Rheinisch-Westfälische
Technische Hochschule Aachen

Lehr- und Forschungsgebiet Mathematische Grundlagen
der Informatik, Prof. Dr. Grädel

# Diplomarbeit
# Reliability of Database Queries

cand.inform.

**Colin Hirsch**

Geboren 8.1.1975
Matr.Nr. 197505

24.2.1998

Diplomarbeit am Lehr- und Forschungsgebiet Mathematische Grundlagen der Informatik der Rheinisch-Westfälischen Technischen Hochschule Aachen, Mathematisch-Naturwissenschaftliche Fakultät.

Ausgegeben und betreut von Prof. Dr. E. Grädel, Zweitgutachter Prof. Dr. F. Baader.

Der Titel der Arbeit lautet "The Reliability of Database Queries".

Colin Hirsch
Weststr. 27
D-52074 Aachen
Telefon: 0241 / 8876 175
E-Mail: `hirsch@informatik.rwth-aachen.de`

Textsatz: $\LaTeX\,2_\varepsilon$ & XEmacs unter Linux

**Erklärung**
Hiermit versichere ich, daß ich die Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Aachen, 24.2.1998

(Colin Hirsch)

# Contents

# Chapter 1

# Background

## 1.1 Introduction

Over the years the problem of query complexity for relational databases has been widely studied. Unlike earlier databases, the relational model can be embedded into a well established formal framework: that of mathematical logic. This allows us to transfer results about logics to (relational) database theory. Regarding queries, their expressibility and complexity, there is a particularly close relationship with finite model theory and descriptive complexity theory. The link the latter creates between classes of logical formulae — which formalize queries — and complexity classes will be of pivotal importance for all things to come here.

However this certain view of relational databases has its limitations. To meet the challenge of an evergrowing range of database applications, some of which would have been considered "non-standard" until recently, it is crucial to develop new theories as background. For one, the classical theories either only allow finite databases, or, when using infinite domains, go to considerable trouble to ensure that the amount of stored data is always finite. A recent development overcoming these limitations replaces finite relations by finitely representable relations (constraint databases, e.g. [9, 18, 7]). The models considered there are especially suited for spatial database applications.

A further problem, sometimes simply ignored in classical relational database theory (cf. [22]), arises from the use of numerical operations in database queries. Only recently have there been efforts to provide a sound logical framework for including infinite domains required even for simple operations like addition. We will introduce the notion of metafinite structures as introduced by Grädel, Gurevich [6], which will serve as basis for two extended database models.

Another aspect is the reliability of information. Even when keeping in mind that every database is only an abstraction or an approximation of some real-world phenomena, the data it contains is generally assumed to be absolutely correct. Further there is no method short of deleting an entry to express that said entry is fully or partially incorrect. The emphasis of this work is to deal with the (un)reliability of database information, building on the initial results developed by de Rougemont [16].

Unreliable data can enter a database through many doors like unreliable sources of data and aging of information. Our approach is of a probabilistic nature, augmenting databases with probabilities about the correctness of stored atomic information. This kind of *probabilistic database* leads to an extended terminology and some new questions. We now have to deal with the *given* database in the context of a set of *possible* databases. A database is possible, if the differences compared with the given database are listed as having positive probability. Most often the given database will be the one with the highest probability of being the true database amongst all possible databases. Viewing the set of possible databases as probability space will enable us to use common stochastical methods along the way. An exact definition of a probabilistic database has to be established. Furthermore it is desirable to find an exact formalization of the quality of a query result. As the given database may differ from the *actual* database, which is available only as a member of a probability space, a user will be interested in how far a query result resembles the actual result. The key point of interest in this work is to determine the complexity of these query reliability questions with respect to the query itself.

Most previous research conducted in the area of probabilistic databases goes into various other directions. Many works are centered around probabilistic deductive databases (e.g. [19, 11]). There classical logic-programming approaches are extended to include unsure information in the shape of rules with probabilities. Recently there has also been work studying more practical aspects of querying probabilistic relational databases (e.g. [26, 12]).

For the remainder of this chapter we recall some definitions from relational databases, complexity theory and stochastics together with some general results required later on. In Chapter 2 we introduce de Rougement's idea of probabilistic databases and a natural extension thereof. Results on the complexity of the query reliability question are given. In Chapter 3 we develop more efficient algorithms for computing query reliabilities using polynomial-time approximation schemes. In Chapter 4 we finally look at the possibilities arising from the metafinite setting. We discuss an embedding of the previous probabilistic databases and a new kind of databases.

## 1.2 Formalism of Relational Databases

As already hinted, the database model of choice for this work is the relational model. Now a relational database can be viewed as a finite relational structure, that is, a finite collection of relations over a finite domain. More formally, let $\sigma = \{R_1, \dots, R_t\}$ be a relational vocabulary consisting of relation symbols $R_i$ with arity $k_i$. Let $A$ be a finite set. Then the finite structure $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_t^{\mathfrak{A}})$, where $R_i^{\mathfrak{A}} \subseteq A^{k_i}$ is the interpretation of $R_i$ in $\mathfrak{A}$, is a relational database.

We use formulae and sentences as queries for our databases. Let $L$ be a logic (e.g. FO). If $\varphi$ is a formula with $k$ free variables in $L(\sigma)$ and $\mathfrak{A}$ is a database with vocabulary $\sigma$, the result of the evaluation of $\varphi$ on $\mathfrak{A}$ is defined as follows:

- If $k = 0$ the result is *true*, iff $\mathfrak{A} \models \varphi$.
- If $k > 0$ the result $\varphi^{\mathfrak{A}}$ is the set of all $k$-tuples in[1] $A$ satisfying $\varphi$;
  $\varphi^{\mathfrak{A}} = \{\bar{a} \mid \mathfrak{A} \models \varphi[\bar{a}]\}$.

Typical query languages considered are QF, FO and SO — the classes of quantifier-free, first-order and second-order formulae, respectively. We will also look at the class of conjunctive queries, a subset of FO. An existential formula $\varphi$ is *conjunctive*, iff it can be equivalently transformed into the form

$$\varphi = \exists x_1 \dots \exists x_k \, (\varphi_1 \wedge \dots \wedge \varphi_n) \ ,$$

where the $\varphi_i$ are atomic formulae.

The general situation henceforth will be that a given database is to be queried and, in complexity theoretic terms, we are interested in how difficult this is. All complexity considerations are with respect to *data complexity*, i.e. a query is fixed and the complexity is given as function in the size of the database. Actually we would rather have to take the size of an appropriate input encoding of the database on a Turing band as input size. However we do not consider any complexity class below P and the size of the input encoding is polynomial in the size of the database (i.e. the underlying domain).

Other possibilities for complexity measures are to study *program complexity*, where the database is fixed and the query or program is modified, and *combined complexity* viewing both the database and the query as input. There, both program and database size are used as parameters for computational complexity. The motivation for ignoring the size of the query is that it will usually be very much smaller than the database.

[1]If $A$ is a set and $\bar{a}$ a $k$-ary tuple, we say "$\bar{a}$ is in $A$" or $\bar{a} \in A$, when we mean $\bar{a} \in A^k$.

## 1.3 Notes on Complexity Theory

The main concern of this work is to deal with complexity theoretic aspects of querying databases and calculating reliabilities of query results. We will mostly use the Turing machine (TM) computation model for complexity considerations. Apart from distinguishing nondeterministic Turing machines (or NTMs) from deterministic Turing machines (or DTMs), we classify TMs according to whether they decide a language or calculate a (partial) function. If $M$ is a TM, we denote by $L(M)$ the language decided by $M$ resp. by $f_M$ the function calculated by $M$.

Most of the complexity classes encountered later on will lie somewhere between P and PSpace. Besides other well known classes such as NP and the polynomial-time hierarchy PH we will take a closer look at UP and especially #P. The class UP, which stands for unambiguous polynomial time, consists of all languages decided by nondetermistic polynomial-time TMs which have either zero or one accepting computation path. It follows from the definition that P $\subseteq$ UP $\subseteq$ NP and the inclusions are generally expected to be strict. The class #P uses polynomial-time NTMs too, but is a class of functions with ranges in the natural numbers. The output of a computation is exactly the number of accepting computation paths.

**Definition 1.1.**
Let $\Sigma$ be an alphabet. A function $f : \Sigma^* \to \mathbb{N}$ belongs to #P, iff there exists a nondeterministic polynomial-time TM $M$ such that for all inputs $x \in \Sigma^*$ the number of accepting computation paths of $M$ on $x$ is exactly $f(x)$.

A language $L \subseteq \Sigma^*$ belongs to UP, iff there exists #P-algorithm $M$ which calculates the characteristic function of $L$. I.e., for all $x \in \Sigma^*$

$$M(x) = \begin{cases} 1 & x \in L \\ 0 & \text{otherwise .} \end{cases} \qquad \diamond$$

The class #P will be of central interest later on in the shape of $\mathrm{FP}^{\#\mathrm{P}}$. One reason for allowing this polynomial-time pre and postprocessing is that we have to handle rationals when performing calculations with probabilities. As #P-algorithms can only return naturals we have to use an appropriate encoding.

We use the corresponding class of languages $\mathrm{P}^{\#\mathrm{P}}$ for comparisons between the strength of $\mathrm{FP}^{\#\mathrm{P}}$and other (better known) complexity classes. The notion of reduction used for #P-hard problems is polynomial-time Turing reducibility, thus a language is #P hard if and only if it is $\mathrm{P}^{\#\mathrm{P}}$ hard. The following “map” shows the situation between the classes mentioned above.

$$\mathrm{P} \subseteq \mathrm{UP} \subseteq \mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{PS{\scriptsize PACE}}$$

It is currently not known which of the inclusions are strict. We will however take a look at stronger results regarding the PH $\subseteq$ $\mathrm{P}^{\#\mathrm{P}}$ inclusion. In [20, 21] Toda showed that PH $\subseteq$ $\mathrm{P}^{\#P[1]}$. A problem is in $\mathrm{P}^{\mathrm{C}[n]}$ if it is $n$-polynomial-time Turing reducible to a problem in C, i.e. each computation is allowed to make at most $n$ oracle queries. Clearly polynomial-time Turing reducibility from problem $A$ to problem $B$ is implied by $n$-polynomial-time Turing reducibility which in turn is implied by polynomial-time many-one reducibility. In [15] Regan and Schwentick give a special type of #P-oracle to be used in the PH $\subseteq$ $\mathrm{P}^{\#P[1]}$ inclusion.

**Theorem 1.2 (Regan, Schwentick).** *Let $D$ be a problem in* PH *and $q$ a polynomial. Then there exists a* #P*-function $f$ and a polynomial $t$ such that for every input $x$, $n = |x|$, the binary representation of $f(x)$ has the form*

$$\{0,1\}^* \cdot 0^{q(n)} \cdot D(x) \cdot 0^{q(n)} \cdot \{0,1\}^{t(n)}$$

*given as regular expression, where $D(x) = 1$ iff $x \in D$.*

There exist a surprising variety of #P-complete problems. Primary candidates are the counting versions of NP-complete problems. Take your favourite NP-complete problem and ask for the number of solutions instead of simply checking for the existence. Additionally there exist problems in P where the counting version is #P-hard. For hardness results later on we will use a #P-hard problem, where the decision problem is obviously trivial. In [23] Valiant showed the problem #Monotone 2-Sat to be #P-hard. Given a propositional formula $\alpha$ in 2-KNF *without negation*, the problem is to compute the number of truth assignments to the variables occuring in $\alpha$ which let $\alpha$ evaluate to *true*. Of course every monotone propositional formula is satisfiable by assigning *true* to all variables.

As we are interested in the data complexity of evaluating logics on databases, we will make use of some well-known results from descriptive complexity theory. For one, it can be easily shown that all first-order formulae can be evaluated in polynomial time. Building on Fagin's way-leading result about NPcorresponding to *existential* second-order logic it has been shown that the class of all second-order queries similarly captures the polynomial hierarchy.

Using these correspondencies we will freely change between these two paradigms. For example when talking about polynomial-time queries, we mean any query belonging to a class of queries which is polynomial-time computable. This case includes besides FO also classes like DataLog and FO+LFP or FO+IFP. Note that on ordered structures the latter two completely capture P.

## 1.4 A Word on Probability

Although the subject of our interest builds on adding probabilities to databases we require only very basic stochastical methods for our means. The probability spaces considered will be finite and probability distributions can be assumed to be given explicitly as probabilities of atomic events. We will use random variables, their expectation and the linearity of expectation in some elementary calculations.

**Lemma 1.3.** *Let $X_i$, $i = 1, \dots, n$ be random variables with values in $[0,1]$ and let $\varepsilon, \delta > 0$. If for all $i, 1 \le i \le n$*

$$Pr\big[X_i > \varepsilon/n\big] < \delta/n,$$

*then*

$$Pr[\sum_{i=1}^{n} X_i > \varepsilon] < \delta.$$

**Proof:** Simply note that

$$\Pr[\sum_{i=1}^{n} X_i > \varepsilon] \ \le\ \Pr[\bigcup_{i=1}^{n}\{X_i > \varepsilon/n\}] \ \le\ \sum_{i=1}^{n} \underbrace{\Pr[X_i > \varepsilon/n]}_{<\,\delta/n} < \delta$$

The following lemma from [10] is about as complicated as it gets.

**Lemma 1.4 (Karp, Luby).** *Let $\{X_i\}$ be a sequence of independent identically distributed random variables with values in $[0,1]$ and expectation $p := \mathrm{E}(X_i) < 0.5$. Then for all $\varepsilon \in (0,1)$,*

$$\Pr\left[\left|\frac{X_1 + \cdots + X_t}{t} - p\right| > \varepsilon \cdot p\right] < 2 \cdot e^{\left(\frac{-2\varepsilon^2 tp}{9(1-p)}\right)}.$$

**Remark 1.5.** One global precondition throuthout this work is that, unless stated otherwise, we assume all probabilities to be rationals. This is necessary in order to perform calculations using the classical Turing machine model. Other machine models like the BSS machine (Blum, Shub, Smale) or abstract state machines which are capable of directly manipulating reals numbers are not considered.

# Chapter 2

# The Finite Case

## 2.1 First Approach: De Rougement's Databases

### 2.1.1 Model

We will now introduce a first version of finite probabilistic databases, as discussed by de Rougemont [16]. We will see that it is defined as a somewhat restricted model, however proving sufficient as basis for all major definitions and some important complexity results. The database vocabulary is restricted to the signature of graphs — one binary relation. We further allow the use of (a finite set of) constants. Errors are only allowed in one direction: For each pair $(x, y) \in E$ there is a probability $\mu((x, y))$ of $(x, y)$ *not* belonging to $E$. On the other hand, a pair of nodes not contained in $E$ will not be in $E$ in all possible databases.

**Definition 2.1.** A *(finite) probabilistic database* $\mathfrak{D}$ is a pair $(\mathfrak{A}, \mu)$, where

- $\mathfrak{A}$ is a finite $\sigma$ structure with domain $A$ and vocabulary $\sigma = \{E, \bar{c}\}$; $E$ is a binary relation and $\bar{c}$ is a sequence of constants,
- $\mu : E \to [0, 1]$ is the error function, where *all probabilities are assumed to be independent.* $\diamond$

In the context of a probabilistic database $\mathfrak{D} = (\mathfrak{A}, \mu)$ we call $\mathfrak{A}$ the *given* database. The given database is used for all query evaluations. Together with the error function $\mu$ the given database induces a set of *possible* databases. A database $\mathfrak{B}$ is possible if it can be obtained by taking $\mathfrak{A}$ and leaving out some of the edges. We call the set of all possible databases $\Omega_{\mathfrak{D}}$. Our intuition is that the given database may not be the *actual database*, which is the database containing the true edge relation. The error function gives us probabilities for some changes to the given database. All possible databases are derivable from the given database by applying such changes, in our case letting out edges. So although not exactly knowing the actual database, we do know that it is one of the possible databases. We can now use $\mu$ to calculate the induced probability $\nu(\mathfrak{B})$ of each $\mathfrak{B} \in \Omega_{\mathfrak{D}}$ being the actual database. It follows, that whenever we want to use the actual database, we will have to use a random variable ranging over $\Omega_{\mathfrak{D}}$ with (probability) distribution $\nu$. The probability distribution $\nu$ over $\Omega_{\mathfrak{D}}$ gives the probability of randomly choosing each $\mathfrak{B} \in \Omega_{\mathfrak{D}}$ as actual database. In the following we will often deal with the probability space[1] $(\Omega_{\mathfrak{D}}, \nu)$ induced by a probabilistic database $\mathfrak{D}$.

The distribution $\nu$ can be directly calculated from $\mu$. We start with single probabilities for single edges and then extend to whole databases. For atomic statements $\beta = Exy$, the probability $\nu(\beta)$ of holding in a randomly chosen $\mathfrak{B} \in \Omega_{\mathfrak{D}}$ is 0, if $\mathfrak{A} \not\models \beta$ — we can not add edges. Else it is $1 - \mu((x, y))$. Further, if $\beta$ is a negative literal, then $\neg\beta$ is atomic and $\nu(\beta) = 1 - \nu(\neg\beta)$. As the error probabilities of events $\beta, \beta'$, where $\beta, \beta'$ are literals built with distinct atoms, are independent, the probability $\nu(\mathfrak{B})$ of $\mathfrak{B}$ being the acual database is easily calculable as

$$\nu(\mathfrak{B}) = \prod_{\beta \in \mathrm{L}it(\mathfrak{B})} \nu(\beta),$$

where $\mathrm{Lit}(\mathfrak{B})$ is the set of all literals *true* in $\mathfrak{B}$.

**Remark 2.2.** The probability of any $\mathfrak{B} \in \Omega_{\mathfrak{D}}$ depends on the probabilities of the differences between $\mathfrak{B}$ and $\mathfrak{A}$. Throughout this work we will use $\mu$ to denote an error function giving the probability of a random event producing a value differing from some given value. Conversely $\nu$ will always be the corresponding probability function directly giving the probability distribution on the possible events. As seen above, to calculate the $\nu$-probability of an edge we need the original value of the edge and the probability of that value changing.

We now have the means to define the expected error of a query. In the boolean case, the expected error is defined naturally. In the general case, we take the expected Hamming distance between the result on the given database and the actual (i.e. real, but only probabilistically known) database. That is the expected number of tuples where the two relations differ.

**Definition 2.3.** Fix a database $\mathfrak{D} = (\mathfrak{A}, \mu)$ and let $\mathfrak{B}$ be a random variable ranging over $\Omega_{\mathfrak{D}}$ with probability distribution $\nu$ (as defined by $\mu$).

[1]Due to our probability spaces being finite we omit the required $\sigma$-algebra. The reader may take the discrete $\sigma$-algebra in all cases.

- Let $\psi$ be a boolean query. The *expected error* of $\psi$ with respect to $\mathfrak{D}$, $H_\psi(\mathfrak{D})$, is the expectation $\mathrm{E}(\psi^{\mathfrak{A}} \oplus \psi^{\mathfrak{B}})$.
- For $k$-ary queries $\psi$, the *expected error* $H_\psi(\mathfrak{D})$ is defined as $\mathrm{E}(|\psi^{\mathfrak{A}} \ \Delta \ \psi^{\mathfrak{B}}|)$.

Here $\oplus$ denotes the bitwise exclusive-or and $\Delta$ the corresponding operation on sets, the symmetric difference: $A\Delta B = (A\backslash B) \cup (B\backslash A)$. $\diamond$

The expected error is now used to calculate the reliability of a query.

**Definition 2.4.** The *reliability* or *fault tolerance* of a $k$-ary query $\psi$ with respect to $\mathfrak{D} = (\mathfrak{A}, \mu)$ with $|A| = n$ is defined as

$$R_\psi(\mathfrak{D}) = 1 - \frac{H_\psi(\mathfrak{D})}{n^k}. \qquad \diamond$$

Note that the reliability is always between 0 and 1. The bordercases, reliability 0 and 1 allow an intuitive interpretation: Reliability 1 expresses that a query always yields the same result, regardless to which of the possible databases is chosen. A reliability of 0 can only be obtained if the result is always exactly the opposite as on the given database. Thus the given database must have probability 0. Although the practical intention is to take one of the most probable possible databases as given, this is not mandatory and databases with zero probability can be taken as the given database.

### 2.1.2 Results

De Rougements discussion of complexity issues of the reliability problem begins with quantifier free formulae. In [16] it is proven that for quantifier-free queries the reliability is polynomial-time computable. The claim that this is still true for first-order queries must however be reverted as being highly unlikely. We do not even need the full power of first-order logic to get #P-hardness of query reliability.

**Theorem 2.5.** *There exists a conjunctive query $\varphi$ such that calculating the expected error $H_\varphi$ is* #P*-hard.*

**Proof:** We will give a reduction from the problem #Monotone 2-Sat to the problem of calculating the expected error for an explicitly given $\varphi$. As noted earlier, #Monotone 2-Sat is #P-complete. The idea is to use the uniform distribution over all truth assignments to the propositional variables to simulate counting.

Let $\alpha = \bigwedge_{i=1}^{n} Y_i \vee Z_i$ be a propositional formula in 2-KNF. We will now encode the structure of $\alpha$ into a probabilistic database $\mathfrak{D} = (\mathfrak{A}, \mu)$, where
$\mathfrak{A} = (A, E, a, f, l, r)$.

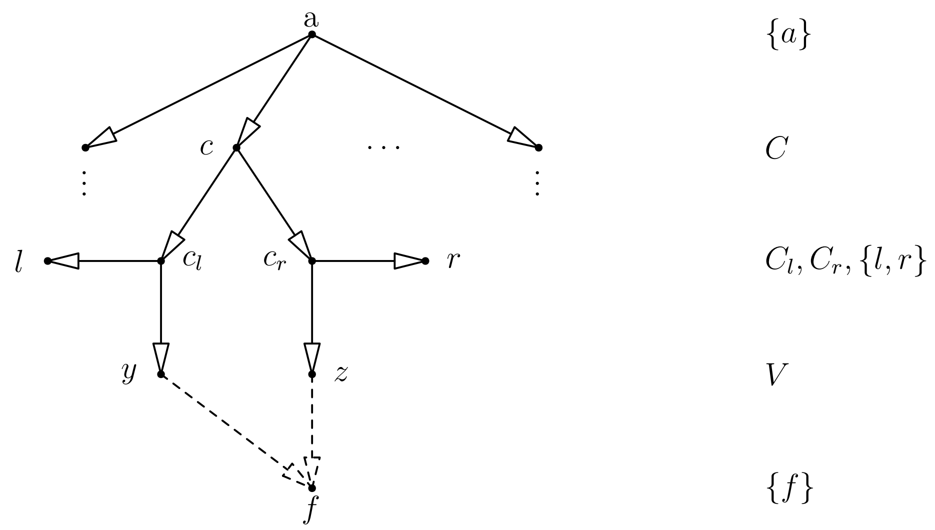
- The universe $A$ is the (disjoint) union of
  - the set $V$ of all variables occuring in $\alpha$,
  - three copies of the set $C$ of all clauses occuring in $\alpha$: one is the set of clauses $C$ itself, one is the "left copy" $C_l = \{c_l \mid c \in C\}$ and one is the "right copy" $C_r = \{c_r \mid c \in C\}$,
  - four *new* elements $a, f, l, r$.
- The edge relation $E$ is the (disjoint) union of
  - $\{(a, c) \mid c \in C\}$, an edge from $a$ to each clause,
  - $\{(c, c_X) \mid c \in C, X \in \{l, r\}\}$, an edge from each clause to the corresponding left and right copies,
  - $\{(c_X, v) \mid c = (Y \vee Z) \in C, [X = l \text{ and } v = Y] \text{ or } [X = r \text{ and } v = Z]\}$, an edge joining the left resp. right copy of $c$ with the left resp. right variable of $c$,
  - $\{(v, f) \mid v \in V\}$, an edge from each variable to $f$,
  - $\{(c_X, X) \mid c \in C, X \in \{l, r\}\}$, edges to distinguish between $c_l$ and $c_r$.
- The constants $a, f, l, r$ are of course interpreted through elements $a, f, l, r$.
- The error function $\mu$ is defined as

$$\mu((x, y)) = \begin{cases} 1/2 & x \in V \text{ is a variable and } y = f \\ 0 & \text{otherwise.} \end{cases}$$

The intuitive meaning is that $f$ stands for *false*. An edge connecting a variable with $f$ is to be interpreted as said variable being set to *false*. The set of all possible truth assignments $\mathfrak{I} = \{I \mid I : \text{Var}(\alpha) \to \{0, 1\}\}$ to the variables of $\alpha$ is exactly represented by the set of all possible databases $\Omega = \{\mathfrak{B}_I \mid I \in \mathfrak{I}\}$. The distribution $\mu$ produces, as induced distribution $\nu$ on $\Omega_{\mathfrak{D}}$, the uniform distribution. The edge relation in $\mathfrak{B}_I$ is defined such that an edge from a variable $v$ to $f$ exists iff $I(v) = \textit{false}$. Now consider the query

$$\begin{aligned} \varphi \ := \ & \exists c \exists y \exists z \exists c_l \exists c_r \\ & (Eac \wedge Ecc_l \wedge Ecc_r \wedge Ec_l y \wedge Ec_r z \wedge Eyf \wedge Ezf \wedge Ec_l l \wedge Ec_r r). \end{aligned}$$

Figure 2.1: Sketch of $\mathfrak{A}$



We ask for the existence of a $c$, which is forced to be a clause via the condition $Eac$. Next we ask for $y$ and $z$ to be the variables of $c$, due to $c$ connecting to $y$ and $z$ through $c_l$ and $c_r$. The conditions $Eyf$ and $Ezf$ ensure that both variables are connected to $f$, i.e. set to *false*. The last two conditions ensure that we do not choose one variable of $c$ twice[2]. So $\varphi$ is equivalent to the existence of a clause in $\alpha$ where both variables are set to *false*. Consequently for each $\mathfrak{B}_I \in \mathfrak{I}$, $\varphi$ holds in $\mathfrak{B}_I$ iff the truth assignment $I$ *does not* satisfy $\alpha$.

Now $\varphi$ evaluates to *true* on the given database $\mathfrak{A}$, as all variables are set to *false*. Under the uniform distribution the expected error of $\varphi$ is exactly the fraction of possible databases where $\varphi$ evaluates to *false*, or equivalently the fraction of truth assignments $I \in \mathfrak{I}$ satisfying $\alpha$.

De Rougement's polynomial-time "proof" for the reliability of first-order queries contains errors in the calculation of probabilities. At one point the probability of the union of some events is calculated via the sum of probabilities. This is of course only possible if the events are pairwise disjoint — which is not verified and indeed not believed to be verifiable in this case. To illustrate this, imagine asking for the existence of a 4-clique in a random graph of some size. Adding up the probabilities for any 4 points to be a 4-clique is obviously not the way to calculate the probability of the graph containing a 4-clique. Further, as many of these 4 point sets will overlap, the probabilities are not independent either. The normal inclusion-exclusion principle used to compute the probability of an arbitrary union of events requires time exponential in the number of sets being combined. Speaking informally, we will see that for quantifier-free formulae we

[2]Note that the case where only one variable occurs in $c$ is handled correctly — the distinction of $c_l$ and $c_r$ has no implications about the equality of $y$ and $z$.

can sit on top of the query and look outwards at the different possible databases. Therefore it is permissible to only look at the parts of the database actually occuring in the query. However this view does not generalize to first-order queries. There it is necessary to take each of the exponentially many possible databases and then look at the formula.

## 2.2 Second Approach: Relational Databases

### 2.2.1 Model

We will now consider a natural extension of the databases considered by de Rougement. First we drop the restriction to graphs and allow any relational structure (with constants) as database. We further allow the error function to give positive probabilities to *all* atomic statements built with database relations and elements of the underlying universe.

**Definition 2.6.** A *(finite) probabilistic database* $\mathfrak{D}$ is a pair $(\mathfrak{A}, \bar{\mu})$, where

- $\mathfrak{A}$ is a finite $\sigma$ structure with domain $A$ and vocabulary $\sigma = \{\bar{R}, \bar{c}\}$; $\bar{R}$ is a finite sequence of relations and $\bar{c}$ is a finite sequence of constants,
- $\bar{\mu}$ is a sequence of error functions containing for each $R_i$ in $\bar{R}$ a corresponding $\mu_{R_i} : A^{k_i} \mapsto [0,1]$, where $k_i$ is the arity of $R_i$. Again all probabilities are assumed to be independent. $\diamond$

We also write $\mu(R\bar{a})$ when we mean $\mu_R(\bar{a})$. Now given a probabilistic database $\mathfrak{D} = (\mathfrak{A}, \mu)$ the set of possible databases $\Omega_{\mathfrak{D}}$ contains all databases sharing the same universe, vocabulary and interpretation of constants. The two-sided errors, probabilities for removing *and adding* tuples to relations, requires an adaption in calculating the corresponding $\nu$ probabilities. Let $R$ be a $k$-ary database relation from $\mathfrak{D}$ and $\bar{a}$ a $k$-tuple in $A$. The probability $\nu(\beta)$ of the atom $\beta = R(\bar{a})$ holding in a randomly chosen $\mathfrak{B} \in \Omega_{\mathfrak{D}}$ is $1 - \mu(\beta)$, if $\mathfrak{A} \models \beta$ and $\mu(\beta)$, otherwise. This construction illustrates the difference between $\nu$ and $\mu$ as mentioned in Remark 2.2. To calculate the absolute $\nu$-probability of an atomic statement holding in a randomly chosen database we need the given value of that atom and the relative probability of that value changing as given by $\mu$. As, again, all probabilities are assumed independent, we can use the same method as in the previous section to extend $\nu$ from atoms to give probabilities for databases.

### 2.2.2 Results

Definitions 2.3 and 2.4 (expected error and reliability of a query) can be applied to our new setting without changes. Clearly the database model used in the previous section is a special case covered by Definition 2.6. It follows that the hardness result of the previous section still applies. Starting again with quantifier-free queries we now take a look at upper-bounds for the reliability problem.

**Theorem 2.7.** *Let $\psi$ be a quantifier-free query. Then the problem of calculating the reliability $R_\psi$ is in* P*.*

We defer the proof to the (even further) extended setting of Theorem 4.4 in Chapter 4, as the differences in the proofs are negligible (and the model considered there is again a true extension of the current version).

We already know that the reliability problem for conjunctive queries is $\mathrm{FP}^{\#\mathrm{P}}$-hard. The following is one of the main results of this work. We show that the scope of query languages for which the reliability problem is $\mathrm{FP}^{\#\mathrm{P}}$-complete ranges from conjunctive queries up to second-order queries. This includes, among many others, important classes such as DataLog, first-order and least-fixpoint queries. The same method used in the proof here will be used again for a similar result in Chapter 4, if reduced to a smaller class of queries.

**Theorem 2.8.** *Let $\psi$ be a second-order query. Then the problem of calculating the reliability $R_\psi$ is in* $\mathrm{FP}^{\#\mathrm{P}}$*.*

**Proof:** Let $\psi(\bar{x})$ be a $k$-ary second order query. Since $\mathrm{FP}^{\#\mathrm{P}}$ is closed under all polynomial-time computable operations and

$$R_\psi(\mathfrak{D}) = 1 - \frac{1}{n^k} \sum_{\bar{a}} H_{\psi(\bar{a})}(\mathfrak{D})$$

we are done if we can prove that the expected error $H_\psi(\mathfrak{D})$ is $\mathrm{FP}^{\#\mathrm{P}}$-computable for Boolean queries. That is, for a given database $\mathfrak{D} = (\mathfrak{A}, \mu)$ we have to calculate the expectation of the event $\psi^{\mathfrak{A}} \neq \psi^{\mathfrak{B}}$ for randomly chosen $\mathfrak{B} \in \Omega_{\mathfrak{D}}$ with a $\mathrm{FP}^{\#\mathrm{P}}$-algorithm. In the following let $\psi$ be a boolean query. The time bounds are with respect to the size $n$ of the representation of $\mathfrak{D}$.

First of all we evaluate $\psi$ on $\mathfrak{A}$ and store the result. Since $\mathrm{SO} = \mathrm{PH} \subseteq \mathrm{FP}^{\#\mathrm{P}}$ this is in $\mathrm{FP}^{\#\mathrm{P}}$. In the following computation of the expected error we will compare this result against the results of evaluating $\psi$ on all other possible databases. The idea is to use the nondeterminism inherent to the #P-part of the computation to create all possible databases in parallel. This is achieved by nondeterministically guessing a truth value for all atomic statements $R\bar{a}$, where $R$ is a database relation and $\bar{a}$ a tuple of the same arity as $R$.

We further want the nondeterministic computation tree to reflect the probability of each of the guessed databases. As our probabilities are rationals, we first have to transform them into integers for our counting environment. There exists a smallest positive integer $g$, such that $\nu(\mathfrak{B}) \cdot g \in \mathbb{N}$ for all $\mathfrak{B} \in \Omega_{\mathfrak{D}}$. We can compute $g$ in polynomial time. Start with the denominator of the first probability $\nu(R\bar{a})$ given on the input Turing band — for simplicity we can assume that the input has been duplicated replacing all $\mu$ probabilities by the induced $\nu$ probabilities. Also all rationals are assumed to be stored in a normalized representation. Then successively calculate the gcd $b$ between the so-far result $g'$ and the next denominator $d$. If $b = d$ then $d$ is a factor of $g'$ and we continue the loop. Otherwise take $g' \cdot d/b$ as $g'$ for the next round. In the end let $g = g'$.

This indeed yields the smallest possible $g$. According to the choice of $g$ there can be no $g' \leq g$ which always produces an integer when multiplied with the denominators of the probabilities. The existence of a $g' \leq g$ which always produces integers when multiplied with the probabilities themselves would imply the existence of a factor $h > 1$ occuring in *all* numerators. Since $g' \mid g$ this contradicts the fact that all probabilities were assumed to be given in a normalized representation.

After guessing a database $\mathfrak{B}$ we calculate $m_{\mathfrak{B}} := \nu(\mathfrak{B}) \cdot g$ and nondeterministically split the computation $m_{\mathfrak{B}}$ times. In the resulting computation tree, for each possible database $\mathfrak{B}$ there are exactly $m_{\mathfrak{B}}$ leafs which "see" $\mathfrak{B}$ as database. We will now first consider the case where $\psi$ is polynomial-time evaluable. Each computation branch evaluates $\psi$ on the database $\mathfrak{B}$ it sees. Then $\psi^{\mathfrak{A}}$ and $\psi^{\mathfrak{B}}$ are compared. In case the two values coincide, the contribution towards the expected *difference* is 0 and the computation rejects. If on the other hand the values differ, the computation accepts. Conclusively, for each database $\mathfrak{B}$ there are accepting computation paths, namely exactly $m_{\mathfrak{B}}$, iff $\psi^{\mathfrak{A}} \neq \psi^{\mathfrak{B}}$. In symbols, the number of accepting computation paths is

$$\sum_{\mathfrak{B} \in \Omega_{\mathfrak{D}}} \nu(\mathfrak{B}) \cdot g \cdot (\psi^{\mathfrak{A}} \oplus \psi^{\mathfrak{B}}) \;=\; H_{\psi}(\mathfrak{D}) \cdot g.$$

Hence dividing the result of the nondeterministic part divided by $g$ yields the desired expected error.

If $\psi$ is more complicated[3], i.e. in PH but not in P, we have to use Theorem 1.2 in order to evaluate $\psi$. Recall that according to 1.2 we can arbitrarily choose a

[3]Assuming P $\subsetneq$ PH.

polynomial $q(n)$, such that there exists a #P-function $f$ and a polynomial $t(n)$, where the output bit telling us whether the input satisfies $\psi$ is padded by $q(n)$ zero bits in either direction. If we can choose $q$ in a way that $q(n)$ is always greater than the length of the binary representation of the number $l$ of leafs in the computation tree above, we can directly extract $H_\psi(\mathfrak{D}) \cdot g$ from the result of the nondeterministic part, as in the simple case. Simply discard the lower $(t(n) + q(n))$ bits, take the (new) lower $(q(n) + 1)$ bits as result and discard all higher-order bits.

Finding a polynomial upper-bound for $l$ is no problem. The first part of the nondeterministic computation splits once for each atomic statement in $\mathfrak{A}$; take $n$ as upper bound. The second part splits $\nu(\mathfrak{B}) \cdot g$ times, where $\nu(\mathfrak{B}) \leq 1$ for all $\mathfrak{B}$ and $g$ is polynomial in $n$. Hence $q(n) := n + g$ is a legal choice.

# Chapter 3

# Approximating Query Reliability

## 3.1 Using an FPTRAS

### 3.1.1 Preliminaries

We have seen, that even when restricted to the very limited class of conjunctive queries, the reliability problem is already $\mathrm{FP}^{\#\mathrm{P}}$-hard. One possible way out of this class of (as far as we know) not efficiently computable functions is to relax the high standards we expect from our reliability calculation. The idea is to find an efficient algorithm approximating the desired result upto some acceptably small error. Considerable research on the approximability of #P functions has already been conducted (e.g. [17, 10]) with mixed results. It has been shown that some #P-hard problems are and some are not approximable in the sense introduced below. There are of course many ways to precisely define what approximating a function is to mean, e.g. regarding the allowed error of the approximation results. Here we will give the definition of and later use a rather strong notion of approximability as generally employed in the context of #P problems. We will unfortunately reach the limits of this kind of approximabiliy rather soon and have to then continue with a more relaxed approach.

**Definition 3.1.** Let $f : \mathrm{dom}(f) \longrightarrow \mathfrak{Q}$ be a function with values in some numerical domain, e.g. $\mathfrak{Q} = \mathbb{N}$ or $\mathbb{Q}$. Let $A$ be a randomized algorithm taking two positive rationals $\varepsilon, \delta$ and an element $x \in \mathrm{dom}(f)$ as input. We call $A$ a *polynomial-time randomized* $(\varepsilon, \delta)$*-approximation algorithm* for $f$ iff there exists a polynomial $p(n)$ such that for some fixed $\varepsilon, \delta > 0$ and all $x \in \mathrm{dom}(f)$

$$\Pr\left[|A(x) - f(x)| > \varepsilon \cdot f(x)\right] < \delta \qquad \text{and} \tag{3.1}$$

$$\mathrm{TIME}_A(x) \leq p(|x|). \tag{3.2}$$

If (3.1) holds *for all* $\varepsilon, \delta > 0$ and in addition there exists a polynomial $p(x, y, z)$ such that $A$ satisfies not only (3.2), but also

$$\mathrm{TIME}_A(x, \varepsilon, \delta) \leq q(|x|, 1/\varepsilon, 1/\delta), \tag{3.3}$$

then we call $A$ a *fully polynomial-time randomized approximation scheme*, or FPTRAS for short. $\diamond$

Using an FPTRAS is a strong idea of approximation in the sense that it enables us to efficiently approximate a given function with values which, with arbitrarily high probability, lie arbitrarily close to the real value.

It is known that a number of #P-complete functions admit an FPTRAS. We will show that this is also the case for the probabilities of existential queries.

We proceed as follows: We will reduce the problem of calculating the probability of an existential query to the problem of calculating the probability that a given propositional formula in $k\,$DNF is true given a probability for each of its variables. For this problem, we can show the existence of an FPTRAS using a result of Karp and Luby [10].

From an FPTRAS for the *probabilities* of existential sentences we obtain a slightly weaker notion of approximability for the *reliability* of any existential or universal query. This weaker version is sufficient for most practical purposes. We then discuss why it cannot be strengthened to an FPTRAS for reliability.

**Definition 3.2.** Let $C$ be a class of propositional formulae. Then $\#C$ is the corresponding counting problem and Prob-$C$ is the corresponding probability problem:

- $\#C$: Given a formula $\varphi \in C$, calculate the number of assignments (to the variables in $\varphi$) that make $\varphi$ *true*.
- Prob-$C$: Given a propositional formula $\varphi \in C$, and a probability function $\nu$ that assigns to each variable $X$ of $\varphi$ a probability $\nu(X) = \Pr[X = 1]$, calculate the induced probability $\nu(\varphi)$ of $\varphi$ being *true*. $\diamond$

In the following we are interested in DNF, the class of propositional formulae in disjunctive normal form, and $k\,$DNF, the class of DNF-formulae where each disjunct has at most $k$ literals.

**Theorem 3.3 (Karp, Luby).** *The problem* #DNF *admits an* FPTRAS.

We use this theorem to derive a similar result for the problems Prob-$k\,$DNF. Remember that one global precondition is the restriction to rational error probabilities.

**Theorem 3.4.** *For all $k \in \mathbb{N}$, the problem* Prob-$k$DNF *admits an* FPTRAS.

**Proof:** Let $\varphi$ be a $k$DNF formula and let $\nu : \{X_1, \ldots, X_m\} \to [0,1]$ be a probability function on the variables of $\varphi$. We will transform $(\varphi, \nu)$ into an appropriate instance $\varphi''$ for #DNF.

For each variable $X$ of $\varphi$, with probability $\nu(X) = p/q$ we introduce new propositional variables $\bar{Y} = Y_{\ell-1}, \ldots, Y_0$, where $\ell = \text{len}(q)$, the length of the shortest binary representation of $q$. We write $\text{val}(\bar{Y})$ for the natural number with binary representation $\bar{Y}$.

We first note that for every $b \in \mathbb{N}$ and $\ell \geq \text{len}(b)$, we can efficiently construct DNF-formulae of length $O(\ell^2)$ expressing "$\text{val}(\bar{Y}) < b$" and "$\text{val}(\bar{Y}) \geq b$", respectively. Indeed, if $b_{\ell-1} \cdots b_0$ represents $b$ in binary then "$\text{val}(\bar{Y}) < b$" is expressed by

$$\bigvee_{\substack{i<\ell \\ b_i=1}} \Big(\neg Y_i \wedge \bigwedge_{\substack{i<j<\ell \\ b_j=0}} \neg Y_j\Big).$$

The formula for "$\text{val}(\bar{Y}) \geq b$" is similar.

We now replace in $\varphi$ every occurence of the literal $X$ by the formula "$\text{val}(\bar{Y}) < p$" and every occurrence of $\neg X_i$ by "$\text{val}(\bar{Y}) \geq p$". This operation is performed for all variables $X$ of $\varphi$ (taking a sequence of fresh variables $\bar{Y}$ for each $X$) and the resulting formula is transformed into DNF to obtain $\varphi'$. Note that this process may increase the length of $\varphi'$ exponentially in $k$, but (since the original formula is in $k$DNF) only polynomially in the length of $\varphi$ and in the number of bits used for the probabilities.

In the case that all probabilities $\nu(X)$ are dyadic rationals, i.e. of the form $p/2^\ell$, we are done. Indeed, for each variable $X$ of the original formula $\varphi$, we have $\ell$ variables $\bar{Y}$; there are $p$ assignments satisfying the formula "$\text{val}(\bar{Y}) < p$" corresponding to the literal $X$, and $2^\ell - p$ assignments satisfying "$\text{val}(\bar{Y}) \geq p$" corresponding to the literal $\neg X$. Hence the probability $\nu(\varphi)$ is just number of assignments satisfying $\varphi'$ divided by the total number of assignments.

However, if the denominators of the probabilities $\nu(X)$ are not powers of two, this is no longer the case. Consider the case that $\nu(X) = p/q$ with $q < 2^\ell$. We still have $p$ assignments satisfying the formula "$\text{val}(\bar{Y}) < p$" (corresponding to $X$) and $2^\ell - p$ assignments satisfying "$\text{val}(\bar{Y}) \geq p$" (corresponding to $\neg X$). However, we should have only $q - p < 2^\ell - p$ assignments corresponding to $\neg X$. Call an assignments to $\bar{Y}$ *illegal* if it satisfies the formula "$\text{val}(\bar{Y}) \geq q$". The probability $\nu(\varphi)$ is the number of *legal* assignments satisfying $\varphi'$ divided by the total number of legal assignments. We can easily calculate the total number of legal assignments (the product of all denominators of the probabilities $\nu(X)$) and hence the total number of illegal assignments. To determine the number of legal assignments satisfying $\varphi'$, we transform $\varphi'$ into a new formula $\varphi''$ that is implied by $\varphi'$ and, in addition, is satisfied by all illegal assignments. For instance, we can take for $\varphi''$ the disjunction of $\varphi'$ with the formulae "$\mathrm{val}(\bar{Y}) \geq q$" for all sequences $\bar{Y}$ that we have introduced for the variables of $\varphi$. Clearly $\varphi''$ is in DNF and can be constructed in polynomial time.

The number of legal assignments satisfying $\varphi'$ is the number of all assignments satisfying $\varphi''$ minus the total number of illegal assigments. Hence, by using the FPTRAS of Karp and Luby for #DNF to approximate the number of assignments for $\varphi''$, we get an FPTRAS for calculating $\nu(\varphi)$.

### 3.1.2 Results

We are now in a position to prove our first approximability result. We will then discuss why we restrict ourselves to existential and universal queries. After introducing our weaker sense of approximation we will give a result regarding a larger class of queries.

**Theorem 3.5.** *Let $\psi$ be any boolean existential query. Then the probability $\nu(\psi)$ (that $\psi$ holds in the actual database) admits an* FPTRAS.

**Proof:** Let $\psi = \exists\bar{y}\varphi(\bar{y})$ be an existential query, where $\varphi$ is quantifier-free and in $k$DNF, for some $k \in \mathbb{N}$.

Given a probabilistic database $\mathfrak{D} = (\mathfrak{A}, \mu)$ with $n$ elements, we first calculate the probabilities $\nu(\alpha)$ for all atomic statements $\alpha$ on $\mathfrak{A}$. We then replace the quantifiers in $\psi$ by disjunctions over all possible values:

$$\psi(\bar{x}) = \exists\bar{y}\varphi(\bar{x},\bar{y}) \;\longmapsto\; \bigvee_{\bar{b}} \varphi(\bar{x},\bar{b}) =: \psi'(\bar{x}).$$

Now we let $\psi''$ be the propositional formula obtained from $\psi'$ by replacing equalities by their truth values and considering atomic statements $R\bar{c}$ as propositional variables. Note that the number of variables per conjunction does not depend on the size of $\mathfrak{D}$, so $\psi''$ is a propositional formula in $k$DNF whose length is polynomial in $n$, and the function $\nu$ defines probabilities for the variables of $\psi''$. Obviously $\nu(\psi'')$ is just the expectation that $\psi$ holds in a randomly chosen $\mathfrak{B} \in \Omega_{\mathfrak{D}}$. By Theorem 3.4 we have an FPTRAS for calculating $\nu(\psi'')$.

Theorem 3.5 implies that the reliability of existential and universal queries can be approximated in the following sense.

**Corollary 3.6.** *Let $\psi$ be an existential or a universal query. Then there exists a polynomial-time randomized algorithm $M$ such that for all unreliable databases $\mathfrak{D}$ and all $\varepsilon, \delta > 0$*

$$Pr[|R_\psi(\mathfrak{D}) - M(\mathfrak{D})| > \varepsilon] < \delta.$$

**Proof:** If $\psi$ is an existential or universal *Boolean* query this is immediate from Theorem 3.5, since for every database $\mathfrak{D} = (\mathfrak{A}, \mu)$ either $H_\psi = H_\psi(\mathfrak{D}) = \nu(\psi)$ and $R_\psi = 1 - \nu(\psi)$ or vice versa, depending on whether $\psi$ holds in $\mathfrak{A}$, or not (cf. Definitions 2.3 and 2.4).

For $k$-ary queries with $k > 0$, we can use Lemma 1.3 and approximate $H_\psi$ by taking the sum of appropriate approximations of the $H_{\psi[\bar{a}]}$ (with error $\varepsilon/n^k$ and probability $\delta/n^k$) for $H_{\psi[\bar{a}]}$ over all $k$-tuples $\bar{a}$. Since $R_\psi = 1 - H_\psi/n^k$ this gives the desired result.

The notion of approximability used in Corollary 3.6 is technically weaker than the existence of an FPTRAS, since $\varepsilon$ bounds the absolute rather than the relative difference of $M(\mathfrak{D})$ and $R_\psi(\mathfrak{D})$ (and both values are between zero and one). In practice, this weaker notion may be sufficient: The user of a database system will be content to know the reliability up to some very small absolute error, even if the actual value is very close to 0 (and we cannot say anything about the relative error). We call such an algorithm a *weak polynomial-time randomized approximation scheme*, or WPTRAS. Formally the definition for a WPTRAS can be derived from an FPTRAS (cf. Definition 3.1) by replacing (3.1) by

$$\Pr\left[|A(x) - f(x)| > \varepsilon\right] < \delta\ . \tag{3.4}$$

Of course we will only keep this "second best" result, if we believe it impossible to strengthen the claim of 3.6 to use an FPTRAS. We will now take a look at the associated decision problem. The following will show that we have most probably already reached the borders of approximability via FPTRAS'.

**Definition 3.7.** Fix any query $\psi$. Let $\mathrm{AR}_\psi$ be the set of all unreliable databases $\mathfrak{D}$ where $R_\psi(\mathfrak{D}) = 1$. We call determining whether a given $\mathfrak{D}$ belongs to $\mathrm{AR}_\psi$ the *absolute reliability problem.* $\diamond$

**Lemma 3.8.**

- *Let $\psi$ be a quantifier free query. Then* $\mathrm{AR}_\psi \in \mathrm{P}$.
- *Let $\psi$ be a polynomial-time evaluable query. Then* $\mathrm{AR}_\psi \in$ Co-NP.

The proofs are simple: In the first case $R_\psi$ is computable in polynomial time. In the second case guess a database $\mathfrak{B}$ and check whether the truth values $\psi^\mathfrak{A}$ and $\psi^\mathfrak{B}$ differ. For our further observations we take a closer look at existential queries.

**Lemma 3.9.** *There exist existential queries $\psi$ such that $AR_\psi$ is* Co-NP*-hard.*

**Proof:** We reduce the problem of 4-colourability of graphs to the complement of $AR_\psi$ for some existential query $\psi$.

Besides the edge relation $E$ we let our language contain two unary relations $R_1$ and $R_2$, together giving one of 4 colours for each node. Now consider the query

$$\psi = \exists x \exists y (Exy \wedge (R_1x \leftrightarrow R_1y) \wedge (R_2x \leftrightarrow R_2y) \wedge x \neq y)$$

expressing that two connected nodes share the same color, i.e. that $R_1, R_2$ do *not* represent a correct 4-colouring.

Given an arbitrary graph $G = (V, E)$ we now construct a database $\mathfrak{D} = (\mathfrak{A}, \mu)$. The universe $V$ and graph relation $E$ are taken unchanged. For our given database $\mathfrak{A}$ we let $R_1, R_2 = \emptyset$, giving all nodes the same colour. The error function $\mu$ is such that $\mu(Euv) = 0$ and $\mu(R_iv) = 1/2$ for all nodes $u, v \in V$, $i = 1, 2$. Note[1] that $\mathfrak{A} \models \psi$. For this construction the existence of a 4-colouring for $G$ is equivalent to $\mathfrak{D} \notin AR_\psi$.

For our existential non-4-colouring query $\psi$ we can approximate the reliability $R_\psi$, however we do not know how to approximate the expected error $H_\psi$. A simple observation shows that the existence of an FPTRAS for the reliability of all existential queries without precondition for $\mathfrak{D}$ (and hence the approximability of both $H_\varphi$ and $R_\varphi$ for all existential or universal queries $\varphi$) is highly unlikely.

**Lemma 3.10.** *Let $f$ be any function such that the associated decision problem $\{x : f(x) > 0\}$ is* NP*-hard. If, for some $\varepsilon, \delta < 1/2$, $f$ admits a randomized polynomial-time $(\varepsilon, \delta)$-approximation algorithm, then* NP $\subseteq$ BPP.

**Proof:** Let $f$, $\varepsilon$ and $\delta$ be as required above and let $A(x)$ be such an approximation algorithm for $f$. We then have $\Pr[|f(x) - A(x)| > \varepsilon \cdot f(x)] < \delta$. Now if $f(x) = 0$, the probability of $A(x) > 0$ is less than $\delta$. Conversely if $f(x) > 0$, the probability of $A(x)$ differing from $f(x)$ by more than $f(x) \cdot \varepsilon < f(x)/2$, and hence of $A(x)$ being zero, is less than $\delta$. As $\delta < 1/2$ we are done.

Note that the decision problem associated with the expected error $H_\psi$ is the complement of the absolute reliability problem $AR_\psi$. Hence there exist existential queries (such as the non-4-colouring query) whose expected error does not admit an FPTRAS, unless NP $\subseteq$ BPP.

[1]Quietly ignoring the case where $E = \emptyset$ or $|V| < 2$.

## 3.2 Using a WPTRAS

### 3.2.1 Results

As we cannot expect any positive results using an FPTRAS for reliability calculations we take a look back at Corollary 3.6. Relaxing our bound on the allowed error gave us a positive, if weaker, approximability result, a WPTRAS for the reliability of existential and universal queries. We will now derive a similar result for all polynomial-time evaluable queries. The proof is based on methods used in [10] for the development of an FPTRAS for #DNF. We will make some modifications in order to obtain a result for all polynomial-time evaluable queries. Some of these modifications destroy the bounds necessary for an FPTRAS, but what is left over turns out to be sufficient for a WPTRAS.

**Theorem 3.11.** *Let $\psi$ be a polynomial-time evaluable query. Then there exists a polynomial-time randomized algorithm $M$ such that for all probabilistic databases $\mathfrak{D}$ and $\varepsilon, \delta > 0$*

$$Pr[|R_\psi(\mathfrak{D}) - M(\mathfrak{D})| > \varepsilon] < \delta.$$

**Proof:** The proof is based on methods used by Karp and Luby in [10] for constructing an FPTRAS for #DNF.

In case $\psi$ is not a Boolean query we use the same idea as in the proof of Lemma 3.6 and approximate the query reliability using the sum over all approximations for each possible valuation of the free variables, alas with stricter bounds.
For the following let $\psi$ be a polynomial-time evaluable Boolean query and let[2] $\xi \in (0, 1/2)$ be a rational.

Let $\mathfrak{D} = (\mathfrak{A}, \mu)$ be a probabilistic database and $\varepsilon, \delta > 0$. We first have to slightliy modify $\mathfrak{D}$ and $\psi$. Let $R$ be a new unary relation and let $c, d$ be two new constants. We define a new database $\mathfrak{D}' = (\mathfrak{A}', \mu')$ and query $\psi'$, where

- $\mathfrak{A}' = (\mathfrak{A}, R, c, d)$ with $R = \emptyset$ and $c \neq d$,
- $\mu'(P\bar{a}) = \begin{cases} \xi & \text{if } P = R \text{ and } \bar{a} = c \text{ or } \bar{a} = d, \\ \mu(P\bar{a}) & \text{otherwise,} \end{cases}$
- $\psi' = (\psi \vee Rc) \wedge Rd.$

[2]It is crucial that $\xi$ can be chosen prior to knowing any one of $\mathfrak{D}$, $\varepsilon$ or $\delta$.

Now let $\mathfrak{B}$' be a random variable taking values in $\Omega_{\mathfrak{D}'}$ with distribution $\nu'$ as induced by $\mu'$. We define a random variable $X$ through $X = \psi^{\mathfrak{B}'}$ and let $\{X_i\}_{i>0}$ be a sequence of independent identically distributed random variables where each $X_i$ is distributed as $X$. It follows that for $p := E(X) = \nu'(\psi')$ we have

$$0 < \xi^2 \leq p \leq \xi < 1/2.$$

For the upper bound of $p$ note that on top-level $\psi'$ contains a conjunction with the atom *Rd*. Since $R$ is empty in $\mathfrak{A}$, $\nu'(Rd) = \mu'(Rd) = \xi$. It follows that $p = q \cdot \xi$ for some $0 \leq q \leq 1$. For the lower bound note that $\psi'$ is *true* if *Rc* and *Rd* hold. For the number of repetitions in Lemma 1.4 we let

$$t = t(\varepsilon, \delta) = \left\lceil \frac{9}{2\,\xi\,\varepsilon^2} \ln \frac{1}{\delta} \right\rceil .$$

Due to $\xi$ being fixed, $t$ is bounded by a polynomial in $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$. Further the $X_i$ are polynomial-time computable, too. It follows that our approximator $\widetilde{X} = \frac{X_1 + \cdots + X_t}{t}$ is polynomial-time computable. The modifications of $\mathfrak{D}$ and $\psi$ lead to our $X_i$ satisfying the preconditions of Lemma 1.4. A few simple transformations on the probability bound in the result of Lemma 1.4 show that for our choice of $t$ we get

$$\Pr\left[\left|\frac{X_1 + \cdots + X_t}{t} - p\right| > \varepsilon \cdot p\right] < \delta. \tag{3.5}$$

It follows from our construction that

$$\nu(\psi) = \frac{(p - \xi^2)}{(\xi - \xi^2)}\ , \tag{3.6}$$

hence using

$$\alpha = \frac{(\widetilde{X} - \xi^2)}{(\xi - \xi^2)} \tag{3.7}$$

as approximation for $\nu(\psi)$ satisfies

$$\Pr\left[|\alpha - \nu(\psi)| > 2 \cdot \varepsilon\right] < \delta. \tag{3.8}$$

To see why (3.8) holds, first note that $\xi < 0.5$ and therefore

$$\frac{\xi^2}{\xi - \xi^2} < 1\ . \tag{3.9}$$

We use (3.6) and (3.7) to substitute $\widetilde{X}$ and $p$ in (3.5) and obtain

$$\Pr[|(\xi - \xi^2)(\alpha - \nu(\psi)| > \varepsilon \cdot (\xi^2 + (\xi - \xi^2)\nu(\psi)] < \delta \ ,$$

or equivalently

$$\Pr\left[\bigl|\alpha - \nu(\psi)\bigr| > \varepsilon \cdot \left(\frac{\xi}{\xi - \xi^2} + \nu(\psi)\right)\right] < \delta \ .$$

Due to (3.9) and $\nu(\psi) < 1$ we are done.

Together with above observations that the $X_i$ are polynomial-time computable and that $t$ bounded by a polynomial in $\frac{1}{\varepsilon}$ (and hence in $\frac{2}{\varepsilon}$) and $\frac{1}{\delta}$, (3.8) shows that our approximator is indeed a WPTRAS. Given an $\varepsilon$ as allowed error we simply use $\varepsilon/2$ in our algorithm.

# Chapter 4

# The Metafinite Case

## 4.1 Metafinite Structures

The database models considered so far are limited in that they are completely finite. In practical applications it is often necessary to deal with infinite domains, e.g. when storing numbers or strings. Also these domains are often used as more than plain sets, e.g. by adding arithmetic operations. We will use parts of the framewok of *metafinite model theory* as presented in [6] to add an infinite part to our databases in a way that preserves some complexity results. Let us now take a closer look at a special case of *metafinite algebras* to be used as databases, and so-called *multiset operations* which will be used as aggregate functions.

We restrict ourselves to so-called numerical structures for the potentially infinite part of our metafinite construction. Omitting an exact definition of what a numerical structure is, we may get an intuition by thinking of typical examples like the ordered field of reals, the field of rationals or the naturals with arithmetic. Finite domains are permitted, but we require at least two distinct elements to interpret two mandatory constants, 0 and 1. We further want all operations to be efficiently computable, i.e. in polynomial time. Without imposing such a limit we are unable to keep the precondition of all first-order queries being polynomial-time evaluable.

**Definition 4.1.** Let $A$ be a finite set and let $\mathfrak{R}$ be a numerical structure. Further let $\mathcal{F}$ be a finite set of functions $f : A^k \to R$. Then the triple $\mathfrak{A} = (A, \mathfrak{R}, \mathcal{F})$ is a *metafinite algebra*. We call $A$ the *primary* part and $\mathfrak{R}$ the *secondary* part of $\mathfrak{A}$. $\diamond$

The interesting part is how logical formulae and terms are built with respect to the two domains. We allow our logics to use a countable set of variables $V$.

Now let $\sigma_{\mathfrak{R}}$ denote the vocabulary of $\mathfrak{R}$. The variables in $V$ only take values in $A$. Terms on the other hand only take values in $R$. Atomic terms are obtained by applying functions[1] in $\mathcal{F}$ to variables. Hence atomic terms perform the step from the primary domain to the secondary domain. Further the set of terms is closed under application of functions in $\sigma_{\mathfrak{R}}$. Anything built differently is not a term. Note that like relations in the classical setting, functions from $\mathcal{F}$ can not be nested. In case we allow constants over the primary part they are of course handled just like variables when building terms.

Atomic formulae are built (as usual) using equations between variables or[2] terms and by applying relations in $\sigma_{\mathfrak{R}}$ to terms. The two explicitly given elements 0 and 1 are needed for the *characteristic function* operator $\chi$. Just as relations perform the step from terms, which live in an underlying domain, into the world of *true* and *false* in the classical setting, this operator performs the step in the other direction. If $\varphi$ is a formula, then $\chi[\varphi]$ is a term with the same free variables as $\varphi$ and the canonical semantics:

$$\chi[\varphi]^{\mathfrak{A}}(\bar{a}) = \begin{cases} 1 & \text{if } \mathfrak{A} \models \varphi(\bar{a}) \\ 0 & \text{otherwise.} \end{cases}$$

Up to this point we have only allowed atomic formulae. We define quantifier-free, first-order and second-order expressions by allowing the corresponding junctors and quantifiers in the construction of (sub)formulae. The top-level symbol of an expression decides whether it is a formula or a term, i.e. whether it yields a truth value or a numerical value, regardless of the nesting of terms and formulae made possible by the characteristic function operator.

The core of what distinguishes metafinite structures from infinite ones is the restricted quantification. First- and second-order variables both live in the primary part, so all quantifiers range over a finite set. One advantage of this approach is that queries can still be effectively evaluated. A further gain is with respect to the use of functions and aggregate operators. When dealing with infinite numerical domains in databases, normally a finite subset, the *active domain* which contains all elements explicitly named in the presentation of the database, is used for query evaluation. If, however, a query contains aggregates like sum or avg or even simple operations like addition, the result will generally not be in the active domain, unless it is closed under such operations. This is unlikely, as the active domain is introduced as finite subset containing all query-relevant constants.

[1]Correctly: function symbols. As often the case — indeed in this work too — we know that there *is* a difference between a function symbol and an actual function, only to never make this distinction explicit once we are past the introductary level.

[2]Note that variables and terms take values in distinct domains and are hence incomparable.

The only case where we need a kind of active domain is to find finite ranges for second-order quantifiers. All first-order constructions go through without active domains and as will be seen the question of results leaving the active domain does not arise.

Let us take a closer look at multiset operations. A multiset operator behaves partly like a quantifier, in that it binds one or more variables. It also acts like a function, as it is applied to terms and again produces a term. Of course it is more general than both, as it maps finite multisets to numbers, i.e. members of the secondary part, and can be used to simulate both quantifiers and functions. We now give an exact definition of how such an operator works.
**Syntax:** Let $\Gamma$ be a multiset operator. Further let $t(\bar{x}, \bar{y})$ be a term and $\varphi(\bar{x}, \bar{y})$ a formula. Then the expression

$$\Gamma_{\bar{x}}(t(\bar{x}, \bar{y}) : \varphi(\bar{x}, \bar{y}))(\bar{y})$$

is a term with free variables $\bar{y}$.
**Semantics:** Let $s(\bar{y}) = \Gamma_{\bar{x}}(t(\bar{x}, \bar{y}) : \varphi(\bar{x}, \bar{y}))(\bar{y})$ be a term to be interpreted in $\mathfrak{A} = (A, \mathfrak{R}, \mathcal{F})$ with $\sigma_s \subseteq \sigma_{\mathfrak{R}}$. With valuation $\bar{b}$ for $\bar{y}$ the value of $s$ is defined as

$$s^{\mathfrak{A}}(\bar{b}) := \Gamma^{\mathfrak{A}}\big(\{\!\{ t^{\mathfrak{A}}(\bar{a}, \bar{b}) \mid \bar{a} \in A^k, \mathfrak{A} \models \varphi(\bar{a}, \bar{b}) \}\!\}\big)$$

Formally the vocabulary $\sigma_{\mathfrak{R}}$ of a structure $\mathfrak{R}$ with multiset operations is extended to include these operators — besides the function and relation symbols. Now just as quantifiers come in a first-order and a second-order version the same is true for multiset operators. The extension is straightforward. Take a new sequence of second-order variables, or relation symbols $\bar{P}$ and a multiset operator $\Xi$. Then $\Xi_{\bar{P}}(t(\bar{y}) : \varphi(\bar{y}))(\bar{y})$ is a term, where the formula $\varphi$ and the term $t$ may contain relations from $\bar{P}$. Said term is interpreted as $\Xi\big(\{\!\{ t^{\mathfrak{A}}(\bar{b}) \mid P_i \subseteq A^{k_i}, P_i \in \bar{P}, (\mathfrak{A}, \bar{P}) \models \varphi(\bar{b}) \}\!\}\big)$, where $k_i$ is the arity of $P_i$ and $\bar{b}$ a valuation for the free variables $\bar{y}$.

The multiset operations considered later on will be some of the prominent aggregates used in SQL: min, max and sum. If desired, this enables us to eliminate all quantifiers by replacing them with aggregates. For example

$$\exists x\, \varphi(x) \quad \widehat{=} \quad (1 = \max_x \ \chi[\varphi(x)])$$

and similarly

$$\forall x\, \varphi(x) \quad \widehat{=} \quad (1 = \min_x \ \chi[\varphi(x)]).$$

## 4.2 Third Approach: Integrating Probability Information

The first application of this new framework is to move the probability information into the database. This brings several advantages. Until now the error function was merely an add-on to the well developed framework of relational databases. We can now take e.g. the rationals as secondary part of our finite database and let the probability functions be a part of the database proper. This also allows the study of definability questions in a well defined surrounding. This is a more natural approach in the sense that query languages are commonly given as logics, not as complexity classes. Hence it is more fitting to see which query language has the power to express the reliability of a query as query again.

The embedding of Definition 2.6 into a metafinite structure is straightforward. Given a database $(\mathfrak{A}, \bar{\mu})$ we take $\mathfrak{A}$ as primary part. The secondary part has to encompass everything needed for reliability calculations. Take $(\mathbb{Q}, +, -, \cdot, /, 1, 0, \leq, \sum)$, the ordered field of rationals with summation as aggregate. Now $\bar{\mu}$ is a sequence of weight functions. Again for each database relation $R \subseteq A^k$ we have an error function $\mu_R : A^k \longrightarrow \mathbb{Q}$ giving the probability of all atoms $R\bar{a}$ differing in the given and a randomly chosen database.

The reader will notice, that contrary to the definition of a metafinite algebra, the primary part is an interpreted structure containing relations instead of merely a plain set. In [6] a metafinite algebra where the primary part is an arbitrary structure is called a *metafinite structure*. We can get around these (in our simplified setting) disturbing relations by changing the relations to functions mapping tuples to $0, 1 \subseteq \mathbb{Q}$. We can then use the supplied operations to simulate logical operators. This idea of using functions into the secondary part as database will be generalized in the next section.

An interesting definability result in [6] is the following.

**Theorem 4.2 (Grädel, Gurevich).** *Let $\psi$ be a quantifier-free query. Then the reliability of $\psi$ is first-order definable.*

This strengthens our earlier result about reliability being polynomial-time computable for quantifier-free queries. Note that this and all further *definability* results apply equally when using real probabilities. The reader may exemplary look at the proof of Lemma 4.8. Since we are interested in complexity theoretic results about query reliability, we will however not pursue this path of taking the ordered real field as secondary part.

## 4.3 Fourth Approach: Functional Databases

### 4.3.1 Model

Now it is time to leave the relational model behind and use the new possibilities of our metafinite setting for the databases themselves. The idea is to replace the database relations by database functions. Relations can of course be seen as functions too, mapping tuples to truth values. Now we let database functions range into the secondary part of our metafinite structures. The domain in this case is the finite primary part.

This extension also leads to a new view about the facts contained in a database. When looking at a relational structure, we can pick a relation and speak of a given tuple either belonging to the relation — or not. In the functional case the question is not whether a tuple belongs to a function[3], but rather what the *value* of the tuple is, under said function. We could of course take the freedom to state that, by convention, for a certain element **nil** of the secondary part, any tuple being mapped to **nil** "does not belong to the (corresponding) function". This idea will not be pursued.

**Definition 4.3.** A *probabilistic functional database* $\mathfrak{D}$ is a tuple $(\mathfrak{A}, \nu)$, where

- $\mathfrak{A}$ is a metafinite algebra $(A, \mathfrak{R}, \mathcal{F})$, where $A$ is a finite set, $\mathfrak{R}$ is a numerical structure and $\mathcal{F}$ is a set of functions of the form $f : A^k \to R$,
- $\nu$ is the probability function. For each $f \in \mathcal{F}$ and $\bar{a} \in \mathrm{dom}(f)$, $\nu_{f(\bar{a})}$ is a probability distribution on $R$ where $\nu_{f(\bar{a})}(r) = \nu_f(\bar{a}, r) = \nu(f(\bar{a}) = r)$ is the probability of the value of $f(\bar{a})$ being $r \in R$. Again all probability distributions are assumed to be independent.

  Further each distribution $\nu_{f(\bar{a})}$ may only contain finitely many non-zero probabilities, i.e. $\{r \in R \mid \nu_{f(\bar{a})}(r) > 0\}$ is finite for all $f$ and $\bar{a}$. $\diamond$

The database relations are replaced by database functions with values in $R$. Instead of one alternative truth value for each atomic statement we now have a whole set of possible alternatives — those given by a positive probability. Consequently queries are terms and query results are functions from the primary into the secondary part. Insofar this model is "closed" just as the relational case: adding a query result to a database yields another database.

The relational case can easily be embedded into the functional model by setting $\mathfrak{R}$ to $(\{0, 1\}, \wedge, \neg, 0, 1)$. The functions in $\mathcal{F}$ then take values in $\{0, 1\}$, effectively degraded to relations.

[3]We do not allow partial functions.

To make this point clear: In the following all queries are terms. These terms may be built using formulae in combination with the characteristic function operator. A query is quantifier-free, if no aggregates or quantifiers are used. A query is first-order, if only first-order aggregates and quantifiers are used. A second-order query may use any first and second-order aggregation and quantification.

Now second-order quantification is not as straightforward as in the usual finite setting. As we are dealing with functions instead of relations, a $k$-ary second-order variable has to be interpreted as a $k$-ary function from the primary into the secondary part of the database. In case the secondary part is infinite, we can no longer enumerate all possibilities. We now introduce our version of active domain / active domain semantics. An element $s$ of the secondary part belongs to the active domain adiff there exists a tuple $\bar{a}$ in the primary part and a database function $f$ such that $f(\bar{a}) = s$. The scope of all second-order operations is the set of functions with ranges in the active domain, i.e. functions of the kind $g : A^k \rightarrow \text{ad}$. This definition is canonical in the sense that it is the most simple version providing the expressive power required for definability issues of qurey reliability in the setup of metafinite databases.

### 4.3.2 Results

In the following we will explicitly consider two variants for the secondary part: The standard arithmetic over the naturals and the field of rationals, both with order. As second-order aggregates we allow $\text{sum}, \text{min}$ and $\text{max}$. With an eye to the practicability of our results we wish our computations to (at least) stay within PSpace. Therefore we do not allow a second-order multiplication aggregate, as the results will generally require exponential space. Regarding first-order operations we have the freedom to allow multiplication and many other common operationas, as long as they are computable in deterministic polynomial-time — we wish first-order queries to be polynomial-time evaluable.

**Theorem 4.4.** *Let $\psi$ be a quantifier-free query. Then for any probabilistic functional database $\mathfrak{D}$ the problem of calculating the reliability $R_\psi(\mathfrak{D})$ is in* FP.

**Proof:** Let $\psi$ be a $k$-ary quantifier-free query. Due to the linearity of expectation we can swap expectation and summation. Hence it suffices to show that $H_\psi(\bar{a})$ can be calculated in polynomial-time for any tuple $\bar{a}$. The claim then follows due to $R_\psi = 1 - \frac{1}{|A|^k} \sum H_{\psi[\bar{a}]}$.

So let $\mathfrak{D} = (\mathfrak{A}, \mu)$ be a probabilistic functional database and $\bar{a} \in A^k$. Now imagine evaluating $\psi[\bar{a}]$ on different databases from $\Omega_{\mathfrak{D}}$. The result depends not on the whole database, but rather on the value of the atoms occuring in $\psi[\bar{a}]$ only. Hence it is not necessary to generate all possible databases; we merely have to look at all alternative values (and their probabilities) for these occuring atoms.

The algorithm proceeds as follows. First $\psi[\bar{a}]$ is evaluated and the result $\psi[\bar{a}]^{\mathfrak{A}}$ stored for later comparisons. Then for the atoms in $\psi[\bar{a}]$ all possible combinations of values[4] with positive probability are generated. Next $\psi[\bar{a}]$ is evaluated using these values. If the result differs from $\psi[\bar{a}]$, the probability for the current valuation of the atoms is added as contribution towards the end result $H_{\psi[\bar{a}]}$.

Although generating all possible valuations for the atoms is exponential in the number of atoms occuring in $\psi$, the algorithm is still polynomial in the size of $\mathfrak{D}$. Note that $\psi$ is fixed, so the number of atoms does not change. If the primary part of the database grows, the number of tuples, and thus the repetitions of above algorithm, grows polynomially (in the exponentiation describing the number of valuations for the atoms, the base depends only on the database and the power only on the query). Calculating the probability for each given valuation is now easy. As all probabilities are independent all we have to do is multiply the probabilities of each of the current values.

**Theorem 4.5.** *Let $\psi$ be a first-order evaluable query. Then the problem of calculating the reliability $R_\psi(\mathfrak{D})$ is in* $\mathrm{FP}^{\#\mathrm{P}}$.

The proof uses the same methods as the first-order part of Theorem 2.8. Also the claim can be strengthened to include all UP evaluable queries. However we do not have a result similar to Theorem 1.2 for the metafinite case allowing us to compute second-order query reliabilty in the same way.

The following results will lead us to a complexity class not encountered so far, the *counting hierarchy*, or CH. We will give an intuitive definition of CH, using PH as guideline. Think of PH as the class of all problems solvable by a polynomial-time bounded algorithm with a fixed, finite amount of nested NP / Co-NP oracle queries. The class CH is defined similarly, with the addition that the nested oracles can also include #P functions. Hence a function is in $\mathrm{FP}^{\mathrm{CH}}$ iff it is computable by a polynomial-time algorithm which may invoke either a NP, a Co-NPor a #P oracle which in turn may again invoke an oracle of one of these three kinds etc. The number of oracle invocations and the oracles themselves are fixed. Obviously the situation is

$$\mathrm{FP}^{\#\mathrm{P}} \subseteq \mathrm{FP}^{\mathrm{CH}} \subseteq \mathrm{PS{\small PACE}}.$$

For all definability issues we have to be able to express probabilities in our logical framework. This can be achieved by letting the secondary part of our databases

[4]This proceeding is in analogy to interpreting the atoms as propositional variables in the relational case.

be the field of rationals (with summation as aggregate operator). The errors can then be put into weight functions as in the previous section. We can then think of the primary part containing a copy of the active domain, or, equivalently, let the error functions be of the kind $\nu_f : A^k \times \mathrm{ad} \longrightarrow R$. There are other conceivable possibilities, but for all further means we will stick to this solution.

In order to not let results of numerical queries explode to much we have already forbidden the use of a second-order multiplication aggregate. It turns out that we require a rather technical restriction regarding the applictaion of second-order addition aggregates.

**Definition 4.6.** Let $\psi$ be a second-order query. We call $\psi$ *(addition) restricted* iff. the result of any division occuring in the scope of a second-order addition aggregate is independent of all second-order variables used in second-order addition aggregation. $\diamond$

**Theorem 4.7.** *Let $\psi$ be a restricted second-order query. Then the problem of calculating the reliability of $\psi$ is in* $\mathrm{FP}^{\mathrm{CH}}$.

We will give the proof of 4.7 in two steps. First we show that the reliabilty of second-order queries is second-order definable. Next we show second-order queries to be $\mathrm{FP}^{\mathrm{CH}}$ evaluable. Note that the precondition of the query being addition restricted is not needed for the first step.

**Lemma 4.8.** *Let $\psi$ be a second-order query. Then the reliability $R_\psi$ is second-order definable.*

**Proof:** Let $\psi$ be a $k$-ary second-order query and let $\mathfrak{D}$ be a functional database. As we know by now,

$$R_\psi(\mathfrak{D}) = 1 - \frac{H_\psi(\mathfrak{D})}{|A|^k} \quad \text{and} \quad H_\psi(\mathfrak{D}) = \sum_{\bar{a}} H_{\psi[\bar{a}]}(\mathfrak{D}).$$

We will show how to stepwise construct a second-order term expressing $R_\psi(\mathfrak{D})$. As our numerical structure considered is a field we have explicit access to a constant "1" and the subtraction and division operations. Further the aggregates allowed include first and second-order summation, so we can easily calculate $|A|^k$ using

$$|A|^k \;=\; \underbrace{\sum_{b\in A} 1 \cdot \;\ldots\; \cdot \sum_{b\in A} 1}_{k \text{ factors}} \;.$$

We now have to show how to express $H_{\psi[\bar{a}]}$. The idea is to use second order operations to simulate generating all possible databases. Our active-domain semantics for second-order variables uses an active domain ad given as

$$\text{ad} = \{r \in R \mid \nu(f(\bar{a}) = r) > 0 \text{ for some } f \in \mathcal{F}, \bar{a} \in A\}.$$

Therefore the set of all functions with the same arity as an $f \in \mathcal{F}$ taking values in ad will include all variations of $f$ in all possible databases.

Let $\mathcal{G}$ be a sequence of second-order variables, i.e. variables taking as values functions from the primary to the secondary part, where for each $f_i \in \mathcal{F}$ there is a $g_i \in \mathcal{G}$ with same arity. Let $\mathfrak{B}$ be the database where all $f_i$ are replaced by the corresponding $g_i$. Then

$$H_{\psi[\bar{a}]} = \sum_{\mathcal{G}} \left(\chi[\psi[\bar{a}]^{\mathfrak{A}} \neq \psi[\bar{a}]^{\mathfrak{B}}] \cdot \nu(\mathfrak{B})\right).$$

And the probability of $\mathfrak{B}$ can be directly calculated as

$$\nu(\mathfrak{B}) = \sum_{g_i \in \mathcal{G}} \sum_{\bar{a}} \nu_{f_i}(\bar{a}, g_i(\bar{a})).$$

The query $\psi$ itself is second-order too, so the reliability can indeed be expressed using a second-order formula using our construction.

**Lemma 4.9.** *Let $\psi$ be a restricted second-order query. Then $\psi$ is* $\text{FP}^{\text{CH}}$ *evaluable.*

**Proof:** Let $\psi$ be a $k$-ary second-order query and let $\mathfrak{D}$ be a functional database. All first-order constructions are polynomial-time evaluable and do not have to be considered. Second-order quantification is handled in the same way as in the canonical proof for SO $\subseteq$ PH. Each existential quantifier leads to an NP oracle invocation and each universal quantifier leads to a Co-NP invocation — the number of quantifier alternations corresponds to the number of alternating nested oracle calls. The interesting cases are the second-order aggregate functions. We will look at the cases max and sum; the proof for min is similar to max.

### max

Let $\psi = \max_R t(R)$, where $R$ is a second-order variable and $t$ a term. By induction hypothesis we can assume to have an $\text{FP}^{\text{CH}}$ algorithm $M_t(R)$, which calculates $t$ with respect to $R$. Note that $t$ has no free first-order variables: When evaluating a query with free variables it is evaluated once for each possible interpretation of these variables. The algorithm given below first finds an $R_{\max}$ such that $t(R_{\max})$ is maximal and then calculates $t(R_{\max})$.

The truth values of all atomic statements $R_{\max}(\bar{a})$ are chosen successively. Let there be $k$ of these truth values, so we have to find $k$ bits representing $R_{\max}$. To find the value of the $i$-th bit, we invoke the following algorithm with the first $(i-1)$ bits as argument.

Construct a local $R_a$ by taking the first $(i-1)$ bits as given and guessing all others. Next let $r_a = t(R_a)$. We now have to find out, whether $r_a$ is maximal, using the algorithm described below. If $r_a$ is not maximal, the $R_a$ we have guessed is uninteresting and we reject. If $r_a$ *is* maximal, we accept iff the $i$-th bit (which was guessed) is 1. I.e. if by extending our already given $(i-1)$ bits by a 1 as $i$-th bit we get some $i$ bits which can be extended to an $R_{\max}$.

To find out whether $r_a$ is not maximal we guess an $R_b$, again taking the $(i-1)$ first bits as given. Then let $r_b = t(R_b)$. We accept iff $r_b > r_a$.

It follows by induction that the so constructed $R_{\max}$ has the desired property and we can calculte the value of $\psi$ as $t(R_{\max})$.

### sum

Now similarly let $\psi = \sum_R t(R)$. In case the secondary part consists of the naturals, we simply invoke a #P-oracle which guesses $R$ and produces $t(R)$ accepting computation paths. As $t(R)$ is syntactically shorter than $\psi$ we can again assume to have an $\mathrm{FP}^{\mathrm{CH}}$ algorithm for $t$.

In case we are dealing with rationals, we have to get around the problem of #P oracles not being able to return fractions. The solution will be to find a polynomial upper bound for the denominator of the result of each second-order sum. Care has to be taken, as the simple algorithm of multiplying all denominators occuring in the sum would produce a double-exponentially large number with an exponentially large representation.

Write $\psi^{\mathfrak{D}}$ as fraction $\frac{v}{w}$. We will first construct $w$ and then calculate $v$. First note that we have an ordering on the set of all possible functions for $R$. We can either use the ordering supplied as part of the secondary structure or look at the encodings on the Turing tape. The initial value of $w'$, which at the end of the computation will have the correct value of $w$, is the denominator of the result of $t$ applied to the first of above possible functions. Next we invoke the following #P-algorithm A until it returns a value of 1. As long as other values are returned, calculate a new $w'$ by multipliyng $w'$ by said return value.

A Guess an $R_a$ as interpretation for $R$ nondeterministically. Then calculate $r_a = t(R_a)$. If the denominator of $r_a$ divides $w'$, reject. Otherwise find out whether $R_a$ is the smallest function with the property that $t(R_a)$ does not divide $w'$ by invoking the NP-algorithm B below. If $R_a$ is not this smallest function, reject. Otherwise calculate the gcd $g$ of $r_a$ and $w'$ and return $g$, i.e. produce exactly $g$ accepting computation paths.

B Guess an $R_b$ nondeterministically and calculate $r_b = t(R_b)$. We then check whether the denominator of $r_b$ does *not* divide $w'$. If that is the case, we accept iff $R_b < R_a$, signalling that $R_a$ is not the smallest function with that property.

It is clear, that if we let $w = w'$ after above loop terminates, then $w$ is the smallest non-negative integer which can be used as denominator for $\psi^{\mathfrak{D}}$ (without knowledge about the numerators).

We can now calculate $v$ by using a #P-algorithm. First all possible functions for $R$ are constructed nondeterministically in parallel. Then $t$ is evaluated using the guessed $R$, which yields some result $r$. Finally a representation of $r$ in the form $\frac{r'}{w}$ is calculated and $r'$ accepting computation paths produced.

Clearly the given algorithm is polynomial-time iff. the length of the binary presentation of $w$ is polynomially bounded. To establish a proof for the "$\Longrightarrow$" direction needed we use the fact that $\psi$ is restricted. But first take a look at the prime decomposition of $w$. Let $d_i$ denote the denominators of the addends. There are two ways, how $w$ could become to large, i.e. require an exponentially long (binary) representation. Any one prime could occur to often as factor of $w$. However this is not possible due to the $d_i$ being of only polynomial size. The other possibility is that the $d_i$ contain to many different primes as factors, which would all have to occur in $w$.

Now let us split the $d_i$ giving $d_i = d \cdot d_i'$, where $d$ is the part of the $d_i$ not depending on $i$. As $d$ is in that sense fixed it does not have to be considered further. The $d_i'$ on the other hand are built only using addition, subtraction and multiplication. These operations are applied (nested) to elements of the active domain and elements come into existence throuth other constructions, like the characteristic function operator, which however all produce integers. It follows that the $d_i'$ can not contain any prime not already present in some denominator of a member of the active domain. However a number of primes exponential in the size of the database representation can not occur simultaneously in the denominators occuring in the active domain without one of these denominators being larger than the whole database it is part of.

# Bibliography

[1] S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*, Addison-Wesley (1995).

[2] M. Benedikt, L. Libkin, *Languages for Relational Databases over Interpreted Structures.*

[3] A. Chandra, D. Harel, *Computable Queries for Relational Databases*, Journal of Computer and System Sciences, **21** No. 2 (1980).

[4] J. Chomicki, G. Kuper, *Measuring Infinite Relations*, Proc. 14th ACM Symp. on Principles of Database Systems PODS (1995).

[5] K. Compton and E. Grädel, *Logical Definability of Counting Functions*, Journal of Computer and System Sciences **53** (1996), 283–297.

[6] E. Grädel and Y. Gurevich, *Metafinite Model Theory*, to appear in Information and Computation.

[7] S. Grumbach, J. Su, *Queries with Arithmetical Constraints*, Proc. on the Int. Conference on Principles and Practice of Constraint Programming (CP95)(1995).

[8] D. Johnson, *A Catalog of complexity Classes*, in: J. van Leeuven (Ed.), *Handbook of Theoretical Computer Science*, Vol. A: Algorithms and Complexity, Elsevier/MIT Press (1990), 67–161.

[9] P. Kanellakis, G. Kuper, P. Revesz, *Constraint Query Languates*, Proc. on the Int. Converence on Pricniples of Database systems (1990), 96–120.

[10] R. Karp and M. Luby, *Monte Carlo algorithms for enumeration and reliability problems*, Proceedings of the 24th Symposium on Foundations of Computer Science FOCS 1983, 56–64.

[11] V. Lakshmanan and F. Sadri, *Probabilistic deductive databases*, Proceedings of the International Logic Programming Symposium, MIT Press (1994), 197–207.

[12] V. Lakshmanan and V. Subrahmanian, *Probview, A flexible probabilistic database system*, ACM Transactions on Database Systems **22** (1997), 419–.

[13] L. Libkin, L. Wong, *Query Languages for Bags and Aggregate Functions.*

[14] C. Papadimitriou, *Computational Complexity*, Addison-Wesley (1994).

[15] K. Regan and T. Schwentick, *On the Power of One Bit of a* #P *Function*, Proceedings of the Fourth Italian Conference on Theoretical Computer Science (1992), 317–329.

[16] M. de Rougemont, *The reliability of queries*, Proc. 14th ACM Symp. on Principles of Database Systems PODS (1995), 286–291.

[17] S. Saluja, K. Subrahmanyam and M. Thakur, *Descriptive Complexity of* #P *Functions*, Journal of Computer and System Sciences **50** (1995), 493–505.

[18] A. Stolboushkin, M. Taitslin, *Linear vs. Order Constraint Queries Over Relational Databases*, Proc. 15th ACM Symp. on Principles of Database Systems PODS (1996).

[19] V. Subrahmanian, *Stable semantics for probabilistic deductive databases*, Information and Computation **110**(1) (1994), 42–.

[20] S. Toda, *On the computational power of PP and* $\oplus P$, Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (1989), 514-519.

[21] S. Toda, *PP is as hard as the polynomial-time hierarchy*, SIAM Journal on Computing **20** (1991), 865-877.

[22] J. Ullman, *Principles of Database and Knowledgebase Systems* **1**, Computer Science Press (1989).

[23] L. Valiant, *The complexity of enumeration and reliability problems*, SIAM J. Computing **8** (1979), 410–421.

[24] V. Vianu / S. Grumbach, L. Libkin, T. Milo, L. Wong, *Database Theory Column / Query Languages for Bags: Expressive Power and Complexity.*

[25] K. Wagner, *Some observations on the connection between counting and recursion*, Theoretical Computer Science **47** (1986),131–147.

[26] E. Zimányi, *Query evaluation on probabilistic relational databases*, Theoretical Computer Science **171** (1997), 179–219.