the determiner fitting operator turns a first-order quantifier of type (1, 1) directly into a second-order quantifier of type $((1),(1))$. Nevertheless, at first sight, there seems to be a problem with identifying collective determiners with second-order generalized quantifiers, as the existential and neutral modifiers produce collective noun phrases of a mixed type $((et)(((et)t)t))$, while our Definition 5.3.1 talks only about quantifiers of uniform types. However, this is not a problem since it is straightforward to extend the definition to allow also quantifiers with mixed types. Below we define examples of second-order generalized quantifiers of mixed types which formalize collective determiners in natural language.

5.4.1. Definition. Denote by Some^{EM} the following quantifier of type $(1,(1))$

$$\{(M,P,G) \mid P \subseteq M;\ G \subseteq \mathcal{P}(M):\ \exists Y \subseteq P(Y \neq \emptyset \text{ and } P \in G)\}.$$

■

Obviously, we can now express the collective meaning of sentence (21), repeated here as sentence (37), by formula (38).

(37) Some students played poker together.

(38) $\mathsf{Some}^{EM}\, x, X[\text{Student}(x), \text{Played-poker}(X)]$.

Analogously, we can define the corresponding second-order quantifier appearing in sentence (20), here as (39).

(39) At least five people lifted the table.

5.4.2. Definition. We take five^{EM} to be the second order-quantifier of type $(1,(1))$ denoting the class:

$$\{(M,P,G) \mid P \subseteq M;\ G \subseteq \mathcal{P}(M):\ \exists Y \subseteq P(\text{card}(Y) = 5 \text{ and } P \in G)\}.$$

■

Now we can formalize the collective meaning of (39) by:

(40) $\mathsf{five}^{EM} x, X[\text{Student}(x), \text{Lifted-the-table}(X)]$.

These simple examples show that it is straightforward to associate a mixed second-order generalized quantifier with every lifted determiner. Also, it is easy to see that for any first-order quantifier Q the lifted second-order quantifiers Q^{EM}, Q^{dfit} and Q^{N} can be uniformly expressed in second-order logic assuming the quantifier Q is also available. In fact, all the lifts discussed in Section 5.2, and, as far as we know, all those proposed in the literature, are definable in second-order logic. This observation can be stated as follows.

5.4.3. Theorem. *Let Q be a Lindström quantifier definable in second-order logic. Then the second-order quantifiers Q^{EM}, Q^{dfit} and Q^{N} are definable in second-order logic, too.*

Proof Let us consider the case of Q^{EM}. Let $\psi(x)$ and $\phi(Y)$ be formulae. We want to express $\mathsf{Q}^{EM} x, Y[\psi(x), \phi(Y)]$ in second-order logic. By the assumption, the quantifier Q can be defined by some sentence $\theta \in \text{SO}[\{P_1, P_2\}]$. We can now use the following formula:

$$\exists Z[\forall x(Z(x) \implies \psi(x)) \wedge (\theta(P_1/\psi(x), P_2/Z) \wedge \phi(Y/Z)].$$

The other lifts can be defined analogously. □

For example, $\mathsf{Some}^{EM}, \mathsf{Most}^{N}, \mathsf{All}^{\text{dfit}}$ are all definable in second-order logic.

Let us notice that the above theorem can be easily generalized to cover not only the three lifts we've discussed but all possible collective operators definable in second-order logic. Namely, using the same idea for the proof we can show the following:

5.4.4. Theorem. *Let us assume that the lift $(\cdot)^*$ and a Lindström quantifier Q are both definable in second-order logic. Then the second-order generalized quantifier Q^* is also definable in second-order logic.*

These theorems show that in the case of natural language determiners — which are obviously definable in second-order logic — the type-shifting strategy cannot take us outside second-order logic. In the next section we show that it is very unlikely that all collective determiners in natural language can be defined in second-order logic. Our argument is based on the close connection between second-order generalized quantifiers and certain complexity classes in computational complexity theory.

5.5 Collective Majority

5.5.1 An Undefinability Result for SOGQ "MOST"

Consider the following sentences:

(41) In the preflop most poker hands have no chance against an Ace and a Two.

(42) Most of the PhD students played Hold'em together.

These sentences can be read collectively. For example, sentence (42) can be formalized using the second-order generalized quantifier MOST (defined in Example 5.3.3) by the following formula:

(43) $\textsc{Most}\, X, Y[\text{PhD-Students}(X), \text{Played-Hold'em}(Y)]$.

Above we assume that the predicates PhD-Students and Play-Hold'em are interpreted as collections of sets of atomic entities of the universe. Obviously, this is just one possible way of interpreting sentence (42). In general, we are aware that when it comes to proportional determiners, like "most", it seems difficult to find contexts where they are definitely read collectively (see e.g. Lønning, 1997, p. 1016). On the other hand, we can not totally exclude the possibility that sentences (41)–(42) can be used in a setting where only a collective reading is possible. Anyway, it seems that MOST is needed in the formalization, assuming that PhD-Students and Play-Hold'em are interpreted as collective predicates.

For the sake of argument, let us assume that our interpretation of sentence (42) is correct. We claim that the lifts discussed above do not give the intended meaning when applied to the first-order quantifier Most. Using them we could only obtain a reading saying something like "there was a group of students, containing most of the students, such that students from that group played Hold'em together". But what we are trying to account for is the meaning where both arguments are read collectively and which can be expressed as follows "most groups of students played Hold'em together".

We shall next show that it is unlikely that *any* lift which can be defined in second-order logic can do the job. More precisely, we show (Theorem 5.5.1 below) that if the quantifier MOST can be lifted from the first-order Most using a lift which is definable in second-order logic then something unexpected happens in computational complexity. This result indicates that the type-shifting strategy used to define collective determiners in the literature is probably not general enough to cover all collective quantification in natural language.

Let us start by discussing the complexity theoretic side of our argument. Recall that second-order logic corresponds in complexity theory to the polynomial hierarchy, PH, (see Theorem 2.4.5 in the Prerequisites chapter, and Section 2.3.5).

The polynomial hierarchy is an oracle hierarchy with NP as the building block. If we replace NP by probabilistic polynomial time (PP) in the definition of PH, then we arrive at a class called the counting hierarchy, CH, (see Section 2.3.6).

Now, we can turn to the theorem which is fundamental for our argumentation.

5.5.1. Theorem. *If the quantifier* Most *is definable in second-order logic, then* CH = PH *and* CH *collapses to its second level.*

Proof The proof is based on the observation in Kontinen and Niemistö (2006) that already the extension of first-order logic by the unary second-order majority quantifier, Most^I, of type ((1)), can define complete problems for each level of the counting hierarchy. The unary second-order majority quantifier is easily definable in terms of the quantifier Most:

$$\text{Most}^I[X]\psi \iff \text{Most}\, X, X[X = X, \psi].$$

Hence, the logic FO(Most) can define complete problems for each level of the counting hierarchy. On the other hand, if the quantifier Most were definable in second-order logic, then by Proposition 5.3.6 we would have that FO(Most) $\leq$ SO and therefore SO would contain complete problems for each level of the counting hierarchy. This would imply that CH = PH and furthermore that CH $\subseteq$ PH $\subseteq C_2P$ (see Toda, 1991). □

5.5.2. Corollary. *The type-shifting strategy is probably not general enough to cover all collective quantification in natural language.*

The following conjecture is a natural consequence of the theorem.

5.5.3. Conjecture. *The quantifier* Most *is not definable in second-order logic.*

5.5.2 Consequences of Undefinability

Does SO Capture Natural Language?

Therefore, it is very likely that second-order logic is not expressive enough to capture natural language semantics. Recall that apart from second-order logic, collective quantification in natural language is also not expressible in $\mathcal{L}^*$ — first-order logic enriched by all Lindström quantifiers (see Kontinen, 2002, and Section 5.3.1). Then we have to look for alternative tools. As we have shown, the natural extension of first-order generalized quantifiers beyond elementary structures leads to the notion of second-order generalized quantifiers. We have outlined how one can account for collectivity using second-order generalized quantifiers. The question arises what kind of insight into language can be obtained by introducing second-order generalized quantifiers into semantics. For example, are there any new interesting generalizations or simplifications available in the theory? We can already notice that the van Benthem problem would disappear simply because we would not try to describe semantics of collective quantifiers in terms of some uniform operation but we would rather define separate second-order generalized quantifiers corresponding to their first-order counterparts. Moreover, with SOGQs we do not restrict ourselves to interpretations with already fixed invariance properties (see Section 5.2.4). We can enjoy the same freedom we have in distributive generalized quantifier theory.

Are Many-sorted Models More Plausible?

Theorem 5.5.1 also shows that the computational complexity of some collective sentences can be enormous when analyzed via second-order logic. In other words, such an approach to collective quantification violates the methodological Σ_1^1-thesis from Section 1.8. However, notice that all these claims are valid only if we restrict a semantic theory to universes containing nothing more than entities. But when we are dealing with collective sentences in our everyday communication we rather tend to interpret them in the universe which contains groups of people and combinations of cards as well as people and cards themselves. Theorem 5.5.1 explains this in terms of complexity — it would be too complicated to understand language, thinking about the world as containing only individual objects. From this point of view many-sorted approaches to natural language semantics seem to be closer to linguistic reality and our observation can be treated as an argument in favor of them. In the light of Theorem 5.5.1 we are inclined to believe that this approach is much more plausible than higher-order approaches. It would be interesting to design psychological experiments throwing light on the mental representation of collective quantification. We conjecture that the results of such experiments will show that subjects use some kind of many-sorted logical representation to comprehend collective interpretations of sentences. Experiments can also help to identify gaps in the semantic theory of collectives and motivate and direct the research effort to fill them in.

Does SOGQ "MOST" Belong to Everyday Language?

Last but not least, let us give an alternative interpretation of our result. As we mentioned, the collective meaning of proportional quantifiers like "most" in natural language is marginal at best. It is not completely clear that one can find situations where sentences like (42) have to be read collectively in the suggested way. It might be the case that everyday language does not realize proportional collective quantification, at all, among other reasons due to its extremely high computational complexity. Therefore, we can also argue that from a linguistic point of view there is no need to extend the higher-order approach to proportional quantifiers. Honestly, this is what we would expect, e.g. formulating Σ_1^1-thesis in Section 1.8, but at that point we can not exclude that possibility. However, if there are no such sentences in everyday English then we would say that we have just encountered an example where computational complexity restricts the expressibility of everyday language.

5.6 Summary

In this chapter we have studied the higher-order approach for collective determiners in natural language. In particular, we have considered type-shifting operators: the existential modifier, the neutral modifier and the determiner fitting operator. The research part of this chapter can be summed up in the following way:

- We observed that all these collective lifts are definable in second-order logic.
- Then we introduced second-order generalized quantifiers and proposed them as a tool for modeling collective quantification in natural language.
- Using second-order generalized quantifiers we considered the collective reading of majority quantifiers and proved that it is likely not definable in second-order logic due to its computational complexity. Hence, the type-shifting approach to collectivity in natural language does not obey the Σ_1^1-thesis formulated in Section 1.8.
- Therefore, the collective reading of quantifier "most" probably cannot be expressed using the type-shifting strategy applied to first-order quantifiers.
- In other words, the type-shifting strategy is not general enough to cover all instances of collective quantification in natural language.
- We see a viable alternative in many-sorted (algebraic) approaches which seem to be much easier from the computational complexity point of view and as a result much more psychologically plausible as a model of processing for collective determiners.
- Another possibility is that the collective reading of proportional quantifiers is not realized in everyday language and therefore there is no need for semantic theory to account for it. In that case we would say that computational complexity restricts everyday language expressibility.

Moreover, some natural research problems have appeared in this chapter. Let us mention a few of them:

5.6.1. Question. Does everyday language contain collective interpretations of proportional quantifiers?

5.6.2. Question. What is the exact computational complexity of many-sorted approaches to collectivity in natural language?

5.6.3. Question. What are the natural invariance properties for collective quantification? We noted in Section 5.2.4 that the standard definitions do not have to work properly in the collective context. After formulating empirically convincing invariance properties for collective quantifiers one may want to revise existing theories.

5.6.4. Question. Moreover, the behavior of SOGQs under different invariance properties has not been studied enough. It might be that under some structural properties definability questions among SOGQs might be easier to solve. Obviously, studying definability questions for SOGQs is a natural enterprise from the perspective of collective semantics for natural language.

5.6.5. Question. Finally, is there a purely semantic proof that the quantifier Most is not definable in second-order logic?

Chapter 6

Hintikka's Thesis Revisited

This chapter is devoted to an investigation of potentially branched combinations of quantifiers in natural language. However, as opposed to the previous chapters we do not stay in the safe domain of theoretical discussion but confront our claims with empirical reality. In this way we are aiming to establish the referential meaning of some controversial sentences. As we have shown in Section 3.2, branching interpretations of some natural language sentences are intractable. Therefore, our quest in this chapter is to check whether in everyday language the intractable meaning of branching sentences is in fact realized.

We start by discussing the thesis formulated by Hintikka (1973) that certain natural language sentences require non-linear quantification to express their meaning. We also sum up arguments in favor and against this thesis which have appeared in the literature.

Then, we propose a novel alternative reading for Hintikka-like sentences, the so-called conjunctional reading. This reading is expressible by linear formulae and tractable. We compare the conjunctional reading to other possible interpretations and argue that it is the best representation for the meaning of Hintikka-like sentences.

Next, we describe the empirical support for the conjunctional reading. The basic assumption here is that a criterion for adequacy of a meaning representation is compatibility with sentence truth-conditions. This can be established by observing the linguistic behavior of language users. We report on our experiments showing that people tend to interpret sentences similar to Hintikka's sentence in a way consistent with the conjunctional interpretation.

This chapter is based on joint work with Nina Gierasimczuk (see Gierasimczuk and Szymanik, 2006, 2007, 2008).

6.1 Hintikka's Thesis

Jaakko Hintikka (1973) claims that the following sentences essentially require non-linear quantification for expressing their meaning.

(1) Some relative of each villager and some relative of each townsman hate each other.

(2) Some book by every author is referred to in some essay by every critic.

(3) Every writer likes a book of his almost as much as every critic dislikes some book he has reviewed.

Throughout the chapter we will refer to sentence (1) as *Hintikka's sentence*. According to Hintikka its interpretation is expressed using Henkin's quantifier (see Section 3.2 for an exposition on branching quantifiers) as follows:

$$(4) \quad \begin{pmatrix} \forall x \exists y \\ \forall z \exists w \end{pmatrix} \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big],$$

where unary predicates V and T denote the set of villagers and the set of townsmen, respectively. The binary predicate symbol $R(x,y)$ denotes the symmetric relation "x and y are relatives" and $H(x,y)$ the relation "x and y hate each other". Informally speaking, the idea of such constructions is that for different rows the values of the quantified variables are chosen independently. According to Henkin's semantics for branching quantifiers, formula (4) is equivalent to the following existential second-order sentence:

$$\exists f \exists g \forall x \forall z \big[(V(x) \wedge T(z)) \implies (R(x,f(x)) \wedge R(z,g(z)) \wedge H(f(x),g(z))\big].$$

Functions f and g (so-called Skolem functions) choose relatives for every villager and every townsman, respectively. As you can see, the value of f (g) is determined only by the choice of a certain villager (townsman). In other words, to satisfy the formula relatives have to be chosen independently.[1] This second-order formula is equivalent to the following sentence with quantification over sets:

$$\exists A \exists B \forall x \forall z \big[(V(x) \wedge T(z)) \implies (\exists y \in A\ R(x,y) \wedge \exists w \in B\ R(z,w) \\ \wedge \forall y \in A \forall w \in B\ H(y,w))\big].$$

The existential second-order sentence is not equivalent to any first-order sentence (see the Barwise-Kunen Theorem in (Barwise, 1979)). Not only universal and existential quantifiers can be branched; the procedure of branching works in

[1]The idea of branching is more visible in the case of simpler quantifier prefixes, like in sentence (5) discussed in Section 6.3.2.

a very similar way for other quantifiers (see Definition 3.2.2). Some examples — motivated by linguistics — are discussed in the next section of this chapter.

The reading of Hintikka's sentence given by formula (4) is called *the strong reading*. However, it can also be assigned *weak readings*, i.e., linear representations which are expressible in elementary logic. Let us consider the following candidates:

(5) $\forall x \exists y \forall z \exists w \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big]$
$\wedge \forall z \exists w \forall x \exists y \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big].$

(6) $\forall x \exists y \forall z \exists w \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big].$

(7) $\forall x \forall z \exists y \exists w \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big].$

In all these formulae the choice of the second relative depends on the one that has been previously selected. To see the difference between the above readings and the branching reading consider a second-order formula equivalent to sentence (6):

$$\exists f \exists g \forall x \forall z \big[(V(x) \wedge T(z)) \implies (R(x, f(x)) \wedge R(z, g(x,z)) \wedge H(f(x), g(x,z)))\big].$$

It is enough to compare the choice functions in this formula with those in the existential second-order formula corresponding to sentence (4) to see the difference in the structure of dependencies required in both readings. Of course, the dependencies in sentences (5) and (7) are analogous to (6). As a result all the weak readings are implied by the strong reading, (4) (where both relatives have to be chosen independently), which is of course the reason for the names. Formulae (5)-(7) are also ordered according to the entailment relation which holds between them. Obviously, formula (5) implies formula (6), which implies formula (7). Therefore, formula (5) is the strongest among the weak readings.

By Hintikka's Thesis we mean the following statement:

Hintikka's Thesis *Sentences like Hintikka's sentence have no adequate linear reading. In particular, Hintikka's sentence should be assigned the strong reading and not any of the weak readings.*

6.1.1. Remark. Let us stress one point here. Of course, every branching quantifier can be expressed by some single generalized quantifier, so in the sense of definability Hintikka's thesis cannot be right. However, the syntax of branching quantification has a particular simplicity and elegance that gets lost when translated into the language of generalized quantifiers. The procedure of branching does not employ new quantifiers. Instead it enriches the accessible syntactic means of arranging existing quantifiers, at the same time increasing their expressive power. Therefore, the general question is as follows: are there sentences with simple determiners such that non-linear combinations of quantifiers corresponding to the determiners are essential to account for the meanings of those sentences? The affirmative answer to this question — suggested by Hintikka — claims the existence of sentences with quantified noun phrases which are interpreted scope independently. We show that for sentences similar to those proposed by Hintikka the claim is not true.

Because of its many philosophical and linguistic consequences Hintikka's claim has sparked lively controversy (see e.g. Jackendoff, 1972; Gabbay and Moravcsik, 1974; Guenthner and Hoepelman, 1976; Hintikka, 1976; Stenius, 1976; Barwise, 1979; Bellert, 1989; May, 1989; Sher, 1990; Mostowski, 1994; Liu, 1996; Beghelli et al., 1997; Janssen, 2002; Mostowski and Wojtyniak, 2004; Szymanik, 2005; Schlenker, 2006; Janssen and Dechesne, 2006; Gierasimczuk and Szymanik, 2006, 2007, 2008). Related discussion on the ambiguity of sentences with multiple quantifiers has been vivid in the more philosophically oriented tradition (see Kempson and Cormack, 1981a; Tennant, 1981; Kempson and Cormack, 1981b; Bach, 1982; Kempson and Cormack, 1982; May, 1985; Jaszczolt, 2002). In the present study some of the arguments presented in the discussion are analyzed and critically discussed. In particular, we propose to allow interpreting Hintikka's sentence by the first-order formula (5):

$$\forall x\exists y\forall z\exists w\big[(V(x)\wedge T(z)) \implies (R(x,y)\wedge R(z,w)\wedge H(y,w))\big] \\ \wedge \forall z\exists w\forall x\exists y\big[(V(x)\wedge T(z)) \implies (R(x,y)\wedge R(z,w)\wedge H(y,w))\big].$$

In the rest of this chapter we will refer to this reading as the *conjunctional reading* of Hintikka's sentence.

Our proposal seems to be very intuitive — as we show in the next section — and it is also consistent with human linguistic behavior. The latter fact is supported by empirical data, which we will present in Section 6.4.

Last but not least, it is worth noticing that our proposition is reminiscent of the linguistic representation of reciprocals. For example, according to the seminal paper on "each other" by Heim et al. (1991), Hintikka's sentence has the following structure: `EACH[[QP and QP] ][V the other]`; that is, "each" simply quantifies over the two conjuncts, which turns the sentence into `[QP1 V the other and QP2 V the other]`, where "the other" picks up the other quantifiers anaphorically. This already resembles our conjunctional representation. See Chapter 4 for more discussion of reciprocity.

Our conclusion is that Hintikka-like sentences allow linear reading. This of course clearly contradicts Hintikka's thesis.

6.2 Other Hintikka-like Sentences

Before we move on to the central problem let us consider more sentences with combinations of at least two determiners such that their branching interpretation is not equivalent to any linear reading. They all fall into the scope of our discussion, and we will call them "Hintikka-like sentences".

Interesting examples of Hintikka-like sentences were given by Jon Barwise (1979).

(8) Most villagers and most townsmen hate each other.

(9) One third of the villagers and half of the townsmen hate each other.

These sentences seem to be more frequent in our communication and more natural than Hintikka's examples, even though their adequate meaning representation is no less controversial.

Many more examples have been given to justify the existence of non-linear semantic structures in natural language; see e.g. sentences (10)–(12).

(10) I told many of the men three of the stories. (Jackendoff, 1972)

(11) A majority of the students read two of those books. (Liu, 1996)

(12) We have been fighting for many years for human rights in China. I recount the story of our failures and successes, and say: "Whenever a representative from each country fought for the release of at least one dissident from each prison, our campaign was a success." (Schlenker, 2006)

6.3 Theoretical Discussion

6.3.1 A Remark on Possible Readings

Let us start with the following remark.

It was observed by Mostowski (1994) that from Hintikka's sentence we can infer that:

(13) Each villager has a relative.

This sentence has obviously the following reading: $\forall x[V(x) \implies \exists y R(x,y)]$. It can be false in a model with an empty town, if there is a villager without a relative. However, the strong reading of Hintikka's sentence (see formula (4)) — having the form of an implication with a universally quantified antecedent — is true in every model with an empty town. Hence, this reading of (13) is not logically implied by the proposed readings of Hintikka's sentence.

Therefore, the branching meaning of Hintikka's sentence should be corrected to the following formula with restricted quantifiers:

(14) $$\begin{matrix}(\forall x : V(x))(\exists y : R(x,y)) \\ (\forall z : T(z))(\exists w : R(z,w))\end{matrix} H(y,w),$$

which is equivalent to:

$$\exists A \exists B \big[\forall x(V(x) \implies \exists y \in A\ R(x,y)) \wedge \forall z(T(z) \implies \exists w \in B\ R(z,w)) \\ \wedge \forall y \in A \forall w \in B\ H(y,w)\big].$$

Observe that similar reasoning can be used to argue for restricting the quantifiers in formulae expressing the different possible meanings of all our sentences. However, applying these corrections uniformly would not change the main point of our discussion. We still would have to choose between the same number of possible readings, the only difference being the restricted quantifiers. Therefore, for simplicity we will forego these corrections. From now on we will assume that all predicates in our formulae have non-empty denotation.

6.3.2 Hintikka-like Sentences are Symmetric

It has been observed that we have the strong linguistic intuition that the two following versions of Hintikka's sentence are equivalent (Hintikka, 1973):

(1) Some relative of each villager and some relative of each townsman hate each other.

(15) Some relative of each townsman and some relative of each villager hate each other.

However, if we assume that formula (6), repeated here:

$$\forall x \exists y \forall z \exists w[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))].$$

is an adequate reading of sentence (1), then we have to analogously assume that an adequate reading of sentence (15) is represented by the formula:

(16) $\forall z \exists w \forall x \exists y[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))].$

However, (6) and (16) are not logically equivalent, therefore it would be wrong to treat them as correct interpretations of sentences (1) or (15). Therefore, we have to reject readings (6) and (16) from the set of possible alternatives.

Notice that a similar argument works when we consider other Hintikka-like sentences. For instance, it is enough to observe that the following sentences are also equivalent:

(8) Most villagers and most townsmen hate each other.

(17) Most townsmen and most villagers hate each other.

However, the possible linear reading of (8):

(18) $\mathsf{Most}\ x\ [V(x), \mathsf{Most}\ y\ (T(y), H(x,y))]$

is not equivalent to an analogous reading of (17). Hence, the linear reading (18) cannot be the proper interpretation.

In general, we are dealing here with the fact that the iteration operator (recall Definition 3.1.2) is not symmetric (see Section 3.1.1).

One of the empirical tests we conducted was aimed at checking whether people really perceive such pairs of sentences as equivalent. The results that we describe strongly suggest that this is the case. Therefore, the argument from symmetry is also cognitively convincing (see Section 6.4.4 for a description of the experiment and Section 6.4.4 for our empirical results).

In spite of this observation we cannot conclude the validity of Hintikka's Thesis so easily. First we have to consider the remaining weak candidates, formulae (5) and (7):

(5) $\forall x \exists y \forall z \exists w \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big]$
$\wedge \forall z \exists w \forall x \exists y \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big],$

(7) $\forall x \forall z \exists y \exists w \big[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))\big].$

Hintikka does not consider them at all, and other authors focus only on formula (7) (see e.g. Barwise, 1979; Mostowski and Wojtyniak, 2004)..

Also for different Hintikka-like sentences we still have to differentiate between some possibilities. As an alternative for formula (18) we can consider not only the branching reading (19) (equivalent to (20)) :

(19) $\begin{pmatrix} \mathsf{Most}\ x : V(x) \\ \mathsf{Most}\ y : T(y) \end{pmatrix} H(x,y).$

(20) $\exists A \exists B \big[\mathsf{Most}\ x\ (V(x), A(x)) \wedge \mathsf{Most}\ y\ (T(y), B(y)) \wedge \forall x \in A\, \forall y \in B\ H(x,y)\big].$

but also the conjunctional meaning:

(21) $\mathsf{Most}\ x\big[V(x), \mathsf{Most}\ y(T(y), H(x,y))\big] \wedge \mathsf{Most}\ y\big[T(y), \mathsf{Most}\ x(V(x), H(y,x))\big].$

Notice that for proportional sentences, e.g., (8), there is no interpretation corresponding to the weakest reading of Hintikka's sentence, formula (7), as proportional sentences contain only two simple determiners, and not four as the original Hintikka's sentence. This very fact already indicates that the conjunctional form — as a uniform representation of all Hintikka-like sentences — should be preferred over the weakest reading. Moreover, we report on another argument against the weakest reading (7) in the next section.

To sum up, the symmetricity argument rules out readings with asymmetric scope dependencies. At this point the adequacy of the weakest reading is also controversial since it is not uniform - it cannot be extended to proportional sentences. Our space of possibilities contains now: the branching and the conjunctional reading. In the next section we give additional possible reasons to reject the weakest reading of Hintikka's sentence. However, we do not take this argument as deciding. We tend to reject the weakest reading as a non-uniform interpretation.

6.3.3 Inferential Arguments

Let us move now to Mostowski's (1994) argument against the weakest reading of Hintikka-like sentences.

Let us consider the following reasoning:

Some relative of each villager and some relative of each townsman hate each other.
Mark is a villager.

Some relative of Mark and some relative of each townsman hate each other.

In other words, if we assume that Mark is a villager, then we have to agree that Hintikka's sentence implies that some relative of Mark and some relative of each townsman hate each other.

If we interpret Hintikka's sentence as having the weakest meaning (7):

$$\forall x \forall z \exists y \exists w[(V(x) \wedge T(z)) \implies (R(x,y) \wedge R(z,w) \wedge H(y,w))],$$

then we have to agree that the following sentence is true in Figure 6.1.

(22) Some relative of Mark and some relative of each townsman hate each other.

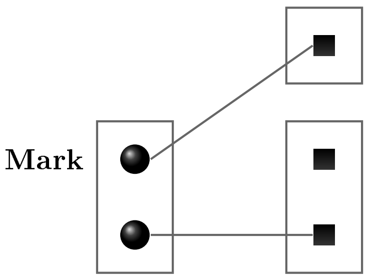


Figure 6.1: Relatives of Mark are on the left, on the right are two town families.

Mostowski (1994) observed that this is a dubious consequence of assigning the weakest interpretation to Hintikka's sentence. He claims that sentence (22) intuitively has the following reading:

(23) $\exists x[R(Mark, x) \wedge \forall y(T(y) \implies \exists z(R(y,z) \wedge H(x,z)))]$.

The above formula (23) is false in the model illustrated in Figure 6.1. Therefore, it cannot be implied by the weakest reading of Hintikka's sentence which is true in the model. However, it is implied by the strong reading which is also false in the model. Hence, Mostowski concludes that Hintikka's sentence cannot have the weakest reading (7).

If we share Mostowski's intuition, then we can conclude from this argument that the weakest reading, (7), should be eliminated from the set of possible alternatives. Otherwise, we can refer to the non-uniformity problem. Then we are left with two propositions: the branching and the conjunctional interpretation. Both of them have the desired inference properties.

6.3.4 Negation Normality

Jon Barwise (1979) in his paper on Hintikka's Thesis refers to the notion of negation normality in a defense of the statement that the proper interpretation of Hintikka's sentence is an elementary formula. He observes that the negations of some simple quantifier sentences, i.e., sentences without sentential connectives other than "not" before a verb, can easily be formulated as simple quantifier sentences. In some other cases this is impossible. Namely, the only way to negate some simple sentences is by prefixing them with the phrase "it is not the case that" or an equivalent expression of a theoretical character.

Sentences of the first kind are called *negation normal*. For example, sentence:

(24) Everyone owns a car.

can be negated normally as follows:

(25) Someone doesn't own a car.

Therefore, this sentence is negation normal. As an example of statement which is not negation normal consider the following (see Barwise, 1979):

(26) The richer the country, the more powerful its ruler.

It seems that the most efficient way to negate it is as follows:

(27) It is not the case that the richer the country, the more powerful its ruler.

Barwise proposed to treat negation normality as a test for first-order definability with respect to sentences with combinations of elementary quantifiers. This proposal is based on the following theorem.

6.3.1. THEOREM. *If φ is a sentence definable in Σ_1^1, the existential fragment of second-order logic, and its negation is logically equivalent to a Σ_1^1-sentence, then φ is logically equivalent to some first-order sentence.*

Barwise claims that the results of the negation normality test suggest that people tend to find Hintikka's sentence to be negation normal, and hence definable in elementary logic. According to him people tend to agree that the negation of Hintikka's sentence can be formulated as follows:

(28) There is a villager and a townsmen that have no relatives that hate each other.

6.3.2. REMARK. Notice that the test works only on the assumption that simple everyday sentences are semantically bounded by existential second-order properties; that is, exactly when we accept the Σ_1^1-Thesis formulated in Section 1.8.

Barwise's claim excludes the branching reading of Hintikka's sentence but is consistent with the conjunctional interpretation. Therefore, in the case of Hintikka's sentence we are left with only one possible reading: the conjunctional reading — at least as far as we are convinced by by the non-uniformity argument Mostowski's and Barwise's arguments. However, in the case of proportional sentences we still have to choose between the branching and the conjunctional interpretation.

6.3.5 Complexity Arguments

Mostowski and Wojtyniak (2004) claim that native speakers' inclination toward a first-order reading of Hintikka's sentence can be explained by means of computational complexity theory. The authors prove that the problem of recognizing the truth-value of the branching reading of Hintikka's sentence in finite models is an NPTIME-complete problem. It can also be shown that proportional branching sentences define an NP-complete class of finite models (see Sevenster, 2006). See Section 3.2 for a discussion of the computational complexity of branching quantifiers.

Assuming that the class of practically computable problems is identical with the PTIME class (i.e., the tractable version of the Church-Turing Thesis; see Edmonds, 1965) they claim that the human mind is not equipped with mechanisms for recognizing NP-complete problems.[2] In other words, in many situations an algorithm for checking the truth-value of the strong reading of Hintikka's sentence is intractable. According to Mostowski and Wojtyniak (2004) native speakers can only choose between meanings which are practically computable.

6.3.3. REMARK. Notice that the argument is similar to the discussion from Chapter 4 where we were considering different interpretations of reciprocal sentences. According to the Strong Meaning Hypothesis the reading associated with

[2]This statement can be given independent psychological support (see e.g. Frixione, 2001). We discus an influence of computational complexity on cognitive abilities in Chapter 1.

the reciprocal in a given sentence is the strongest available reading which is consistent with the context (see Section 4.1.1). However, we have shown that in some cases the strong interpretation of reciprocal sentences is intractable and suggested in Section 4.5 that in such situations interpretation will shift to a weaker (tractable) inferential meaning.

The conjunctional reading is PTIME computable and therefore — even taking into account computational restrictions — can reasonably be proposed as a meaning representation.

6.3.6 Theoretical Conclusions

We discussed possible obstacles against various interpretations of Hintikka-like sentences. Our conjunctional version for Hintikka-like sentences seems to be very acceptable according to all mentioned properties. It is the only reading satisfying all the following properties:

- symmetry;
- uniformity for all Hintikka-like sentences;
- passing Mostowski's inferential test;
- being negation normal for Hintikka's sentence;
- having truth-value practically computable in finite models.

In the case of Hintikka's sentence the conjunctional reading is arguably the only possibility. In general, for proportional sentences it competes only with the intractable branching reading. The next section is devoted to empirical arguments that the conjunctional reading is consistent with the interpretations people most often assign to Hintikka-like sentences.

6.4 Empirical Evidence

Many authors — taking part in the dispute on the proper logical interpretation of Hintikka-like sentences — have argued not only from their own linguistic intuitions but also from the universal agreement of native speakers. For instance, Barwise claims that:

> In our experience, there is almost universal agreement rejecting Hintikka's claim for a branching reading. (Barwise, 1979, p. 51)

However, none of the authors have given real empirical data to support their claims. We confronted this abstract discussion with linguistic reality through experiments.

6.4.1 Experimental Hypotheses

Our hypotheses are as follows:

1. Hypothesis. *People treat Hintikka-like sentences as symmetric sentences.*

This was theoretically justified in the paper of Hintikka (1973) and discussed in Section 6.3.2. To be more precise we predict that subjects will treat sentences like (29) and (30) as equivalent.

(29) More than 3 villagers and more than 5 townsmen hate each other.

(30) More than 5 townsmen and more than 3 villagers hate each other.

2. Hypothesis. *In an experimental context people assign to Hintikka-like sentences meanings which are best represented by the conjunctional formulae.*

We predict that subjects will tend to interpret Hintikka-like sentences in the conjunctional way, i.e. they will accept the sentence when confronted with a model that satisfies its conjunctional interpretation. Arguments for that were given throughout the previous section and were summed up in Section 6.3.6.

In addition we conduct a cross-linguistic comparison. We predict that the comprehension of Hintikka-like sentences is similar in English and Polish — in both languages native speakers allow the conjunctional reading.

6.4.2 Subjects

Subjects were native speakers of English and native speakers of Polish who volunteered to take part in the experiment. They were undergraduate students in computer science at Stanford University and in philosophy at Warsaw University. They all had elementary training in logic so they could understand the instructions and knew the simple logical quantifiers. The last version of the experiment, the one we are reporting on here, was conducted on thirty-two computer science students and ninety philosophy students. However, in the process of devising the experiment we tested fragments of it on many more subjects, getting partial results on which we reported for example at the Logic Colloquium 2006 (see Gierasimczuk and Szymanik, 2006, 2007).

The choice of students with some background in logic was made so that our subjects could be trusted to understand instructions which assume some familiarity with concepts of validity and truth. In that manner, we could formulate the task using such phrases as "one sentence implies the other", "inference pattern is valid", and "sentence is a true description of the picture".

6.4.3 Materials

It was suggested by Barwise and Cooper (1981) and empirically verified by Geurts and van der Silk (2005) that the monotonicity of quantifiers influences how difficult they are to comprehend. Moreover, our results from Chapter 7 also suggest this dependency (see Section 7.4). In particular, sentences containing downward monotone quantifiers are more difficult to reason with than sentences containing only upward monotone quantifiers.[3] Therefore, in the experiment — as we are interested rather in combinations of quantifiers than in simple determiners — we used only monotone increasing quantifiers of the form "More than n" and their combinations in simple grammatical sentences. We used determiners, that are relatively easy to process, because we wanted our subjects to focus on combinations of quantifiers and not on individual ones.

In our tasks the quantifiers referred to the shape of geometrical objects (circles and squares). The sentences were Hintikka-like sentences (for the whole test in English see Appendix A and in Polish Appendix B). All sentences were checked for grammaticality by native speakers.

6.4.4 Experiments

The study was conducted in two languages and consists of two parts. It was a "pen and paper" study. There were no time limits and it took 20 minutes on average for all students to finish the test. Below we present descriptions of each part of the English version of the test. The Polish test was analogous (compare Appendices A and B).

Experiment I: Are Hintikka-like Sentences Symmetric?

The first part of the test was designed to check whether subjects treat Hintikka-like sentences as symmetric (see Section 6.3.2 for a discussion).

Let us recall the notion of symmetry for our sentences. Let $\mathsf{Q}_1, \mathsf{Q}_2$ be quantifiers and ψ a quantifier-free formula. We will say that sentence $\varphi := \mathsf{Q}_1 x\ \mathsf{Q}_2 y\ \psi(x, y)$ is symmetric if and only if it is equivalent to $\varphi' := \mathsf{Q}_2 y\ \mathsf{Q}_1 x\ \psi(x, y)$. In other words, switching the whole quantifier phrase (determiner + noun) does not change its meaning.

We checked whether subjects treat sentences with switched quantifier prefixes as equivalent. We presented subjects with sentence pairs φ, φ' and asked whether the first sentence implies the second sentence. There were 20 tasks. Ten of them were valid inference patterns based on the symmetry. The rest were fillers. Six were invalid patterns similar to the symmetric case. In three we changed the

[3]A quantifier Q_M is upward monotone (increasing) iff the following holds: if $\mathsf{Q}_M(A)$ and moreover $A \subseteq B \subseteq M$ then $\mathsf{Q}_M(B)$. The downward monotone (decreasing) quantifiers are defined analogously as being closed on taking subsets. See Section 2.2.5.

nouns following the quantifiers, i.e. we had $\varphi := \mathsf{Q}_1 x\ \mathsf{Q}_2 y\ \psi(x, y)$ and $\varphi' := \mathsf{Q}_1 y\ \mathsf{Q}_2 x\ \psi(x, y)$. In the second three we switched the determiners and not the whole quantifier phrases, i.e. $\varphi := \mathsf{Q}_1 x\ \mathsf{Q}_2 y\ \psi(x, y)$ and $\varphi' := \mathsf{Q}_2 x\ \mathsf{Q}_1 y\ \psi(x, y)$. Four of the tasks were simple valid and invalid inferences with the quantifiers "more than", "all", and "some".

We constructed our sentences using nonexistent nouns to eliminate any pragmatic influence on subjects' answers. For example, in the English version of the test we quantified over non-existing nouns proposed by Soja et al. (1991): mells, stads, blickets, frobs, wozzles, fleems, coodles, doffs, tannins, fitches, and tulvers. In Polish we came up with the following nouns: strzew, memniak, balbasz, protorożec, melarek, krętowiec, stular, wachlacz, fisut, bubrak, wypsztyk. Our subjects were informed during testing that they are not supposed to know the meaning of the common nouns occurring in the sentences. Therefore, subjects were aware that they have to judge an inference according to its logical form and not by semantic content or pragmatic knowledge.

Figure 6.2 gives examples of each type of task in English.

More than 12 fleems and more than 13 coodles hate each other.

More than 13 coodles and more than 12 fleems hate each other.

VALID NOT VALID

More than 20 wozzles and more than 35 fitches hate each other.

More than 20 fitches and more than 35 wozzles hate each other.

VALID NOT VALID

More than 105 wozzles and more than 68 coodles hate each other.

More than 68 wozzles and more than 105 coodles hate each other.

VALID NOT VALID

Some tulvers are mells.

Some mells are tulvers.

VALID NOT VALID

Figure 6.2: 4 tasks from the first experiment: symmetry pattern, two invalid patterns and a simple inference.

We excluded the possibility of interpreting the sentences as being about the relations between objects of the same kind (e.g. "68 coodles hate each other") by explicitly telling the subjects that in this setting the relation can occur only between objects from two different groups.

Results We got 90% correct answers in the group consisting of philosophy undergraduates at Warsaw University and 93% correct answers among Stanford University computer science students, where by "correct" we mean here "correct according to our prediction about symmetricity". In simple inferences we got 83% (χ^2=153.4, df=1, $p < 0.001$) and 97% (χ^2=110.63, df=1, $p < 0.001$), respectively. In valid symmetricity tasks the result was: 94% (χ^2=709.33, df=1, $p < 0.001$) and 98% (χ^2=286.90, df=1, $p < 0.001$), and in invalid symmetricity inferences: 86% (χ^2=286.02, df=1, $p < 0.001$) and 93% (χ^2=138.38, df=1, $p < 0.001$) (see Figure 6.3). This is a statistically significant result for both groups. Therefore, our first hypothesis — that people treat Hintikka-like sentences as symmetric sentences — was confirmed.

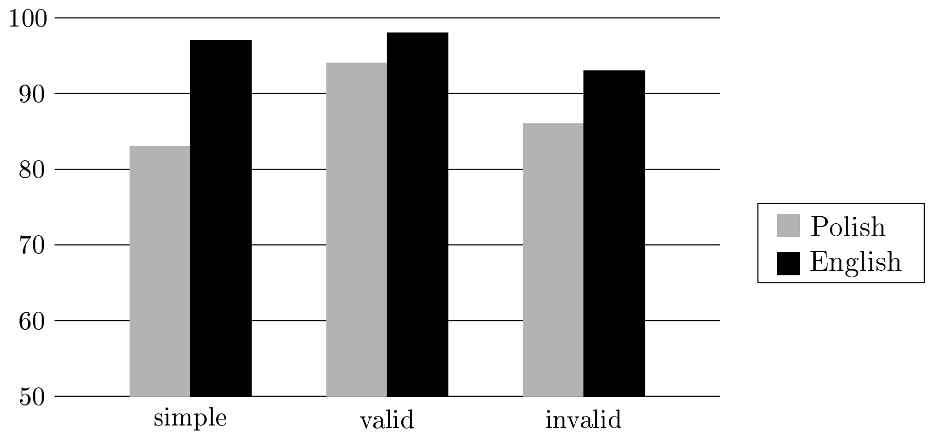


Figure 6.3: Percentage of correct answers in the first test.

Moreover, additional analysis reveals that with respect to the simple inferences 45 philosophy (50%) and 28 computer science (88%) students answered correctly all questions. Focusing on the proper symmetricity tasks, 71 subjects among the philosophers (79%, χ^2 = 30.04, p<0.001, df=1) and 29 computer scientists (91%, χ^2 = 21.13, p<0.001, df=1) recognized correctly all valid and invalid reasoning with a combination of two quantifiers (see Table 6.1 for summary of the results).

Discussion Let us shortly justify our statistical analysis of the results. We were only interested in the frequency of correct answers among all answers to the tasks based on the valid symmetric inference pattern (simple inferences and inferences based on the logically invalid schema were treated as fillers) and that is why we

Groups	Polish philosophers	American computer scientists
number of subjects	90	32
simple inferences correct	83%	97%
symmetricity inferences correct	94%	98%
invalid inferences correct	86%	93%
all simple inferences correct	45 (50%)	28 (88%)
all symmetricity items correct	71 (79%)	29 (91%)

Table 6.1: Percentage of correct answers in the first test.

used χ^2 to analyze our data and not a statistical model, like MANOVA, in which the observed variance is partitioned into components due to different independent (explanatory) variables (e.g. 2 groups of subjects, 4 types of tasks). We did not analyze the data with MANOVA because the assumptions were violated (see e.g. Ferguson and Takane, 1990). According to our hypothesis we had expected that the number of answers "valid" will dominate. In other words, the normality assumption of MANOVA was not satisfied, i.e., the distribution of the answers is not normal but skewed (-4.728) towards validity. That is another reason to use non-parametric test like χ^2. Additionally, the conditions (within-subject) for each kind of tasks were different (the number of problems varied between 10, 4, 3, 3) and the groups were not equal (90 philosophers, 32 computer scientists) what also indicates the use of non-parametric statistical model.

However, we did compare between-subjects the performance of two unequal groups (philosophers vs computer scientists) with respect to the three tests and found no statistically significant differences. To be more precise, there was no difference neither in the task based on valid symmetric inference schema (χ^2=6.583, df=6, p=0.361), in the simple inferences (χ^2=8.214, df=4, p=0.084), nor in invalid inference patterns (χ^2=3.888, df=4, p=0.421).

Experiment II: Branching vs. Conjunctional Interpretation

The second questionnaire was the main part of the experiment, designed to discover whether people agree with the conjunctional reading of Hintikka-like sentences. Subjects were presented with nine non-equivalent Hintikka-like sentences. Every sentence was paired with a model. All but two sentences were accompanied by a picture satisfying the conjunctional reading but not the branching reading. The remaining two control tasks consisted of pictures in which the associated sentences were false, regardless of which of the possible interpretations was chosen.

Every illustration was black and white and showed irregularly distributed squares and circles. Some objects of different shapes were connected with each other by lines. The number of objects in the pictures varied between 9 and 13 and the number of lines was between 3 and 15. Moreover, the number of objects in pictures where the sentences were false was similar to the number of objects in the rest of the test items. Almost all subjects solved these tasks according to our predictions (90% correct answers).

The sentences were of the following form, where $1 \leq n, m \leq 3$:

(31) More than n squares and more than m circles are connected by lines.

(32) Więcej niż n kwadraty i więcej niż m koła są połączone liniami.

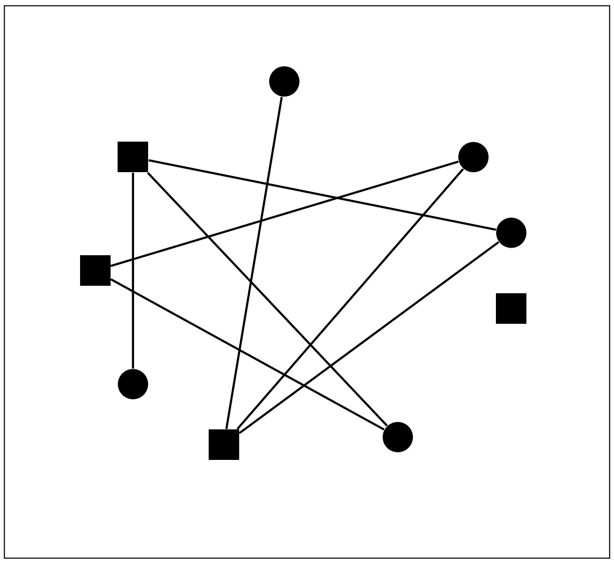
Notice that the Hintikka-like sentences discussed in Chapter 6.1 as well as the items in the symmetricity test contain the phrase "each other". However, we decided not to use this phrase in the sentences tested in the main part of the experiments. This was because our previous experiments (Gierasimczuk and Szymanik, 2006, 2007) indicated that the occurrence of reciprocal expressions in these sentences made people interpret them as statements about the existence of lines between figures of the same geometrical shape. This surely is not the interpretation we wanted to test.

In the first the the usage of the phrase "each other" was meant for making "hating" relation symmetric. In the case of this experiment the relation "being connected by a line" is already symmetric in itself. Moreover, interviews with native speakers suggest that in the context of the relation "being connected by lines" omitting "each other" leads to more natural sentences. Additionally, in the Polish version of the sentences there is no possible phrase corresponding to "each other". This is a grammatical difference between Polish and English Hintikka-like sentences. Even though we assert the possibility of the influence that reciprocals can have on the interpretation of the Hintikka-like sentences (see e.g. Dalrymple et al., 1998) this discussion falls outside the scope of the chapter.

Figures 6.4 and 6.5 show two examples of our tasks. In the first picture the conjunctional reading is true and the branching reading is false. In the second picture the associated sentence is false, regardless of interpretation. The subjects were asked to decide if the sentence is a true description of the picture.

6.4.1. Remark. Let us give here a short explanation why we did not show pictures with a branching interpretation. The theoretical arguments given in Section 6.1 justify the following opposition: either Hintikka-like sentences are interpreted in the conjunctional or in the branching way. We want empirical evidence for acceptability of the conjunctional reading. In principle we have to compare this with the branching meaning. Notice however, that the branching reading implies the conjunctional reading so it is impossible to achieve consistent results rejecting branching readings and confirming conjunctional reading — at least as long as subjects recognize the inference relations between branching and conjunctional readings, and in our experience most of them do (see Gierasimczuk and Szymanik,

More than 1 square and more than 2 circles are connected by lines.



TRUE FALSE

Figure 6.4: Conjunctional task from the second part of the experiment.

2006, 2007). Therefore, we want to prove that people accept the conjunctional reading and not that they reject the branching one. In other words, we are looking for the weakest (theoretically justified) meaning people are ready to accept. To do this it is sufficient to have tasks with pictures for which the conjunctional reading is true, but the branching reading is false. As long as subjects accept them we know that they agree with the conjunctional reading and there is no need to confront them with the branching pictures. Of course this does not mean that people in principle reject the branching reading (but see the computational complexity argument from Section 6.3.5).

Results We got the following results.[4] 94% (χ^2=444.19, df=1, $p < 0.001$) of the answers of the philosophy students and 96% (χ^2=187.61, df=1, $p < 0.001$) of the answers of the computer science students were conjunctional, i.e., "true" when the picture represented a model for a conjunctional reading of the sentence. When it comes to 2 sentences that were false in the pictures no matter how subjects interpreted them we got the following results 92% (χ^2=136.94, df=1, $p < 0.001$) and 96% (χ^2=50.77, df=1, $p < 0.001$) (see Figure 6.6). All these differences are statistically significant. Therefore, our second hypothesis — that in an empirical

[4]We used non-parametric statistical test χ^2 because of the analogous reasons like those explained when discussing the first experiment.

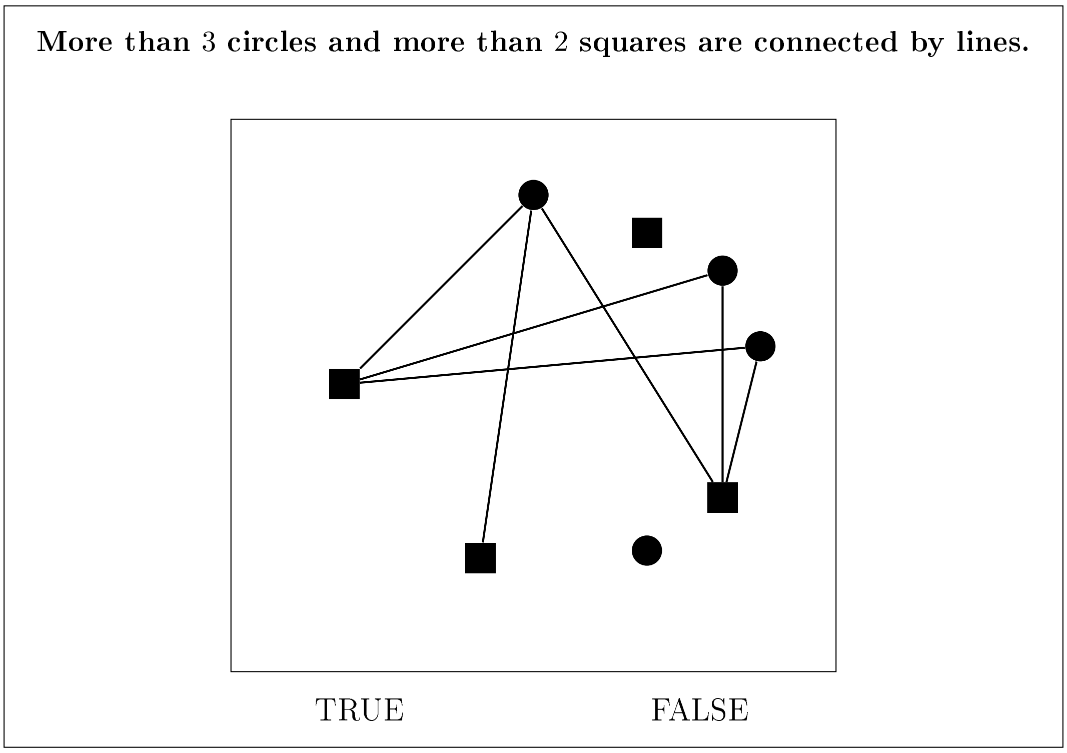

More than 3 circles and more than 2 squares are connected by lines.
TRUE
FALSE

Figure 6.5: An example of a false task from the second part of the experiment.

context people can assign to Hintikka-like sentences meanings which are best represented by the conjunctional formulae — was confirmed.

Additonally, analysis of the individual subjects' preferences revealed what follows. 85 (94%, χ^2 = 71.11, p<0.001, df=1) philosophers and 31 (97%, χ^2 = 28.12, p<0.001, df=1) computer scientists agreed on the conjunctional reading in more than half of the cases. Moreover, 67 (74%, χ^2 = 21.51, p<0.001, df=1) philosophers and 28 (88%, χ^2 = 18, p<0.001, df=1) computer scientists chose conjunctional readings in all tasks (see Table 6.2 for presentation of all data).

Groups	Polish philosophers	American computer scientists
number of subjects	90	32
conjunctional answers	**94%**	**95%**
recognized falsity	92%	96%
most conjunctional answers	85 (94%)	31 (97%)
only conjunctional answers	67 (74%)	28 (88%)

Table 6.2: Results of the second test.

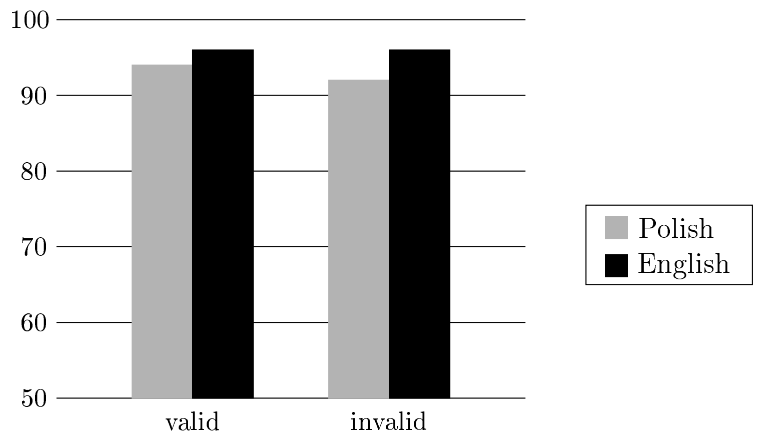


Figure 6.6: Percentage results of the second test.

Moreover, we did not observe any differences between our two subject groups neither in judging obviously false situations (χ^2=0.188, df=1, p=0.664) nor in the conjunctional preferences (χ^2=3.900, df=7, p=0.791). Therefore, we conclude that with respect to interpretation of quantifier combinations in Hintikka-like sentences there is no difference between English and Polish.

6.5 Summary

Conclusions

We argue that Hintikka-like sentences have readings expressible by linear formulae that satisfy all conditions which caused the introduction of branching interpretation, despite what Hintikka (1973) and many of his followers have claimed. The reasons for treating such natural language sentences as having Fregean (linear) readings are twofold.

In Section 6.1 we discussed theoretical arguments. We can sum up them as follows.

- For Hintikka's sentence we should focus on four possibilities: a branching reading (4), and three weak readings: (5), (6), (7).
- Hintikka's argument from symmetricity given in Section 6.3.2, together with the results of our first experiment, allows us to reject asymmetric formulae. A similar argument leads to rejecting the linear readings of other Hintikka-like sentences.
- What about the weakest reading? It does not exist for some Hintikka-like sentences so it cannot be viewed as a universal reading for all of them.

Moreover, the inferential argument from Section 6.3.3 suggests that the weakest meaning is also not an appropriate reading of Hintikka's sentence.

- Therefore, there are only two alternatives — we have to choose between the conjunctional (5) and the branching readings (4).

In section 6.4 we discussed our empirical results. They indicate that people interpret Hintikka-like sentences in accordance with the conjunctional reading, at least in an experimental context. Moreover, we observed no statistically significant differences in preferences of native English and native Polish subjects.

Additionally, our experimental arguments can be supported by the following observations.

- The argument by Barwise from negation normality, discussed in Section 6.3.4, agrees with our empirical results.
- Branching readings — being NP-complete — can be too difficult for language users. Conjunctional readings being PTIME computable are much easier in this sense.

Hence, even though we in principle agree that Hintikka-like sentences are ambiguous between all proposed readings, our experiments and theoretical considerations convince us that in some situations the proper reading of Hintikka-like sentences can be given by conjunctional formulae. This clearly contradicts Hintikka's thesis.

Perspectives

We have tested one of the best known among non-Fregean combinations of quantifiers, the so-called Hintikka-like sentences. As a result we came up with arguments that those sentences can be interpreted in natural language by Fregean combinations of quantifiers. However, there is still some research to be done here.

6.5.1. Question. One can find and describe linguistic situations in which Hintikka-like sentences demand a branching analysis. For example, the work of Schlenker (2006) goes in this direction (recall example 12).

6.5.2. Question. Moreover, it is interesting to ask which determiners allow a branching interpretation at all (see e.g. Beghelli et al., 1997).

6.5.3. Question. Finally, we did not discuss the interplay of our proposition with a collective reading of noun phrases (see e.g. Lønning, 1997, and Chapter 5) and different interpretations of reciprocal expressions (see Dalrymple et al., 1998, and Chapter 4).

As to the empirical work, we find a continuation toward covering other quantifier combinations exciting and challenging. Some ideas we discussed in the context of Hintikka-like sentences, such as inferential meaning, negation normality, and computational complexity perspective, seem universal and potentially useful for studying other quantifier combinations. For instance, they can be used to investigate the empirical reality of the influence of computational complexity on the Strong Meaning Hypothesis in the domain of reciprocal expressions discussed in Chapter 4.

Chapter 7
Comprehension of Simple Quantifiers

In this chapter we step back from complex quantifiers to consider a simpler case. We are interested here in quantifiers binding two unary variable (monadic quantifiers of type (1, 1)). As we will see their meanings can be explained in terms of finite-state and push-down automata. Therefore, from the computational complexity perspective we are very low in the hierarchy of computational problems (see Section 2.3.3).

Restricting ourselves to tractable natural language constructions we will be able to directly argue for the relevance of computational complexity theory for cognitive science. Actually, we will empirically show that more complex quantifiers are more difficult to comprehend. What do we mean by that?

In this chapter we compare the time needed for understanding different types of quantifier. Our results are consistent with an automata-theoretic model of processing monadic quantifiers in natural language which has been posited by several linguists and logicians. We show that the distinction between quantifiers recognized by finite automata and push-down automata is psychologically relevant. Moreover, our results point out the influence of computational resources on the complexity of cognitive tasks as discussed in Chapter 1. Additionally, our research clarifies the recent neuropsychological research on quantifier comprehension. In particular, we directly consider the computational properties of quantifiers and not their logical definability. We also throw more light on the involvement of working memory by comparing comprehension of quantifiers over discourse universes with randomly placed objects and those where objects are ordered in some specific way, simplifying the computational task with respect to memory resources.

This chapter is based on the paper (Szymanik, 2007a) and joint work with Marcin Zajenkowski (see Szymanik and Zajenkowski, 2008).

7.1 Motivations

One of the primary objectives of cognitive sciences is to explain human information processing. As we have mentioned in Chapter 1 we are mainly concerned with the computational level (see Marr, 1983) of this task. Cognitive science has put a lot of effort into investigating the computational level of linguistic competence (see e.g. Isac and Reiss, 2008; Sun, 2008). Today computational restrictions are taken very seriously when discussing cognitive capacities (see Chapter 1 for more discussion). Unfortunately, there are not many empirical studies directly linking the complexity predictions of computational models with psychological reality. The present research aims at increasing our empirical evidence in favor of this connection.

We are concerned here with the very basic linguistic ability of understanding sentences, and consistently identify meaning with a computing procedure (see Chapter 1). In particular, we are dealing here with the capacity of recognizing the truth-value of sentences with simple quantifiers (like “some”, “an even number of”, “more than 7”, “less than half”) in finite situations illustrated in pictures. We show that a simple computational model describing the processing of such sentences — presented in Section 7.1.2 — is psychologically plausible with respect to reaction time predictions.

Our research was motivated — among other things — by a recent neuropsychological investigation into the same problem and, accordingly, by some problems with the interpretation of its results. We discuss these matters in the next section before we provide readers with some mathematical details of the automata-theoretic model of quantifier processing we are working with. Sections 7.2 and 7.3 present our empirical studies of some predictions that can be drawn from that model. We end with a summary and an outline of future work. The basic notions of mathematical linguistics and automata theory we are using can be found in Section 2.3.1 of the Prerequisites chapter.

7.1.1 Previous Investigations in the Area

Quantifiers have been widely treated from the perspective of cognitive psychology (see e.g. Sanford et al., 1994). However, the research presented by McMillan et al. (2005) was the first attempt to investigate the neural basis of natural language quantifiers (see also Mcmillan et al. (2006) for evidence on quantifier comprehension in patients with focal neurodegenerative disease, and Clark and Grossman (2007); Troiani et al. (2009) for a more general discussion). This paper was devoted to a study of brain activity during comprehension of sentences with quantifiers. Using neuroimaging methods (BOLD fMRI) the authors examined the pattern of neuroanatomical recruitment while subjects were judging the truth-value of statements containing natural language quantifiers. According to the authors their results verify a particular computational model of natural language quantifier comprehension posited by several linguists and logicians (see e.g. van Benthem, 1986). We have challenged this statement by invoking the computational difference between first-order quantifiers and divisibility quantifiers (see Szymanik, 2007a). The starting point of the recent research is this very criticism. Let us have a closer look at it.

McMillan et al. (2005) were considering the following two standard types of quantifiers: first-order and higher-order quantifiers. First-order quantifiers are those definable in first-order predicate calculus, which is the logic containing only quantifiers $\exists$ and $\forall$ binding individual variables. In the research, the following first-order quantifiers were used: "all", "some", and "at least 3". Higher-order quantifiers are those not definable in first-order logic, for example "most", "every other". The subjects taking part in the experiment were presented with the following higher-order quantifiers: "less than half of", "an even number of", "an odd number of".

The expressive power of higher-order quantifiers is much greater than the expressibility of first-order quantifiers. This difference in expressive power corresponds to a difference in the computational resources required to check the truth-value of a sentence with those quantifiers.

In particular, to recognize first-order quantifiers we only need computability models which do not use any form of internal memory (data storage). Intuitively, to check whether sentence (1) is true we do not have to involve short-term memory (working memory capacity) (see e.g. Baddeley, 2007, for a psychological model).

(1) All sentences in this chapter are correct.

It suffices to read the sentences from this chapter one by one. If we find an incorrect one, then we know that the statement is false. Otherwise, if we read the entire chapter without finding any incorrect sentence, then statement (1) is true (see Figure 7.2 for an illustration of a relevant automaton). We can proceed in a similar way for other first-order quantifiers. Formally, it has been proved by van Benthem (1986) that all first-order quantifiers can be computed by such simple devices as finite automata (see Theorem 7.1.8 in Section 7.1.2, which contains the mathematical details of the correspondence between quantifiers and automata).

However, for recognizing some higher-order quantifiers, like "less than half" or "most", we need computability models making use of internal memory. Intuitively, to check whether sentence (2) is true we must identify the number of correct sentences and hold it in working memory to compare with the number of incorrect sentences.

(2) Most of the sentences in this chapter are correct.

Mathematically speaking, such an algorithm can be realized by a push-down automaton (see Theorem 7.1.10 in the next section).

From this perspective, McMillan et al. (2005) have hypothesized that all quantifiers recruit the right inferior parietal cortex, which is associated with numerosity. Taking the distinction between the complexity of first-order and higher-order quantifiers for granted, they also predicted that only higher-order quantifiers recruit the prefrontal cortex, which is associated with executive resources, like working memory. In other words, they believe that computational complexity differences between first-order and higher-order quantifiers are also reflected in brain activity during processing quantifier sentences (McMillan et al., 2005, p. 1730). This hypothesis was confirmed.

In our view the authors' interpretation of their results is not convincing. Their experimental design may not provide the best means of differentiating between the neural bases of the various kinds of quantifiers. The main point of criticism is that the distinction between first-order and higher-order quantifiers does not coincide with the computational resources required to compute the meaning of quantifiers. There is a proper subclass of higher-order quantifiers, namely divisibility quantifiers, which corresponds — with respect to memory resources — to the same computational model as first-order quantifiers.

McMillan et al. (2005) suggest that their study honours a distinction in complexity between classes of first-order and higher-order quantifiers. They also claim that:

> higher-order quantifiers can only be simulated by a more complex computing device — a push-down automaton — which is equipped with a simple working memory device.
>
> (McMillan et al., 2005, p. 1730)

Unfortunately, this is not true. In fact, most of the quantifiers identified in the research as higher-order quantifiers can be recognized by finite automata. As we will see in the next section both "an even number" and "an odd number" are quantifiers recognized by two-state finite automata with a transition from the first state to the second and *vice versa*.

7.1.2 Monadic Quantifiers and Automata

In what follows we give a short description of the relevant mathematical results. We assume familiarity with the basic terminology of automata theory which we surveyed in Section 2.3.1.

Monadic Generalized Quantifiers

In this chapter we are concerned with generalized quantifier theory restricted to its simplest form. We want to assign a meaning for sentences like the following:

(3) All poets have low self-esteem.

(4) Some dean danced nude on the table.

(5) At least 7 grad students prepared presentations.

(6) An even number of the students saw a ghost.

(7) Most of the students think they are smart.

(8) Less than half of the students received good marks.

We have explained in detail the semantics assigned to these quantifiers in Section 2.2, where we also formally defined the notion of a generalized quantifier. Below we give a formal definition of monadic generalized quantifiers of type (1, 1). These are the quantifiers we are working with in this chapter.

7.1.1. Definition. A monadic generalized quantifier of type (1, 1) is a class Q of models of the form $\mathbb{M} = (M, A, B)$, where A and B are subsets of the universe M. Additionally, Q is closed under isomorphisms. ■

Representation of Finite Models

Having a quantified sentence and a model we would like to know how to compute the truth-value of this sentence in that model. The first step is to represent finite situations (models) as strings over some finite alphabet. In other words, we need to encode our finite models in a linear form. Here is the idea for doing it.

We restrict ourselves to finite models of the form $\mathbb{M} = (M, A, B)$. For instance, let us consider the model from Figure 7.1. We list all elements of the model in some order, e.g., $c_1, \ldots, c_5$. Then we replace every element in that sequence with one of the symbols from alphabet $\Gamma = \{a_{\bar{A}\bar{B}}, a_{A\bar{B}}, a_{\bar{A}B}, a_{AB}\}$, according to the constituents to which it belongs. This means that we put the string of letters $a_{\bar{A}\bar{B}}$ in place of element c_1 as it belongs to the complement of the set A (denoted as $\bar{A}$) and to the complement of the set B. We write $a_{A\bar{B}}$ for element c_2 because it belongs to the set A and to the complement of the set B, and so on. As a result, in our example, we get the word $\alpha_M = a_{\bar{A}\bar{B}}a_{A\bar{B}}a_{AB}a_{\bar{A}B}a_{\bar{A}B}$. The word α_M corresponds to the model in which: $c_1 \in \bar{A}\bar{B}, c_2 \in A\bar{B}c_3 \in AB, c_4 \in \bar{A}B, c_5 \in \bar{A}B$. Hence, it uniquely (up to isomorphism) describes the model from Figure 7.1.

7.1.2. Definition. The class Q corresponding to a quantifier is represented by the set of words (language) L_{Q} describing all elements (models) of the class. ■

We can extend this idea of coding to generalized quantifiers of any type (see e.g. Mostowski, 1998). For a monadic quantifier binding n variables we will in general need an alphabet consisting of 2^n letters, one letter for every constituent. However, if we restrict ourselves to so-called CE-quantifiers of type (1, 1), i.e., satisfying isomorphism closure, extension (domain independence) and conservativity

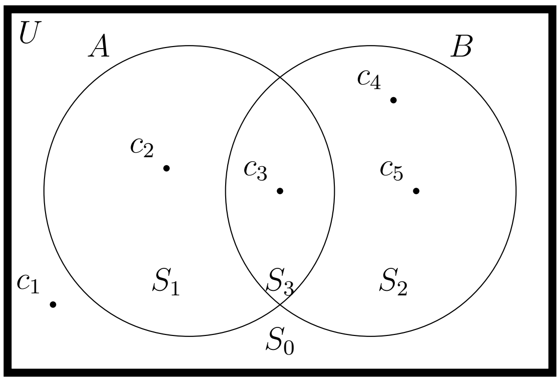


Figure 7.1: This model is uniquely described by $\alpha_M = a_{\bar{A}\bar{B}}a_{A\bar{B}}a_{AB}a_{\bar{A}B}a_{\bar{A}B}$.

(see Section 2.2.5) — which arguably cover all determiners in natural language — then we would need only two letters, one for the elements belonging to the first argument but not the second one and the other for the elements in the intersection of the arguments. This observation follows from Theorem 2.2.19. It suggests that this coding is a quite psychologically plausible representation in case of CE-quantifiers.

Quantifier Automata

Having these definitions we would like to know what kind of automata correspond to particular quantifiers.

Aristotelian Quantifiers The Aristotelian quantifiers "all", "some", "no", and "not all", are first-order definable. They need finite automata with a fixed number of states. Let us consider an example.

7.1.3. EXAMPLE. All *As are B* is true if and only if $A \subseteq B$. In other words, the sentence is true as long as there is no element belonging to A but not B. Having representation α_M of a finite model M over alphabet Γ we can easily recognize whether M satisfies sentence All *As are B*. The following finite automaton from Figure 7.2 does the job. The automaton gets α_M as its input. It inspects the

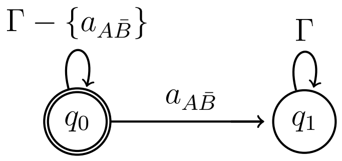


Figure 7.2: Finite automaton recognizing L_{All}

word letter by letter starting in the accepting state. As long as it does not find letter $a_{A\bar{B}}$ it stays in the accepting state, because this means that there was no element belonging to A but not to B. If it finds such an element (letter), then it already "knows" that the sentence is false and moves to the rejecting state, where it stays no matter what happens next.

In other words, the quantifier "All" corresponds to the following regular language:

$$L_{\mathsf{All}} = \{\alpha \in \Gamma^* : \#a_{A\bar{B}}(\alpha) = 0\},$$

where $\#c(\alpha)$ is the number of occurrences of the letter c in the word α.

Cardinal Quantifiers Cardinal quantifiers, e.g., "at least 3", "at most 7", and "between 8 and 11", like the Aristotelian quantifiers are also first-order definable. However, the number of states of a finite automaton recognizing a cardinal quantifier increases in proportion to the number that needs to be represented.

7.1.4. Example. Consider for example the following automaton for At least three *As are B*:

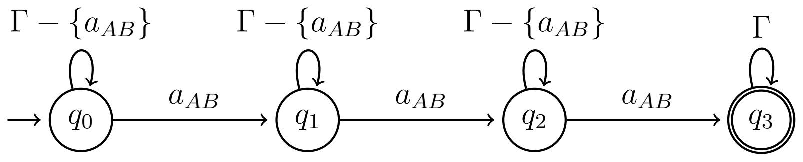

$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[loop above, "\Gamma - \{a_{AB}\}"] \arrow[r, "a_{AB}"] & q_1 \arrow[loop above, "\Gamma - \{a_{AB}\}"] \arrow[r, "a_{AB}"] & q_2 \arrow[loop above, "\Gamma - \{a_{AB}\}"] \arrow[r, "a_{AB}"] & q_3 \arrow[loop above, "\Gamma"]
\end{tikzcd}$$

Figure 7.3: Finite automaton recognizing $L_{\mathsf{At\ least\ three}}$

This automaton needs four states and it corresponds to the language:

$$L_{\mathsf{At\ least\ three}} = \{\alpha \in \Gamma^* : \#a_{AB}(\alpha) \geq 3\}.$$

Furthermore, to recognize "at least 8" we would need nine states and so on.

Parity Quantifiers

7.1.5. Example. What about the quantifier "an even number of"? It corresponds to the following regular language:

$$L_{\mathsf{Even}} = \{\alpha \in \Gamma^* : \#a_{AB}(\alpha) \text{ is even }\}.$$

The finite automaton from Figure 7.4 checks whether the number of occurrences of the letter a_{AB} in the string coding a given model is of even parity. It needs to remember whether it is in the "even state" (q_0) or the "odd state" (q_1) and loops between these states.

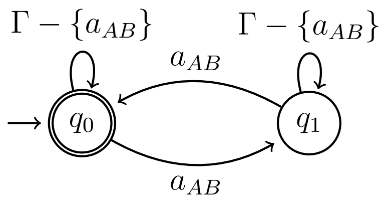


Figure 7.4: Finite automaton recognizing L_{Even}

Proportional Quantifiers

7.1.6. EXAMPLE. Finally, let us have a look at the quantifier "most". The sentence Most *As are B* is true if and only if $\text{card}(A \cap B) > \text{card}(A - B)$. Therefore, the quantifier corresponds to the following context-free language:

$$L_{\mathsf{Most}} = \{\alpha \in \Gamma^* : \#a_{AB}(\alpha) > \#a_{A\bar{B}}(\alpha)\}.$$

There is no finite automaton recognizing all such languages. As models might be of arbitrary finite cardinality so also the length of the coding strings is unbounded. In such a case it is impossible to compute "most" having only a fixed finite number of states as we are not able to predict how many states are needed (see Section 2.3.1 for mathematical justification). To give a computational device for this problem, some kind of internal memory, which allows the automaton to compare any number of occurrences of the symbols a_{AB} and $a_{A\bar{B}}$, is needed. A Push-down automata is a computational model that can achieve this by implementing the idea of a stack (the particular push-down automaton needed for recognizing L_{Most} is similar to the automaton for the language L described in Section 2.3.1).

Characterization The examples considered above already give us the flavor of what is going on. Below we give a general answer to the question about computing devices recognizing particular quantifiers. We start by saying what it means that a class of monadic quantifiers is recognized by a class of devices.

7.1.7. DEFINITION. Let $\mathcal{D}$ be a class of recognizing devices, Ω a class of monadic quantifiers. We say that $\mathcal{D}$ accepts Ω if and only if for every monadic quantifier Q:

$$\mathsf{Q} \in \Omega \iff \text{there is a device } A \in \mathcal{D} \text{ such that } A \text{ accepts } L_{\mathsf{Q}}.$$

■

Now we are ready to state the relevant results. Quantifiers definable in first-order logic, FO, can be recognized by acyclic finite automata, which are a proper subclass of the class of all finite automata (van Benthem, 1986).

7.1.8. Theorem. *A quantifier* Q *is first-order definable iff* L_{Q} *is accepted by an acyclic finite automaton.*

A less well-known result due to Mostowski (1998) says that exactly the quantifiers definable in divisibility logic, $\mathrm{FO}(\mathsf{D}_\mathsf{n})$ (i.e. first-order logic enriched by all quantifiers "divisible by n", for $n \geq 2$), are recognized by finite automata (FAs).

7.1.9. Theorem. *A monadic quantifier* Q *is definable in divisibility logic iff* L_{Q} *is accepted by a finite automaton.*

For instance, the quantifier D_2 can be used to express the natural language quantifier "an even number of". An example of a quantifier falling outside the scope of divisibility logic is "most". Hence, it cannot be recognized by a finite automaton.

A partial characterization of the quantifiers recognized by push-down automata is also known. For example, quantifiers of type (1) expressible in the arithmetic of addition (additive quantifiers, semi-linear quantifiers), so-called Presburger Arithmetic (PrA), are recognized by push-down automata (PDA) (van Benthem, 1986).

7.1.10. Theorem. *A quantifier* Q *of type (1) is definable in PrA iff* L_{Q} *is accepted by a push-down automaton.*

More results on push-down automata and quantifiers can be found in a survey by Mostowski (1998), where the class of quantifiers recognized by deterministic push-down automata is characterized. This class seems particularly interesting from a cognitive point of view.

Obviously, the semantics of many natural language quantifier expressions cannot be modeled by such a simple device as a PDA. Just think about sentence (9) whose meaning cannot be computed by any push-down automaton (it corresponds to the language L_{abc} from Section 2.3.1).[1] This fact follows from the Pumping Lemma for context-free languages (see Theorem 2.3.11).

(9) An equal number of logicians, philosophers, and linguists climbed K2.

There are of course much more complex expressions in natural language, like those discussed in the previous sections of the thesis.

To sum up, first-order and higher-order quantifiers do not always differ with respect to memory requirements. For example, "an even number of" is a higher-order quantifier that can still be recognized by a finite automaton. Therefore, differences in processing cannot be explained based solely on definability properties, as those are not fine-grained enough. A more careful perspective — taking into account all the results we've mentioned, which are summed up in Table 7.1 — will have to be applied to investigate quantifier comprehension. In what follows we present research exploring the subject empirically with respect to the computational model described in this section.

[1] At least without assuming any additional invariance properties for the quantifier in question.

Definability	Examples	Recognized by
FO	“all cars”, “some students”, “at least 3 balls”	acyclic FA
FO(D_n)	“an even number of balls”	FA
PrA	“most lawyers”, “less than half of the students”	PDA

Table 7.1: Quantifiers, definability, and complexity of automata.

7.1.3 The Present Experiment

Our experiment consists of two separate studies. The first study, described in Section 7.2, compares the reaction times needed for the comprehension of different types of quantifiers. In particular, it improves upon the hypothesis of McMillan et al. (2005) by taking directly into account the predictions of computational model and not only definability considerations. Additionally, we compare two classes of quantifiers inside the first-order group: Aristotelian and cardinal quantifiers.

The second study described in Section 7.3 dwells more on the engagement of working memory in quantifier comprehension by using ordered and random distributions of the objects in pictures presented to participants.

General Idea of the First Study: Comparing Quantifiers

First, we compared reaction time with respect to the following classes of quantifiers: those recognized by an acyclic FA (first-order), those recognized by an FA (parity), and those recognized by a PDA. McMillan et al. (2005) did not report any data on differences between first-order and parity quantifiers.

We predict that reaction time will increase along with the computational power needed to recognize quantifiers. Hence, parity quantifiers (even, odd) will take more time than first order-quantifiers (all, some) but not as long as proportional quantifiers (less than half, more than half) (see Definition 4.3.3).

Moreover, we have additionally compared the Aristotelian quantifiers with cardinal quantifiers of higher rank, for instance “less than 8”. In the study of McMillan et al. (2005) only one cardinal quantifier of relatively small rank was taken into consideration, namely “at least 3”. We predict that the complexity of the mental processing of cardinal quantifiers depends on the number of states in the relevant automaton. Therefore, cardinal quantifiers of high rank should be more difficult than Aristotelian quantifiers. Additionally, we suggest that the number of states in an automaton (size of memory needed) influences comprehension more directly than the use of loops. Hence, we hypothesize that the reaction time for the comprehension of cardinal quantifiers of higher rank is between that for parity and proportional quantifiers.

General Idea of the Second Study: Quantifiers and Ordering

There are many possible ways of verifying the role of working memory capacity in natural language quantifier processing. One way we have taken into account is as follows.

In the first study, sentences with pictures were presented to subjects, who had to decide whether the sentence was true. The array elements were randomly generated. However, the ordering of elements can be treated as an additional independent variable in investigating the role of working memory. For example, consider the following sentence:

(10) Most As are B.

Although checking the truth-value of sentence (10) over an arbitrary universe needs a use of working memory, if the elements of a universe are ordered in pairs (a, b) such that $a \in A$, $b \in B$, then we can easily check it without using working memory. It suffices to go through the universe and check whether there exists an element a not paired with any b. This can be done by a finite automaton.

We have compared reaction times while subjects are judging the truth-value of statements containing proportional quantifiers, like sentence (10), over ordered and arbitrary universes. We predict that when dealing with an ordered universe working memory is not activated as opposed to when the elements are placed in an arbitrary way. As a result reaction time over ordered universes should be much shorter.

7.2 The First Study: Comparing Quantifiers

7.2.1 Participants

Forty native Polish-speaking adults took part in this study. They were volunteers from the University of Warsaw undergraduate population. 19 of them were male and 21 were female. The mean age was 21.42 years (SD = 3.22) with a range of 18–30 years. Each subject was tested individually and was given a small financial reward for participation in the study.

7.2.2 Materials and Procedure

The task consisted of eighty grammatically simple propositions in Polish containing a quantifier that probed a color feature of cars on display. For example:

(11) Some cars are red.

(12) Less than half of the cars are blue.

Eighty color pictures presenting a car park with cars were constructed to accompany the propositions. The colors of the cars were red, blue, green, yellow, purple and black. Each picture contained fifteen objects in two colors (see Figure 7.5).

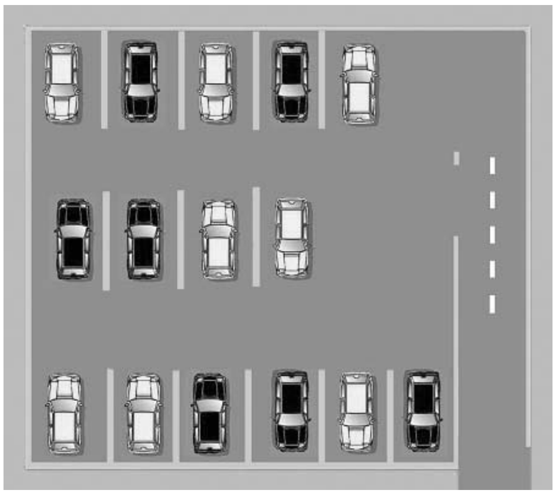


Figure 7.5: An example of a stimulus used in the first study.

Eight different quantifiers divided into four groups were used in the study. The first group of quantifiers were first-order Aristotelian quantifiers (all, some); the second were parity quantifiers (odd, even); the third were first-order cardinal quantifiers of relatively high rank (less than 8, more than 7); and the fourth were proportional quantifiers (less than half, more than half) (see Table 7.2). Each quantifier was presented in 10 trials. Hence, there were in total 80 tasks in the study. The sentence matched the picture in half of the trials. Propositions with "less than 8", "more than 7", "less than half", "more than half" were accompanied with a quantity of target items near the criterion for validating or falsifying the proposition. Therefore, these tasks required a precise judgment (e.g. seven target objects and fifteen in total) for "less than half"). Debriefing following the experiment revealed that none of the participants had been aware that each picture consisted of fifteen objects.

The experiment was divided into two parts: a short practice session followed immediately by the experimental session. Each quantifier problem was given one 15.5 s event. In the event the proposition and a stimulus array containing 15 randomly distributed cars were presented for 15000 ms followed by a blank screen for 500 ms. Subjects were asked to decide if the proposition was true of the presented picture. They responded by pressing the key "P" if true and the key "F" if false. The letters refer to the first letters of the Polish words for "true" and "false".

The experiment was performed on a PC computer running E-Prime version 1.1.

7.2.3 Results

Analysis of Accuracy

As we expected the tasks were quite simple for our subjects and they made only a few mistakes. The percentage of correct answers for each group of quantifiers is presented in Table 7.2.

Quantifier group	Examples	Percent
Aristotelian FO	all, some	99
Parity	odd, even	91
Cardinal FO	less than 8, more than 7	92
Proportional	less than half, more than half	85

Table 7.2: The percentage of correct answers for each group of quantifiers.

Comparison of Reaction Times

To examine the differences in means we used a repeated measures analysis of variance with type of quantifier (4 levels) as the within-subject factor. The assumption of normality was verified by the Shapiro-Wilk test. Because the Mauchly's test showed violation of sphericity, Greenhouse-Geiser adjustment was applied. Moreover, polynomial contrast analysis was performed for the within-subject factor. SPSS 14 was used for the analysis.

Table 7.3 presents mean (M) and standard deviation (SD) of the reaction time in milliseconds for each type of quantifier.

Group	Quantifiers	M	SD
Aristotelian FO	all, some	2257.50	471.95
Parity	even, odd	5751.66	1240.41
Cardinal FO	less than 8, more than 7	6035.55	1071.89
Proportional	less than half, more than half	7273.46	1410.48

Table 7.3: Mean (M) and standard deviation (SD) of the reaction time in milliseconds for each type of quantifier.

We found out that the increase in reaction time was determined by the quantifier type ($F(2.4, 94.3) = 341.24$, $p < 0.001$, η^2=0.90). Pairwise comparisons among means indicated that all four types of quantifiers differed significantly from one another ($p < 0.05$). Polynomial contrast analysis showed the best fit for a linear trend ($F(1, 39) = 580.77$, $p < 0.001$). The mean reaction time increased as follows: Aristotelian quantifiers, parity quantifiers, cardinal quantifiers, proportional quantifiers (see Figure 7.6).

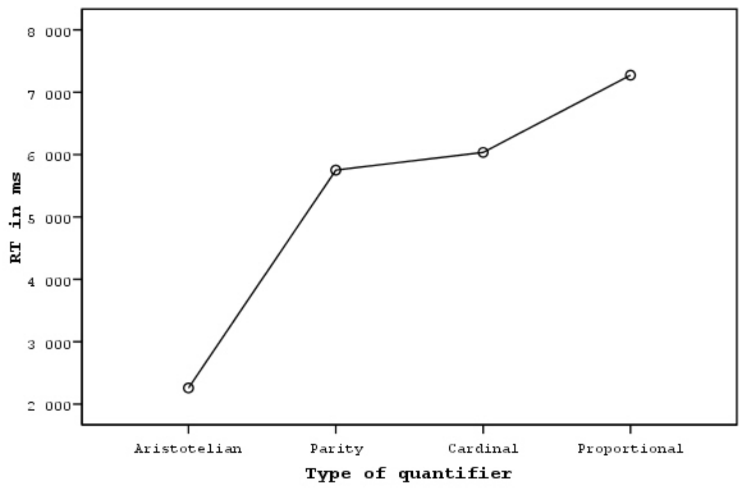


Figure 7.6: Average reaction times in each type of quantifiers in the first study.

7.3 The Second Study: Quantifiers and Ordering

7.3.1 Participants

Thirty native Polish-speaking adults took part in the second study. They were undergraduate students from two Warsaw universities. 12 were male and 18 were female. The mean age was 23.4 years (SD = 2.51) with a range of 20–28 years. Each subject was tested individually.

7.3.2 Materials and Procedure

In the task, we used sixteen grammatically simple propositions in Polish containing proportional quantifiers that probed a color feature of cars on a display (e.g. "More than half of the cars are blue"). Color pictures presenting a car park with eleven cars were constructed to accompany the propositions. As in the first study, the colors used for the cars were: red, blue, green, yellow, purple and black. Each picture contained objects in two colors.

Two different proportional quantifiers (less than half, more than half) were presented to each subject in 8 trials. Each type of sentence matched the picture in half of the trials. Moreover, each quantifier was accompanied by four pictures presenting cars ordered in two rows with respect to their colors (see Figure 7.7) and four pictures presenting two rows of randomly distributed cars. The rest of the procedure was the same as in the first study.

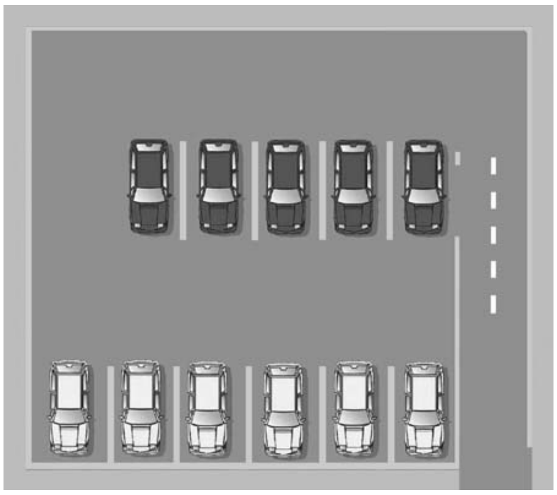


Figure 7.7: An example of a stimulus used in the second study. A case when cars are ordered.

7.3.3 Results

Analysis of Accuracy

The behavioral data showed higher accuracy of subjects' judgments for ordered universes (90% correct) than for unordered universes (79% correct) (see Table 7.4).

Comparison of Reaction Times

Since there were only two types of situations (random and ordered) in the study, a paired-samples t-test was used to analyze differences in the reaction times. Proportional quantifiers over randomized universes (M=6185.93; SD=1759.09) were processed significantly longer than these over ordered models (M=4239.00; SD=1578.26) ($t(29) = 5.87$ $p < 0.001$; $d = 1.16$).

Table 7.4 summarizes the results of this study.

Situation	Accuracy	M	SD
Randomized	79%	6185.93	1759.09
Ordered	90%	4239.000	1578.27

Table 7.4: Accuracy, mean (M) and standard deviation (SD) of the reaction time in milliseconds for proportional quantifiers over randomized and ordered universes.

7.4 Summary

Conclusions

We have been studying the comprehension of natural language quantifiers from the perspective of simple, automata-theoretic computational models. Our investigation is a continuation of previous studies. In particular, it enriches and explains some data obtained by McMillan et al. (2005) with respect to reaction times. Our results support the following conclusions:

- The automata-theoretic model described in Section 7.1.2 correctly predicts that quantifiers computable by finite automata are easier to understand than quantifiers recognized by push-down automata. It improves the results of McMillan et al. (2005), which compared only first-order quantifiers with higher-order quantifiers, putting in one group quantifiers recognized by finite automata and those recognized by push-down automata.

- We have observed a significant difference in reaction time between Aristotelian and divisible quantifiers, even though they are both recognized by finite automata. This difference may be accounted for by observing that the class of Aristotelian quantifiers is recognized by acyclic finite automata, whereas in the case of divisible quantifiers we need loops. Therefore, loops are another example of a computational resource having an influence on the complexity of cognitive tasks.

- We have shown that processing first-order cardinal quantifiers of high rank takes more time than comprehension of parity quantifiers. This suggests that the number of states in the relevant automaton plays an important role when judging the difficulty of a natural language construction. Arguably, the number of states required influences hardness more than the necessity of using cycles in the computation.

- Decreased reaction time in the case of proportional quantifiers over ordered universes supports the findings of McMillan et al. (2005), who attributed the hardness of these quantifiers to the necessity of using working memory.

- Last but not least, our research provides direct evidence for the claim that human linguistic abilities are constrained by computational resources (internal memory, number of states, loops).

Perspectives

There are many questions we leave for further research. Below we list a few of them.

7.4.1. Question. Our experimental setting can be used for neuropsychological studies extending the one by McMillan et al. (2005). On the basis of our research and the findings of McMillan et al. (2005) we predict that comprehension of parity quantifiers — but not first-order quantifiers — depends on executive resources that are mediated by the dorsolateral prefrontal cortex. This would correspond to the difference between acyclic finite automata and finite automata. Moreover, we expect that only quantifiers recognized by PDAs but not FAs activate working memory (inferior frontal cortex). Additionally, the inferior frontal cortex should not be activated when judging the truth-value of sentences with proportional quantifiers over ordered universes. Are these predictions correct? Further studies answering this question would contribute to extending our understanding of simple quantifier comprehension on Marr's implementation level.

7.4.2. Question. What about the algorithmic level of explanation? It would be good to describe the procedures actually used by our subjects to deal with comprehension. In principle it is possible to try to extract real algorithms by letting subjects manipulate the elements, tracking their behavior and then drawing some conclusions about their strategies. This is one of the possible future directions to enrich our experiments.

7.4.3. Question. Before starting any neuropsychological experiments it would be useful to measure memory involvement for different types of quantifiers using some more classical methods known from cognitive psychology, like a dual-task paradigm combining a memory span measure with a concurrent processing task. Will these methods confirm working memory engagement according to the predictions?

7.4.4. Question. We find it interesting to explore the differences in comprehension of Aristotelian and cardinal quantifiers in more detail, both from the empirical and theoretical points of view. This would provide us with an opportunity to better understand the connection between the number of states as a part of computational models and the real cognitive capacities described by these models. How does the number of states in the minimal automaton influence the difficulty of the quantifier?

7.4.5. Question. It has been observed by Geurts (2003) that monotonicity plays a crucial role in reasoning with quantifiers (see also Geurts and van der Silk, 2005). Upward monotone quantifiers are easier than downward monotone ones with respect to reasoning. It is a matter of empirical testing to check whether the same holds for comprehension. Our study was not designed to explore this possibility. However, we compaired pairs of quantifiers with respect to monotonicity in the right argument and observed the following. In the case of the Aristotelian quantifiers "all" and "some" monotonicity influences reaction time for comprehension in a way close to being significant. Parity quantifiers are non-monotone, but we have observed that "odd" is more difficult. For cardinal first-order quantifiers we have a significant result: the decreasing quantifier "less than 8" is more difficult than its increasing counterpart. Unfortunately, we did not observe any statistical dependencies between proportional quantifiers of different monotonicity. What is the role of monotonicity in quantifier comprehension?

7.4.6. Question. Finally, the automata-theoretic model can be extended for other notions than simple quantifiers. For example — as was already suggested by van Benthem (1987) — by considering richer data structures it can account for conditionals, comparatives, compound expressions in natural language, and non-elementary combinations of quantifiers (like branching); also it can form a link with learnability theory (see e.g. Gierasimczuk, 2007) and others. Are such possible extensions of any value for cognitive science?

Chapter 8

Conclusions and Perspectives

8.1 General Summary

In the thesis we have pursued the topic of the computational complexity of natural language quantifier constructions. Our perspective combined logical research with linguistic insights and an empirical, cognitive foundation.

In Chapter 1 we discussed computational semantics as a part of the Dynamic Turn in the philosophy of language. In particular, we gave a short history and argued for identifying a basic element of meaning — confronting it with the actual world — with algorithms. Assuming the procedural theory of meaning allows us to view natural language comprehension in a broader perspective of cognitive science. Moreover, it suggests that we should look for theoretical constraints in computational complexity theory. Aiming for that general perspective we recalled the three levels of explanation in cognitive psychology proposed by Marr: computational, algorithmic, and neurological. Next, we noticed that studying the computational complexity of natural language constructions can contribute to the computational level of explanation. Among other possibilities, as computational complexity theory deals with abstract, inherent, and hardware-independent properties of information processing tasks it can be used for explaining the cognitive difficulty of human performance. Following the literature, we proposed treating problems computable in polynomial time as easy (tractable) and problems not computable in polynomial time (NP-hard) as difficult (intractable). Our technical work, in the following chapters, was mostly devoted to drawing tractability distinctions between the meanings of various natural language quantifier constructions, and evaluating the plausibility of the proposal.

Additionally, assuming the distinction between tractable and intractable and following Ristad (1993), we have argued in Section 1.8 that a good semantic theory of the everyday fragment of natural language should be expressible in the existential fragment of second-order logic. Simply put, its descriptive power should be high enough to account for the flexibility of everyday language, but, on the other hand, a theory can not be too strong or it will overgeneralize the linguistic data and need to postulate implausible linguistic mechanisms.

Let us briefly review how our technical results throw light on these grand issues identified in Chapter 1.

In Chapters 3 and 5 we studied the logical and computational structures of existing linguistic theories and obtained some new complexity results. In Chapter 3 we studied polyadic quantifiers in natural language. We showed that the most common operations creating polyadic quantifiers in everyday language — Boolean operations, iteration, cumulation, and resumption — do not lead outside the tractable class of semantic constructions. On the other hand, more sophisticated polyadic lifts playing an important role in linguistics — branching and Ramseyfication — can produce expressions with intractable referential meanings. In Chapters 4, 5, and 6 we investigated fragments of natural language semantics, drawing some linguistic conclusions from complexity observations. First of all, our results show that computational complexity might be a factor constraining the expressive power of everyday language. For example, it might be the case that collective quantification in natural language cannot really express all possible facts about collections. The interpretation process has to be restricted by some other linguistic factors to keep its complexity reasonable. On the other hand, we have shown that linguistic theories have to take computational complexity into account as long as they want to serve as an input for plausible cognitive theories.

Additionally, in Chapter 4 we studied some linguistic cases where model-checking is intractable and the inferential aspect of comprehension comes into the game. This was the case with the different readings of reciprocal expressions which are connected by inferential properties. Our results suggest that the two basic aspects of meaning, model-theoretic and inferential, as identified in Chapter 1, are strongly linked and that their interaction might be triggered by computational complexity effects. This shows how computational complexity can influence pragmatics.

Finally, in Chapter 7 we described an empirical study supporting the computational complexity perspective on meaning. These observations directly linked complexity to difficulty for natural language. They also show the strength of the procedural approach to semantics with respect to formulating interesting hypotheses of psychological relevance. This clear link between algorithmic semantics and cognitive science makes a dynamic approach to meaning extremely attractive and proves its relevance.

In general, our research has explored the advantages of identifying meaning with an algorithm. We have shown the fruitfulness of this approach for linguistics and cognitive science.

There are of course many related issues waiting for an explanation. We have discussed some open questions following directly from our work at the end of each chapter. Below we review some more general questions and directions for future research. This may help in forming a better picture of the research endeavor this dissertation is a part of.

8.2 Outline

8.2.1 Different Complexity Measures

Recall that in Chapter 3 we showed that proportional Ramsey quantifiers define NP-complete classes of finite models. On the other hand, we have also observed that bounded Ramsey quantifiers are in PTIME. It is an open problem where the precise border lies between tractable and mighty Ramsey quantifiers. As we already noted in Section 3.4 the above open problem directs us towards parametrized complexity theory. In general the following question arises:

8.2.1. Question. What is the parametrized complexity of iteration, cumulation, resumption, branching, and Ramseyfication (the operations studied in Chapter 3)?

Using parametrized complexity can help us to find the boundary between tractable and mighty Ramsey quantifiers. Moreover, it can answer some doubts about worst-case complexity as a measure of linguistic difficulty (see Section 1.5.3). We can already notice that also from the parametrized complexity perspective the clique problem is believed to be intractable. Therefore, our general claims about the intractability of strong reciprocity would probably be preserved under this measure.

Broadly speaking we would like to study the complexity of natural language semantics from various perspectives. Therefore, we ask:

8.2.2. Question. What is the complexity of quantifiers under different measures, like parametrized, circuit, and average-case complexity?

Sevenster (2006) has already noted that it would be interesting to characterize the circuit complexity of quantifiers. The other interesting measure would be average-case complexity. It could help to answer questions like the following: How difficult is it to compute quantifiers on average (random) graphs? Here the situation might be different than for worst-case complexity. For instance, the clique problem (the strong interpretation of reciprocal expressions) can be solved on average in sub-exponential time on random graphs. Therefore, from this perspective Ramsey quantifiers can become “almost” tractable in most cases.

8.2.2 Quantifiers and Games

We have motivated our interest in the computational complexity of quantifiers by the insight it gives into the possibilities for processing natural language determiners. However, besides studying various measures of complexity we can also investigate the properties of evaluation games for quantifiers. In the case of existential and universal quantifiers and their combinations such games have been extensively studied (see e.g. Hintikka and Sandu, 1997). However, the issue has never been satisfactory worked out for a wider class of generalized quantifiers, including polyadic constructions (see e.g. Clark, 2007).

8.2.3. Remark. Our suggestion for future work is to identify simple quantifiers with games and then investigate combinations of these games coinciding with polyadic lifts on simple quantifiers.

There is already some work in that direction. For instance, Peter Aczel (1975) formulated a game-theoretical interpretation for infinite strings of monotone quantifiers, more than 30 years ago. Recently, similar ideas were proposed by van Benthem (2002, 2003). He considered a so-called game logic which encodes the algebra of game operations. In particular, he has shown that the algebra of sequential operations, like choice, dual and composition, coincides with the evaluation games for first-order logic (see van Benthem, 2003). Recently, van Benthem et al. (2007) have extended this idea and proposed a complete logic, which they call Concurrent Dynamic Game Logic, to formalize simultaneous games. This logic — with the product operator representing parallel games — has applications in studying the branching operation. An immediate open question stemming from that research is as follows:

8.2.4. Question. Does concurrent dynamic logic coincide with evaluation games for first-order logic extended by all Henkin quantifiers (see van Benthem et al., 2007)?

Studying the game algebra corresponding to polyadic combination of quantifiers can help to understand the structure of natural operations creating complex games for compound quantifier expression out of simple ones. This fresh perspective might be valuable for solving some of the notoriously difficult problems of compositionality.

Moreover, games might be useful for formulating an intuitive semantics for second-order definable quantifiers in arbitrary weak models (see Section 1.6). In a finite universe second-order definable quantifiers correspond to alternating computing (see Theorem 2.4.5) which is similar to game-theoretical semantics. Maybe this idea can be naturally extended to the case of infinite weak models.

Additionally, one can think about using generalized quantifiers to define solution concepts in games. For example, we can express the existence of an extreme Nash equilibrium in a game using a first-order formula.

8.2.5. Question. What about branching quantifiers? Can they account for the existence of mixed equilibria in non-determined games? Maybe studying other generalized quantifiers in that context can lead to conceptually new game solutions. If we enrich first-order logic with generalized quantifiers we can count, or talk about proportions. Does this give rise to any interesting game concepts?

8.2.3 Cognitive Difficulty and Complexity

We need to investigate the interplay between cognitive difficulty and computational complexity in more detail. Do the differences in computational complexity really play an important role in natural language processing as our data from Chapter 7 suggests? Can we show empirically the influence of computational complexity on the difficulty of other cognitive tasks? For example:

8.2.6. Question. Can we design experiments confirming our speculations from Section 4.5, i.e., that shifts in reciprocal meaning are sometimes triggered by the computational complexity of sentences?

8.2.7. Question. Is there an empirical way to prove the implausibility of the higher-order approach to collective quantification studied in Chapter 5?

Moreover, it is very important to extend the complexity explanation beyond the first level of Marr's hierarchy (described in Section 1.5.1). We should not only study the abstract computational properties of natural language constructions but also try to grasp the procedures really used by humans to deal with the comprehension of natural language. It would in principle be possible to extract real strategies by letting subjects manipulate the elements, tracking their behavior and then drawing some conclusions about their strategies. This is one of the possible directions for enriching our experiments from Chapter 7. Obviously, the most ambitious question is to describe comprehension at the neural implementation level using a brain scanning technique. Hopefully, at some point there will be enough cognitive data to start broad fMRI studies of the problem.

8.2.4 Future of GQT and Beyond

Finally, we would like to share one general impression on the state of the art in generalized quantifier theory. Recently, Peters and Westerståhl (2006) have published an excellent monograph in the field. The book focuses on definability questions and their relevance for linguistics. It gives the impression that the research field is almost complete and there are not so many interesting open questions. In fact, one can observe decreasing interest in generalized quantifier theory since the eighties. On the other hand, the above-mentioned handbook contains no chapter that deals with computational complexity issues, collective quantification or cognitive science. It was already noticed in the review by van Benthem (2007) that "classical" generalized quantifier theory as presented in the book considers logic and language, but misses out on computation, which should be the third pillar.

We hope that our dissertation may be seen as laying the groundwork for strengthening generalized quantifier theory with the computational pillar, which might help to revive the field. We have tried to convey a general message that there is still a lot to do in the field of generalized quantifiers. We have mainly focused on complexity questions and their interplay with cognitive science but there are many more possibilities. For instance, the problem of investigating the invariance properties of collective quantifiers (second-order generalized quantifiers), as we have mentioned in Chapter 5, is not only very classical in spirit but also potentially would have a large impact on linguistics and theoretical computer science. Other examples of interesting research directions which can be taken in the context of generalized quantifiers include game-theoretical analysis, learnability theory for quantifiers and its connections with invariance properties like monotonicity.

Last but not least, it is necessary to extend the computational complexity analysis to different natural language constructions as well as quantifiers; for instance, exploiting the automata approach involving richer data structures, as was already proposed in the eighties (see e.g. van Benthem, 1987). Moreover, we have argued in the first chapter that it would be natural to treat dynamic theories of language, like belief-revision, theories of context-dependence, signalling games, and discourse representation theories, with a computational complexity analysis. Presumably we need a new model of computation which is better conceptual fit to the task of describing communication in language. One obvious candidate would be a model based on games, which have already been embraced by computer scientists as a rich model of computations.

Summing up, a fresh computational approach is needed to evaluate the cognitive plausibility of dynamic theories. And, if necessary, this can lead to more natural reformulations. Moreover, complexity analysis can be a first step towards connecting linguistic theories of communication with cognitive science. After all, the ability to use language is one of many human cognitive processes and as such should not be analyzed in isolation.

Appendix A
English Version of the Test from Chapter 6

FIRST TEST

Instruction: Over the next pages you will find 20 tasks. Each task represents some inference. Your aim is to decide whether this inference is valid.

In other words, each task consists of 2 sentences with a horizontal line between them. You must decide whether a sentence above the line implies a sentence below the line.

If you think that inference pattern is valid (second sentence is implied by the first one) encircle: "VALID", otherwise encircle: "NOT VALID".

Example 1:

$$\frac{\text{At least 5 mells are stads.}}{\text{At least 3 mells are stads.}}$$

[✓ VALID] NOT VALID

Example 2:

$$\frac{\text{At most 5 blickets are frobs.}}{\text{At most 3 blickets are frobs.}}$$

VALID [✓ NOT VALID]

More than 6 fleems are tulvers.
___
More than 5 fleems are tulvers.

VALID NOT VALID

More than 12 fleems and more than 13 coodles hate each other.
___
More than 13 coodles and more than 12 fleems hate each other.

VALID NOT VALID

More than 16 stads and more than 9 blickets hate each other.
___
More than 9 blickets and more than 16 stads hate each other.

VALID NOT VALID

More than 16 mells and more than 25 blickets hate each other.
___
More than 25 blickets and more than 16 mells hate each other.

VALID NOT VALID

More than 10 mells are fleems.
___
More than 11 mells are fleems.

VALID NOT VALID

More than 9 frobs and more than 8 coodles hate each other.

More than 8 coodles and more than 9 frobs hate each other.

VALID NOT VALID

More than 20 wozzles and more than 35 fitches hate each other.

More than 20 fitches and more than 35 wozzles hate each other.

VALID NOT VALID

All wozzles are fleems.

All fleems are wozzles.

VALID NOT VALID

More than 100 wozzles and more than 150 stads hate each other.

More than 150 stads and more than 100 wozzles hate each other.

VALID NOT VALID

More than 105 wozzles and more than 68 coodles hate each other.

More than 68 wozzles and more than 105 coodles hate each other.

VALID NOT VALID

More than 6 doffs and more than 5 fitches hate each other.

More than 5 fitches and more than 6 doffs hate each other.

VALID NOT VALID

More than 47 stads and more than 55 tannins hate each other.

More than 47 tannins and more than 55 stads hate each other.

VALID NOT VALID

More than 58 frobs and more than 49 tannins hate each other.

More than 49 frobs and more than 58 tannins hate each other.

VALID NOT VALID

More than 7 coodles and more than 6 doffs hate each other.

More than 6 doffs and more than 7 coodles hate each other.

VALID NOT VALID

Some tulvers are mells.

Some mells are tulvers.

VALID NOT VALID

More than 99 coodles and more than 68 tulvers hate each other.

More than 68 tulvers and more than 99 coodles hate each other.

VALID NOT VALID

More than 7 tannins and more than 8 fitches hate each other.

More than 8 fitches and more than 7 tannins hate each other.

VALID NOT VALID

More than 19 frobs and more than 11 fleems hate each other.

More than 11 fleems and more than 19 frobs hate each other.

VALID NOT VALID

More than 159 stads and more than 25 fitches hate each other.

More than 159 fitches and more than 25 stads hate each other.

VALID NOT VALID

More than 8 frobs and more than 27 doffs hate each other.

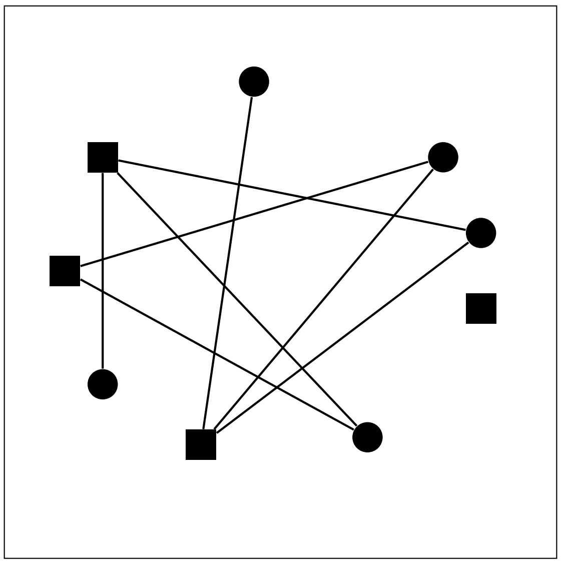
More than 27 frobs and more than 8 doffs hate each other.

VALID NOT VALID

SECOND TEST

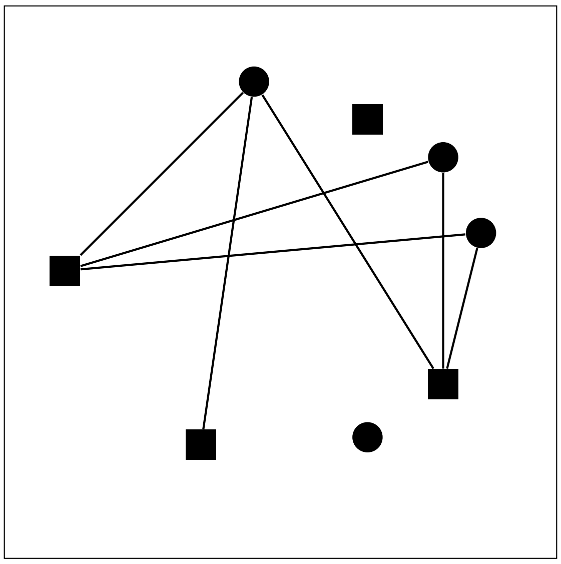
Instruction: Over the next few pages you will find 9 tasks to solve. Each task consists of a picture. Above every picture there is exactly one sentence. Encircle TRUE if and only if the sentence is a true description of the picture. Otherwise, encircle FALSE.

More than 1 square and more than 2 circles are connected by lines.



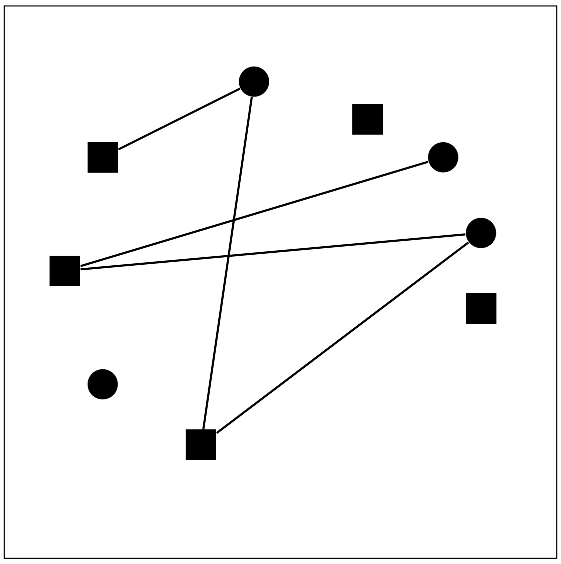
TRUE FALSE

More than 3 circles and more than 2 squares are connected by lines.



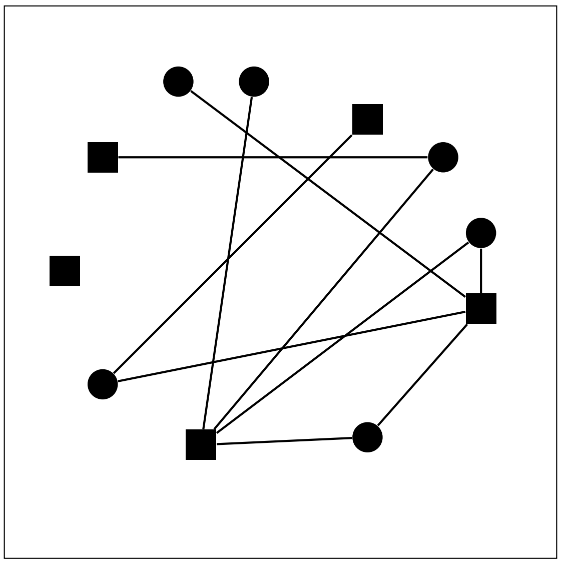
TRUE FALSE

More than 1 square and more than 1 circle are connected by lines.



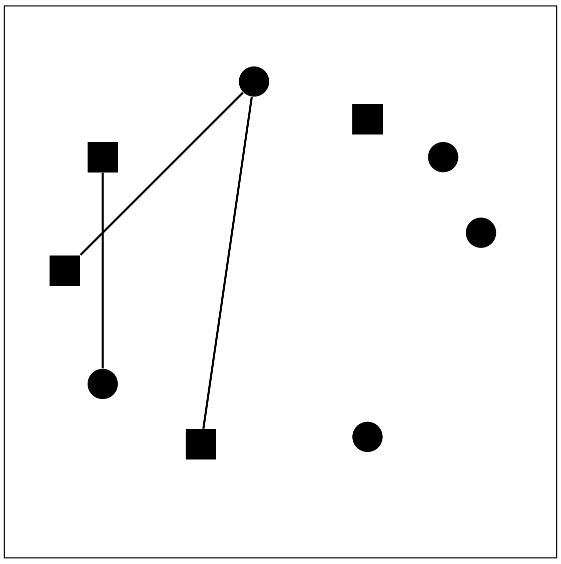
TRUE FALSE

More than 3 circles and more than 1 square are connected by lines.



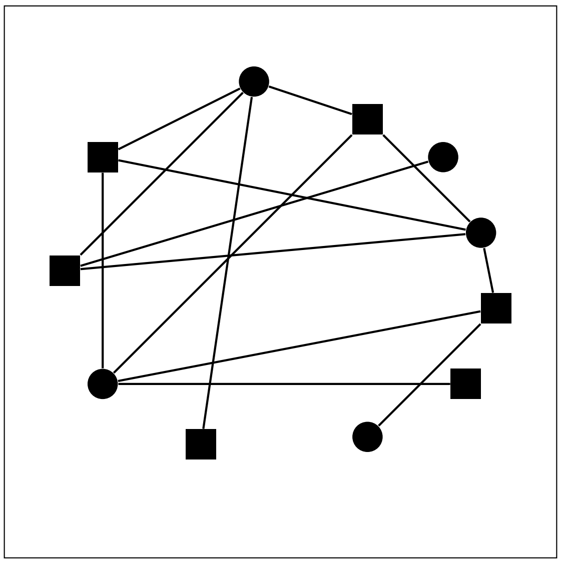
TRUE FALSE

More than 3 circles and more than 3 squares are connected by lines.



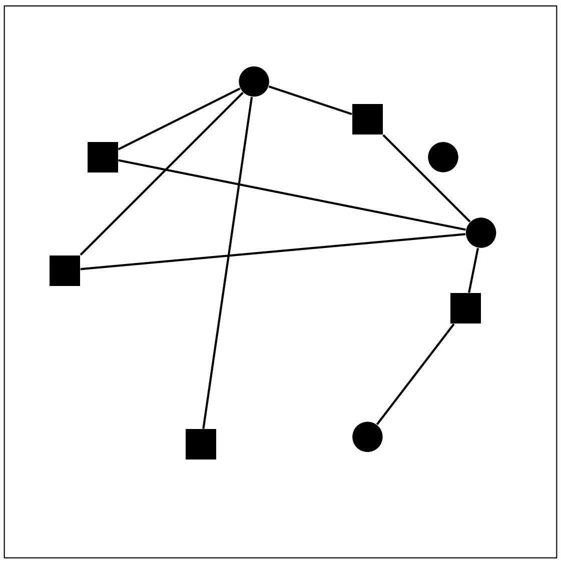
TRUE FALSE

More than 2 circles and more than 3 squares are connected by lines.



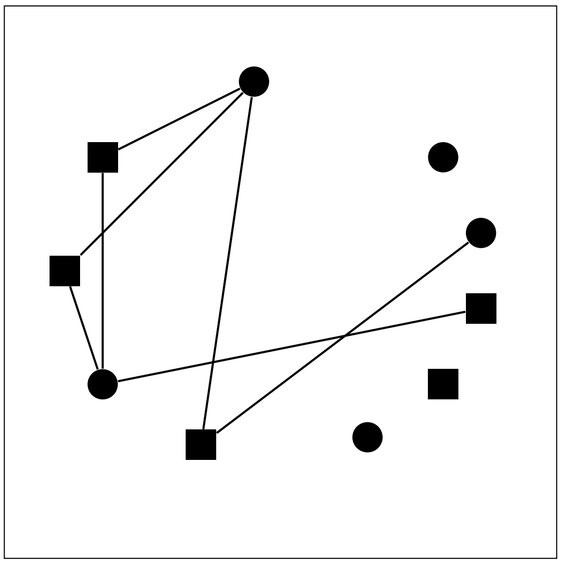
TRUE FALSE

More than 1 circle and more than 3 squares are connected by lines.



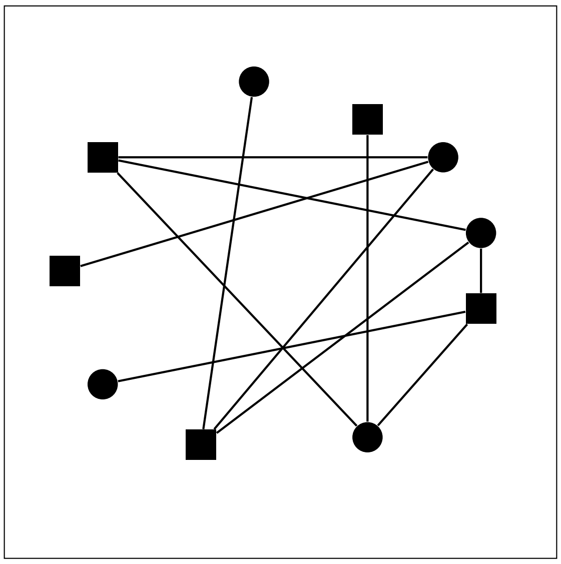
TRUE FALSE

More than 2 squares and more than 1 circle are connected by lines.



TRUE FALSE

More than 2 circles and more than 2 squares are connected by lines.



TRUE FALSE

Appendix B

Polish Version of the Test from Chapter 6

PIERWSZY TEST

Instrukcja: Na następnych stronach znajduje się 20 zadań. W każdym zadaniu przedstawione jest pewne wnioskowanie. Twoim celem jest stwierdzić, czy jest to wnioskowanie poprawne. Innymi słowy, każde zadanie składa się z 2 zdań oddzielonych od siebie poziomą linią. Musisz zdecydować czy zdanie znajdujące się pod linią wynika ze zdania znajdującego się nad linią.

Jeśli uważasz, że wnioskowanie jest poprawne (drugie zdanie wynika z pierwszego) zakreśl: „POPRAWNE”, w przeciwnym przypadku zakreśl: „NIE POPRAWNE”.

Przykład 1:

$$\frac{\text{Co najmniej 5 strzew jest kretowcami.}}{\text{Co najmniej 3 strzewa są kretowcami.}}$$

Co najmniej 5 strzew jest krętowcami.

Co najmniej 3 strzewa są krętowcami.

[✓ POPRAWNE] NIE POPRAWNE

Przykład 2:

Co najwyżej 5 memniaków jest stularami.

Co najwyżej 3 balbasze są stularami.

POPRAWNE [✓ NIE POPRAWNE]

Więcej niż 6 protorożców jest wypsztykami.

Więcej niż 5 protorożców jest wypsztykami.

POPRAWNE NIE POPRAWNE

Więcej niż 12 protorożców i więcej niż 13 melarków nienawidzi się wzajemnie.

Więcej niż 13 melarków i więcej niż 12 protorożców nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 16 krętowców i więcej niż 9 memniaków nienawidzi się wzajemnie.

Więcej niż 9 memniaków i więcej niż 16 krętowców nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 16 strzew i więcej niż 25 memniaków nienawidzi się wzajemnie.

Więcej niż 25 memniaków i więcej niż 16 strzew nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 10 strzew jest protorożcami.

Więcej niż 11 strzew jest protorożcami.

POPRAWNE NIE POPRAWNE

Więcej niż 9 stularów i więcej niż 8 melarków nienawidzi się wzajemnie.

Więcej niż 8 melarków i więcej niż 9 stularów nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 20 wachlaczy i więcej niż 35 fisutów nienawidzi się wzajemnie.

Więcej niż 20 fisutów i więcej niż 35 wachlaczy nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Wszystkie wachlacze są protorożcami.

Wszystkie protorożce są wachlaczami.

POPRAWNE NIE POPRAWNE

Więcej niż 100 wachlaczy i więcej niż 150 kretowców nienawidzi się wzajemnie.

Więcej niż 150 kretowców i więcej niż 100 wachlaczy nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 105 wachlaczy i więcej niż 68 melarków nienawidzi się wzajemnie.

Więcej niż 68 wachlaczy i więcej niż 105 melarków nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 6 balbaszy i więcej niż 5 fisutów nienawidzi się wzajemnie.

Więcej niż 5 fisutów i więcej niż 6 balbaszy nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 47 krętowców i więcej niż 55 burbaków nienawidzi się wzajemnie.

Więcej niż 47 burbaków i więcej niż 55 krętowców nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 58 stularów i więcej niż 49 bubraków nienawidzi się wzajemnie.

Więcej niż 49 stularów i więcej niż 58 bubraków nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 7 melarków i więcej niż 6 balbaszy nienawidzi się wzajemnie.

Więcej niż 6 balbaszy i więcej niż 7 melarków nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Pewne wypsztyki są strzewami.

Pewne strzewa są wypsztykami.

POPRAWNE NIE POPRAWNE

Więcej niż 99 melarków i więcej niż 68 wypsztyków nienawidzi się wzajemnie.

Więcej niż 68 wypsztyków i więcej niż 99 melarków nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 7 burbaków i więcej niż 8 fisutów nienawidzi się wzajemnie.

Więcej niż 8 fisutów i więcej niż 7 burbaków nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 19 stularów i więcej niż 11 protorożców nienawidzi się wzajemnie.

Więcej niż 11 protorożców i więcej niż 19 stularów nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 159 krętowców i więcej niż 25 fisutów nienawidzi się wzajemnie.

Więcej niż 159 fisutów i więcej niż 25 krętowców nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

Więcej niż 8 stularów i więcej niż 27 balbaszy nienawidzi się wzajemnie.

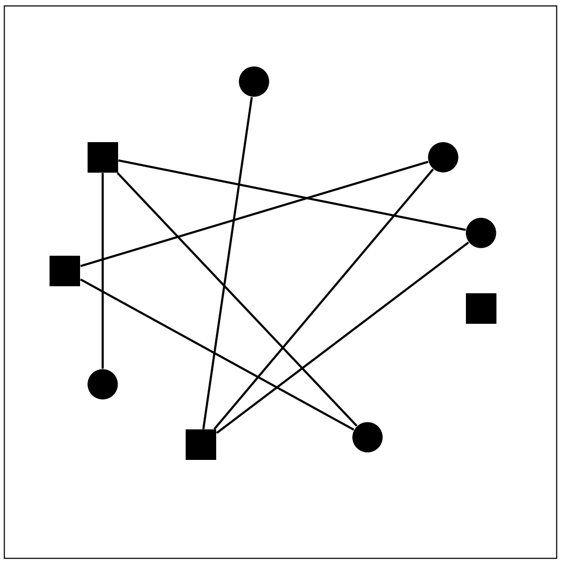
Więcej niż 27 stularów i więcej niż 8 balbaszy nienawidzi się wzajemnie.

POPRAWNE NIE POPRAWNE

DRUGI TEST

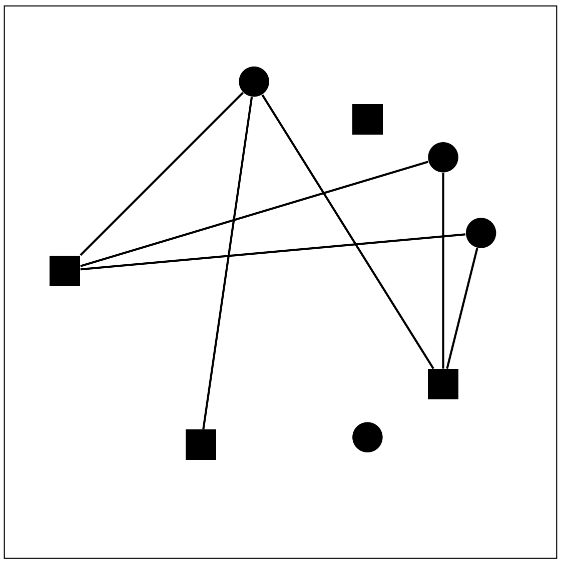
Instrukcja: Na następnych stronach znajdziesz 9 zadań. Na każde zadanie składa się obrazek. Nad każdym obrazkiem jest jedno zdanie. Zakreśl PRAWDA gdy zdanie prawdziwe opisuje obrazek. W przeciwnym przypadku zakreśl FAŁSZ.

Więcej niż 1 kwadrat i więcej niż 2 koła są połączone liniami.



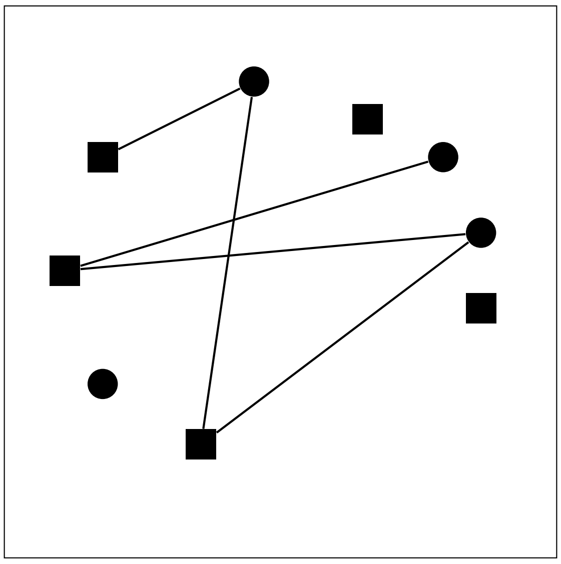
PRAWDA FAŁSZ

Więcej niż 3 koła i więcej niż 2 kwadraty są połączone liniami.



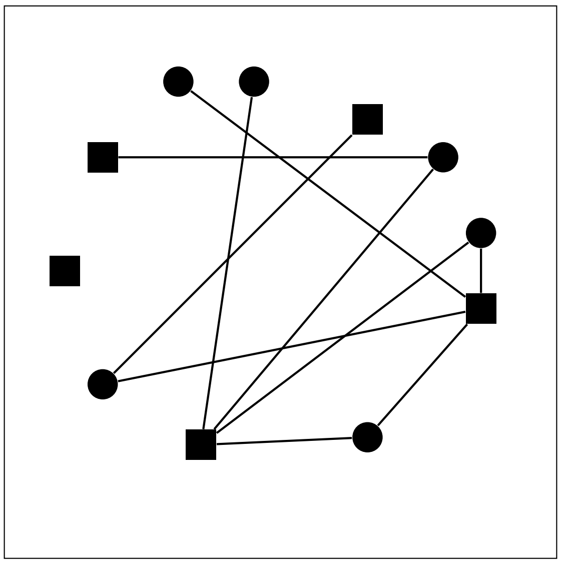
PRAWDA FAŁSZ

Więcej niż 1 kwadrat i więcej niż 1 koło są połączone liniami.



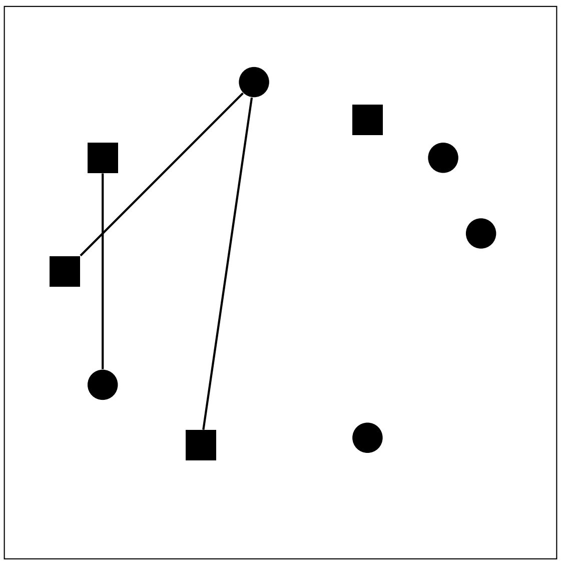
PRAWDA FAŁSZ

Więcej niż 3 koła i więcej niż 1 kwadrat są połączone liniami.



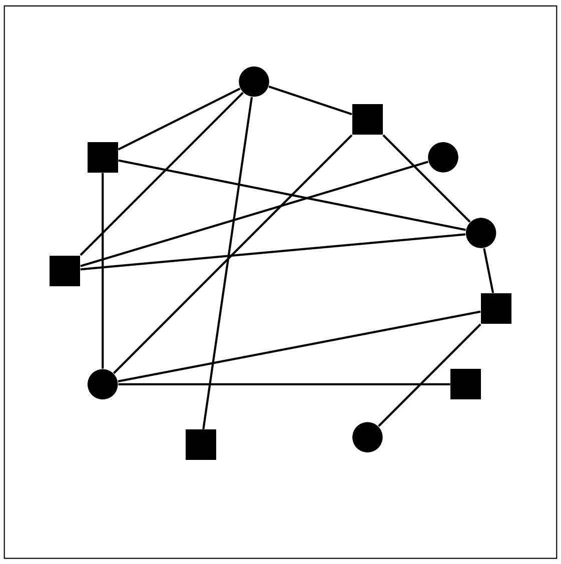
PRAWDA FAŁSZ

Więcej niż 3 koła i więcej niż 3 kwadraty są połączone liniami.



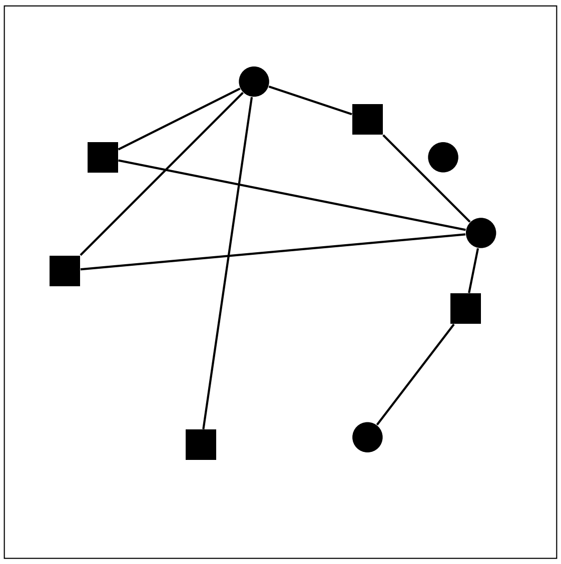
PRAWDA FAŁSZ

Więcej niż 2 koła i więcej niż 3 kwadraty są połączone liniami.



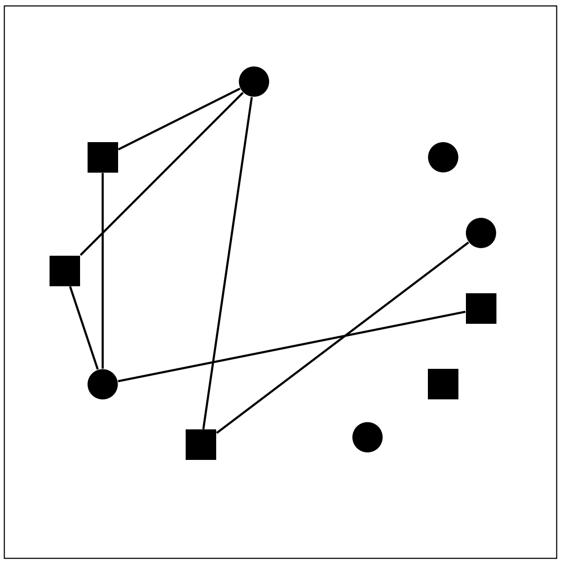
PRAWDA FAŁSZ

Więcej niż 1 koło i więcej niż 3 kwadraty są połączone liniami.



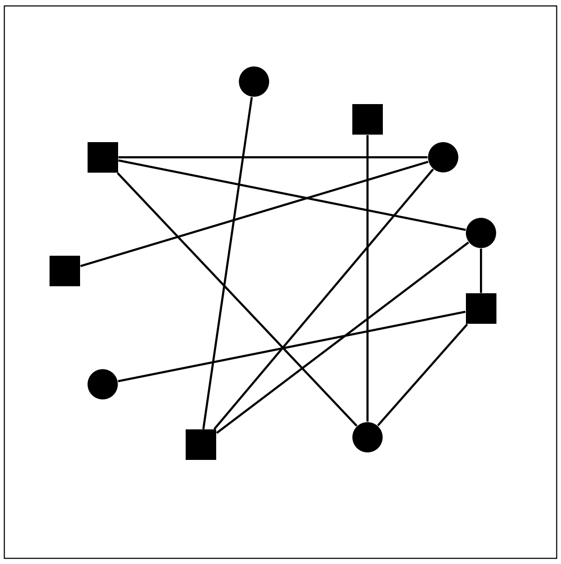
PRAWDA FAŁSZ

Więcej niż 2 kwadraty i wiecej niż 1 koło są połączone liniami.



PRAWDA FAŁSZ

Więcej niż 2 koła i więcej niż 2 kwadraty są połączone liniami.



PRAWDA FAŁSZ

Bibliography

Aczel, P. (1975). Quantifiers, games and inductive definitions. In Kanger, S., editor, *Proceedings of the Third Scandinavian Logic Symposium*, volume 82 of *Studies in Logic and Foundations of Mathematics*, pages 1–14. Elsevier.

Agrawal, M., Kayal, N., and Saxena, N. (2004). Primes in P. *Annals of Mathematics*, 160(2):781–793.

Ajdukiewicz, K. (1931). O znaczeniu wyrażeń. In *Księga Pamiątkowa Polskiego Towarzystwa Filozoficznego we Lwowie.* PTF, Lviv.

Andersson, A. (2002). On second-order generalized quantifiers and finite structures. *Annals of Pure and Applied Logic*, 115(1–3):1–32.

Austin, J. (1975). *How to do things with words.* The William James Lectures delivered at Harvard University in 1955. Cambridge, Harvard UP.

Bach, K. (1982). Semantic nonspecificity and mixed quantifiers. *Linguistics and Philosophy*, 4(4):593–605.

Baddeley, A. (2007). *Working Memory, Thought, and Action.* Oxford Psychology Series. Oxford University Press, USA, 1st edition.

Bar-Hillel, Y. (1954). Indexical expressions. *Mind*, 63:359–379.

Barton, E. G., Berwick, R., and Ristad, E. S. (1987). *Computational Complexity and Natural Language.* Bradford Books. The MIT Press.

Bartsch, R. (1973). The semantics and syntax of number and numbers. In Kimball, J. P., editor, *Syntax and Semantics 2*, pages 51–93. Seminar Press, New York.

Barwise, J. (1979). On branching quantifiers in English. *Journal of Philosophical Logic*, 8:47–80.

Barwise, J. and Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219.

Beck, S. (2000). The semantics of different: Comparison operator and relational adjective. *Linguistics and Philosophy*, 23(2):101–139.

Beghelli, F., Ben-Shalom, D., and Szabolcsi, A. (1997). Variation, distributivity, and the illusion of branching. In Szabolcsi, A., editor, *Ways of Scope Taking*, volume 65 of *Studies in Linguistic and Philosophy*, pages 29–69. Kluwer Academic Publisher.

Bellert, I. (1989). *Feature System for Quantification Structures in Natural Language.* Foris Publications, Dordrecht.

Ben-Avi, G. and Winter, Y. (2003). Monotonicity and collective quantification. *Journal of Logic, Language and Information*, 12(2):127–151.

Ben-Avi, G. and Winter, Y. (2004). A characterization of monotonicity with collective quantifiers. *Electronic Notes in Theoretical Computer Science*, 53:21–33.

Benacerraf, P. (1967). God, the devil, and Gödel. *The Monist*, 51:9–32.

Bennett, M. R. (1974). *Some extensions of a Montague fragment of English.* PhD thesis, University of California, Los Angeles.

van Benthem (2007). Review of Stanley Peters and Dag Westerståhl 'Quantifiers in Language and Logic'. *Notre Dame Philosophical Reviews.*

van Benthem and Pacuit, E. (2006). The tree of knowledge in action: Towards a common perspective. In Governatori, G., Hodkinson, I. M., and Venema, Y., editors, *Advances in Modal Logic*, pages 87–106. College Publications.

van Benthem, J. (1983). The semantics of variety in categorial grammar. Report 83–29, Department of Mathematics, Simon Fraser University. Published in Buszkowski, W., Marciszewski, W., and van Benthem, J. (Eds.) (1988). Categorial grammar, linguistic and literary studies in Eastern Europe Volume 25. Amsterdam: John Benjamins, pp. 37–55.

van Benthem, J. (1986). *Essays in logical semantics.* Reidel.

van Benthem, J. (1987). Towards a computational semantics. In Gärdenfors, P., editor, *Generalized Quantifiers*, pages 31–71. Reidel Publishing Company.

van Benthem, J. (1989). Polyadic quantifiers. *Linguistics and Philosophy*, 12(4):437–464.

van Benthem, J. (1995). *Language in action.* MIT Press, Cambridge, MA.

van Benthem, J. (2002). Extensive games as process models. *Journal of Logic, Language and Information*, 11(3):289–313.

van Benthem, J. (2003). Logic games are complete for game logics. *Studia Logica*, 75(2):183–203.

van Benthem, J., Ghosh, S., and Liu, F. (2007). Modelling simultaneous games with concurrent dynamic logic. In van Benthem, J., Ju, S., and Veltman, F., editors, *A Meeting of the Minds-Proceedings of the Workshop on Logic, Rationality and Interaction*, Beijing.

Benz, A., Jager, G., and van Rooij, R., editors (2005). *Game Theory and Pragmatics.* Palgrave Studies in Pragmatics, Language & Cognition. Palgrave Macmillan.

van den Berg, M. (1996). Dynamic generalized quantifiers. In van der Does, J. and van Eijck, J., editors, *Quantifiers, Logic, and Language*, CSLI Lecture Notes, pages 63–94. Stanford University, California.

Blass, A. and Gurevich, Y. (1986). Henkin quantifiers and complete problems. *Annals of Pure and Applied Logic*, 32:1–16.

Blass, A. and Gurevich, Y. (2003). Algorithms: A quest for absolute definitions. *Bulletin of European Association for Theoretical Computer Science*, 81:195–225.

Bogdanov, A. and Trevisan, L. (2006). Average-case complexity. *Foundational Trends in Theoretical Computer Science*, 2(1):1–106.

Büchi, J. (1960). Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grudlagen der Mathematik*, 6:66–92.

Burtschick, H. J. and Vollmer, H. (1998). Lindström quantifiers and leaf language definability. *International Journal of Foundations of Computer Science*, 9(3):277–294.

Carnap, R. (2007). *Meaning And Necessity - A Study In Semantics And Modal Logic.* Clarke Press.

Chalmers, D. (1994). A computational foundation for the study of cognition. See: http://consc.net/papers/computation.html.

Cherniak, C. (1981). Minimal rationality. *Mind*, 90(358):161–183.

Chomsky, N. (1969). *Aspects of the Theory of Syntax.* The MIT Press.

Chomsky, N. (2002). *Syntactic Structures*. Walter de Gruyter, 2nd edition.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363.

Church, A. (1973). Outline of a revised formulation of the logic of sense and denotation (part i). *Noûs*, 7(1):24–33.

Church, A. (1974). Outline of a revised formulation of the logic of sense and denotation (part ii). *Noûs*, 8(2):135–156.

Clark, R. (2007). Games, quantifiers and pronouns. In Pietarinen, A. V., editor, *Game Theory and Linguistic Meaning*, Current Research in the Semantics/Pragmatics Interface, pages 207–228. Elsevier Science.

Clark, R. and Grossman, M. (2007). Number sense and quantifier interpretation. *Topoi*, 26(1):51–62.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of Computing*, pages 151–158, New York, NY, USA. ACM Press.

Cooper, B. S. (2003). *Computability Theory*. Chapman Hall/Crc Mathematics Series. Chapman & Hall/CRC.

Culy, C. (1985). The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, 8(3):345–351.

Dalrymple, M., Kanazawa, M., Kim, Y., Mchombo, S., and Peters, S. (1998). Reciprocal expressions and the concept of reciprocity. *Linguistics and Philosophy*, 21:159–210.

Dekker, P. (2008). A guide to dynamic semantics. Preprint series PP-2008-42, Institute for Logic, Language and Information. Manuscript available at http://www.illc.uva.nl/Publications/ResearchReports/PP-2008-42.text.pdf.

van Ditmarsch, H., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic*. (Synthese Library). Springer, 1st edition.

van der Does, J. (1992). *Applied quantifier logics. Collectives and naked infinitives.* PhD thesis, Universiteit van Amsterdam.

van der Does, J. (1993). Sums and quantifiers. *Linguistics and Philosophy*, 16(5):509–550.

Downey, R. G. and Fellows, M. R. (1998). *Parameterized Complexity*. Monographs in Computer Science. Springer.

Dummett, M. (1978). *Truth and Other Enigmas.* Harvard University Press.

Ebbinghaus, H.-D. and Flum, J. (2005). *Finite Model Theory.* Springer Monographs in Mathematics. Springer.

Ebbinghaus, H. D., Flum, J., and Thomas, W. (1996). *Mathematical Logic.* Undergraduate Texts in Mathematics. Springer.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467.

Fagin, R. (1974). Generalized first-order spectra an polynomial time recognizable sets. In Karp, R., editor, *Complexity of Computation*, volume 7 of *SIAM—AMS Proceedings*, pages 43–73. American Mathematical Society.

Ferguson, G. A. and Takane, Y. (1990). *Statistical Analysis in Psychology and Education.* McGraw-Hill Education.

Flum, J. and Grohe, M. (2006). *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 1st edition.

Frege, G. (1879). *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.* Halle.

Frege, G. (1892). Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100:25–50.

Frege, G. (1980). *Philosophical and Mathematical Correspondence of Gottlob Frege.* University Of Chicago Press.

Frixione, M. (2001). Tractable competence. *Minds and Machines*, 11(3):379–397.

Gabbay, D. M. and Moravcsik, J. M. E. (1974). Branching quantifiers, English and Montague grammar. *Theoretical Linguistics*, 1:140–157.

Gärdenfors, P. (2003). *Belief Revision.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability.* W. H. Freeman and Co., San Francisco.

Geurts, B. (2003). Reasoning with quantifiers. *Cognition*, 86(3):223–251.

Geurts, B. and van der Silk, F. (2005). Monotonicity and processing load. *Journal of Semantics*, 22:97–117.

Gierasimczuk, N. (2007). The problem of learning the semantics of quantifiers. In Ten Cate, B. and Zeevat, H., editors, *Logic, Language, and Computation, 6th International Tbilisi Symposium on Logic, Language, and Computation, TbiLLC 2005*, volume 4363 of *Lecture Notes in Computer Science*, pages 117–126, Batumi, Georgia. Springer.

Gierasimczuk, N. (2009). Identification through inductive verification. Application to monotone quantifiers. In Bosch, P., Gabelaia, D., and Lang, J., editors, *Logic, Language, and Computation, 7th International Tbilisi Symposium on Logic, Language, and Computation, TbiLLC 2007*, volume 5422 of *Lecture Notes on Artificial Inteligence*, pages 193–205, Tbilisi, Georgia. Springer.

Gierasimczuk, N. and Szymanik, J. (2006). Hintikka's thesis revisited. Prepublications series PP-2006-35, Institute for Logic, Language and Information. Manuscript avaiable at: `http://www.illc.uva.nl/Publications/ResearchReports/PP-2006-35.text-Dec-2006.pdf`.

Gierasimczuk, N. and Szymanik, J. (2007). Hintikka's thesis revisited. *The Bulletin of Symbolic Logic*, 17:273–273.

Gierasimczuk, N. and Szymanik, J. (2008). Interpreting quantifier combinations. Hintikka's thesis revisited. Prepublications series PP-2006-35 august 2008, Institute for Logic, Language and Information. Manuscript avaiable at: `http://www.illc.uva.nl/Publications/ResearchReports/PP-2006-35.text-Aug-2008.pdf`.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10:447–474.

Gottlob, G., Leone, N., and Veith, H. (1999). Succinctness as a source of complexity in logical formalisms. *Annals of Pure and Applied Logic*, pages 231–260.

Grädel, E. and Gurevich, Y. (1998). Metafinite model theory. *Information and Computation*, 140(1):26–81.

Grädel, E., Kolaitis, P. G., Libkin, L., Marx, M., Spencer, J., Vardi, M. Y., Venema, Y., and Weinstein, S. (2007). *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Graham, R. L., Rothschild, B. L., and Spencer, J. H. (1990). *Ramsey Theory*. John Wiley & Sons, New York.

Grice, P. (1991). *Studies in the Way of Words*. Harvard University Press.

Groenendijk, J. and Stokhof, M. (1991). Dynamic predicate logic. *Linguistics and Philosophy*, 14(1):39–100.

Guenthner, F. and Hoepelman, J. P. (1976). A note on the representation of branching quantifiers. *Theoretical Linguistics*, 3:285–289.

Gurevich, Y. (1995). The Value, if any, of Decidability. *Bulletin of European Association for Theoretical Computer Science*, pages 129–135.

Hamm, F. and Lambalgen, M. (2004). Moschovaki's notion of meaning as applied to linguistics. In Baaz, M., Friedman, S., Krajicek, J., and Peters, A. K., editors, *ASL Lecture Notes in Logic, Logic Colloqium'01*, pages 167–183. A. K. Peters Publishers.

Heim, I., Lasnik, H., and May, R. (1991). Reciprocity and plurality. *Linguistic Inquiry*, 22(1):63–101.

Hella, L. (1996). Logical hierarchies in PTIME. *Information and Computation*, 129(1):1–19.

Hella, L., Kolaitis, P. G., and Luosto, K. (1996). Almost everywhere equivalence of logics in finite model theory. *Bulletin of Symbolic Logic*, 2(4):422–443.

Hella, L., Väänänen, J., and Westerståhl, D. (1997). Definability of polyadic lifts of generalized quantifiers. *Journal of Logic, Language and Information*, 6(3):305–335.

Henkin, L. (1961). Some remarks on infinitely long formulas. In *Infinistic Methods*, pages 167–183. Pergamon Press.

Hintikka, J. (1973). Quantifiers vs. quantification theory. *Dialectica*, 27:329–358.

Hintikka, J. (1976). Partially ordered quantifiers vs. partially ordered ideas. *Dialectica*, 30:89–99.

Hintikka, J. (1996). *Principles of Mathematics Revisited.* Cambridge University Press.

Hintikka, J. (1997). *Paradigms for Language Theory and Other Essays.* Jaakko Hintikka Selected Papers. Springer, 1st edition.

Hintikka, J. and Sandu, G. (1997). Game-theoretical semantics. In van Benthem, J. and ter Meulen, A., editors, *Handbook of Logic and Language*, pages 361–410. Elsevier, Amsterdam.

Hodges, W. (1997). Compositional semantics for a language of imperfect information. *Journal of the Interest Group in Pure and Applied Logics*, 5 (4):539–563.

Hofstadter, D. (2007). *Gödel, Escher, Bach.* Tusquets, 3rd edition.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2000). *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2nd edition.

Immerman, N. (1982). Relational queries computable in polynomial time (extended abstract). In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 147–152, New York, NY, USA. ACM.

Immerman, N. (1998). *Descriptive Complexity*. Texts in Computer Science. Springer.

Impagliazzo, R. (1995). A personal view of average-case complexity. In *SCT '95: Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, Washington, DC, USA. IEEE Computer Society.

Isac, D. and Reiss, C. (2008). *I-Language: An Introduction to Linguistics as Cognitive Science*. Core Linguistics. Oxford University Press, USA.

Jackendoff, R. (1972). *Semantic Interpretation and Generative Grammar*. MIT Press, Cambridge, Massachuset.

Janssen, T. (2002). Independent choices and the interpretation of IF-logic. *Journal of Logic, Language and Information*, 11:367–387.

Janssen, T. and Dechesne, F. (2006). Signalling in IF games: a tricky business. In Johan van Benthem, Gerhard Heinzmann, M. R. and Visser, H., editors, *The age of alternative logics. Assessing philosophy of logic and mathematics today*, volume 3 of *Logic, epistemology, and the unity of science*, chapter 15, pages 221–241. Springer, Dordrecht, the Netherlands.

Jaszczolt, K. (2002). *Semantics and Pragmatics: Meaning in Language and Discourse*. Longman Linguistics Library. Longman.

Kager, R. (1999). *Optimality Theory*. Cambridge Textbooks in Linguistics. Cambridge University Press.

Kamp, H. (1971). Formal properties of 'now'. *Theoria*, 37:227–273.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Studies in Linguistics and Philosophy. Springer.

Kamp, H. and Stokhof, M. (2008). Information in natural language. In van Benthem, J. and Adriaans, P., editors, *Philosophy of Information*, Handbook of the Philosophy of Science. North-Holland.

Kandel, E. R., Schwartz, J. H., and Jessell, T. M. (2000). *Principles of Neural Science*. McGraw-Hill Medical.

Kaplan, D. (1979). On the logic of demonstratives. *Journal of Philosophical Logic*, 8(1):81–98.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Kempson, R. M. and Cormack, A. (1981a). Ambiguity and quantification. *Linguistics and Philosophy*, 4(2):259–309.

Kempson, R. M. and Cormack, A. (1981b). On 'formal games and forms for games'. *Linguistics and Philosophy*, 4(3):431–435.

Kempson, R. M. and Cormack, A. (1982). Quantification and pragmatics. *Linguistics and Philosophy*, 4(4):607–618.

Kontinen, J. (2002). Second-order generalized quantifiers and natural language. In Nissim, M., editor, *Proceedings of the Seventh ESSLLI Student Session*, pages 107–118. Elsevier.

Kontinen, J. (2004). *Definability of second order generalized quantifiers*. PhD thesis, Helsinki University.

Kontinen, J. and Niemistö, H. (2006). Extensions of MSO and the monadic counting hierarchy. Manuscript available at http://www.helsinki.fi/~jkontine/.

Kontinen, J. and Szymanik, J. (2008). A remark on collective quantification. *Journal of Logic, Language and Information*, 17(2):131–140.

Kozen, D. C. (2006). *Theory of Computation: Classical and Contemporary Approaches*. Texts in Computer Science. Springer.

Kracht, M. (2003). *The Mathematics of Language*, volume 63 of *Studies in Generative Grammar*. Walter de Gruyter.

Kripke, S. (2008). Frege's theory of sense and reference: Some exegetical notes. *Theoria*, 74(3):181–218.

Krynicki, M. and Mostowski, M. (1995). Henkin quantifiers. In Krynicki, M., Mostowski, M., and Szczerba, L., editors, *Quantifiers: Logics, Models and Computation*, pages 193–263. Kluwer Academic Publishers.

Krynicki, M. and Mostowski, M. (1999). Ambigous quantifiers. In Orłowska, E., editor, *Logic at work*, pages 548–565. Heidelberg.

Kugel, P. (1986). Thinking may be more than computing. *Cognition*, 22(2):137–198.

Ladusaw, W. (1979). *Polarity Sensitivity as Inherent Scope Relations.* PhD thesis, University of Texas, Austin.

Lambalgen, M. V. and Hamm, F. (2005). *The Proper Treatment of Events.* Explorations in Semantics. Wiley-Blackwell.

Landman, F. (2000). Against binary quantifiers. In *Events and Plurality*, Studies in Linguistic and Philosophy, pages 310–349. Kluwer Academic Publisher.

Levesque, H. J. (1988). Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17(4):355–389.

Lewis, D. (2002). *Convention: A Philosophical Study.* Wiley-Blackwell.

Libkin, L. (2004). *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer.

Lindström, P. (1966). First order predicate logic with generalized quantifiers. *Theoria*, 32:186–195.

Link, G. (1983). The logical analysis of plurals and mass terms: A lattice-theoretical approach. In Bäuerle, R., Schwarze, C., and von Stechow, A., editors, *Meaning, Use, and Interpretation of language*, pages 302–323. Gruyter, Berlin.

Liu, F.-H. (1996). Branching quantification and scope independence. In van der Does, J. and van Eijck, J., editors, *Quantifiers, Logic and Language*, pages 155–168. Center for the Study of Language and Information.

Lønning, J. T. (1997). Plurals and collectivity. In van Benthem, J. and ter Meulen, A., editors, *Handbook of Logic and Language*, pages 1009–1053. Elsevier.

Lorentzen, P. (1955). *Einführung in die Operative Logik und Mathematik.* Springer, Berlin.

Lucas, J. R. (1961). Minds, machines and Gödel. *Philosophy*, 36(137):112–127.

Luo, H., Su, W., and Li, Z. (2002). The properties of self-complementary graphs and new lower bounds for diagonal Ramsey numbers. *Australasian Journal of Combinatorics*, 25:103–116.

Luosto, K. (1999). Ramsey theory is needed for solving definability problems of generalized quantifiers. In *Lecture Notes in Computer Science*, volume 1754, pages 121–134.

Macintyre, A. (1980). Ramsey quantifiers in arithmetic. In Pacholski, L., Wierzejewski, J., and Wilkie, A. J., editors, *Model theory of algebra and arithmetics*, volume 834 of *Lecture Notes in Mathematics*, pages 186–210. Springer–Verlag.

Magidor, M. and Malitz, J. I. (1977). Compact extensions of L(Q). *Annals of Mathematical Logic*, 11:217–261.

Makowsky, J. and Pnueli, Y. (1995). Computable quantifiers and logics over finite structures. In Krynicki, M., Mostowski, M., and Szczerba, L., editors, *Quantifiers: Logics, Models and Computation*, pages 313–357. Kluwer Academic Publishers.

Manaster-Ramer, A. (1987). Dutch as a formal language. *Linguistics and Philosophy*, 10(2):221–246.

Marr, D. (1983). *Vision: A Computational Investigation into the Human Representation and Processing Visual Information.* W.H. Freeman, San Francisco.

May, R. (1985). *Logical Form: Its Structure and Derivation.* Linguistic Inquiry Monographs. The MIT Press.

May, R. (1989). Interpreting logical form. *Linguistics and Philosophy*, 12(4):387–435.

McMillan, C. T., Clark, R., Moore, P., Devita, C., and Grossman, M. (2005). Neural basis for generalized quantifier comprehension. *Neuropsychologia*, 43:1729–1737.

Mcmillan, C. T., Clark, R., Moore, P., and Grossman, M. (2006). Quantifiers comprehension in corticobasal degeneration. *Brain and Cognition*, 65:250–260.

McNaughton, R. and Papert, S. A. (1971). *Counter-Free Automata.* M.I.T. Research Monograph no. 65. The MIT Press.

Miestamo, M., Sinnemäki, K., and Karlsson, F., editors (2008). *Language Complexity: Typology, contact, change.* Studies in Language Companion Series. John Benjamins Publishing Company.

Montague, R. (1970). Pragmatics and intensional logic. *Dialectica*, 24(4):277–302.

Moschovakis, Y. (1990). Sense and denotation as algorithm and value. In Oikkonen, J. and Väänänen, J., editors, *Lecture Notes in Logic 2*, pages 210–249. Springer.

Moschovakis, Y. (2001). What is an algorithm? In Enquist, B. and Schmid, W., editors, *Mathematics unlimited – 2001 and beyond*, pages 919–936. Springer.

Moschovakis, Y. (2006). A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29(1):27–89.

Moss, L. and Tiede, H. J. (2006). Applications of modal logic in linguistics. In Blackburn, P., van Benthem, J. F. A. K., and Wolter, F., editors, *Handbook of Modal Logic*, Studies in Logic and Practical Reasoning, pages 1031–1077. Elsevier Science.

Mostowski, A. (1957). On a generalization of quantifiers. *Fundamenta Mathematicae*, 44:12–36.

Mostowski, M. (1991). Arithmetic with the Henkin quantifier and its generalizations. In Gaillard, F. and Richard, D., editors, *Seminaire du Laboratoire Logique, Algorithmique et Informatique Clermontois*, volume II, pages 1–25.

Mostowski, M. (1994). Kwantyfikatory rozgałęzione a problem formy logicznej. In Omyła, M., editor, *Nauka i język*, pages 201–242. Biblioteka Myśli Semiotycznej.

Mostowski, M. (1995). Quantifiers definable by second order means. In Krynicki, M., Mostowski, M., and Szczerba, L., editors, *Quantifiers: Logics, Models and Computation*, pages 181–214. Kluwer Academic Publishers.

Mostowski, M. (1998). Computational semantics for monadic quantifiers. *Journal of Applied Non-Classical Logics*, 8:107–121.

Mostowski, M. (2008). Potential infinity and the Church Thesis. *Fundamenta Informaticae*, 81(1-3):241–248.

Mostowski, M. and Szymanik, J. (2005). Semantical bounds for everyday language. *Semiotica*, to appear. See also ILLC Preprint Series, PP-2006-40; Manuscript avaiable at: http://www.illc.uva.nl/Publications/ResearchReports/PP-2006-40.text.pdf.

Mostowski, M. and Szymanik, J. (2007). Computational complexity of some Ramsey quantifiers in finite models. *The Bulletin of Symbolic Logic*, 13:281–282.

Mostowski, M. and Wojtyniak, D. (2004). Computational complexity of the semantics of some natural language constructions. *Annals of Pure and Applied Logic*, 127(1-3):219–227.

Muskens, R. (2005). Sense and the computation of reference. *Linguistics and Philosophy*, 28(4):473–504.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V., editors (2007). *Algorithmic Game Theory*. Cambridge University Press.

Otto, M. (1997). *Bounded Variable Logics and Counting. A study in Finite Models.*, volume 9 of *Lecture Notes in Logic*. Springer-Verlag.

Pagin, P. (2009). Communication and the complexity of semantics. In Hinzen, W., Machery, E., and Werning, M., editors, *Oxford Handbook of Compositionality*. Forthcoming; Manuscript available at http://people.su.se/~ppagin/pagineng.htm.

Papadimitriou, C. H. (1993). *Computational Complexity*. Addison Wesley.

Paris, J. and Harrington, L. (1977). A mathematical incompleteness in Peano Arithmetics. In Barwise, J., editor, *Handbook for mathematical logic*, pages 1133–1142. North–Holland.

Partee, B. and Rooth, M. (1983). Generalized conjunction and type ambiguity. In Bäuerle, R., Schwarze, C., and von Stechow, A., editors, *Meaning, use, and interpretation of language*, pages 361–383. Gruyter, Berlin.

Partee, B. H., ter Meulen, A. G., and Wall, R. (1990). *Mathematical Methods in Linguistics*. Studies in Linguistics and Philosophy. Springer.

Penrose, R. (1996). *Shadows of the Mind : A Search for the Missing Science of Consciousness*. Oxford University Press, USA.

Peregrin, J. (2003). *Meaning: The Dynamic Turn*. Elsevier Science.

Peters, S. and Westerståhl, D. (2006). *Quantifiers in Language and Logic*. Clarendon Press, Oxford.

Piaget, J. (2001). *Psychology of Intelligence*. Routledge Classics. Routledge, 1st edition.

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 Clark, A., Fox, C., and Lappin, S., editors, *Computational Linguistics and Natural Language Processing Handbook*. Blackwell. Forthcoming; Manuscript avaiable at: http://www.cs.man.ac.uk/~ipratt/papers/nat_lang/handbook.ps.

Pudlak, P. (1999). A note on applicability of the incompleteness theorem to human mind. *Annals of Pure and Applied Logic*, 96(1-3):335–342.

Pullum, G. K. and Gazdar, G. (1982). Natural languages and context-free languages. *Linguistics and Philosophy*, 4(4):471–504.

Putnam, H. (1975/1985). The meaning of 'meaning'. In *Philosophical Papers. Mind, Language and Reality.*, volume 2, pages 215–271. Cambridge University Press.

Quine, W. V. (1964). *Word and Object.* Studies in Communication. The MIT Press.

Ramsey, F. (1929). On a problem of formal logic. In *Proceedings of the London Mathematical Society*, volume 30 of *2*, pages 338–384.

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. CSLI Publications, Stanford, CA.

van Rooij, I. (2004). *Tractable cognition: Complexity theory in cognitive psychology.* PhD thesis, University of Victoria, BC, Canada.

van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science: A Multidisciplinary Journal*, 32(6):939–984.

van Rooij, I., Stege, U., and Kadlec, H. (2005). Sources of complexity in subset choice. *Journal of Mathematical Psychology*, 49(2):160–187.

Russell, B. (1903). *The Principles of Mathematics.* Cambridge University Press, Cambridge.

Sabato, S. and Winter, Y. (2005). From semantic restrictions to reciprocal meanings. In *Proceedings of FG-MOL 2005.* CSLI Publications.

Sanford, A. J., Moxey, L. M., and Paterson, K. (1994). Psychological studies of quantifiers. *Journal of Semantics*, 11(3):153–170.

Savitch, W. (1970). Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 4:177–192.

Scha, R. (1981). Distributive, collective and cumulative quantification. In Groenendijk, J. A. G., Janssen, T. M. V., and Stokhof, M. B. J., editors, *Formal methods in the study of language, Part 2*, pages 483–512. Mathematisch Centrum, Amsterdam.

Schaefer, M. (2001). Graph Ramsey theory and the polynomial hierarchy. *Journal of Computer and System Sciences*, 62:290–322.

Schlenker, P. (2006). Scopal independence: A note on branching and wide scope readings of indefinites and disjunctions. *Journal of Semantics*, 23(3):281–314.

Schmerl, J. H. and Simpson, S. G. (1982). On the role of Ramsey Quantifiers in First Order Arithmetic. *Journal of Symbolic Logic*, 47:423–435.

Sevenster, M. (2006). *Branches of imperfect information: logic, games, and computation.* PhD thesis, Universiteit van Amsterdam.

Sher, G. (1990). Ways of branching quantifiers. *Linguistics and Philosophy*, 13:393–442.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

Soja, N., Carey, S., and Spelke, E. (1991). Ontological categories guide young children's induction of word meaning: Object terms and substance terms. *Cognition*, 38:179–211.

Spencer, J. H. (1987). *Ten lectures on the probabilistic methods.* Society for Industrial and Applied Mathematics.

Stalnaker, R. C. (2003). *Ways a World Might Be: Metaphysical and Anti-Metaphysical Essays.* Oxford University Press, USA.

Stenius, E. (1976). Comments on Jaakko Hintikka's paper "Quantifiers vs. quantification theory". *Dialectica*, 30:67–88.

Sternberg, R. J. (2008). *Cognitive Psychology.* Wadsworth Publishing, 5th edition.

Stockmeyer, L. J. (1976). The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1).

Sun, R., editor (2008). *The Cambridge Handbook of Computational Psychology.* Cambridge University Press, 1st edition.

Suppes, P. (1982). Variable-free semantics with remark on procedural extensions. In Simon and Scholes, editors, *Language, Mind and Brain*, pages 21–34. Hillsdale, Erlbaum.

Syropoulos, A. (2008). *Hypercomputation: Computing Beyond the Church-Turing Barrier.* Monographs in Computer Science. Springer, 1st edition.

Szymanik, J. (2005). Problemy z formą logiczną. *Studia Semiotyczne*, 25:187–200.

Szymanik, J. (2007a). A comment on a neuroimaging study of natural language quantifier comprehension. *Neuropsychologia*, 45(9):2158–2160.

Szymanik, J. (2007b). A strong meaning hypothesis from a computational perspective. In Aloni, M., Dekker, P., and Roelofsen, F., editors, *Proceedings of the Sixteenth Amsterdam Colloquium*, pages 211–216. University of Amsterdam.

Szymanik, J. (2008). The computational complexity of quantified reciprocals. In Bosch, P., Gabelaia, D., and Lang, J., editors, *Logic, Language, and Computation, 7th International Tbilisi Symposium on Logic, Language, and Computation, TbiLLC 2007*, volume 5422 of *Lecture Notes in Computer Science*, pages 139–152, Tbilisi, Georgia. Springer.

Szymanik, J. and Zajenkowski, M. (2008). Comprehension of simple quantifiers. Empirical evaluation of a computational model. Prepublications series PP-2008-49, Institute for Logic, Language and Information. Manuscript available at `http://www.illc.uva.nl/Publications/ResearchReports/PP-2008-49.text.pdf`.

Tarski, A. (1944). The semantic conception of truth: and the foundations of semantics. *Philosophy and Phenomenological Research*, 4(3):341–376.

Tennant, N. (1981). Formal games and forms for games. *Linguistics and Philosophy*, 4(2):311–320.

Tichý, P. (1969). Intension in terms of Turing machines. *Studia Logica*, 24(1):7–21.

Tichý, P. (1988). *The Foundations of Frege's Logic*. De Gruyter, Berlin.

Tiede, H. J. (1999). Identifiability in the limit of context-free generalized quantifiers. *Journal of Language and Computation*, 1:93–102.

Toda, S. (1991). PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877.

Troiani, V., Peelle, J., Clark, R., and Grossman, M. (2009). Is it logical to count on quantifiers? Dissociable neural networks underlying numerical and logical quantifiers. *Neuropsychologia*, 47(1):104–111.

Tsotsos, J. (1990). Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, 13(3):423–469.

Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265.

Väänänen, J. (1997a). Generalized quantifiers, an introduction. In *ESSLLI: European Summer School in Logic, Language, and Information, ESSLLI Workshop*, volume 1754 of *Lecture Notes in Computer Science*, pages 1–17. Springer.

Väänänen, J. (1997b). Unary quantifiers on finite models. *Journal of Logic, Language and Information*, 6(3):275–304.

Väänänen, J. (2007). *Dependence Logic — A New Approach to Independence Friendly Logic.* London Mathematical Society Student Texts. Cambridge University Press.

Väänänen, J. and Westerståhl, D. (2002). On the expressive power of monotone natural language quantifiers over finite models. *Journal of Philosophical Logic*, 31:327–358.

Vardi, M. Y. (1982). The complexity of relational query languages (extended abstract). In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 137–146, New York, NY, USA. ACM Press.

Westerståhl, D. (1984). Some results on quantifiers. *Notre Dame Journal of Formal Logic*, 25:152–169.

Winter, Y. (2001). *Flexibility principles in Boolean semantics.* The MIT Press, London.

Wittgenstein, L. (1922). *Tractatus Logico Philosophicus.* Routledge Classics. Routledge.

Wittgenstein, L. (1953). *Philosophical Investigations.* Blackwell Publishing, Incorporated, 50th anniversary commemorative edition edition.

Index

Samenvatting

In deze dissertatie bestuderen we de complexiteit van gegeneraliseerde kwantoren in de natuurlijke taal. We doen dit vanuit een interdisciplinair perspectief: we combineren filosofische inzichten met theoretische informatica, experimentele cognitiewetenschap en met theorieën uit de linguïstiek.

In het hoofdstuk 1 beargumenteren we dat een deel van de betekenis van een zin - de zogenaamde referentiële betekenis (*model-checking*) - beschreven moet worden met algoritmen. We bespreken ook de inbreng van complexiteitstheorie voor de analyse van cognitieve taken. We beargumenteren dat alleen die problemen die berekend kunnen worden in polynomiale tijd, cognitief bruikbaar *cognitively tractable* zijn. Ook verdedigen we dat semantische theorieën van de alledaagse natuurlijke taal geformuleerd kunnen worden in het existentiële fragment van de tweede-orde logica.

Hoofdstuk 2 bevat een overzicht van de basisnoties van de theorie van gegeneraliseerde kwantoren, berekenbaarheids-theorie, en van de beschrijvende complexiteitstheorie.

We beargumenteren in het hoofdstuk 3 dat PTIME kwantoren gesloten zijn onder iteratie (*iteration*), cumulatie (*cumulation*) en resumptie (*resumption*). Vervolgens bespreken we de NP-volledigheid van vertakkende kwantoren (*branching quantifiers*). We laten zien dat sommige Ramsey-kwantoren NP-volledige klassen van eindige modellen definiëren, terwijl andere alleen PTIME klassen definieren. We geven ook een voorwaarde voor een Ramsey kwantor om berekenbaar te zijn in polynomiale tijd. We beëindigen het hoofdstuk met een vraag betreffende het verschil in de complexiteit van verschillende Ramsey kwantoren.

Het hoofdstuk 4 bevat een onderzoek naar de rekenkundige complexiteit van *polyadic lifts* die verschillende lezingen uitdrukken van wederkerige zinnen (*reciprocal sentences*) met gekwantificeerde antecedenten. We laten een dichotomie zien tussen deze twee lezingen. De sterke wederkerige lezing kan namelijk NP-complete constructies creëren, terwijl de zwakke, bemiddeld wederkerige lezingen dit niet kunnen. We beargumenteren dat dit verschil verdisconteerd moet worden in de bekende 'Strong Meaning Hypothesis' (Sterke Betekenis Hypothese).

De definieerbaarheid en de complexiteit van de *type-schifting approach* van collectieve kwantificatie in de natuurlijke taal staat centraal in het hoofdstuk 5. We laten zien dat onder redelijke aannamen over de complexiteit, de kwantificatie niet algemeen genoeg is om de semantiek van alle collectieve kwantoren in de natuurlijke taal te omvatten. De *type-shifting approach* kan de tweede-orde logica niet overschrijden, terwijl sommige collectieve kwantoren kunnen niet in de tweede-orde logica worden uitgedrukt.

Vervolgens verdedigen we dat algebraïsche *many-sorted* formalismen die betrekking hebben op collectiviteit (*collectivity*) geschikter zijn dan de *type-shifting* benadering om collectieve kwantificatie in de natuurlijke taal te definieren. Het kan zo zijn dat sommige collectieve kwantoren niet in de natuurlijke taal aanwezig zijn, omdat ze een te grote complexiteit hebben. Ten slotte introduceren we zogenaamde tweede-orde gegeneraliseerde kwantoren in het onderzoek naar de collectieve semantiek.

Het hoofdstuk 6 handelt over de stelling van Hintikka, welke zegt dat zinnen zoals "de meeste jongens en de meeste meisjes haten elkaar" niet uitgedrukt kunnen worden door lineaire eerste orde formules, en dat vertakte kwantificatie (*branching quantification*) noodzakelijk is. We bespreken verschillende lezingen van zulke zinnen en beargumenteren dat ze de lezing hebben die *wel* uitgedrukt kan worden door lineaire formules, in tegenstelling tot wat Hintikka beweert. We presenteren empirisch bewijs ter ondersteuning van deze theoretische overwegingen.

In het hoofdstuk 7 bespreken we de semantiek van monadische kwantoren in de natuurlijke taal. Deze kan worden uitgedrukt in zogenaamde 'finite-state' en 'push-down' automata. We presenteren en bekritiseren vervolgens het neurologisch onderzoek dat zich baseert op dit model. Bouwend op deze discussie voltrekken we een experiment, dat empirisch bewijs levert dat de voorspellingen van het rekenkundige model bevestigt. We laten zien dat de verschillen in de tijd die een mens nodig heeft om zinnen met monadische kwantoren te begrijpen, consistent is met de verschillen in complexiteit die door het model voorspeld worden.

In het laatste hoofdstuk, 8, bespreken we een paar algemene, open vragen en mogelijke richtingen van verder onderzoek; met name het gebruik van verschillende maten van complexiteit, het betrekken van de speltheorie, et cetera.

Samenvattend; we onderzoeken, vanuit verschillende perspectieven, de gevolgen van het analyseren van betekenis als een algoritme, en het toepassen van complexiteitanalyse van semantische vraagstukken. We hopen dat dit onderzoek de vruchtbaarheid van een abstracte en complexiteitstheoretische benadering van de linguïstiek en cognitiewetenschap laat zien.

Abstract

In the dissertation we study the complexity of generalized quantifiers in natural language. Our perspective is interdisciplinary: we combine philosophical insights with theoretical computer science, experimental cognitive science and linguistic theories.

In Chapter 1 we argue for identifying a part of meaning, the so-called referential meaning (model-checking), with algorithms. Moreover, we discuss the influence of computational complexity theory on cognitive tasks. We give some arguments to treat as cognitively tractable only those problems which can be computed in polynomial time. Additionally, we suggest that plausible semantic theories of the everyday fragment of natural language can be formulated in the existential fragment of second-order logic.

In Chapter 2 we give an overview of the basic notions of generalized quantifier theory, computability theory, and descriptive complexity theory.

In Chapter 3 we prove that PTIME quantifiers are closed under iteration, cumulation and resumption. Next, we discuss the NP-completeness of branching quantifiers. Finally, we show that some Ramsey quantifiers define NP-complete classes of finite models while others stay in PTIME. We also give a sufficient condition for a Ramsey quantifier to be computable in polynomial time. We end this chapter with a question about the complexity dichotomy between Ramsey quantifiers.

In Chapter 4 we investigate the computational complexity of polyadic lifts expressing various readings of reciprocal sentences with quantified antecedents. We show a dichotomy between these readings: the strong reciprocal reading can create NP-complete constructions, while the weak and the intermediate reciprocal readings do not. Additionally, we argue that this difference should be acknowledged in the Strong Meaning Hypothesis.

In Chapter 5 we study the definability and complexity of the type-shifting approach to collective quantification in natural language. We show that under reasonable complexity assumptions it is not general enough to cover the semantics of all collective quantifiers in natural language. The type-shifting approach cannot lead outside second-order logic and arguably some collective quantifiers are not expressible in second-order logic. As a result, we argue that algebraic (many-sorted) formalisms dealing with collectivity are more plausible than the type-shifting approach . Moreover, we suggest that some collective quantifiers might not be realized in everyday language due to their high computational complexity. Additionally, we introduce the so-called second-order generalized quantifiers to the study of collective semantics.

In Chapter 6 we study the statement known as Hintikka's thesis: that the semantics of sentences like "Most boys and most girls hate each other" is not expressible by linear formulae and one needs to use branching quantification. We discuss possible readings of such sentences and come to the conclusion that they are expressible by linear formulae, as opposed to what Hintikka states. Next, we propose empirical evidence confirming our theoretical predictions that these sentences are sometimes interpreted by people as having the conjunctional reading.

In Chapter 7 we discuss a computational semantics for monadic quantifiers in natural language. We recall that it can be expressed in terms of finite-state and push-down automata. Then we present and criticize the neurological research building on this model. The discussion leads to a new experimental set-up which provides empirical evidence confirming the complexity predictions of the computational model. We show that the differences in reaction time needed for comprehension of sentences with monadic quantifiers are consistent with the complexity differences predicted by the model.

In Chapter 8 we discuss some general open questions and possible directions for future research, e.g., using different measures of complexity, involving game-theory and so on.

In general, our research explores, from different perspectives, the advantages of identifying meaning with algorithms and applying computational complexity analysis to semantic issues. It shows the fruitfulness of such an abstract computational approach for linguistics and cognitive science.