SUPERFINITENESS OF QUERY ANSWERS IN DEDUCTIVE DATABASES: AN AUTOMATA-THEORETIC APPROACH

Laks V.S. Lakshmanan* and Daniel A. Nonen

Dept. of Computer Science, Concordia University, Montreal, Quebec

Abstract: Deciding finiteness of query answers is a problem of fundamental importance to query systems supporting declarative logic based languages with function symbols. Unfortunately, this problem is undecidable in general. One of the recently proposed techniques for coping with this situation consists in (i) approximating the original program with function symbols by a datalog program with infinite base relations, together with finiteness constraints (FCs) acting on them (FCs say that if certain columns of a relation are finite, so are other columns), and (ii) using an approximation for finiteness, called superfiniteness. A query is finite if it has finite answers in the least fixpoint model of the program, whereas it is superfinite if it has a finite answer in every fixpoint model of the program. While superfiniteness is decidable, the only known procedure for it takes time exponential in the size of the program and associated constraints. The main contribution of this paper is the development of automata-theoretic techniques for superfiniteness analysis and polynomial time decision procedures for certain classes of linear programs in the presence of unary FCs.

1 Introduction

Deductive databases, with their increased expressive power over relational databases have been recognized as one of the important data models for next generation applications [T 91]. However, Datalog, the query language on which much of deductive database research is based, lacks the power of function symbols. Function symbols are very important for (i) better data structuring ability, (ii) applications involving streams or list constructors, and (iii) applications like temporal deductive databases (e.g., see [CI 88]). A query to a deductive database is expressed using a set of Horn clause rules, often called a query program. The query is then answered against the least fixpoint model (e.g., see [Ull 89]) of this program, which intuitively is the set of facts derived by the rules from the base facts in the database. In the presence of function symbols, answers to certain queries can be infinite. Detection of finite queries is fundamental to the design of query systems. Indeed, finiteness analysis is an integral part of such systems as NAIL! [Ull 89], LDL [Chi 89], and SYGRAF [KL 88]. Besides, recent works (e.g., see Brodsky and Sagiv [BS 89, BS 91], Sohn and van Gelder [SG 91]) also show applications of finiteness analysis to the detection

*This research was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada and in part by a grant from the Fonds Pour Formation De Chercheurs Et L'Aide À La Recherche of Québec.
of termination of top-down evaluation of logic query programs.

Shmueli [Sh 87] showed that query finiteness is in general undecidable for programs with function symbols. References to other decidability/undecidability results on finiteness for different classes of queries can be found in [KRS 88]. Ramakrishnan et al. [RBS 87] developed an elegant framework for finiteness analysis. They approximate programs with function symbols by function-free programs with infinite base relations satisfying finiteness constraints (FCs). Intuitively, FCs assert that if certain columns in a relation have a finite number of values, so will other column(s). For example, consider an infinite relation \( s(A, B, C) \). The relation \( s \) satisfies the FC \( AB \rightarrow C \) exactly when it associates a finite number of \( C \)-values with any given \( AB \)-value. Among other things [RBS 87] also showed that finiteness is decidable for monadic programs.

Sagiv and Vardi [SV 89] showed that finiteness (in this framework) can be viewed as the conjunction of a property called weak finiteness and termination. They showed that while weak finiteness is decidable, termination is in general undecidable. They also furnished a polynomial time algorithm for detecting finiteness for monadic programs. It should be noted that the decidability of finiteness in the presence of infinite base relations and FCs is still open [R 81].

Kifer et al. [KRS 88] proposed a stronger notion of finiteness called superfiniteness which refers to a query answer being finite in all fixpoint models of the program, as opposed to only in the least model. Intuitively, a model \( M \) of a program \( \Pi \) is a fixpoint model if for every fact in \( M \) there is a rule in \( \Pi \) that justifies this fact. It turns out that superfiniteness, which is stronger than finiteness, is decidable. Kifer et al. have developed a complete axiom system, and a decision procedure for superfiniteness. They also extend their procedure to detect finiteness for certain class of query programs. While their contribution is fundamental and significant, the time complexity of their algorithm (for superfiniteness) is exponential in the size of the input program and constraints 1.

The methodology for handling the finiteness problem that is embodied by the works [RBS 87, KRS 88] is to (i) approximate a given logic program with function symbols by a datalog program with infinite base relations together with FCs acting on them, and (ii) use superfiniteness as a sufficient condition for detecting finiteness. It would thus be desirable to have an efficient algorithm for detecting superfiniteness. The main motivation for this paper is the development of such an algorithm. A thorough understanding of superfiniteness from the standpoint of efficient detection is needed for developing polynomial time algorithms for this problem. Throughout the paper, we restrict attention to linear programs 2 with one idb predicate. We first develop a simple proof procedure using rule/goal (R/G) trees for reasoning about superfiniteness (Section 3). In addition to shedding some light on superfiniteness analysis, this procedure is useful in many of our proofs. We develop a notion of compositionality of (linear) programs and show that this property can be tested in polynomial time (Section 4). The significance of compositionality is that it characterizes programs (together with FCs) for which superfiniteness analysis can be performed by using the local information at the nodes of a R/G tree. Both Sections 5 and 6 assume unary FCs. In Section 5, we consider the class of compositional linear programs and develop an automata-theoretic technique for detecting superfiniteness of predicates defined by such programs. Our technique leads to a polynomial time decision procedure. In Section 6, we extend this technique for the class of linear single recursive rule programs.

---

1 The main concern of that paper was proving decidability and axiomatizability of superfiniteness.

2 These are programs in which each recursive rule has at most one subgoal mutually recursive with the head, in its body.