Quantifiers in TIME and SPACE: computational complexity of generalized quantifiers in natural language

Szymanik, J.K.

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

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

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

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

Therefore, in the last section of this chapter — motivated by the search for non-controversial NP-complete semantic constructions in natural language — we investigate the so-called Ramsey quantifiers. We outline some links between them and branching quantifiers. Then we prove that some Ramsey quantifiers, e.g. the proportional, define NP-complete classes of finite models. Moreover, we observe that so-called bounded Ramsey quantifiers are PTIME computable.
Chapter 3. Complexity of Polyadic Quantifiers

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

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

3.1 Standard Polyadic Lifts

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

(1) Some logicians smoke.

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

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

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

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

### 3.1.1 Iteration

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

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

(2) Most logicians criticized some papers.

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

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

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

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

$\lt(Q, Q')[A, B, R] \iff Q[A, \{a \mid Q'(B, R(a))\}]$, where $R(a) = \{b \mid R(a, b)\}$.

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

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

The observation that quantifiers are order dependent will play a crucial role when we discuss possible readings of determiner combinations and scope dominance between them in Chapter 6.
Chapter 3. Complexity of Polyadic Quantifiers

3.1.2 Cumulation

Consider the following sentence:

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

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

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

$$\text{Cum}(Q, Q')[A, B, R] \iff \text{lt}(Q, \text{Some})[A, B, R] \land \text{lt}(Q', \text{Some})[B, A, R^{-1}].$$

3.1.3 Resumption

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

Below we give a formal definition of the resumption operator.

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

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

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

(4) Most twins never separate.
3.1.4 PTIME GQs are Closed under It, Cum, and Res

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

Sevenster (2006) claims that if quantifiers are definable in the Presburger arithmetic of addition, then the computational complexity of their Boolean combinations, iteration, cumulation, and resumption stay in LOGSPACE.\footnote{Moreover, he conjectures that the circuit complexity class \( ThC_0 \) (see e.g. Chapter 5.4 Immerman, 1998, for definition) is also closed under taking these operations on quantifiers.} We do not know whether all natural language determiners are definable in the arithmetic of addition. Therefore, we do not want to restrict ourselves to this class of quantifiers. Hence, we show that PTIME computable quantifiers are closed under Boolean combinations and the three lifts defined above. As in the dissertation we are interested in the strategies people may use to comprehend quantifiers we show a direct construction of the relevant procedures. In other words, we show how to construct a polynomial model-checker for our polyadic quantifiers from PTIME Turing machines computing monadic determiners.

**Proposition.** Let \( Q \) and \( Q' \) be monadic quantifiers computable in polynomial time with respect to the size of a universe. Then the quantifiers: (1) \( \neg Q \); (2) \( Q \neg \); (3) \( Q \land Q' \); (4) \( \text{It}(Q, Q') \); (5) \( \text{Cum}(Q, Q') \); (6) \( \text{Res}(Q) \) 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 \( \neg Q \) is like \( M \). The only difference is that we change accepting states into rejecting states and \textit{vice versa}. In other words, we accept \( \neg Q \) whenever \( M \) rejects \( Q \) and reject whenever \( M \) accepts. The working time of a so-defined new Turing machine is exactly the same as the working time of machine \( M \). Hence, the outer negation of PTIME quantifiers can be recognized in polynomial time.

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

(3) To compute \( Q \land Q' \) 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 $\text{lt}(Q, Q')[A, B, R] \iff Q[A, A']$, where $A' = \{a \mid Q'(B, R(a))\}$, for $R(a) = \{b \mid R(a, b)\}$. Notice that for every $a$ from the universe, $R(a)$ is a monadic predicate. Having this in mind we construct in polynomial time $A'$. To do this we execute the following procedure for every element from the universe. We initialize $A' = \emptyset$. Then we repeat for each $a$ from the universe the following: Firstly we compute $R(a)$. Then using the polynomial machine $M'$ we compute $Q'[B, R(a)]$. 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 $Q[A, A']$.

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

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

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

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

3.1.6. Theorem. Let $Q$ and $Q'$ be generalized quantifiers computable in polynomial time with respect to the size of a universe. Then the quantifiers: (1)
We have argued that PTIME quantifiers are closed under Boolean operations as well as under the polyadic lifts occurring frequently in natural language. In other words, these operations do not increase the complexity of quantifier semantics. As we can safely assume that most of the simple determiners in natural language are PTIME computable then the semantics of the polyadic quantifiers studied above is tractable. This seems to be good news for the theory of natural language processing. Unfortunately, not all natural language lifts behave so nicely from a computational perspective. In the next section we show that branching can produce NP-complete quantifier constructions from simple determiners. Speaking in the terminology of Blass and Gurevich (1986), introduced in Definition 2.5.3, some branching quantifiers are mighty.

3.2 Branching Quantifiers

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

3.2.1 Henkin’s Quantifiers are Mighty

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

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

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

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

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

(10) \( \left( \forall x \exists y \left( \forall z \exists w \right) \left( [V(x) \land T(z)) \implies (R(x, y) \land R(z, w) \land H(y, w)) \right) \right) \),

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”.
Chapter 3. Complexity of Polyadic Quantifiers

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 $Zxy [\varphi(x, y), \psi(x, y)]$ can be interpreted in a second-order language as:

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

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

3.2.1. Theorem. The quantifier $Z$ is mighty.

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

3.2.2 Proportional Branching Quantifiers are Mighty

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

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

$$\exists X \subseteq A \exists Y \subseteq B [(X, A) \in Q \land (Y, B) \in Q' \land X \times Y \subseteq R].$$

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

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

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

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

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

It has been shown by Merlijn Sevenster (2006) that the problem of recognizing the truth-value of formula (11) in finite models is NP-complete. Actually, it can also be proven that all proportional branching sentences, like (12), define an NP-complete class of finite models. In other words the following holds.
3.2.3. Theorem. Let $Q$ and $Q'$ be proportional quantifiers, then the quantifier $B\rho(Q, Q')$ is mighty.

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

3.2.3 Branching Counting Quantifiers are Mighty

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

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

Their branching readings have the following form:

\[
\left( \begin{array}{c}
\text{More than } k \ x : V(x) \\
\text{More than } m \ y : T(y)
\end{array} \right) H(x, y),
\]

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.

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

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 $C^{\geq A}$ of type (1) which says that the number of elements satisfying a given formula in a model $M$ is greater than the cardinality of a set $A \subseteq M$. Alternatively we could introduce a two-sorted variant of finite structures, augmented by an infinite number sort. Then we can define counting quantifiers in such a way that the numeric constants in a quantifier refer to the number domain (see e.g. Otto, 1997; Grädel and Gurevich, 1998).

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

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

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

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

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

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

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

3.2.7. Proposition. The quantifier \( \text{Br}(C^\geq A, C^\geq B) \) is mighty.

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

This constitutes another class of branching mighty quantifiers.

3.2.4 Linguistic Remark and Branching of Dissertation

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

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

3.3.1 Ramsey Theory and Quantifiers

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

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

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

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

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

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

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

²More details may be found in the monograph on Ramsey Theory by Graham et al. (1990).
Imagine an alien force, vastly more powerful than us landing on Earth and demanding the value of $R(5, 5)$ or they will destroy our planet. In that case, we should marshal all our computers and all our mathematicians and attempt to find the value. But suppose, instead, that they asked for $R(6, 6)$, we should attempt to destroy the aliens.\(^3\)

(see e.g. Spencer, 1987)

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

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

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

Let us start with the general definition of Ramsey quantifiers.

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

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

3.3.2 The Branching Reading of Hintikka’s Sentence

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

\(^3\)Currently the number $R(19, 19)$ is known (Luo et al., 2002). It is equal to $178,859,075,135,299$.\n
3.3. **Ramsey Quantifiers**

3.3.2. **Definition.** $\mathcal{M} \models R_{xy} \varphi(x, y)$ means that there is a set $A \subseteq M$ such that $\forall a \in M \exists b \in A \; E(a, b)$ and for each $a, b \in A$, $\mathcal{M} \models \varphi(a, b)$. ■

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

3.3.3. **Theorem.** The quantifier $R_{e}$ is mighty.

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

3.3.3 **Clique Quantifiers**

Let us start with simple Ramsey quantifiers expressing the **clique** problem.

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

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

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

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

3.3.5. **Definition.** Let us consider models of the form $\mathcal{M} = (M, A, \ldots)$. We define the Ramsey counting quantifier, $R_{A}$, as follows: $\mathcal{M} \models R_{Axy} \varphi(x, y)$ iff there is $X \subseteq M$ such that $\text{card}(X) \geq \text{card}(A)$ and for all $a, b \in X$, $\mathcal{M} \models \varphi(a, b)$. ■

The Ramsey quantifier $R_{A}$ expresses the general **clique** problem and as a result it inherits its complexity.

---

4 Compare this definition with Definition 3.2.4.
3.3.6. Proposition. The Ramsey quantifier $\mathcal{R}_A$ is mighty.

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

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

3.3.4 Proportional Ramsey Quantifiers

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

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

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

In this sense $q$ determines the proportional Ramsey quantifier $\mathcal{R}_q$.

3.3.8. Definition. $\mathcal{M} \models \mathcal{R}_q xy \varphi(x, y)$ iff there is a $q$-large (relative to $M$) $A \subseteq M$ such that for all $a, b \in A$, $\mathcal{M} \models \varphi(a, b)$. □

We will prove that for every rational number $0 < q < 1$ the corresponding Ramsey quantifier $\mathcal{R}_q$ defines an NP-complete class of finite models.\(^5\)

3.3.9. Theorem. For every rational number $q$, such that $0 < q < 1$, the corresponding Ramsey quantifier $\mathcal{R}_q$ is mighty.

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

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

\(^5\)The following result was obtained in cooperation with Marcin Mostowski (see Mostowski and Szymanik, 2007).
3.3. Ramsey Quantifiers

Now we can state the following lemma.

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

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

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

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

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

For proving our claim we need the following:

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

**Proof:**

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

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

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

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

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

So $m(1-q) < qn - k + 1$ and $m(1-q) + k - 1 < qn$.

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

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

Therefore, the following are equivalent:

1. In $G$ there is a clique of size at least $k$;
2. $\text{card}(Cl) \geq k$;
3. $\text{card}(Cl') \geq k + m$;
4. $\text{card}(Cl') \geq \lceil q(n + m) \rceil$;
5. $\text{card}(Cl') \geq q(n + m)$;
6. $\frac{\text{card}(Cl')}{\text{card}(V')} \geq \frac{q(n+m)}{n+m}$
7. The clique $Cl'$ is $q$-large in $G'$.

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

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

### 3.3.5 Tractable Ramsey Quantifiers

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

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

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

$$\text{card}(A) \geq f(\text{card}(U)).$$

Then we define Ramsey quantifiers corresponding to the notion of “$f$-large”.

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

Notice that the above definition is very general and covers all previously defined Ramsey quantifiers. For example, we can reformulate Theorem 3.3.9 in the following way:
3.3.14. **Corollary.** Let \( f(n) = \lceil rn \rceil \), for some rational number \( r \) such that \( 0 < r < 1 \). Then the quantifier \( R_f \) is mighty.

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

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

3.3.15. **Definition.** We say that a function \( f \) is **bounded** if

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

Otherwise, \( f \) is **unbounded.** ■

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

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

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

3.3.16. **Theorem.** If \( f \) is PTIME computable and bounded, then the Ramsey quantifier \( R_f \) is PTIME computable.
Proof Assume that \( f \) is PTIME computable and bounded. Then there exists a number \( m \) such that for every \( n \) the following disjunction holds:
\[
[f(n) < m \text{ or } n - m < f(n)].
\]
Let us fix a graph model \( G = (V, E) \), where \( \text{card}(V) = n \).

In the first case assume that \( f(n) < m \). First observe that if there exists a clique of size greater than \( f(n) \) then there has to also be a clique of size exactly \( f(n) \). Thus to decide whether \( G \in R_f \), it is enough to check if there is a clique of size \( f(n) \) in \( G \). We know that \( f(n) < m \). Hence we only need to examine all subgraphs up to \( m \) vertices. For each of them we can check in polynomial time whether it forms a clique. Hence, it is enough to observe that the number of all subgraphs of size between 1 up to \( m \) is bounded by a polynomial. In fact this is the case as the number of \( k \)-combinations from a set is smaller than the number of permutations with repetitions of length \( k \) from that set. Therefore, we have:
\[
\binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{m} \leq n^1 + n^2 + \ldots + n^m \leq m(n^m).
\]

Let us consider the second case; assume that \( n - m < f(n) \). This time we have to only check large subgraphs; to be precise, we need to examine all subgraphs containing from \( n \) down to \( n - m \) vertices. Again, the number of such subgraphs is bounded by a polynomial for fixed \( m \). We use the following well known equality \( \binom{n}{n-k} = \binom{n}{k} \) to show that we have to inspect only a polynomial number of subsets:
\[
\binom{n}{n} + \binom{n}{n-1} + \ldots + \binom{n}{n-m} = \binom{n}{n} + \binom{n}{1} + \ldots + \binom{n}{m} \leq 1 + n^1 + n^2 + \ldots + n^m \leq m(n^m).
\]

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

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

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

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

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

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

### 3.4 Summary

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

- PTIME quantifiers are closed under Boolean operations, iteration, cumulation, and resumption.

- When branching PTIME quantifiers we may arrive at NP-complete polyadic quantifiers, e.g. branching proportional quantifiers are mighty.

- Ramsey counting quantifiers are mighty.

- Proportional Ramsey quantifiers are mighty.

- Bounded Ramsey quantifiers (and branching quantifiers) are PTIME computable.

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

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

We have shown that proportional Ramsey quantifiers define NP-complete classes of finite models. On the other hand, we also observed that bounded Ramsey quantifiers are in PTIME. It is an open problem where the precise border lies between tractable and mighty Ramsey quantifiers.
3.4.1. **QUESTION.** Can we prove under some complexity assumptions that the PTIME Ramsey quantifiers are exactly the bounded Ramsey quantifiers?

3.4.2. **QUESTION.** Is it the case that for every function \( f \) from some class we have a duality theorem, i.e., \( R_f \) is either PTIME computable or NP-complete?

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

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

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

In the next Chapter we will argue that they do.