Journal of Computer and System Sciences 78 (2012) 1045–1098

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

The implication problem for 'closest node' functional dependencies in complete XML documents

M.W. Vincent [a,*], J. Liu [a], M. Mohania [b]

[a] University of South Australia, Adelaide, Australia
[b] IBM Research Laboratory, New Delhi, India

ARTICLE INFO

Article history:
Received 11 January 2009
Received in revised form 15 November 2011
Accepted 9 January 2012
Available online 25 January 2012

Keywords:
XML
DTD
Functional dependency
Implication
Axiom system

ABSTRACT

With the growing use of XML as a format for the permanent storage of data, the study of functional dependencies in XML (XFDs) is of fundamental importance in a number of areas such as understanding how to effectively design XML databases without redundancy or update problems, and data integration. In this article we investigate a particular type of XFD, called a *weak 'closest node' XFD*, that has been shown to extend the classical notion of a functional dependency in relational databases. More specifically, we investigate the implication problem for weak 'closest node' XFDs in the context of XML documents with no missing information. The implication problem is the most important one in dependency theory, and is the problem of determining if a set of dependencies logically implies another dependency. Our first, and main, contribution is to provide an axiom system for XFD implication. We prove that our axiom system is both sound and complete, and we then use this result to develop a sound and complete quadratic time closure algorithm for XFD implication. Our second contribution is to investigate the implication problem for XFDs in the presence of a Document Type Definition (DTD). We show that for a class of DTDs called *structured* DTDs, the implication problem for a set of XFDs and a structured DTD can be converted to the implication problem for a set of XFDs alone, and so is axiomatizable and efficiently solvable by the first contribution. We do this by augmenting the original set of XFDs with additional XFDs generated from the structure of the DTD.

1. Introduction

Integrity constraints are one of the oldest and most important topics in database research, and they find application in a variety of areas such as database design, data translation, query optimization and data storage [1]. With the adoption of the eXtensible Markup Language (XML) [2] as the industry standard for data interchange over the internet, and its increasing usage as a format for the permanent storage of data in database systems [3], the study of integrity constraints in XML has increased in importance in recent years (a recent survey of the topic has been given by Fan [4]). It has also been argued that integrity constraints in XML might be of even greater importance than integrity constraints in relational database systems [4]. This is because of the adoption of XML as the standard for electronic data interchange, and the importance of integrity constraints in specifying the semantics of data – a topic that has been identified as crucial in effective data integration [5].

While a number of different types of XML integrity constraints have been studied [4], the growing use of XML as a permanent storage format for data has recently motivated the study of *functional dependencies in XML* (XFDs), because of their central role in designing XML documents that avoid redundancy and update problems. The formal study of XFDs and

* Corresponding author.
E-mail addresses: millist.vincent@unisa.edu.au (M.W. Vincent), jixue.liu@unisa.edu.au (J. Liu), mkmukesh@in.ibm.com (M. Mohania).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2012.01.003

normalization in XML was initiated by Arenas and Libkin [6–8], who defined an XFD using the notion of a *'tree tuple'*, which in turn is an extension of the total unnesting operator used in nested relations [9]. An alternative approach to defining an XFD, based on the concept of a *'closest node'*, has been introduced by us in recent work [10]. Our approach has similarities with the one adopted by Buneman et al. in their study of keys in XML [11], and we have shown that a 'closest node' XFD generalizes an absolute XML key as defined by Buneman et al. for the case of simple XML paths [10]. While a 'closest node' XFD has the same syntax as a 'tree tuple' XFD, i.e. a set of simple paths on the l.h.s. of the XFD and a single simple path on the r.h.s. of the XFD, the 'closest node' approach differs from the 'tree tuple' approach in how one chooses the nodes in the tree representation of an XML document to test for XFD satisfaction. In the 'closest node' approach, nodes must satisfy a spatial property called 'closest', which in turn has been shown to be an extension of the property that two data values appear in the same tuple of a relation [12], whereas in the 'tree tuple' approach the nodes must belong to a tuple generated from the total unnest of the XML tree.

More recently [12], we have extended the study of 'closest node' XFDs, but using slightly different semantics to that originally proposed in [10].[1] To distinguish between these two approaches, we refer to an XFD as defined in [10] as a *strong* 'closest node' XFD, and that used in [12] and in this article as a *weak* 'closest node' XFD. This terminology is deliberate, since it will be shown later that if an XML document satisfies a 'closest node' XFD using strong semantics, then it also satisfies the XFD using weak semantics but the converse is not true. The motivation for defining a weak 'closest node' XFD was to preserve the semantics of an FD when a complete relation is mapped to an XML document, and we showed that if a complete relation is first mapped to a nested relation by an arbitrary sequence of nest operations [9], and then directly to an XML document, the resulting XML document satisfies a weak 'closest node' XFD if and only if the relation satisfies the corresponding FD [12]. In contrast, the motivation for the strong 'closest node' XFD definition was to extend the semantics of an XML key as proposed by Buneman et al. [11], in a similar fashion to the way that an FD extends the notion of a key in relational databases [1].

In general, it is not possible to compare a 'tree tuple' XFD to either a strong, or weak, 'closest node' XFD. This is because a *Document Type Definition* (DTD) [2] is mandatory in the 'tree tuple' approach, but optional in the 'closest node' approach, and because the two approaches use different semantics for missing information. However, the two approaches are comparable when a DTD is present and the XML document contains no missing information with respect to the paths of the DTD. In this case, the definition of a 'tree tuple' XFD has been shown to be equivalent to that of a weak 'closest node' XFD (but not in general to a strong 'closest node' XFD) [12].

In this article we focus on the *implication problem* for weak 'closest node' XFDs, that is determining if a set of XFDs logically implies another XFD. The implication problem is probably the most important one in dependency theory, irrespective of the data model or type of integrity constraint, since a solution to it lies at the heart of any automated procedure for reasoning about integrity constraints. In the context of 'tree tuple' XFDs, the implication problem has been investigated by Arenas and Libkin [6], and more recently by Kot and White [13]. Arenas and Libkin showed that the implication problem was efficiently decidable for restricted classes of DTDs, but was intractable for more general classes of DTDs. Kot and White developed a variant of the classical relational chase algorithm [1] to solve the XFD implication problem, both with and without a DTD. They then used their chase algorithm to develop sound and complete axiom systems for several classes of DTDs.

1.1. Approach

In this section we outline the general features of our approach. First, our definition of a weak 'closest node' XFD is the same as that given by us in related work [12]. In this approach a DTD is optional, and we investigate the implication problem for XFDs both in XML documents without a DTD, and in the presence of a new type of DTD that we call a *structured* DTD, which is a special case of a *simple* DTD as defined by Arenas and Libkin [6]. Our approach is similar to that of Kot and White [13] just mentioned in that they also consider XFD implication both in the presence and absence of a DTD, but differs from the approach of Arenas and Libkin who only consider XFD implication in the presence of a DTD. However, we only consider structured DTDs, rather than the more general classes of DTDs considered by both Arenas and Libkin and Kot and White, since it is the most appropriate class for the applications of XML that are the focus of this article, namely *data-centric* applications[14–17]. We shall discuss this point in more detail in Section 6.

Another feature of our approach is that we consider the XFD implication problem in the context of a new subclass of XML documents that we call *complete* XML documents,[2] which differs from the approach of Arenas and Libkin, and Kot and White, who allow incomplete XML documents. Intuitively, a complete XML document is one with 'no missing information' and is an extension of the notion of a complete relation. We now outline our motivation for investigating the implication problem in this context.

While one of the goals in the design of XML was to explicitly cater for irregularly structured data, XML is also being widely used in more traditional business applications involving regularly structured data, often referred to as data-centric XML [14–17]. For example, a recent survey of several hundred large companies in the U.S. found that around 70% were now using XML enabled databases, or native XML databases, for their core data processing applications [18]. In this setting,

[1] A detailed discussion of the differences between the two definitions will be given in a later section.

[2] A precise definition will be given later.