Linguist and Philos (2010) 33:215–250
DOI 10.1007/s10988-010-9076-z

RESEARCH ARTICLE

Computational complexity of polyadic lifts of generalized quantifiers in natural language

Jakub Szymanik

Published online: 13 November 2010

Abstract We study the computational complexity of polyadic quantifiers in natural language. This type of quantification is widely used in formal semantics to model the meaning of multi-quantifier sentences. First, we show that the standard constructions that turn simple determiners into complex quantifiers, namely Boolean operations, iteration, cumulation, and resumption, are tractable. Then, we provide an insight into branching operation yielding intractable natural language multi-quantifier expressions. Next, we focus on a linguistic case study. We use computational complexity results to investigate semantic distinctions between quantified reciprocal sentences. We show a computational dichotomy between different readings of reciprocity. Finally, we go more into philosophical speculation on meaning, ambiguity and computational complexity. In particular, we investigate a possibility of revising the Strong Meaning Hypothesis with complexity aspects to better account for meaning shifts in the domain of multi-quantifier sentences. The paper not only contributes to the field of formal semantics but also illustrates how the tools of computational complexity theory might be successfully used in linguistics and philosophy with an eye towards cognitive science.

Keywords Generalized quantifier theory · Computational complexity · Polyadic quantification · Multi-quantifier sentences · Strong Meaning Hypothesis

1 Introduction

Quantifier expressions occur whenever we describe the world, and communicate about it. Generalized quantifier theory is therefore one of the basic tools of linguistics

J. Szymanik (✉)
Department of Philosophy, Stockholm University, Stockholm, Sweden
e-mail: jakub.szymanik@gmail.com

today, studying the possible meanings and the inferential power of quantifier expressions by logical means. The classical version of the theory was developed in the 1980s, at the interface of linguistics, mathematics and philosophy. Until now, advances in "classical" generalized quantifier theory has focused mainly on definability questions and their applications to linguistics (see e.g. Keenan and Westerståhl 1997). This misses out on computation, the third pillar of language use and logical activity which can help to bridge logical perspective on linguistics with cognitive science results.

Moreover, even though there have been interesting papers devoted to polyadic quantifiers (see e.g. van Benthem 1989; Keenan 1992, 1996) most research in generalized quantifier theory has been directed towards monadic quantification. In the theory monadic quantifiers are simply the relations between subsets of the universe. They express meanings of simple determiners, like "some". The recent book by Peters and Westerståhl (2006) bears witness to this tendency, devoting more than 90 of its volume to the discussion of monadic quantifiers. It is then clear that definability and complexity of monadic quantifiers has been extensively studied. For example, it is known that on finite models monadic quantifiers definable in first-order logic are recognizable by acyclic finite-automata (van Benthem 1986); first-order logic enriched by all quantifiers of the form "divisible by n" corresponds to the class of regular languages (Mostowski 1998); and that proportional quantifiers, like "most", can be recognized by push-down automata (van Benthem 1986). Those results suggest that the verification of monadic quantifiers in natural language should be relatively easy. Indeed, a corpus of empirical studies shows that the cognitive task of recognizing the logical-value of sentences with monadic quantifiers is simple. In fact, recently the automata-theoretic model for processing monadic quantifiers has been confronted with human comprehension and the results show that the model captures many important cognitive aspects of the problem. For example, on the basis of differences between minimal automata recognizing various quantifiers one can predict the reaction time needed by subjects to solve a task (see e.g. Szymanik and Zajenkowski 2010a) as well as working memory activation both on the behavioral (see Szymanik and Zajenkowski 2010b) and neurological level (see e.g. McMillan et al. 2005) during the processing of a quantifier sentence. Those studies—linking computational complexity with cognitive processing—lead to a natural interest in computational properties of polyadic quantification with an eye towards cognitive applications.

In this paper we study the semantic constructions that turn simple monadic quantifiers into more complex ones: Boolean operations, iteration, cumulation, resumption, branching, and Ramseyfication. Those complex quantifier structures are used extensively in linguistics for giving the semantics of multi-quantifier sentences. The task is non-trivial as sentences with many quantifiers are semantically ambiguous. They can have many possible readings given by various interpretations of scope dependencies. Those interpretations may depend not just on the structure but also on the context. Various accounts of quantifier scope have been developed and the theoretical models predict different patterns of scope preferences (see e.g. Bach 1982; Bott and Radó 2009; Gierasimczuk and Szymanik 2009; Jaszczolt 2002; Kempson and Cormack 1981a, b, 1982; May 1985; Robaldo 2009; Tennant 1981).

Our approach rooted in computational complexity theory and cognitive science provides new complexity arguments enriching that classical discussion.

Moreover, polyadic constructions have been extensively discussed in logic mostly with respect to their definability properties (see e.g. Hella et al. 1997). As we will see those definability results very often imply complexity bounds. However, as we are driven by a psychological question about the strategies people may use to comprehend quantifiers we usually show a direct construction of the relevant procedures leading to complexity results. For example, we prove that Boolean operations, iteration, cumulation, and resumption do not increase computational complexity when applied to determiners by constructing a polynomial model-checkers for polyadic quantifiers from polynomial-time bounded Turing machines computing corresponding monadic determiners. As most of the natural language determiners may be modeled by monadic quantifiers computable in polynomial time this observation suggests that typically polyadic quantifiers in natural language are tractable. However, the next two polyadic lifts we are studying, branching and Ramseyfication, can lead to intractable constructions. Those complex quantifiers find their use, for instance, in capturing the semantics of reciprocal sentences with quantified antecedents.

The paper is organized as follows. In Sect. 2 we present preliminaries, including a quick tour into computational complexity theory. Next in Sect. 3 we discuss computational complexity of standard lifts turning tractable monadic determiners into more complex quantifier constructions being still easy to compute: Boolean combinations, iteration, cumulation, and resumption. Then, in Sect. 3.6 we briefly discuss branching as an intractable polyadic lift. Section 3.6 is addressed to more logically inclined readers. Next, we provide a linguistic case study in Sect. 4, where we apply Ramsey quantifiers to study complexity and meaning shifts in the domain of quantified reciprocal sentences. This section, though strongly grounded in the first part of the paper, can be read independently from the previous sections. We end with the more philosophical Sect. 5, where we consider the potential influence of computational complexity on human interpretation of multi-quantifier sentences.

We hope that the paper will help lay a groundwork for extending natural language semantics with computational and cognitive aspects by underlining issues of computational complexity and their possible interplay with "difficulty" as experienced by subjects interpreting sentences. We also believe that our considerations will directly contribute to the semantic problem of scopal-ambiguity and multiple quantification.

2 Preliminaries

2.1 Generalized quantifiers

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) An even number of the students saw a ghost.

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

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 (2). 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 (3). Let us assume that the quantifier "most" simply means "more than half". Hence, sentence (3) is true if and only if the 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.

Therefore, one can propose the following formalization. A quantifier Q, like "every", "an even number of", "most", is a way of associating with each set M a function from pairs of subsets of M into $\{0, 1\}$ (False, True). Specifically,

$$\mathsf{Every}_M[A, B] = 1 \text{ iff } A \subseteq B$$

$$\mathsf{Even}_M[A, B] = 1 \text{ iff } card(A \cap B) \text{ is even}$$

$$\mathsf{Most}_M[A, B] = 1 \text{ iff } card(A \cap B) > card(A - B)$$

The above approach may be extended to cover not only quantifiers of type (1, 1) (binding two unary predicates) but all quantifiers of any type $t = (n_1, \ldots, n_k)$ (binding k predicates each of arity n_i for $1 \leq i \leq k$). Here simply, Q_M maps each k-tuple of relations where each R_i is a subset of M^{n_i}. However, following the historical line of development below we introduce different but equivalent definition of generalized quantifier. The new definition, identifying quantifiers with classes of models, comes handy when considering computational properties.

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 present, particularly in linguistics, and we briefly introduced it above. However, historically speaking the notion of 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.

Definition 1 *Let* $t = (n_1, \ldots, n_k)$ be a k-tuple of positive integers. A generalized 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

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

Definition 2 If in the above definition for all $i : n_i = 1$, then we say that a quantifier is monadic, otherwise we call it polyadic.

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 *As are B* is true if and only if $card(A \cap B) > card(A - B)$ and therefore the quantifier corresponds to the class of models where this inequality holds.

Therefore, formally speaking:

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

It was realized by Montague (1970) that the notion of 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.

Notice that 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.

Definition 3 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:

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

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

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

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

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

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

Peters and Westerståhl (2006, Chap. 4.4) discuss this operator in great depth—among others—they show that it preservers isomorphism invariance (ISOM) and that all ISOM natural language quantifiers that satisfy extensions (EXT) and conservativity (CONS) are relativizations of type (1) quantifiers. CONS (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.

Definition 4 A quantifier of type $(n_1, \ldots, n_k)$ satisfies domain independence (EXT) iff the following holds:

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

The property which in a sense extends EXT is conservativity:

Definition 5 A type (1, 1) quantifier Q is conservative (CONS) iff for all M and all $A, B \subseteq M$:

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

Quantifiers closed under isomorphisms (ISOM) 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).

2.2 Computing quantifiers

For the purposes of presenting a computational complexity discussion on quantifiers we restrict ourselves to ISOM quantifiers over finite universes. It allows us to develop some suitable representation of quantifiers. Finite models can be encoded as finite strings over some vocabulary as follows. 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: $U = \{1, \ldots, n\}$. A natural way of encoding a model $\mathbb{M}$ (up to isomorphism) is by listing its universe, U, and storing the interpretation of the symbols in τ by writing down their truth-values on all tuples of objects from U.

Definition 6 Let $\tau = \{R_1, \ldots, R_k\}$ be a relational vocabulary and $\mathbb{M}$ a τ-model of the following form: $\mathbb{M} = (U, R_1^M, \ldots, R_k^M)$, where $U = \{1, \ldots, n\}$ is the universe of model $\mathbb{M}$ and $R_i^M \subseteq U^{n_i}$ is an n_i-ary relation over U, 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 $O((card(U))^c)$, where c is the maximal arity of the predicates in τ (or $c = 1$ if there are no predicates).

The code has the following form:

$$\tilde{n}\#\tilde{R_1^M}\#\cdots\#\tilde{R_n^M}, \text{ where:}$$

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

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 $\mathbb{M} = \{1, 2, 3\}$, 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 step by step construct the code of the model:

- $\tilde{n}$ 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.

Therefore, according to Definition 1, generalized quantifiers can be treated as classes of finite strings, i.e., as languages. Now we can easily fit the notions into the descriptive complexity paradigm (see e.g. Immerman 1998) and the next section of the paper introduces basic concepts of computational complexity theory.

Definition 7 By the complexity of a quantifier Q we mean the computational complexity of the corresponding class of finite models.

Consider a quantifier of type (1, 2). 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 first considered from the computational complexity point of view by Blass and Gurevich (1986). They investigated

various forms of branching (Henkin) quantifiers (see Sect. 3.6) defining NP or NLOGSPACE complete graph problems.

Definition 8 We say that a quantifier Q is NP-hard if the corresponding class of finite models is NP-hard. Q is NP-complete if the corresponding class belongs to NP and is NP-hard.

In the rest of the paper we will investigate which natural language polyadic quantifiers are NP-hard (intractable). But before we turn to it we need to recall some basic concepts of the computational complexity theory (Papadimitriou 1993, for an extensive treatment see).

2.3 Complexity classes

The basic device of computation in the computational complexity theory is a multi-tape Turing (1936) machine. Most of the particulars of Turing machines are not of direct interest to us. Nevertheless, we review 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.

Definition 9 Let Γ be some finite alphabet and $L \subseteq \Gamma^*$ a language (a subset of the set of all finite strings over alphabet Γ). 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 note 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.

Theorem 1 *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.[1] The question whether this simulation can be done without exponential growth in computation time leads us to computational complexity theory.

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

Definition 10 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, $L \in$ 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 some 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.

Definition 11

- LOGSPACE $= \bigcup_{k\in\omega}$ SPACE$(k \log n)$
- PTIME $= \bigcup_{k\in\omega}$ TIME(n^k)
- NPTIME $= \bigcup_{k\in\omega}$ NTIME(n^k)

[1] 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, p. 49 for details).

If $L \in$ NPTIME, then we say that L is decidable (computable, solvable) in non-deterministic polynomial time and likewise for other complexity classes.

The question whether PTIME (P for short) is strictly contained in NPTIME (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 PTIME which are conceived of as *efficiently solvable*. In the paper we take this distinction for granted and investigate semantic constructions in natural language from that perspective.

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.

Definition 12 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.

Definition 13 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*

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

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

Definition 14 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 is complete for NP. Many other now famous problems were then proven NP-complete by Karp (1972)—including some versions of satisfiability as well as some graph problems, e.g. CLIQUE, which we will use further in the text. Garey and Johnson (1979) contains a list of NP-complete languages.

Moreover, we will need a concept of relativization defined via 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.

Definition 15 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 $\mathcal{C}$.

3 Standard polyadic lifts

As we pointed out in the introduction most research in generalized quantifier theory has been directed towards monadic quantification in natural language. 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. One way to deal with polyadic quantification in natural language is to define it in terms of monadic quantifiers using Boolean combinations and so-called polyadic lifts: iteration, cumulation, and resumption (see e.g. van Benthem 1989). We observe that these operations do not increase the computational complexity of quantifiers.

3.1 Boolean combinations

To account for complex noun phrases, like those occurring in sentences (4)–(7), we define disjunction, conjunction, outer negation (complement) and inner negation (post-complement) of generalized quantifiers.

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

(5) At least 100 and not more than 200 students started in the marathon. (conjunction)

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

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

Definition 16 Let Q, Q' be generalized quantifiers, both of type $(n_1, \ldots, n_k)$. We define:

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

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

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

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

3.2 Iteration

The Fregean nesting of first-order quantifiers, e.g., $\forall\exists$, can be applied to any generalized quantifier by means of iteration. For example, iteration may be used to express the meaning of the following sentence in terms of its constituents.

(8) Most logicians criticized some papers.

The above 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:

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

Of course the sentence can have a different reading corresponding to other lifts than iteration. We will introduce another possibility in Sect. 3.3. But first let us define iteration precisely.

Definition 17 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:

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

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

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, p. 347).

3.3 Cumulation

Consider the following sentence:

(9) 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'08. 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 80 professors taught at least one from 60 courses and each of the courses was taught by at least one professor. The reading might be linguistically forced as follows:

(10) Eighty professors taught *a total of* sixty courses at ESSLLI'08

Cumulation is easily definable in terms of iteration and the existential quantifier as follows.

Definition 18 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:

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

3.4 Resumption

The next lift we are about to introduce—resumption (vectorization)—has found many applications in theoretical computer science (see e.g. Immerman 1998). 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 instance consider:

(11) Men seldom make passes at girls who wear glasses

(12) People usually are grateful to firemen who rescue them.

In (11), for example, the first argument is the product of the set of men and the set of girls wearing glasses and (12) expresses a relation between the set of pairs: (a person, a fireman) such that the latter rescues the former. Additionally, resumption is sometimes used for interpretation of certain cases of adverbial quantification and "donkey" anaphora (see e.g. Peters and Westerståhl 2006, Chap. 10.2 for a criticism).

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

Definition 19 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:

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

That is, $\mathsf{Res}^k(\mathsf{Q})$ is just Q applied to a universe, U^k, containing k-tuples. In particular, $\mathsf{Res}^1(\mathsf{Q}) = \mathsf{Q}$.

3.5 PTIME GQs are closed under Bool, It, Cum, and Res

When studying the computational complexity of quantifiers a natural question arises in the context of polyadic lifts. Do they increase complexity? In particular, is it possible that two tractable determiners can be turned into an intractable quantifier?

We show that PTIME computable quantifiers are closed under Boolean combinations and the three lifts defined above. As we are interested in the strategies people may use to comprehend quantifiers we show a direct construction of the relevant procedures.

Proposition 1 *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 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: $(\mathsf{Q}\neg)_U[R_1, \ldots, R_k] \Longleftrightarrow \mathsf{Q}_U[R_1, \ldots, R_{k-1}, U - R_k]$. 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 $\mathsf{Q}_U[R_1, \ldots, R_{k-1}, U - R_k]$.
(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, 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 $\mathsf{It}(\mathsf{Q}, \mathsf{Q}')[A, B, R] \Longleftrightarrow \mathsf{Q}[A, A']$, 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. Now, to construct A' in polynomial time 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 18). 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 6.

Additionally, let us give an argument that the above proposition holds for all generalized quantifiers, and not only for the 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 e.g. Immerman 1998). 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 $\mathrm{FO}(A)$ is in $\mathrm{LOGSPACE}^{\mathcal{C}}$ (deterministic logarithmic space with an oracle from $\mathcal{C}$). 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 $\mathrm{LOGSPACE}^{\mathrm{PTIME}} = \mathrm{PTIME}$.

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). However, notice that our argument given in point (6) of the proof does 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.

Theorem 2 *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 shown 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 quantifiers. As we can safely assume that most of the simple determiners in natural language are PTIME computable the semantics of the polyadic quantifiers studied above is tractable. This seems to be good news for the computational theory of natural language processing for multi-quantifier sentences. However, not all polyadic quantifiers in natural language are polynomial-time computable. In the next section we study two notorious examples of complex quantification.

3.6 An intractable polyadic lift: branching

3.6.1 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:

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

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:

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

Via simple transformations this sentence is equivalent to the following:

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

and therefore, it expresses infinity: there exists an injective function from the universe to the universe which is not surjective. It is true in a model $\mathbb{M}$ if and only if the domain of $\mathbb{M}$ is infinite (Dedekind).

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).

Henkin's quantifiers are intractable. The famous linguistic application of branching quantifiers is for the study of scope dependencies in multi-quantifier sentences. Consider the following statements:

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

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

(15) 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 (13) is expressed as follows:

$$(16) \quad \begin{pmatrix} \forall x \exists y \\ \forall z \exists w \end{pmatrix} [(V(x) \wedge T(z)) \Rightarrow (R(x,y) \wedge R(z,w) \wedge H(y,w))],$$

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 $\mathrm{Z}xy\ [\varphi(x,y), \psi(x,y)]$ can be interpreted in a second-order language as:

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

We state that the problem of recognizing the truth-value of formula (16) in a finite model is NP-complete (Mostowski and Wojtyniak 2004). In other words:

Theorem 3 *The quantifier* Z *is intractable.*

Therefore, branching—as opposed to iteration, cumulation, and resumption—substantially effects computational complexity. As a result sentences (13)–(15) under interpretation defended by Hintikka are intractable but see remarks in the concluding Sect. 3.7.

Basic proportional branching quantifiers are intractable. 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. Recall that a quantifier Q of type (1, 1) is monotone increasing in its right argument (MON↑) whenever: if $Q_M[A,B]$ and $B \subseteq B' \subseteq M$, then $Q_M[A,B']$.

Definition 20 *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 the pairs of non-monotone quantifiers (see e.g. Robaldo 2009; Sher 1990).

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

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

(18) At least one third of all villagers and half of all townsmen hate each other.

For instance, an intended branching reading of (17) is as follows:

(#) There is a set V containing a majority of the villagers and a set T containing a majority of the townsmen and $(V \times T) \cup (T \times V) \subseteq HATE$.

Let us consider computational complexity of interpretations like this.

It has been shown by Merlijn Sevenster (2006) that the problem of recognizing the truth-value of sentence (17) under interpretation (#) in finite models is NP-complete. Actually, it can also be proven that all basic proportional branching sentences, like (18), define an NP-complete classes of finite models. In other words the following holds.

Theorem 4 *Let* Q *and* Q' *be basic proportional quantifiers, then the quantifier* $\mathsf{Br}(\mathsf{Q},\mathsf{Q}')$ *is intractable.*

By basic proportional branching sentences (e.g. (18)), we mean the branching interpretations of sentences containing basic proportional quantifiers, i.e., quantifiers saying that some fraction of a universe has a given property (see e.g. Keenan and Westerståhl 1997) according to the following definition:

Definition 21

$$\mathbb{M} \models \mathsf{Q}_{\mathsf{q}}[A,B] \text{ iff } \frac{card\ (A \cap B)}{card\ (A)} \geq q, \text{ where } 0<\mathrm{q}<1 \text{ is a rational number.}$$

Let us give two examples of basic proportional quantifiers.

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

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

Therefore, the above result gives another example of a polyadic quantifier construction in natural language which has an intractable reading.

Branching counting quantifiers are intractable. Below we will briefly discuss the complexity of branching readings of sentences, which, among others, play an important role in the empirical study of Gierasimczuk and Szymanik (2009) devoted to finding out the proper reading for sentences being discussed here. We show, under the cognitively plausible assumption that people use uniform strategies for processing all quantifiers of the form "more than k", the branching reading of sentences used in the experiment is intractable. This observation partially explains why subjects tended to avoid branching reading in favor of linear, tractable interpretation in the study by Gierasimczuk and Szymanik (2009) (see Sect. 5 for further discussion).

Consider the schema of a sentence:

(19) More than k villagers and more than n townsmen hate each other.

Its branching reading has the following form:

$$(20) \quad \begin{pmatrix} \text{More than k } x : V(x) \\ \text{More than m } y : T(y) \end{pmatrix} H(x,y),$$

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.

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

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 $\mathsf{C}^{\geq A}$ 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 $A \subseteq M$.[2]

[2] 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 numerical constants in a quantifier refer to the number domain (see e.g. Grädel and Gurevich 1998; Otto 1997).

Definition 22 Let $\mathbb{M} = (M, A, \ldots)$. We define the counting quantifier of type (1) as follows:

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

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 $card(V') \geq card(A)$, $card(T') \geq card(B)$ 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.

Definition 23 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 $E \subseteq V_1 \times V_2$.

Definition 24 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.

Proposition 2 *The quantifier* $\mathsf{Br}(\mathsf{C}^{\geq \mathsf{A}}, \mathsf{C}^{\geq \mathsf{B}})$ *is intractable.*

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

This constitutes another class of branching intractable quantifiers.

3.7 Conclusions so far

We have investigated the computational complexity of polyadic quantifiers, preparing the ground for a linguistic case study in the following section. We have shown that some polyadic constructions do not increase computational complexity, while others—such as branching 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 determiners we may arrive at NP-complete polyadic quantifiers, e.g. branching basic proportional quantifiers are intractable.

There is one linguistic proviso concerning all multi-quantifier sentences presented in that section. Namely, they are ambiguous. Moreover, such sentences can hardly be found in a linguistic corpus (see Sevenster 2006, footnote 8, p. 140). In the

paper (Gierasimczuk and Szymanik 2009) it has been shown that their readings vary between easy (PTIME) and difficult (branching) interpretations. More precisely, Gierasimczuk and Szymanik (2009) propose a novel alternative two-way reading expressible by linear formulae. This interpretation is based on linguistic and logical observations. They report on experiments showing that people tend to interpret sentences similar to Hintikka sentence in a way consistent with linear, non-branching, interpretation. Additionally, it is argued that the non-branching reading is psychologically the dominant one. This proviso motivates us to look for intractable natural language quantifiers which not only occur frequently in everyday English but are also one of the sources of its complexity. Section 4 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.

Last, but not least, there is the question of possible applications of a kind of analysis we gave in that part of the paper. Do differences in computational complexity of polyadic quantifiers play any role in natural language interpretation? In the next section we will argue that they do.

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 section 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 polyadic lifts of simple generalized quantifiers.

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, Chap. 7). In other words, we will consider quantified reciprocal sentences, like "Five professors discuss with each other", where reciprocal phrase "each other" refers to a quantified noun phrase, in this case "five professors".

Next we study the computational complexity of reciprocal lifts with respect to the quantifiers in the antecedents. 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 between 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 basic 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 Sect. 3.6.1).

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 (21)–(23).

(21) At least four members of parliament refer to each other indirectly.

(22) Most Boston pitchers sat alongside each other.

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

The possible interpretations of reciprocity exhibit a wide range of variation. For example, sentence (21) 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 (22) and (23) have different meanings. Sentence (22) 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 (23) 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 (21)–(23) are illustrated in Fig. 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, R, should cover the domain, A, (in our case restricted by a quantifier in the antecedent). We have three possibilities:

FUL (strong) Each pair of elements from A participates in R directly.
LIN (intermediate) Each pair of elements from A participates in R directly or indirectly.
TOT (weak) 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,

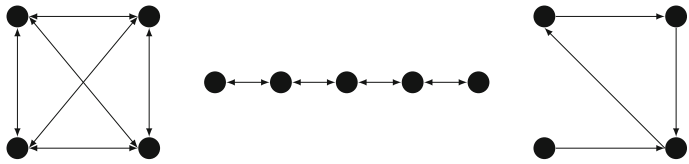

Fig. 1 On the left is a typical model satisfying sentence (21) under the so-called *strong reciprocal* interpretation. Each element is related to each of the other elements. In the middle is an example of a model satisfying sentence (22) in a context with at most nine pitchers. This is the *intermediate 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 (23), assuming the so-called *weak reciprocal* interpretation. For each element there exists a different related element

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 (24) and weak alternative reciprocity occurring in sentence (25) (see Fig. 2 for typical models).

(24) Most of the stones are arranged on top of each other.
(25) All the 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 Fig. 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 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 Fig. 3 on the right.

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).

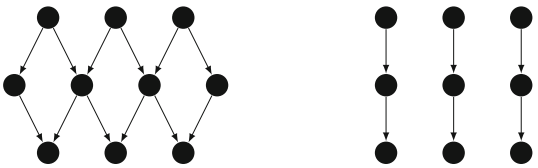

Fig. 2 On the left is a typical model satisfying sentence (24) under the so-called *intermediate alternative reciprocal* interpretation. Ignoring the direction of arrows, every element in the witness set of the quantifier Most is connected directly or indirectly. On the right is an example of a model satisfying sentence (25) 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

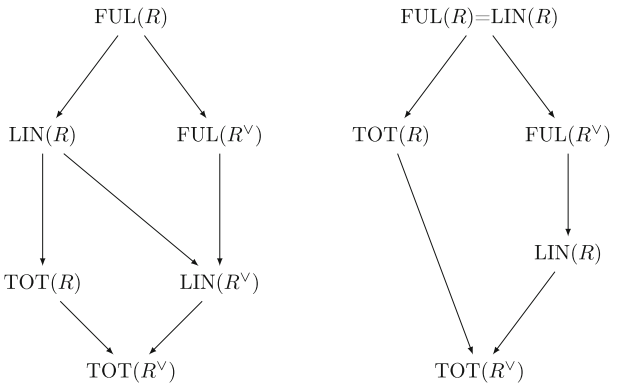

Fig. 3 On the left, inferential dependencies between the six interpretations of reciprocity. On the right, the situation when the reciprocal relation is transitive. Implications are represented by arrows

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.

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:

(26) The children followed each other.

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

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

The relation "follow 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 two 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:

(28) 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 the strongest possibility and according to SMH it properly describes the meaning of that sentence.

Our results show that 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 the paper, where we also treat SMH in a broad way covering not only reciprocity phenomena but also other multi-quantifier sentences. We investigate the cognitive status of so understood SMH and argue that if one assumes some kind of algorithmic theory of meaning, then the shifts between different interpretations of sentences, predicted by SMH, have to be extended by accommodating the possible influence of differences in computational complexity between various readings of sentences.

4.2 Reciprocals as polyadic quantifiers

Sentences with reciprocal expressions transform monadic quantifiers into polyadic ones. We will analyze reciprocal expressions in that spirit by defining appropriate lifts on monadic quantifiers. For the sake of simplicity we will restrict ourselves to reciprocal sentences with right monotone increasing quantifiers in their antecedents. 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).

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).

Definition 25 Let Q be a right monotone increasing quantifier of type (1, 1). We define:

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

We will call the result of such lifting a *Ramsey quantifier*. It says that there exists a subset X of the domain A, restricted by 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:

Definition 26

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

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}.

4.2.2 Intermediate reciprocal lift

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

Definition 27

$$\mathsf{Ram}_{\mathsf{I}}(\mathsf{Q})[A,R] \Longleftrightarrow \exists X \subseteq A[\mathsf{Q}(A,X) \wedge \forall x,y \in X(x \neq y \Rightarrow \exists \text{ sequence } z_1,\ldots,z_\ell \in X \text{ such that } (z_1 = x \wedge R(z_1,z_2) \wedge \cdots \wedge R(z_{\ell-1},z_\ell) \wedge z_\ell = y)].$$

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.

Definition 28

$$\mathsf{Ram}_{\mathsf{I}}^{\vee}(\mathsf{Q})[A,R] \Longleftrightarrow \exists X \subseteq A[\mathsf{Q}(A,X) \wedge \forall x,y \in X(x \neq y \Rightarrow \exists \text{ sequence } z_1,\ldots,z_\ell \in X \text{ such that}(z_1 = x \wedge (R(z_1,z_2) \vee R(z_2,z_1)) \wedge \cdots \wedge (R(z_{\ell-1},z_\ell) \vee R(z_\ell,z_{\ell-1})) \wedge z_\ell = y)].$$

In other words, $\mathsf{Ram}_{\mathsf{I}}^{\vee}$ says that any two elements from X are in the relation $R^{\vee}$ directly or indirectly. The property of graph connectedness is not first-order expressible; we need a universal monadic second-order formula. Hence from the definability point of view $\mathsf{Ram}_{\mathsf{I}}(\mathsf{Ram}_{\mathsf{I}}^{\vee})$ seems more complicated than $\mathsf{Ram}_{\mathsf{S}}$ $(\mathsf{Ram}_{\mathsf{S}}^{\vee})$. 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.

Definition 29

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

Definition 30

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

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.

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.

The linguistic application of reciprocal lifts is straightforward. For example, using them we can account for the meanings of the reciprocal sentences (21)–(25) discussed in Sect. 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.

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

(29) $\mathsf{Ram_S}(\mathsf{At\ least4})[\mathsf{MP}, \mathsf{Refer\text{-}indirectly}]$.

(22) Most Boston pitchers sat alongside each other.

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

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

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

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

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

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

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

It is easy to see that our formulas 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 Figs. 1 and 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 some facts proven by Szymanik (2009) without giving all details.

Recall that we can 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

$\mathsf{Ram}_{\mathsf{S}}^{\vee}$ 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 $\mathsf{Ram}_{\mathsf{S}}(\mathsf{At\ least\ k})[A,R]$ is true we have to solve the CLIQUE problem for M and k. In other words, we ask whether there exists $A \subseteq M$ of cardinality greater than k such that each pair of elements from A is connected by R. 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 $\mathsf{Ram}_{\mathsf{S}}(\mathsf{At\ least\ k})$ is in PTIME. For instance, $\mathsf{Ram}_{\mathsf{S}}(\mathsf{At\ least\ 5})$ is computable in polynomial time. In general, notice that the strong reciprocal sentence $\mathsf{Ram}_{\mathsf{S}}(\exists^{\geq k})[A,R]$ is equivalent to the following first-order formula:

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

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 22 that the counting quantifier $\mathsf{C}^{\geq A}$ 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 $\mathsf{C}^{\geq A}$ such that $card(A) = k$. Therefore, the determiners $\mathsf{At\ least}\ 1, \mathsf{At\ least}\ 2, \mathsf{At\ least}\ 3, \ldots$ are interpreted by one counting quantifier $\mathsf{C}^{\geq A}$—the set A just has to be chosen differently in every case.

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

Proposition 3 *The quantifier* $\mathsf{Ram}_{\mathsf{S}}(\mathsf{C}^{\geq A})$ *is intractable.*

Proof Let as take any model $\mathbb{M} = (M, A, \ldots)$. We have to decide whether $\mathbb{M} \models \mathsf{Ram}_{\mathsf{S}}(\mathsf{C}^{\geq A})$. This is equivalent to the CLIQUE problem for $\mathbb{M}$ and $card(A)$.

Therefore, the Ramsey quantifier $\mathsf{Ram_S}(\mathsf{C}^{\geq A})$ defines an NP-complete class of finite models. □

Corollary 1 *The quantifier* $\mathsf{Ram}^{\vee}_{\mathsf{S}}(\mathsf{C}^{\geq A})$ *is intractable.*

These results indicate that even though in a given situation checking the truth-value of a sentence with a fixed number, such as (21), is tractable, the general schema characterizing strong reciprocal sentences with counting quantifiers is NP-complete.

4.3.2 Basic 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:

(34) Most members of parliament refer to each other indirectly.

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

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

We will call these sentences *strong reciprocal sentences with basic proportional quantifiers*. Their general form is given by the sentence schema (36), 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 $card(Cl) \geq q \times card(A)$.

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 $\frac{card(A)}{card(U)} \geq q$. In this sense q determines a basic proportional quantifier $\mathsf{Q_q}$ of type (1, 1) as stated in Definition 21.

The strong reciprocal lift of a basic proportional quantifier, $\mathsf{Ram_S}(\mathsf{Q_q})$, is of type (1, 2) and obviously might be used to express the meaning of sentences like (34)–(36). We will call quantifiers of the form $\mathsf{Ram_S}(\mathsf{Q_q})$ *basic proportional Ramsey quantifiers*. Mostowski and Szymanik (2007) have shown the following:

Proposition 4 *If q is a rational number and $0 < q < 1$, then the quantifier* $\mathsf{Ram_S}(\mathsf{Q_q})$ *is intractable.*

Corollary 2 *If q is a rational number and $0 < q < 1$, then the quantifier* $\mathsf{Ram}^{\vee}_{\mathsf{S}}(\mathsf{Q_q})$ *is intractable.*

Therefore, strong reciprocal sentences with basic proportional quantifiers in the antecedent, like (34) or (35), 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 (1997) we will identify monotone simple unary quantifiers with number-theoretic functions, $f : \omega \to \omega$, 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.

Definition 31 Given $f : \omega \to \omega$, we define:

$$(\mathsf{Q}_f)_M[A] \Longleftrightarrow card(A) \geq f(card(M)).$$

As an example consider the following:

- $\exists = (\mathsf{Q}_f)_M$, where $f(card(M)) \geq 1$.
- $\forall = (\mathsf{Q}_g)_M$, where $g(card(M)) = card(M)$.
- $\mathsf{At\ least\ half} = (\mathsf{Q}_h)_M$, where $h(card(M)) \geq \frac{card(M)}{2}$.

Given a monotone increasing quantifier we can easily find the function corresponding to it.

Definition 32 Let Q be a monotone increasing quantifier of type (1). Define:

$$f(n) = \begin{cases} \textit{least } k \textit{ such that}: \\ \exists U \exists A \subseteq U[card(U) = n \wedge card(A) = k \wedge \mathsf{Q}_U(A)] & \textit{if such a } k \textit{ exists} \\ n + 1 & \textit{otherwise}. \end{cases}$$

Proposition 5 *If* Q *is a monotone increasing quantifier of type (1) and* f *is defined as in Definition 32 then*

$$\mathsf{Q} = \mathsf{Q}_f.$$

Proof The equality follows directly from the definitions. □

Väänänen (1997) considers so called bounded functions:

Definition 33 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: $\lceil \frac{n}{2} \rceil$, $\lceil \sqrt{n} \rceil$, $\lceil \log n \rceil$. Szymanik (2009) proves that polynomial computable bounded quantifiers are closed under the strong reciprocal lift. Therefore, we have the following:

Proposition 6 *If a monotone increasing quantifier* Q_f *is PTIME computable and bounded, then the reciprocal quantifier* $\mathsf{Ram_S}(\mathsf{Q}_f)$ *is PTIME computable.*

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.

Corollary 3 *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.*

Moreover, notice, that the relativization, Q_f^{rel}, of Q_f is the right monotone type (1, 1) quantifier:

$$(Q_f^{rel})_M\ [A,B] \Longleftrightarrow card(A \cap B) \geq f(card(A)).$$

Thus, the restriction to unary quantifiers is not essential and the result carries over to type (1, 1) determiners.

What are the possible conclusions from Proposition 6? 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 computable in practice. 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:

(37) Some members of parliament refer to each other indirectly.

(38) 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 tractable.

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.

Proposition 7 *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram}_{\mathsf{I}}(\mathsf{Q})$ *is PTIME computable.*

Proof Let $\mathcal{G} = (V, A, E)$ be a directed colored graph-model, where V is the set of verticies, E the set of edges, and $A \subseteq V$. To check whether $\mathcal{G} \in \mathsf{Ram}_{\mathsf{I}}(\mathsf{Q})$, one has to compute all connected components of the subgraph determined by A. For example, one 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, the algorithm 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 $\mathcal{G}$ is $O(card(V) + card(E))$. Moreover, the number of the components in A is bounded by $card(A)$. 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, $\mathsf{Ram}_{\mathsf{I}}(\mathsf{Q})$ is in PTIME. □

Corollary 4 *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram}_{\mathsf{I}}^{\vee}(\mathsf{Q})$ *is PTIME computable.*

The next proposition follows immediately.

Proposition 8 *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram}_{\mathsf{W}}(\mathsf{Q})$ *is PTIME computable.*

Proof To check whether a given graph-model $\mathcal{G} = (V, A, E)$ is in $\mathsf{Ram}_{\mathsf{W}}(\mathsf{Q})$, 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, $\mathsf{Ram}_{\mathsf{W}}(\mathsf{Q})$ is in PTIME. □

Corollary 5 *If a monotone increasing quantifier* Q *is PTIME computable, then the quantifier* $\mathsf{Ram}_{\mathsf{W}}^{\vee}(\mathsf{Q})$ *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 (22), (23), (24), and (25) we discussed in the introduction are tractable. Hence from a computational complexity perspective the intermediate and weak reciprocal lifts behave similar to iteration, cumulation and resumption (discussed in Sect. 3).

4.5 Conclusions on reciprocals

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 with intractable semantics. For instance, strong reciprocal sentences with basic proportional quantifiers in the antecedent, e.g. sentence (34), are NP-complete.
- For PTIME computable quantifiers the intermediate and weak reciprocal interpretations (see e.g. sentences (22) and (23) 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 (37) and (38).

Therefore, we argue that:

- The semantic distinctions of Dalrymple et al. (1998) seem solid from a computational complexity perspective.

• Shifts in meaning may be triggered by the computational complexity of sentences (see the next section of the paper).

Many questions arise which are to be answered in a future work. Here we will mention only two of them.

There is a vast literature on the definability of polyadic lifts of generalized quantifiers (e.g. Väänänen 1997; 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.

We can also empirically compare the differences in shifts from the strong interpretation of reciprocal sentences with bounded and basic proportional quantifiers in antecedents. Our approach predicts that subjects will shift to easier interpretations more frequently in the case of sentences with basic proportional quantifiers.

In the next section we discuss the potential influence of computational complexity on the shifts in meaning of multi-quantifier sentences as predicted by the Strong Meaning Hypothesis.

5 A complexity perspective on shifts in meaning

Recall that according to the SMH a reciprocal expression is interpreted as having the logically strongest truth conditions that are consistent with the given context. Otherwise, the interpretation will shift toward the logically weaker intermediate or weak readings, depending on context (see Sect. 4.1.1). Notice, that SMH can be extended to cover other multi-quantifier sentences which we considered in Sects. 3 and 3.6. In that case it claims, that a scope ambiguous sentence should take the logically strongest interpretation unless there are some other factors forcing shifts in meaning.

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.[3] Such algorithms can be described by investigating how language users evaluate the truth-value of sentences in various situations (see e.g. Hackl 2009; Gierasimczuk and Szymanik 2009; Pietroski et al. 2009; Szymanik and Zajenkowski 2010a). 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

[3] This approach goes back to Frege (1892) and exists in the linguistic literature at different levels of transparency (see e.g. Moschovakis 2006; Szymanik 2009).

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[4] too hard for comprehension, we can find their inferential relations with relatively easier sentences.

According to the SMH any reciprocal sentence should be interpreted as a strong reciprocal sentence, if this interpretation is only available. 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 verify. 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. Analogously, for multi-quantifier sentences the branching reading is usually intractable. 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. We only recall that computational restrictions for cognitive abilities are widely treated in the literature (see e.g. Cherniak 1981; Ristad 1993; Levesque 1988; van Rooij 2008; Szymanik 2009). 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 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. In fact experimental results on human interpretation of multi-quantifier sentences presented by Gierasimczuk and Szymanik (2009) are consistent with that claim. In the experiment subjects were confronted with a verification task for sentences containing two counting quantifiers (see Sect. 3.6.1). They were avoiding intractable branching reading in favor of a linear, polynomial-time computable interpretation. Another possibility is that the P-cognition thesis runs counter to the SMH: subjects do not formulate branching readings as they are too difficult. This stronger claim seems to be supported by the feeling that the supposed cases of branching readings in English are not very convincing.

Summing up, our computational complexity perspective on natural language semantics suggests that it might not always be possible to interpret a sentence in the strong way given an appropriate context, as SMH suggests. If the sentence in

[4] The fact that the general problem is hard does not show that all instances normally encountered are hard. It is a matter for empirical study to provide 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 Szymanik (2009) for a more substantial discussion.

question is intractable under the strong interpretation, then people will turn to tractable readings like the intermediate and weak reciprocal meaning instead of the strong interpretation, or iterative and cumulative reading but not branching interpretation. Our observations give a cognitively reasonable argument for some shifts to occur, even though they are not predicted by SMH. For example, SMH assumes that the following sentence should be interpreted as a strong reciprocal statement.

(39) 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.

The main question remains whether our complexity perspective on multi-quantifier sentences can be coupled with experimental results as it has had place in the case of computational semantics for monadic determiners. For example, we would like to know whether subjects entertain intractable readings and shift to tractable ones or intractable interpretations are not considered at all. We leave these problems for a future research.

Acknowledgments I would like to thank Johan van Benthem, Paul Dekker, Nina Gierasimczuk, Theo Janssen, Marcus Kracht, Marcin Mostowski, Rick Nouwen, Remko Scha, Jouko Väänänen, Dag Westerståhl, and Yoad Winter for many comments and inspiration. I am also grateful to the audience and anonymous referees of 7th International Symposium on Language, Logic and Computation, Tbilisi 2007, 16th Amsterdam Colloquium 2007, and Eleventh Meeting on Mathematics of Language, Bielefeld 2009, where parts of that work were presented. I would also like to acknowledge an anonymous reviewer of L&P for very insightful comments. The research was supported by Grant NN 101 410835 'Semantyka i pragmatyka zdań, filozofia języka i jego meta-filozofia' from Ministry of Science and Higher Education of the Republic of Poland and by a research grant funded by the Swedish Research Council.

References

Bach, K. (1982). Semantic nonspecificity and mixed quantifiers. *Linguistics and Philosophy, 4*(4), 593–605.

Barwise, J., & 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.

Ben-Avi, G., & Winter, Y. (2003). Monotonicity and collective quantification. *Journal of Logic, Language and Information, 12*(2), 127–151.

Blass, A., & Gurevich, Y. (1986). Henkin quantifiers and complete problems. *Annals of Pure and Applied Logic, 32*, 1–16.

Bott, O., & Radó, J. (2009). How to provide exactly one interpretation for every sentence, or what eye movements reveal about quantifier scope. In S. Winkler, & S. Featherson, (Eds.), *The fruits of empirical linguistics*, Vol. 1. Berlin: Walther de Gruyter.

Cherniak, C. (1981). Minimal rationality. *Mind, 90*(358), 161–183.

Cook, S.A. (1971). The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on theory of computing* (pp. 151–158). ACM Press: New York, NY.

Dalrymple, M., Kanazawa, M., Kim, Y., Mchombo, S., & Peters, S. (1998). Reciprocal expressions and the concept of reciprocity. *Linguistics and Philosophy, 21*, 159–210.

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.
Frixione, M. (2001). Tractable competence. *Minds and Machines, 11*(3), 379–397.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability*. San Francisco: W. H. Freeman and Co.
Gierasimczuk, N., & Szymanik, J. (2009). Branching quantification vs. two-way quantification. *The Journal of Semantics, 26*(4), 329–366.
Grädel, E., & Gurevich, Y. (1998). Metafinite model theory. *Information and Computation, 140*(1), 26–81.
Hackl, M. (2009). On the grammar and processing of proportional quantifiers: Most versus more than half. *Natural Language Semantics, 17*(1), 63–98.
Hella, L., Väänänen, J., & 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* (pp. 167–183). Oxford: Pergamon Press.
Hintikka, J. (1973). Quantifiers vs. quantification theory. *Dialectica*, 27, 329–358.
Hintikka, J. (1996). *Principles of mathematics revisited*. Cambridge: Cambridge University Press.
Hintikka, J., & Sandu, G. (1997). Game-theoretical semantics. In J. van Benthem & A. ter Meulen, (Eds.), *Handbook of logic and language* (pp. 361–410). Amsterdam: Elsevier.
Immerman, N. (1998). *Descriptive complexity*. Texts in Computer Science. Newyork: Springer.
Jaszczolt, K. (2002). *Semantics and pragmatics: Meaning in language and discourse*. London: Longman Linguistics Library, Longman.
Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller, & J. W. Thatcher, (Eds.), *Complexity of computer computations* (pp. 85–103). New York: Plenum Press.
Keenan, E. (1992). Beyond the Frege boundary. *Linguistics and Philosophy, 15*(2), 199–221.
Keenan, E. (1996). Further beyond the Frege boundary. In J. van der Does, & J. van Eijck, (Eds.), *Quantifiers, logic, and language*. CSLI Lecture Notes (pp. 179–201). California: Stanford University.
Keenan, E., & Westerståhl, D. (1997). Generalized quantifiers in linguistics and logic. In J. van Benthem, & A. ter Meulen, (Eds.), *Handbook of logic and language* (pp. 837–895). Elsevier: Amsterdam.
Kempson, R. M., & Cormack, A. (1981a). Ambiguity and quantification. *Linguistics and Philosophy, 4*(2), 259–309.
Kempson, R. M., & Cormack, A. (1981b). On 'formal games and forms for games'. *Linguistics and Philosophy, 4*(3), 431–435.
Kempson, R. M., & Cormack, A. (1982). Quantification and pragmatics. *Linguistics and Philosophy, 4*(4), 607–618.
Krynicki, M., & Mostowski, M. (1995). Henkin quantifiers. In M. Krynicki, M. Mostowski, & L. Szczerba, (Eds.), *Quantifiers: logics, models and computation* (pp. 193–263). Dordercht: Kluwer Academic Publishers.
Landman, F. (2000). Against binary quantifiers. In *Events and plurality*. Studies in Linguistic and Philosophy (pp. 310–349). Dordercht: Kluwer Academic Publisher.
Levesque, H. J. (1988). Logic and the complexity of reasoning. *Journal of Philosophical Logic, 17*(4), 355–389.
Lindström, P. (1966). First order predicate logic with generalized quantifiers. *Theoria, 32*, 186–195.
May, R. (1985). *Logical form: Its structure and derivation*. Linguistic Inquiry Monographs Cambridge, MA: The MIT Press.
McMillan, C. T., Clark, R., Moore, P., Devita, C., & Grossman, M. (2005). Neural basis for generalized quantifier comprehension. *Neuropsychologia, 43*, 1729–1737.
Montague, R. (1970). Pragmatics and intensional logic. *Dialectica, 24*(4), 277–302.
Moschovakis, Y. (2006). A logical calculus of meaning and synonymy. *Linguistics and Philosophy, 29*(1), 27–89.
Mostowski, A. (1957). On a generalization of quantifiers. *Fundamenta Mathematicae, 44*, 12–36.
Mostowski, M. (1998). Computational semantics for monadic quantifiers. *Journal of Applied Non-Classical Logics, 8*, 107–121.
Mostowski, M., & Szymanik, J. (2007). Computational complexity of some Ramsey quantifiers in finite models. *The Bulletin of Symbolic Logic, 13*, 281–282.
Mostowski, M., & Wojtyniak, D. (2004). Computational complexity of the semantics of some natural language constructions. *Annals of Pure and Applied Logic, 127*(1–3), 219–227.

Otto, M. (1997). *Bounded variable logics and counting. A study in finite models.* Volume 9 of Lecture Notes in Logic. Berlin: Springer-Verlag.
Papadimitriou, C. H. (1993). *Computational complexity*. Redwood City, CA: Addison Wesley.
Peters, S., & Westerståhl, D. (2006). *Quantifiers in language and logic*. Oxford: Clarendon Press.
Pietroski, P., Lidz, J., Hunter, T., & Halberda, J. (2009). The meaning of 'most': Semantics, numerosity, and psychology. *Mind and Language, 24*, 54–85.
Ristad, E. S. (1993). *The language complexity game*. Artificial Intelligence. Cambridge, MA: The MIT Press.
Robaldo, L. (2009). Independent set readings and generalized quantifiers. *Journal of Philosophical Logic, 39*(1), 23–58.
Sabato, S., & Winter, Y. (2005). From semantic restrictions to reciprocal meanings. In *Proceedings of FG-MOL 2005*. CSLI Publications.
Sevenster, M. (2006). *Branches of imperfect information: Logic, games, and computation*. PhD thesis, Universiteit van Amsterdam. http://www.illc.uva.nl/Publications/Dissertations/DS-2006-06.text.pdf.
Sher, G. (1990). Ways of branching quantifiers. *Linguistics and Philosophy, 13*, 393–442.
Szymanik, J. (2009). *Quantifiers in TIME and SPACE. Computational complexity of generalized quantifiers in natural language*. PhD thesis, Universiteit van Amsterdam. http://www.illc.uva.nl/Publications/ResearchReports/DS-2009-01.text.pdf.
Szymanik, J., & Zajenkowski, M. (2010a). Comprehension of simple quantifiers. Empirical evaluation of a computational model. *Cognitive Science: A Multidisciplinary Journal, 34*(3), 521–532.
Szymanik, J., & Zajenkowski, M. (2010b). Quantifiers and working memory. In M. Aloni, & K. Schulz, (Eds.), *Amsterdam Colloquium 2009*. Lecture Notes in Artificial Intelligence 6042 (pp. 456–464). Berlin: Springer.
Tennant, N. (1981). Formal games and forms for games. *Linguistics and Philosophy, 4*(2), 311–320.
Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*(2), 230–265.
Väänänen, J. (1997). 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: Cambridge University Press.
van Benthem, J. (1986). *Essays in logical semantics*. Dordercht: Reidel.
van Benthem, J. (1989). Polyadic quantifiers. *Linguistics and Philosophy, 12*(4), 437–464.
van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science: A Multidisciplinary Journal, 32*(6), 939–984.