Vardi’s Law and Some Exceptions

The Gap between Data and Expression Complexity Reasoning Formalisms

Georg Gottlob
University of Oxford
& TU Wien

FUNDAMENTAL INSIGHT AND DEFINITIONS

- **Data Complexity**
- **Expression Complexity (a.k.a. Program Complexity)**
- **Combined Complexity**

THE COMPLEXITY OF RELATIONAL QUERY LANGUAGES

Extended Abstract

Moshe Y. Vardi†

Department of Computer Science
Stanford University
Stanford, California 94305

Abstract

Two complexity measures for query languages are proposed. *Data complexity* is the complexity of evaluating a query in the language as a function of the size of the database, and *expression complexity* is the complexity of evaluating a query in the language as a function of the size of the expression defining the query. We study the data and expression complexity of logical languages - relational calculus and its extensions by transitive closure, fixpoint and second order existential quantification - and algebraic languages - relational algebra and its extensions by bounded and unbounded looping. The pattern which will be shown is that the expression complexity of the investigated languages is one exponential higher then their data complexity, and for both types of complexity we show completeness in some complexity class.

1. Introduction

In the last years there has been a lot of interest in query languages for relational databases. Following Codd's pioneering work [Codd] on the relational calculus and algebra, a lot of work has been done on studying and comparing the expressive power of several query languages [AU,Ba,CH1,CH2,CH3,Chan,Coop,Pa]. The approach taken here is to compare query languages by investigating the complexity of evaluating queries in these languages.

There are three ways to measure the complexity of evaluating queries in a specific language. First, one can fix a specific query in the language and study the complexity of applying this query to arbitrary databases. The complexity is then given as a function of the size of the databases. We call this complexity *data complexity*.

Alternatively, one can fix a specific database and

Finally, one can study the complexity of applying queries represented by arbitrary expressions in the language to arbitrary databases. The complexity is then given as a function of the combined size of the expressions and the databases. We call this complexity *combined complexity*.

It turns out that combined complexity is pretty closed to expression complexity, and for this reason we

VARDI'S LAW

… This happens to be quite a typical pattern. The expression complexity of the investigated languages is usually one exponential higher than the data complexity…

This is indeed very often so, but let us see some exceptions

CASE 1: Double-Exponential Gap

Language: Guarded Datalog[∃]

Semantics: Chase (possibly infinite)

Problem: Query-of-Tuple Problem (QOT)

$\text{reports}(x,y) \;\&\; \text{consultant}(y) \rightarrow \exists z\,(\text{emp}(z) \wedge \text{reports}(y,z))$

CASE 1: Double-Exponential Gap

Language: Guarded Datalog[∃]

Semantics: Chase (possibly infinite)

Problem: Query-of-Tuple Problem (QOT)

Data complexity: PTIME-complete

Expression/combined Complexity: 2EXPTIME complete.

[Cali, G., Kifer, 2012]

CASE 2: Expression Complexity ≅ Data Complexity

Language: Monadic Datalog over trees

Semantics: Datalog semantics

Problem: Query-of-Tuple Problem (QOT)

Combined complexity: **O(|Program| x |Tree|)**

Data Complexity: in LINTIME (as expressive as MSO over trees)

Expression/Combined Complexity: in LINTIME, PTIME-complete.

[G., Koch 2012]

However, Vardi's Law is true in 95% of the languages I analysed.

Exception just confirm the rule,

Vardi's law is almost always true!

Thank you, Moshe, for your insights.