Generalized Quantifiers *

Jouko Väänänen
Department of Mathematics
University of Helsinki
Helsinki, Finland

Abstract

We review recent work in the field of generalized quantifiers on finite models. We give an idea of the methods that are available in this area. Main emphasis is on definability issues, such as whether there is a logic for the PTIME properties of unordered finite models.

1 Introduction

The ordinary quantifiers "for some" and "for all" are not sufficient for expressing some basic mathematical concepts. This led **Mostowski** [22] to introduce in 1957 *generalized quantifiers*, such as "for infinitely many" and "for uncountably many". In the 1960's and 1970's these and other similar quantifiers were intensively studied by logicians. In this decade generalized quantifiers re-emerged in the framework of finite structures. Researchers in *descriptive complexity theory* and *natural language semantics* were looking at ways to formalize expressions like "for at least half" and "for an even number". It turned out that the concept of generalized quantifier introduced by Mostowski, and further extended by **Lindström** [21], was the right answer.

The goal of descriptive complexity theory is to classify problems, not according to how much resources they need when solved by means of a Turing

*Appeared in the Bulletin of the European Association for Theoretical Computer Science, 62 (1997), 115–136. Reprinted in Generalized Quantifiers and Computation, Lecture Notes in Computer Science, vol. 1754, Springer, 1999.

machine, but according to how powerful logical languages are necessary for describing the problems. For a quick example, let us consider some graph problems, say the problem whether a graph is triangle-free.

We are given a graph, that is, a pair (G, E), where G is a set and E is a binary predicate representing the edge-relation. Even before we think about triangle-freeness, we may ask how difficult it is to express the fact that (G, E) is indeed a graph. We have to check that E is symmetric and anti-reflexive. These two axioms can be easily expressed in a logical language as follows: In what follows, x and y are variables for elements of the domain G and $E(x, y)$ stands for the assertion that x and y are in the edge-relation. Using standard logical notation, $\forall$ means "for all", $\rightarrow$ means "implies", $\neg$ means "not" and $\wedge$ means "and", the following sentence expresses the fact that (G, E) is a graph:

$$\forall x \forall y (E(x, y) \rightarrow E(y, x)) \wedge \forall x \neg E(x, x). \tag{1}$$

Likewise, the sentence

$$\forall x \forall y \forall z \neg (E(x, y) \wedge E(y, z) \wedge E(z, x)). \tag{2}$$

expresses triangle-freeness of the graph (G, E). From the point of view of descriptive complexity theory we may now say that to express the graph-axioms, and also the triangle-freeness of a graph, we do not need more than a logical language which has variables for elements and permits the use of $\forall$, $\neg$, $\wedge$ and $\rightarrow$. The smallest such language is called *first order logic* FO.

The logical language FO has become an extremely successful tool in logic in this century. However, this success is almost solely due to its handiness in expressing completeness properties of infinite structures such as "for every number there is a bigger number" or "between any two distinct numbers there is another number" etc. Problems arising in computer science do not usually involve structures with universal completeness properties, not least because these structures (databases, computations, etc) are *finite*. Respectively, all the usual methods in the theoretical study of FO almost systematically fail in the finite context (see e.g. [11, 12]). Indeed, first order logic is not at all handy in expressing interesting properties of finite structures. Let us take the path-problem as an example. From a programming point of view it is natural to express the existence of a path from x to y in a graph (G, E) by a formula $\mathrm{Conn}(x, y)$ such that

$$\mathrm{Conn}(x, y) \leftrightarrow [E(x, y) \vee \exists z (E(x, z) \wedge \mathrm{Conn}(z, y))]. \tag{3}$$

This sentence features $\leftrightarrow$ ("if and only if"), $\exists$ ("there exists") and $\vee$ ("or"), which are all definable in terms of the operations of FO. But more interestingly, it seems that in order to get the formula $\mathrm{Conn}(x, y)$ we have to "solve" the equivalence (3) since Conn occurs on both sides. Such solutions are called *fixed points*. In general there may be several different fixed points, but there is a simple way of guaranteeing that a unique least fixed point exists. In (3) the existence of a unique least fixed point characteristically follows from the fact that on the right hand side the predicate Conn occurs only positively. The smallest logic extending first order logic where such fixed points can be expressed is called *fixed point logic* FP. It was introduced by **Aho** and **Ullman** [1] in 1979. They also proved that the least fixed point of the equivalence (3) is not first order definable.

Fixed point logic is definitely very different from first order logic. Expressions of first order logic can be written down on a piece of paper, but how to write down the expression Conn? If we make a try, changing variables to avoid confusion, the result is likely to look like this: $\mathrm{Conn}(x, y) \leftrightarrow$

$$E(x, y) \vee \exists z_1(E(x, z_1) \wedge (E(z_1, y) \vee \exists z_2(E(z_1, z_2) \wedge (E(z_2, y) \vee \ldots))).$$

This sentence is never-ending! The expressions of fixed point logic are not sentences in the ordinary sense of the word but some kind of self-referential recursive generalized sentences. Despite this difficulty in construing fixed point queries as sentences, fixed point logic has a very clear computational content. Whenever a graph and a fixed point expression is given, it is immediate how to check whether the graph satisfies the expression, and this can be done in polynomial time in the size of the graph.

It would be tempting to conjecture that not only is every fixed point query in polynomial time, but conversely every polynomial time graph property is expressible as a fixed point query. Indeed, **Immerman** [19] and **Vardi** [24] showed in 1982 that this is true in the special case that the graph is endowed with a linear ordering of the vertices. In such a case it is possible to use tuples of vertices to build a model of a Turing machine inside the graph and imitate the polynomial time property by a suitable fixed point sentence. So in the presence of an ordering the fixed point approach is very powerful. However, what if we do not have an ordering of the vertices of the graph?

It is relatively easy to see with present-day game-theoretic techniques that the polynomial time query "the number of vertices is even" is not expressible

in fixed point logic on unordered graphs. The same is true of *all* non-trivial counting queries. This observation has led to the following problem:

> Is there some natural extension of fixed point logic which expresses exactly the polynomial time queries on unordered graphs? (4)

raised first by **Chandra** and **Harel** [3].

If it were to be the case that P = NP, then such a natural extension would exist, namely existential second order logic Σ^1_1, defined below. First let us observe that first order logic FO makes sense in a framework that is much more general than just graphs. It is customary in logic to consider structures of a very general type, such as ordered structures, directed graphs, hypergraphs, groups, fields, etc. Common to all these structures is that there is one domain and one or more relations (functions can be treated as relations) and constants on this domain. **Codd** [4] defined on such finite structures the so called *relation algebra* as a kind of minimal database query language. Relation algebra is essentially the same thing as FO. More exactly, a *structure* consists of a set A, a sequence of relations on A, and a sequence of distinguished constants on A. Each relation is a subset of some Cartesian product A^k of A. The number k is called the *arity* of the relation. First order logic for such abstract structures has a name for each relation and constant of the structure. (We use the same symbol for an object and its name, whenever no confusion arises.) The names of the relations and constant of a structure is called the *vocabulary* of the structure.

For example, if (G, E) is a graph and X is a subset of G, we can form a new structure (G, E, X) which has one relation E of arity 2 and one relation X of arity 1. We can say in first order logic that X contains neighbors of its elements (i.e. X is a union of connected components):

$$\forall x \forall y((X(x) \wedge E(x, y)) \rightarrow X(y)).$$

Here $X(t)$ is interpreted as "t is in X". In existential second-order logic Σ^1_1 we can form expressions such as

$$\exists X(X(x) \wedge \neg X(y) \wedge \forall z \forall u((X(z) \wedge E(z, u)) \rightarrow X(u))). \quad (5)$$

Here $\exists X$ is a so called second order quantifier because it binds a relation variable X rather than an element variable like $\exists z$ in (3). Note that (5) says

that there is no path from x to y. Thus it is equivalent to $\neg\mathrm{Conn}(x, y)$. This shows that Σ^1_1 can express things which are not first order definable.

No-one knows whether Σ^1_1 is closed under negation or not in the framework of finite models. In infinite models infinity itself is a Σ^1_1 concept. Its complement - finiteness - is not Σ^1_1, as follows easily from the so called Compactness Theorem of infinite model theory.

Fagin [9] proved that a query is NP if and only if it is expressible in Σ^1_1, in symbols NP = Σ^1_1. The role of guessing, inherent in non-deterministic computations, is played by existential second order quantifiers. Fagin's result holds on all structures, ordered or not, because if an order was not present, we could guess an ordering and continue as if the ordering was present. (Recall that ordering is used to simulate a Turing machine inside a structure).

It is possible to express every fixed point query in Σ^1_1. Thus if P = NP, we have the extension Σ^1_1 of FP which expresses exactly the polynomial time queries on graphs or indeed on any structures. The message is, that if we were able to answer question (4) in the negative, we would have proved P $\neq$ NP.

The point of generalized quantifiers (to be defined below) is that they provide a very general yet coherent and mathematically exact approach to extending FO and FP. Proof techniques developed for them provide one possible road to analyzing question (4) and other open questions of descriptive complexity theory.

2 Generalized quantifiers – definition

We pointed out above, that there is no fixed point expression (and no first order expression) which would say that the number of vertices of a graph is even. Neither can we say in FP (or FO) that the degree of a vertex is even, or that at last half of the vertices have degree ≥ 3, or that two vertices have the same degree, etc. There is an endless list of examples of simple properties which cannot be captured by FP. This motivates the following idea: We extend FO (and FP) by allowing a new operation $Q_{\mathrm{even}}x(\ldots x\ldots)$ with the interpretation

$$Q_{\mathrm{even}}x(\ldots x\ldots) \Leftrightarrow \text{ the number of } x \text{ with } \ldots x\ldots \text{ is even.} \qquad (6)$$

Now the expression $Q_{\text{even}}xE(x,y)$ says in a graph that the vertex y has even degree. By adding another new operation

$$Q_{\text{half}}x(\ldots x\ldots) \Leftrightarrow \text{ at least half of all elements } x \text{ satisfy } \ldots x\ldots,$$

we can say things like at least half of the vertices of a graph have degree ≥ 3:

$$Q_{\text{half}}x\exists y\exists z\exists u(E(x,y)\wedge E(x,z)\wedge E(x,u)\wedge y\neq z\wedge y\neq u\wedge z\neq u).$$

The operations Q_{even} and Q_{half} are examples of generalized quantifiers. The extension of FO by Q_{even} is denoted by $\text{FO}(Q_{\text{even}})$ and the extension by Q_{half} is denoted by $\text{FO}(Q_{\text{half}})$. It is easy to verify that all queries that can be expressed in $\text{FO}(Q_{\text{even}})$ or $\text{FO}(Q_{\text{half}})$ are polynomial time computable, but not every polynomial time computable query is expressible in $\text{FO}(Q_{\text{even}})$ or $\text{FO}(Q_{\text{half}})$.

There is an element of arbitrariness in the definition of Q_{even} and Q_{half} above. One gets the feeling that something was needed and it was just thrown in *ad hoc*. The point is that generalized quantifiers provide a way of extending a language in a *minimal* way. Any extension of FO in which we can say one way or other that a predicate is satisfied by an even number of elements, and which satisfies some natural regularity properties, actually contains $\text{FO}(Q_{\text{even}})$.

Let us consider Hamiltonicity of a graph as an example. The probability that a randomly chosen finite graph is Hamiltonian tends to 1 when the size of the graph increases. There are logics, like FO and FP for which a *zero-one law* holds, that is, whatever sentence of the logic we consider, the probability that a randomly chosen finite structure satisfies that sentence tends to 0 or 1 when the size of the graph increases (see [13] for an informal discussion of this). Since Hamiltonicity cannot be expressed either in FO or in FP, the question was raised whether there is some extension of these logics in which Hamiltonicity can be expressed and which has a zero-one law. **Dawar** and **Grädel** [7] proved that the extension of FO obtained by adding the generalized quantifier

$$Q_{\text{Ham}}xy(\ldots x\ldots y\ldots) \quad \Leftrightarrow \quad \begin{array}{l}\text{the graph with the edge-relation}\\ \{(x,y):\ldots x\ldots y\ldots\} \text{ is Hamiltonian}\end{array} \tag{7}$$

does not have the zero-one law. Thus no extension of FO capable of expressing Hamiltonicity can have a zero-one law.

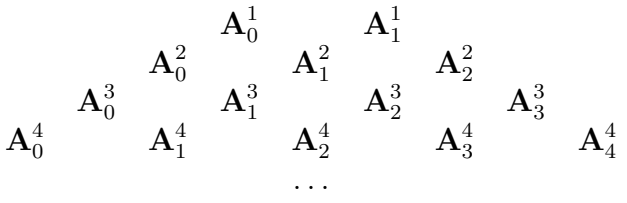

Figure 1: Models with one unary predicate

Definition 1 ([21]) *Suppose L is a vocabulary. Any collection Q of structures of vocabulary L, that is closed under isomorphisms, is called a generalized quantifier of vocabulary L.*

The concept of generalized quantifier seems to be extremely general, and does not appear to have anything to do with the ordinary quantifiers $\forall$ and $\exists$.

Let us discuss generality first. The important part of the definition is closure under isomorphisms. To see what this means, and how it limits generality, suppose L consists of one predicate symbol R which is unary i.e. of arity one. Up to isomorphism there are just $n+1$ non-isomorphic L-structures of size n, namely for each $m \leq n$ the structure $\mathbf{A}_m^n$ with $\{1, \ldots, n\}$ as universe and $\{1, \ldots, m\}$ as the interpretation of R. We can picture these structures as a pyramid (see Figure 1). On vertical rows the size of the model is constant. On rows parallel to the left side of the pyramid the size of the predicate R is constant. On rows parallel to the right side the size of the complement of R is constant.

Now a generalized quantifier of vocabulary L is essentially just a subset of this pyramid. Thus we have a pretty good picture of what kind of generalized quantifiers of vocabulary L there are. For more general vocabularies the picture becomes less and less clear.

The relation between generalized quantifiers and ordinary quantifiers becomes evident from the actual use of generalized quantifiers. Suppose Q is as in the above definition. For simplicity, assume L consists of a relation symbol R, which is binary, i.e. of arity two, and of X, which is unary, i.e. of arity one. Using the quantifier Q we can talk about pairs (x, y) of elements satisfying some condition $\ldots x \ldots y \ldots$ and of elements z satisfying another

condition $\_\_\_z\_\_\_$. The expression

$$Qxyz(\ldots x \ldots y \ldots)(\_\_\_z\_\_\_)$$

says in a structure $\mathbf{A}$ with universe A that the L-structure (A, R, X) with universe A and

$$\begin{aligned} R &= \{(x, y) \in A : \ldots x \ldots y \ldots \text{ holds in } \mathbf{A}\} \\ X &= \{z \in A : \_\_\_z\_\_\_ \text{ holds in } \mathbf{A}\}, \end{aligned}$$

is in the set Q.

In this framework the ordinary quantifier $\exists$ can be defined as a generalized quantifier of vocabulary $\{X\}$, X unary, as follows:

$$\exists = \{(A, X) : X \subseteq A, X \neq \emptyset\}.$$

Thus $\exists x(\ldots x \ldots)$ holds in a structure $\mathbf{A}$ if and only if $(A, X) \in \exists$ where

$$X = \{x \in A : \ldots x \ldots \text{ holds in } \mathbf{A}\}.$$

Similarly

$$\forall = \{(A, X) : X = A\}.$$

Let us denote by $\mathrm{FO}(Q)$ the extension of first order logic by the generalized quantifier Q. There is no special difficulty in adding a generalized quantifier Q to FP, although some attention has to be paid to the details. (Technically speaking, we should speak about the so called inflationary fixed point logic, unless Q is a so called monotone quantifier, but we disregard this detail here.) We denote the result by $\mathrm{FP}(Q)$.

Do generalized quantifiers lead to a solution of question (4)? The answer is "yes and no", as we shall see below. However, it is interesting to note already here that on *almost all* structures PTIME = $\mathrm{FP}(Q_{\mathrm{even}})$ by a recent result of **Hella, Kolaitis and Luosto** [16]. This means that there is a representation of PTIME queries in terms of $\mathrm{FP}(Q_{\mathrm{even}})$ which holds on a randomly chosen finite structure with a probability which tends to one as the size of the structure increases. Even such a weak representation would not be possible with FO or FP alone.

3 A hierarchy of generalized quantifiers

To evaluate the merits of the concept of generalized quantifier we have to look at the results that it gives rise to. For example, hierarchy results are in general rather rare in complexity theory, while strong methods exist for proving hierarchy results for generalized quantifiers.

The hierarchies of generalized quantifiers are based on counting how many variables and in how many formulas the quantifier binds. For example, the quantifier Q_{even} of (6) binds one variable x in one formula. We say that it has type (1). The quantifier

$$\mathrm{I}xy(\ldots x\ldots)(\_\_y\_\_) \iff \begin{array}{l}\text{there are as many } x \text{ with } \ldots x \ldots \\ \text{as there are } y \text{ with } \_\_y\_\_\end{array}$$

binds two variables (x and y) in two formulas, one variable per formula. We say that it has type (1,1). Finally Q_{Ham} of (7) binds two variables in one formula, we say that it has type (2).

In general, a *type* is a finite sequence $\tau = (t_1, \ldots, t_k)$ of positive integers with $t_1 \geq \ldots \geq t_k$. A *vocabulary* of type τ has an t_i-ary predicate symbol R_i for each $i = 1, \ldots, k$. A generalized quantifier has type τ if its vocabulary is of type τ.

The type of a vocabulary determines how many models (up to isomorphism) there are of that vocabulary in each size of the domain. If the type τ is *unary*, that is, $t_1 = \ldots = t_k = 1$, then there are

$$\left(\begin{array}{c} n + 2^k - 1 \\ 2^k - 1 \end{array}\right)$$

non-isomorphic models of vocabulary L of size n. If the vocabulary L is non-unary, that is, $\max\{t_1, \ldots, t_k\} \geq 2$, then the number of non-isomorphic models of vocabulary L and of size n is harder to compute, but asymptotically it is ([8])

$$\frac{1}{n!} 2^{n^{t_1} + \ldots + n^{t_k}}.$$

These formulas give immediately an idea of the number of quantifiers of a given type. The number *per se* is infinite, but we can get a finite number by considering the restrictions of the quantifier to a fixed finite domain A of

cardinality n. If L is unary of type $(t_1, \ldots, t_k)$, the number of restrictions of quantifiers of vocabulary L to A is, of course

$$2^{\binom{n+2^k-1}{2^k-1}}$$

and if L is non-unary, the number is

$$2^{\frac{1}{n!}2^{n^{t_1}+\ldots+n^{t_k}}}.$$

The point of calculating these numbers is that they give a simple counting method (invented by **Per Lindström**) for proving results about generalized quantifiers. Suppose we want to construct a generalized quantifier Q of type (1,1) which is not definable in FO(Q') for any generalized quantifier Q' of type (1). Let us make a list

$$\phi_0(Q'), \phi_1(Q'), \phi_2(Q'), \ldots$$

of the possible sentences of FO(Q') that could give rise to a definition of such a Q. All we have to take care of is that in size n the quantifier Q is defined differently than what $\phi_n(Q')$ says. But (roughly speaking) there are $2^{(n+3)(n+2)(n+1)/3}$ possibilities for Q and only 2^{n+1} possibilities for Q'. So sheer counting shows that such a Q can be constructed. By elaborating this idea it is possible to prove a hierarchy theorem which demonstrates the existence of genuinely new quantifiers on each level of the type hierarchy (see below).

If $\tau = (t_1, \ldots, t_k)$ and $\tau' = (t'_1, \ldots, t'_{k'})$ are types, we let $\tau < \tau'$ if τ precedes τ' in the lexicographic order, that is, $t_i < t'_i$ for the least i such that $t_i \neq t'_i$ (or $k < k'$ if $t_i = t'_i$ for all $i = 1, \ldots, k$). Thus

$$(1) < (1,1) < \ldots < (2) < (2,1) < (2,1,1) < \ldots < (2,2) < \ldots (3) < \ldots.$$

The order-type of this ordering is the infinite ordinal number ω^ω.

Theorem 2 (The Hierarchy Theorem [17]) *For every type $\tau = (t_1, \ldots, t_k)$ there is a generalized quantifier Q of type τ such that Q is not definable in* FO(Q') *for any Q' of type $< \tau$.*

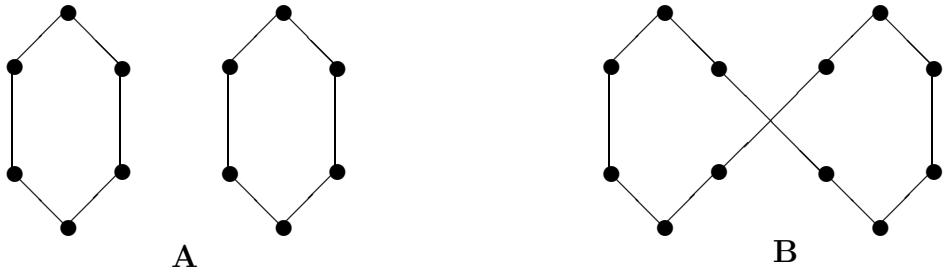

Figure 2: Two graphs

In loose terms: we get something genuinely new on every level of types. This result permits many refinements and variations. The most important of them says that Q can be chosen to be LOGSPACE, at least if $t_1 = \ldots = t_k$. Various other constraints can be imposed on Q. An essential feature of this result is that it is a pure existence result, reminiscent of Cantor's diagonalization method. We may force the abstract object Q to satisfy some nice properties but we cannot "put our finger" on Q.

4 Games and quantifiers

The Hierarchy Theorem of the previous section establishes the richness of the family of all generalized quantifiers. To study properties of individual quantifiers more refined methods are needed. Before introducing the main technical tool in the study of generalized quantifiers, the method of games, let us discuss the general problem of separating models by logical means.

Suppose $\mathbf{A}$ and $\mathbf{B}$ are two models of the same vocabulary. We may assume they are both graphs, or we may assume they are two databases (of similar type). Picture 2 shows a simple example of the case that $\mathbf{A}$ and $\mathbf{B}$ are graphs. Graph $\mathbf{B}$ is connected while $\mathbf{A}$ is not, so the graphs are not isomorphic. What is the simplest way of describing the difference between $\mathbf{A}$ and $\mathbf{B}$? Certainly the difference can be described in FO with the sentence

$$\exists x_1 \exists x_2 \ldots \exists x_{12}(E(x_1, x_2) \wedge E(x_2, x_3) \wedge \ldots \wedge E(x_{11}, x_{12}) \wedge E(x_{12}, x_1))$$

but this raises the new question, whether we can do the same with a sentence which would work for larger cycles, too.

On the other hand, the sentence (recall (5))

$$\exists x \exists y (x \neq y \wedge \exists X (X(x) \wedge \neg X(y) \wedge \forall z \forall u ((X(z) \wedge E(z,u)) \rightarrow X(u))) \quad (8)$$

is a sentence of Σ^1_1 which is true in $\mathbf{A}$ and false in $\mathbf{B}$. To see that this sentence is true in $\mathbf{A}$, choose x from the left-hand cycle, y from the right-hand cycle, and let X be the whole left-hand cycle. Note that (8) works regardless of the sizes of $\mathbf{A}$ and $\mathbf{B}$ as long as $\mathbf{A}$ consists of two cycles and $\mathbf{B}$ of one. So (8) looks like a good logical method to describe the difference between $\mathbf{A}$ and $\mathbf{B}$. But (8) is unsatisfactory as it uses the "second order" quantifier $\exists X$, and no PTIME algorithm is known for checking the truth of sentences involving $\exists X$.

Then there is the fixed point formula $\mathrm{Conn}(x,y)$ of (3), but it is (in a sense) infinitary and therefore unsatisfactory. Of course we can also take the generalized quantifier of type (2)

$$Q_{\mathrm{conn}} xy(\ldots x \ldots y \ldots) \iff \begin{array}{l} \text{the graph which has an edge between} \\ \text{every } x \text{ and } y \text{ with } \ldots x \ldots y \ldots \\ \text{is connected.} \end{array}$$

Then $\mathrm{FO}(Q_{\mathrm{conn}})$ can in a trivial way describe a difference between $\mathbf{A}$ and $\mathbf{B}$. The concept of generalized quantifier provides one possible framework for asking questions like: is $\mathrm{FO}(Q_{\mathrm{conn}})$ really the simplest logic in which the difference of $\mathbf{A}$ and $\mathbf{B}$ of Picture 2 can be expressed? So what is simpler than $\mathrm{FO}(Q_{\mathrm{conn}})$? What if we replace Q_{conn} by a quantifier of simpler type? The types simpler than that of Q are the unary types (1), (1,1), ...

Proposition 3 ([15]) *No generalized quantifier of a unary type can express connectivity of graphs.*

The main idea behind the proof is the use of the following "bijective" game. The game $BG_k(\mathbf{A}, \mathbf{B})$ is played on two models $\mathbf{A}$ and $\mathbf{B}$. There are two players called I and II. Player II starts by choosing a bijection f_1 between the domain A of $\mathbf{A}$ and the domain B of $\mathbf{B}$. (If there is no such f_1, Player II loses.) Then Player I chooses an element a_1 of A. Now Player II chooses again a bijection f_2 and Player I chooses again an element a_2 of A. The game

continues like this for k moves. In the end we have a correspondence

$$\begin{array}{c} a_1 \mapsto f_1(a_1) \\ a_2 \mapsto f_2(a_2) \\ \vdots \\ a_k \mapsto f_k(a_k). \end{array} \tag{9}$$

Player II is the winner if he has been able to play all the k moves and the produced correspondence (9) is a partial isomorphism, i.e., preserves structure. For example, if $\mathbf{A}$ and $\mathbf{B}$ are graphs, this means that there is an edge between a_i and a_j in $\mathbf{A}$ if and only if there is and edge between $f_i(a_i)$ and $f_j(a_j)$ in $\mathbf{B}$ (and $a_i = a_j \iff f_i(a_i) = f_j(a_j)$). A winning strategy of Player II in $BG_3(\mathbf{A}, \mathbf{B})$ for the graphs of Picture 2 can be described as follows. The choice of f_1 can be arbitrary (because every single element looks the same in both models: edge to the left and edge to the right). After Player I has chosen a_1, Player II chooses f_2 so that it maps any element at a distance $d \leq 2$ from a_1 to an element at the same distance d from $f_1(a_1)$ preserving orientation. Otherwise f_2 can be quite arbitrary as long as it is a bijection. After Player I chooses a_2, Player II chooses f_3 so that it maps a_1, a_2, and their immediate neighbors to $f_1(a_1)$, $f_2(a_2)$, and their immediate neighbors, preserving again orientation. After Player I has chosen a_3 it is easy to see that the mapping $a_i \mapsto f_i(a_i)$ preserves the edge-relation.

Let $\mathbf{Q}_1$ be the family of all unary generalized quantifiers and $\mathrm{FO}(\mathbf{Q}_1)$ the extension of FO by all quantifiers in $\mathbf{Q}_1$.

Theorem 4 ([14]) *A sufficient condition for a property P of finite models to be undefinable in* $\mathrm{FO}(\mathbf{Q}_1)$ *is that for all numbers k there are models* $\mathbf{A}$ *and* $\mathbf{B}$ *such that*

(i) $\mathbf{A}$ *has property P,*

(ii) $\mathbf{B}$ *does not have property P,*

(iii) Player II has a winning strategy in the game $BG_k(\mathbf{A}, \mathbf{B})$.

This theorem reduces the logical question whether a property P is expressible in terms of propositional connectives and quantifiers of a very general albeit unary nature, to the purely mathematical question whether certain

models with certain combinatorial properties exist. For example, graphs like in Picture 2 together with Theorem 4 provide a proof of Proposition 3.

The proof of Theorem 4 is lengthy but not particularly difficult. The sufficiency of the condition for the undefinability of P is based on the following idea: Let us assume the model $\mathbf{A}$ satisfies a sentence $Qx(\dots x\dots)$, supposedly defining P, where Q is a generalized quantifier of type (1). (In general, the defining sentence need not start with Q.) Furthermore, suppose Player II has a winning strategy in $BG_k(\mathbf{A},\mathbf{B})$ with k sufficiently large. The actual choice of k is based on an inspection of the formula $\dots x\dots$. This strategy advises Player II to choose some bijection f_1. Since $Qx(\dots x\dots)$ is true in $\mathbf{A}$, the structure (A,X) is in Q, where A is the domain of $\mathbf{A}$ and X is the set of elements x of A which satisfy $\dots x\dots$. Let B be the domain of $\mathbf{B}$ and Y the set of elements x of B which satisfy the same condition $\dots x\dots$ in $\mathbf{B}$. Now comes the fundamental idea of the game BG_k. We claim that

$$f_1 : (A,X) \cong (B,Y). \tag{10}$$

Suppose $x \in A$. We have to prove $a \in X \iff f_1(a) \in Y$. Suppose therefore e.g. $x \in X$. Thus x satisfies the condition $\dots x\dots$ in $\mathbf{A}$. Let us let Player I choose $a_1 = x$ in the game $BG_k(\mathbf{A},\mathbf{B})$. Because Player II is playing a winning strategy, we know that a_1 satisfies the same basic relations of $\mathbf{A}$ as $f_1(a_1)$ satisfies in $\mathbf{B}$. By formulating the induction hypothesis appropriately, we actually know that a_1 satisfies the same "definable" relations in $\mathbf{A}$ as $f_1(a_1)$ satisfies in $\mathbf{B}$. Here "definable" means definability in a restricted sense, which however includes the relation $\dots x\dots$. Since we know that a_1 satisfies condition $\dots a_1\dots$ in $\mathbf{A}$, we can conclude that $f_1(a_1)$ satisfies condition $\dots f_1(a_1)\dots$ in $\mathbf{B}$. In particular, $f_1(a_1) \in Y$, as desired.

Now that we know (10), we may draw from $(A,X) \in Q$ the conclusion $(B,Y) \in Q$, which means, by definition, that the sentence $Qx(\dots x\dots)$ is true in $\mathbf{B}$. Thus $Qx(\dots x\dots)$ cannot, after all, define the property P.

Theorem 4 has been successfully used to show that various graph properties (e.g. planarity) are not expressible in terms of unary quantifiers [23]. Such results can be seen as formalizations of the vague intuition that some properties of binary relations cannot be reduced to properties of cardinalities of definable sets.

What is really remarkable about Theorem 4 is that it generalizes both to non-unary quantifiers and to extensions of *fixed point logic* by generalized

quantifiers. Let us call a type $\tau = (t_1, \ldots, t_k)$ *r-ary* if $\max\{t_1, \ldots, t_k\} \leq r$. A vocabulary (and a generalized quantifier) is called *r-ary* if its type is r-ary. The word "binary" is generally used for 2-ary.

The game $BG_k^r(\mathbf{A}, \mathbf{B})$, an r-ary version of $BG_k(\mathbf{A}, \mathbf{B})$, is defined like $BG_k(\mathbf{A}, \mathbf{B})$ except that when Player I moves, he chooses an r-tuple $(a_i^1, \ldots, a_i^r)$ rather than a single element a_i. So after k moves we have the correspondence

$$\begin{array}{c} a_1^j \mapsto f_1(a_1^j), j = 1, \ldots, r \\ \vdots \\ a_k^j \mapsto f_k(a_k^j), j = 1, \ldots, r \end{array} \tag{11}$$

and Player II wins if this is a partial isomorphism. This game is immensely more difficult to win for Player II than $BG_k(\mathbf{A}, \mathbf{B})$ since already f_1 has to preserve all r-tuples. For graphs this means that Player II cannot even make the first move without losing unless $\mathbf{A}$ and $\mathbf{B}$ are isomorphic.

Let $\mathbf{Q}_r$ denote the family of all r-ary generalized quantifiers, and $\mathrm{FO}(\mathbf{Q}_r)$ the corresponding extension of FO.

Theorem 5 ([14]) *Theorem 4 holds for r-ary generalized quantifiers, that is, if* $\mathbf{Q}_1$ *is replaced by* $\mathbf{Q}_r$ *and* $BG_k(\mathbf{A}, \mathbf{B})$ *by* $BG_k^r(\mathbf{A}, \mathbf{B})$.

By means of this criterion it is possible to show that the following "Ramsey-quantifier" of type (r)

$$Ram^r \mathbf{x}(\ldots \mathbf{x} \ldots) \iff \exists X(X \text{ large } \wedge \forall \mathbf{x} \in [X]^r(\ldots \mathbf{x} \ldots)) \tag{12}$$

is not definable in $\mathrm{FO}(\mathbf{Q}_{r-1})$[18]. Here "large" can be anything reasonable, e.g. $n/2$ or $\log(n)$, where n is the size of the model. We have denoted an r-tuple $(x_1, \ldots, x_r)$ by the bold face $\mathbf{x}$. $[X]^r$ means the set of all r-tuples of distinct elements of X. The Ramsey-quantifier is not known to be PTIME in general. Indeed, in many natural instances it is NP-complete. But there are other explicit r-ary quantifiers, even in FP, which are not definable in $\mathrm{FO}(\mathbf{Q}_{r-1})$. One example is the transitive closure quantifier on r-tuples (see below) [10]. An interesting conclusion from this is:

Theorem 6 ([14]) *Fixed point logic* FP *cannot be represented as the extension of* FO *by finitely many generalized quantifiers.*

Namely, the arities of the finitely many quantifiers would have a common upper bound r, so such a representation would contradict the existence of FP queries, which are not definable in $\mathrm{FO}(\mathbf{Q}_r)$. This is a strong manifestation of the inherent incapability of (finitely many) generalized quantifiers to express recursion.

What about *combining* recursion and generalized quantifiers? We have already remarked that any generalized quantifier can be added to fixed point logic FP. Let $\mathrm{FP}(\mathbf{Q}_r)$ denote the extension of FP by all r-ary quantifiers.

We can replace the logic $\mathrm{FO}(\mathbf{Q}_1)$ by the logic $\mathrm{FP}(\mathbf{Q}_r)$ in Theorem 4, if we simultaneously replace the game $BG_k(\mathbf{A}, \mathbf{B})$ by a new game $BPG_k^r(\mathbf{A}, \mathbf{B})$, called the *bijective pebble game*, which we now define. This game is like $BG_k^r(\mathbf{A}, \mathbf{B})$ but it is potentially infinitely long (although it does not make sense to play more than n^k moves, where n is the size of $\mathbf{A}$). Therefore it is even harder for Player II than $BG_k^r(\mathbf{A}, \mathbf{B})$. To counterbalance the length there is a special mechanism to limit the size of the final correspondence (cf. (11)). There are k pebbles. While in $BG_k^r(\mathbf{A}, \mathbf{B})$ Player I chooses some r-tuple, in the game $BPG_k^r(\mathbf{A}, \mathbf{B})$ he puts pebbles on elements of an r-tuple. What is the difference? Eventually he runs out of pebbles! Then he can take back some pebbles that he has used already and reuse them. At some point Player I decides that the game has lasted long enough. Now we look at the correspondence generated by the elements that have a pebble on them in the final position. So we form the mapping

$$\begin{aligned} a_{i_1}^{j_1} &\mapsto f_{i_1}(a_{i_1}^{j_1}) \\ &\vdots \\ a_{i_k}^{j_k} &\mapsto f_{i_k}(a_{i_k}^{j_k}), \end{aligned} \tag{13}$$

where $a_{i_1}^{j_1}, \ldots, a_{i_k}^{j_k}$ is the sequence of elements with a pebble on them at the end. If this is a partial isomorphism, then Player II has won.

The challenge that $BPG_k^r(\mathbf{A}, \mathbf{B})$ presents for Player II is that he should find a strategy which works no matter how long the game has been going on and no matter how long it will go on. For example, if $\mathbf{A}$ and $\mathbf{B}$ are as in Picture 2, Player II wins $BPG_k^1(\mathbf{A}, \mathbf{B})$ if and only if $k = 1$.

Theorem 7 ([14]) *Theorem 4 holds for fixed point logic and r-ary generalized quantifier, that is, if* $\mathrm{FO}(\mathbf{Q}_1)$ *is replaced by* $\mathrm{FP}(\mathbf{Q}_r)$ *and* $BG_k(\mathbf{A}, \mathbf{B})$ *by* $BPG_k^r(\mathbf{A}, \mathbf{B})$.

With this criterion it is possible to prove that certain properties of r-ary relations are not expressible in terms of $(r-1)$-ary generalized quantifiers even if taking least fixed points of formulas are allowed. The first result in this direction, due to **Cai**, **Fürer** and **Immerman** [2], exhibited a LOGSPACE property of graphs that is not expressible in $\mathrm{FP}(\mathbf{Q}_1)$. This demonstrates in a powerful way the impossibility of solving question (4) by means of fixed points and quantifiers which merely count sizes of definable sets. A further result of **Hella** [14] showed that for every r there are LOGSPACE properties of finite models that are not expressible in $\mathrm{FP}(\mathbf{Q}_r)$. Hence:

Theorem 8 ([14]) *On unordered finite models, PTIME is not the extension of fixed point logic by finitely many generalized quantifiers.*

This shows that if we want to answer (4) affirmatively, we have to look beyond fixed point logic and finite collections of generalized quantifiers.

The difference between the Hierarchy Theorem (Theorem 2) and the results of this section is that the counting method of the Hierarchy Theorem simply gives the existence of a quantifier with certain properties, with no concern to whether the quantifier has any intuitive meaning. The more elaborate game-theoretic methods of this section make it possible to take a concrete meaningful quantifier, like the Ramsey-quantifier, and prove that it cannot be reduced to simpler quantifiers.

5 Quantifier schemata

We have pointed out that PTIME can be expressed in logical formalism as fixed point logic, provided that we restrict ourselves to ordered models. We have also pointed out that there is an extension of fixed point logic by a single generalized quantifier, which captures PTIME on almost all unordered finite models. Finally, we have concluded that in the framework of all unordered finite models there is no extension of fixed point logic by finitely many generalized quantifiers that would give all of PTIME.

Can we capture PTIME by adding an *infinite* number of generalized quantifiers to fixed point logic? There is a trivial answer. We can take one new quantifier for each PTIME query and add the resulting infinitely many quantifiers to FO. Surely we get PTIME, but we have not gained anything. It

makes more sense to look at infinite collections of quantifiers arising from some effective process, and then ask, can we get all of PTIME.

There are several ways in which a single generalized quantifier can give rise to an infinite sequence of quantifiers. Recall the Ramsey-quantifier Ram^r defined in (12). We can think of it as arising from the much simpler quantifier

$$Q_f x(\ldots x \ldots) \iff \exists X \left(|X| \geq f(n) \wedge \forall x \in X(\ldots x \ldots)\right),$$

where $f : \mathbf{N} \to \mathbf{N}$ and n is the size of the model. With this quantifier one can say (by choosing $f(n) = \lfloor n/2 \rfloor + 1$) things like:

"Most vertices have a green neighbor."

Let us denote Ram^r by $Ram^r(Q_f)$ if "X large" in (12) is defined as $|X| \geq f(n)$, where again n denotes the size of the model. $Ram^r(Q_f)$ is called a *Ramsey lift* of Q_f. So with the sentence

$$Ram^2(Q_f)xyE(x, y)$$

we can say

"Most vertices are neighbors of each other." (14)

For every choice of f, the quantifier Q_f of type (1) gives rise to the infinite sequence of quantifiers $Ram^r(Q_f)$ of higher and higher type. It turns out that for non-trivial f the quantifier $Ram^n(Q_f)$ cannot be defined in terms of quantifiers of smaller type even if fixed points are used. But we would not get all of PTIME even if we added all possible Ramsey lifts of quantifiers of type (1) [17]. Other lifts of the nature of the Ramsey lift have been considered, especially in the study of natural language semantics, where they are used to formalize expressions like

> Most boys in my class and most girls in your class
> have all dated each other.

However, the most interesting lift from the point of view of descriptive complexity theory is the *resumption* or *vectorization* lift. In a few words, the vectorization of a quantifier Q says about tuples what Q itself says about elements. Thus the second vectorization of Q_{even} says "for an even number of pairs (x, y) we have $\ldots x \ldots y \ldots$. In general, let Q be a quantifier of type

$\tau = (t_1, \ldots, t_k)$. Then its m'th vectorization $Res^m(Q)$ is a quantifier of type $(mt_1, \ldots, mt_k)$. So the arity of the quantifier increases by a factor of m. A quantifier Q of type τ is a class of models of vocabulary L, where L consists of a t_i-ary relation symbol $R_i(x_1, \ldots, x_{t_i})$ for each $i = 1, \ldots, k$. To define $Res^m(Q)$ we use the vector notation $\mathbf{z}$ to denote a sequence $(z_1, \ldots, z_m)$ of m variables. Let the language L' consist of an (mt_i)-ary relation symbol $R'_i(x_1, \ldots, x_{mt_i})$ for each $i = 1, \ldots, k$. Then $Res^m(Q)$ is the class of models $(A, R'_1, \ldots, R'_k)$ of vocabulary L' for which the L-structure $(A^m, R_1, \ldots, R_k)$ is in Q, where

$$\begin{aligned} R_i &= \{\big((a_1^1, \ldots, a_{t_1}^1), \ldots, (a_1^m, \ldots, a_{t_i}^m)\big) \in (A^m)^{t_i} : \\ &\quad R'_i(a_1^1, \ldots, a_{t_1}^1, \ldots, a_1^m, \ldots, a_{t_i}^m)\}. \end{aligned}$$

For example, suppose Q is the type (2,1) quantifier which consists of structures (A, E, U), where $U \subseteq A$ and (A, E) is a graph with a clique of the size of the set U. So in a model with domain A the expression

$$Qxyz(\ldots x \ldots y \ldots)(\_\_\_z\_\_\_)$$

says that $(A, \{(a, b) \in A^2 : \ldots a \ldots b \ldots\})$ is a graph with a clique with as many elements as there are $a \in A$ with $\_\_\_a\_\_\_$, while $Res^m(Q)$ is of type $(2m, m)$ and

$$Q\mathbf{xyz}(\ldots \mathbf{x} \ldots \mathbf{y} \ldots)(\_\_\_\mathbf{z}\_\_\_)$$

says in the same model that $(A^m, \{(\mathbf{a}, \mathbf{b}) \in (A^m)^2 : \ldots \mathbf{a} \ldots \mathbf{b} \ldots\})$ is a graph with a clique with as many elements as there are sequences $\mathbf{a} \in A^m$ with $\_\_\_\mathbf{a}\_\_\_$.

Note that if Q is PTIME, then so is every $Res^m(Q)$, so the vectorization lift is computationally simpler than the Ramsey lift. We denote by $\mathrm{FO}(Q^{<\omega})$ the extension of FO by all the vectorizations $Res^m(Q)$, $m = 1, 2, \ldots$, of Q.

As an important concrete example, consider the *Transitive Closure* quantifier of type $(2, 1, 1)$:

$$\begin{aligned} TC &= \{(A, E, X, Y) : (A, E) \text{ is a graph}, X \subseteq A, Y \subseteq A \\ &\quad \text{and from every } x \in X \text{ there is a path in the graph} \\ &\quad \text{to some } y \in Y.\} \end{aligned}$$

Theorem 9 ([20]) *NLOGSPACE*$= \mathrm{FO}(TC^{<\omega})$ *on ordered models.*

It is interesting to note that before this result it was not even known whether NLOGSPACE is closed under complements. It is an open problem, whether there is some natural logic L such that NLOGSPACE= FO(L) holds on unordered models.

The *Alternating Transitive Closure* quantifier ATC consists of models (A, E, X, Y), where (A, E) is a graph, $X \subseteq A$, $Y \subseteq A$, and every $x_0 \in X$ has a neighbor x_1 whose every neighbor x_2 has a neighbor x_3 whose every neighbor x_4 has a neighbor x_5 ...etc... until we come to an element of Y. Immerman [20] proved that PTIME= FO($ATC^{<\omega}$) on ordered finite models. This result has the following interesting version on *all* models, ordered or unordered:

Theorem 10 ([5]) FP = FO($ATC^{<\omega}$).

So although no finite sequence of generalized quantifiers can capture all of FP on unordered models (Theorem 6), the infinite sequence $ATC^{<\omega}$ is capable of the job.

Generalized quantifiers are thus, after all, able to express recursive definitions, as soon as sufficient arities, i.e. sufficiently long tuples are available. Still, it remains an open problem, whether PTIME = FO($Q^{<\omega}$) for some Q on unordered models. But Dawar has proved the following interesting result, which shows that the approach of (vectorizations) of generalized quantifiers is at least as good as any other:

Theorem 11 ([6]) *If question (4) has a positive answer (in an exact sense), then there is a generalized quantifier Q so that PTIME = FO($Q^{<\omega}$) on all finite models.*

The methods available at the moment in the study of generalized quantifiers can be effectively used to study definability questions concerning individual quantifiers and families of quantifiers with a bound on arities. Unfortunately the same methods become extremely hard when applied to quantifier schemata.

References

[1] A.V. Aho and J.D. Ullman, Universality of data retrieval languages, Sixth ACM Symposium on Principles of Programming Languages, 1979, 110–117.

[2] J. Cai, M. Fürer and N. Immerman, An Optimal Lower Bound on the Number of Variables for Graph Identification, Combinatorica 12:4 (1992), 389-410.

[3] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

[4] E.F. Codd, Relational completeness of database sublanguages, in: Database Systems (R.Rustin, ed.), Prentice-Hall, 1972, 65–98.

[5] E. Dahlhaus, Skolem normal forms concerning the least fixpoint, in: Computation theory and logic (E. Börger, ed.), 101–106, Lecture Notes in Comput. Sci., 270, Springer, Berlin-New York, 1987.

[6] A. Dawar, Generalized quantifiers and logical reducibilities, Journal of Logic and Computation, 5(1995), 213–226.

[7] A. Dawar and E. Grädel, Generalized quantifiers and 0-1 laws, Proc. 10th IEEE Symp. on Logic in Computer Science, 1995, 54–64.

[8] R. Fagin, The number of finite relational structures, Discrete Mathematics 19(1977), 17–21.

[9] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, in: Complexity of Computation (R. Karp, ed.) SIAM-AMS Proc. 7, 1974, 27–41.

[10] M. Grohe and L. Hella, A double arity hierarchy theorem for transitive closure logic, Archive for Mathematical Logic, 35(3): 157-172, 1996.

[11] Y. Gurevich. Toward logic tailored for computational complexity. In M. M. Richter et al., editor, *Computation and Proof Theory, Lecture Notes in Mathematics 1104*, pages 175–216. Springer-Verlag, 1984.

[12] Y. Gurevich. Logic and the challenge of computer science. In E. Börger, editor, *Current trends in theoretical computer science*, pages 1–57. Computer Science Press, 1988.

[13] Y. Gurevich, Zero-one laws, This column in the EATCS Bulletin 46 (1992), 90-106.

[14] L. Hella, Logical hierarchies in PTIME, *Information and Computation* 129: 1-19, 1996.

[15] L. Hella and G. Sandu, Partially ordered connectives and finite graphs, in: Quantifiers: Logics, models and computation (M. Krynicki, M. Mostowski and L. Szczerba, eds.), vol II, Kluwer Academic Publishers 1995, 79–88.

[16] L. Hella, Ph. Kolaitis and K. Luosto, Almost everywhere equivalence of logics in finite model theory, Bulletin of Symbolic Logic 2(4), 1996, 422-443.

[17] L. Hella, K. Luosto and J. Väänänen, The hierarchy theorem for generalized quantifiers. J. Symbolic Logic 61 (1996), no. 3, 802–817.

[18] L. Hella, J. Väänänen and D. Westerståhl, Definability of polyadic lifts of generalized quantifiers, to appear in the *Journal of Logic, Language and Information*.

[19] N. Immerman, Relational queries computable in polynomial time, Information and Control 68 (1986), 86–104.

[20] N. Immerman, Languages that capture complexity classes, SIAM J. Comput. 16, No. 4 (1987), 760-778.

[21] P. Lindström, First order predicate logic with generalized quantifiers, *Theoria* 32 (1966), 186–195.

[22] A. Mostowski, On a generalization of quantifiers. *Fundamenta Mathematicae* 44 (1957) 12–36.

[23] J. Nurmonen, On winning strategies with unary quantifiers, Journal of Logic and Computation, 6(6): 779-798, 1996.

[24] M. Vardi, Complexity of relational query languages, 14th ACM STOC Symposium (1982), 137–146.