Querying Sequence Databases with Transducers

Anthony J. Bonner$^1$ and Giansalvatore Mecca$^2$

$^1$ University of Toronto – Department of Computer Science – Toronto, ON, Canada http://www.cs.toronto.edu/~bonner


Abstract. This paper develops a database query language called Transducer Datalog motivated by the needs of a new and emerging class of database applications. In these applications, such as text databases and genome databases, the storage and manipulation of long character sequences is a crucial feature. The issues involved in managing this kind of data are not addressed by traditional database systems, either in theory or in practice. To address these issues, in recent work, we introduced a new machine model called a generalized sequence transducer. These generalized transducers extend ordinary transducers by allowing them to invoke other transducers as "subroutines." This paper establishes the computational properties of Transducer Datalog, a query language based on this new machine model. In the process, we develop a hierarchy of time-complexity classes based on the Ackermann function. The lower levels of this hierarchy correspond to well-known complexity classes, such as polynomial time and hyper-exponential time. We establish a tight relationship between levels in this hierarchy and the depth of subroutine calls within Transducer Datalog programs. Finally, we show that Transducer Datalog programs of arbitrary depth express exactly the sequence functions computable in primitive-recursive time.

1 Introduction

This paper develops a database query language, called Transducer Datalog, motivated by the needs of a new and emerging class of database applications. In these applications, such as text databases and genome databases, the storage and manipulation of long sequences is a crucial feature [12, 10]. The issues involved in managing this kind of data are not addressed by traditional database systems, either in theory or in practice. In particular, traditional database query languages are poor at manipulating sequence data. This is because they treat data items as atomic entities without any internal structure. Thus, they cannot interrogate or manipulate the sequential structure of items such as documents, DNA sequences and protein sequences.

To address this limitation, sequence operations have been introduced into recent data models and query languages (e.g. [4, 7]). In many cases, however, the sequence operations are ad hoc and are not investigated in a theoretical framework. In other cases, (e.g. [17, 26]) the operations are designed for pattern
extraction but not for sequence restructuring. Although pattern recognition is a fundamental feature of any language for querying sequences, sequence restructurings are equally important. For example, in genome databases, one frequently needs to concatenate sequences together, to splice out selected subsequences, and to compute the reverse of a sequence.

Introducing sequence operations into database query languages presents interesting theoretical challenges. For instance, the resulting query language should be expressive both in terms of pattern matching and sequence restructurings. At the same time, it should be safe; that is, a query should always return a finite answer and terminate in a finite amount of time. Unlike traditional databases, safety and finiteness of computations are major problems in sequence databases, since sequences can grow to arbitrary length, and the set of possible sequences is infinite, even when the underlying alphabet—or domain—is finite. This means that, unlike traditional database queries, sequence queries can end up in non-terminating computations.

To achieve expressiveness, database researchers have developed sequence query languages based on abstract machines, such as automata or transducers. Unfortunately, the resulting languages are unsafe, and to achieve safety, syntactic restrictions have been imposed, restrictions that severely limit expressiveness (e.g., [11, 14]). To resolve these problems, we recently developed a query language for sequence data called Transducer Datalog [22, 6, 20], which extends classical Datalog [8] so that it can handle sequence data. A key element in Transducer Datalog is a new machine model called a generalized sequence transducer. Intuitively, a generalized transducer is a transducer that can invoke other transducers as "subroutines." These subtransducers can likewise invoke other transducers as sub-subroutines. Etc. Like ordinary transducers, generalized transducers always terminate. The result is a query language for sequence databases that is safe; moreover, because generalized transducers are more powerful than ordinary transducers, the query language has considerable expressive power, and its computational complexity can be tuned by restricting the transducers used in the language. These properties of the query language, which are a result of the computational properties of the generalized transducers, are studied in this paper.

By coupling a declarative query language, like Datalog, with a procedural model of computation, like generalized transducers, we have been able to develop a flexible paradigm for querying sequence databases. In Transducer Datalog, set-oriented database operations are easily expressed using a semantics based on Datalog, while powerful sequence manipulation procedures can be programmed using transducers. The language itself combines database query primitives with a powerful computational model for sequences, which is an important requirement for sequence databases. For example, in genome databases, because genometechnology is rapidly evolving, new sequence operations are constantly needed, operations that cannot be anticipated in advance. Genome databases thus need to combine a query language with arbitrary procedures for executing sequence operations [13]. Clearly, different models of computation could be embedded in