On XML Integrity Constraints in the Presence of DTDs

Wenfei Fan
Bell Labs and Temple University
600 Mountain Avenue
Murray Hill, NJ 07974, USA
wenfei@research.bell-labs.com

Leonid Libkin*
Department of Computer Science
University of Toronto
Toronto ON M5S 3H5, Canada
libkin@cs.toronto.edu

Abstract

The paper investigates XML document specifications with DTDs and integrity constraints, such as keys and foreign keys. We study the consistency problem of checking whether a given specification is meaningful: that is, whether there exists an XML document that both conforms to the DTD and satisfies the constraints. We show that DTDs interact with constraints in a highly intricate way and as a result, the consistency problem in general is undecidable. When it comes to unary keys and foreign keys, the consistency problem is shown to be NP-complete. This is done by coding DTDs and integrity constraints with linear constraints on the integers. We consider the variations of the problem (by both restricting and enlarging the class of constraints), and identify a number of tractable cases, as well as a number of additional NP-complete ones. By incorporating negations of constraints, we establish complexity bounds on the implication problem, which is shown to be coNP-complete for unary keys and foreign keys.

1 Introduction

Although a number of dependency formalisms were developed for relational databases, functional and inclusion dependencies are the ones used most often. More precisely, only two subclasses of functional and inclusion dependencies, namely, keys and foreign keys, are commonly found in practice. Both are fundamental to conceptual database design, and are supported by the SQL standard [23]. They provide a mechanism by which one can uniquely identify a tuple in a relation and refer to a tuple from another relation. They have proved useful in update anomaly prevention, query optimization and index design [1, 30].

XML (eXtensible Markup Language [6]) has become the prime standard for data exchange on the Web. XML data typically originates in databases. If XML is to represent data currently residing in databases, it should support keys and foreign keys, which are an essential part of the semantics of the data. A number of key and foreign key specifications have been proposed for XML, e.g., the XML standard (DTD) [6], XML Data [21] and XML Schema [29]. Keys and foreign keys for XML are important in, among other things, query optimization [27], data integration [16], and in data exchange for converting databases to an XML encoding.

XML data usually comes with a DTD that specifies how a document is organized. Thus, a specification of an XML document may consist of both a DTD and a set of integrity constraints, such as keys and foreign keys. A legitimate question then is whether such a specification is *consistent*, or meaningful: that is, whether there exists a (finite) XML document that both satisfies the constraints and conforms to the DTD.

In the relational database setting, such a question would have a trivial answer: one can write arbitrary (`primary`) `key` and `foreign key` specifications in SQL, without worrying about consistency. However, DTDs (and other schema specifications for XML) are more complex than relational schemas: in fact, XML documents are typically modeled as node-labeled trees, e.g., in XSL [11, 31], XQL [28], XML Schema [29], XPath [12] and DOM [3]. Consequently, DTDs may interact with keys and foreign keys in a rather nontrivial way, as will be seen shortly. Thus, we shall study the following family of problems, where $\mathcal{C}$ ranges over classes of integrity constraints:

XML SPECIFICATION CONSISTENCY ($\mathcal{C}$)

INPUT:	A DTD D, a set Σ of $\mathcal{C}$-constraints.
QUESTION:	Is there an XML document that conforms to D and satisfies Σ?

*Research affiliation: Bell Laboratories.

Throughout the paper, we only consider *finite* documents (trees). We shall study the following four classes of constraints:

- $\mathcal{C}_{K,FK}$: a class of keys and foreign keys defined in terms of XML attributes;
- $\mathcal{C}^{Unary}_{K,FK}$: unary keys and foreign keys in $\mathcal{C}_{K,FK}$, i.e., those defined in terms of a single attribute;
- $\mathcal{C}^{Unary}_{K^{\neg},IC}$: unary keys, unary inclusion constraints and negations of unary keys;
- $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$: unary keys, unary inclusion constraints and their negations.

It should be mentioned that unary keys and foreign keys considered in this paper are similar to but more general than XML ID and IDREF specifications.

The complement of a special case of the consistency problem for $\mathcal{C}^{Unary}_{K^{\neg},IC}$ (resp. $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$) is the *implication problem*: given any DTD D and any finite set Σ of unary keys and inclusion constraints, whether is it the case that all XML trees satisfying Σ and conforming to D must also satisfy some other unary key (resp. unary key or inclusion constraint)? This question is important in, among other things, data integration. For example, one may want to know whether a constraint φ holds in a mediator interface, which may use XML as a uniform data format [4, 26]. This cannot be verified directly since the mediator interface does not contain data. One way to verify φ is to show that it is implied by constraints that are known to hold [16].

These problems, however, turn out to be far more intriguing than their counterparts in relational databases. In the XML setting, DTDs do interact with keys and foreign keys, and this interaction may lead to problems with XML specifications.

Examples. To illustrate the interaction between XML DTDs and key/foreign key constraints, consider a DTD D_1, which specifies a (nonempty) collection of teachers:

```
<!ELEMENT  teachers  (teacher+)>
<!ELEMENT  teacher   (teach, research)>
<!ELEMENT  teach     (subject, subject)>
```

It says that a teacher teaches two subjects. Here we omit the descriptions of elements whose type is string (e.g., PCDATA in XML).

Assume that each teacher has an attribute `name` and each subject has an attribute `taught_by`. Attributes are single-valued. That is, if an attribute l is defined for an element type τ in a DTD, then in a document conforming to the DTD, each element of type τ must have a unique l attribute with a string value. Consider

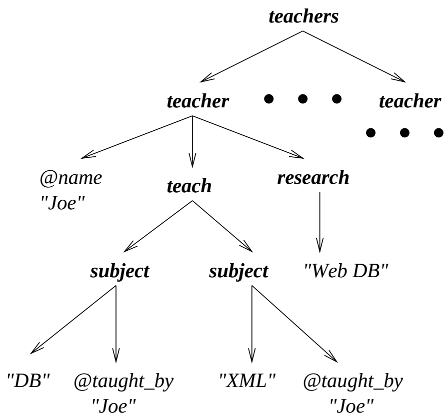

Figure 1: An XML tree conforming to D_1

a set of unary key and foreign key constraints, Σ_1:

$$\begin{aligned} teacher.name &\rightarrow teacher, \\ subject.taught\_by &\rightarrow subject, \\ subject.taught\_by &\subseteq teacher.name. \end{aligned}$$

That is, `name` is a key of `teacher` elements, `taught_by` is a key of `subject` elements and it is also a foreign key referencing `name` of `teacher` elements. More specifically, referring to an XML tree T, the first constraint asserts that two distinct `teacher` nodes in T cannot have the same `name` attribute value: the (string) value of `name` attribute uniquely identifies a `teacher` node. It should be mentioned that two notions of equality are used in the definition of keys: we assume string *value* equality when comparing `name` attribute values, and *node* identity when it comes to comparing `teacher` elements. The second key states that `taught_by` attribute uniquely identifies a `subject` node in T. The third constraint asserts that for any `subject` node x, there is a `teacher` node y in T such that the `taught_by` attribute value of x equals to the `name` attribute value of y. Since `name` is a key of `teacher`, the `taught_by` attribute of any `subject` node refers to a `teacher` node.

Obviously, there exists an XML tree conforming to D_1, as shown in Figure 1. However, there is no XML tree that both conforms to D_1 and satisfies Σ_1. To see this, let us first define some notations. Given an XML tree T and an element type τ, we use $ext(\tau)$ to denote the set of all the nodes labeled τ in T. Similarly, given an attribute l of τ, we use $ext(\tau.l)$ to denote the set of l attribute values of all τ elements. Then immediately from Σ_1 follows a set of dependencies:

$$\begin{aligned} |ext(teacher.name)| &= |ext(teacher)|, \\ |ext(subject.taught\_by)| &= |ext(subject)|, \\ |ext(subject.taught\_by)| &\leq |ext(teacher.name)|, \end{aligned}$$

where $|\cdot|$ is the cardinality of a set. Therefore, we have

$$|ext(subject)| \leq |ext(teacher)|. \qquad (1)$$

On the other hand, the DTD D_1 requires that each teacher must teach two subjects. Since no sharing of nodes is allowed in XML trees and the collection of `teacher` elements is nonempty, from D_1 follows:

$$1 \ < \ 2\,|ext(teacher)| = |ext(subject)|. \qquad (2)$$

Thus $|ext(teacher)| < |ext(subject)|$. Obviously, (1) and (2) contradict with each other and therefore, there exists no XML tree that both satisfies Σ_1 and conforms to D_1. In particular, the XML tree in Figure 1 violates the key $subject.taught\_by \rightarrow subject$.

This example demonstrates that a DTD may impose dependencies on the cardinalities of certain sets of objects in XML trees. These *cardinality constraints* interact with keys and foreign keys. More specifically, keys and foreign keys enforce classes of cardinality constraints that interact with those imposed by DTD. This makes the consistency analysis of keys and foreign keys for XML far more intriguing than that for relational databases. Because of the interaction, simple key and foreign key constraints (e.g., Σ_1) may not be satisfiable by XML trees conforming to certain DTDs (e.g., D_1).

Note that some XML DTDs do not have finite XML trees conforming to them even in the absence of keys and foreign keys. For instance, there exists no finite tree conforming to the DTD D_2 given below:

```
<!ELEMENT  db  (foo)>
<!ELEMENT  foo (foo)>
```

Contributions. The main contributions of the paper are the following:

1. For the class $\mathcal{C}_{K,FK}$ of keys and foreign keys, we show that both the consistency and the implication problems are undecidable.
2. These negative results suggest that we look at the restriction $\mathcal{C}^{Unary}_{K,FK}$ of *unary* keys and foreign keys (which are most typical in XML documents). We provide a coding of DTDs and these unary constraints by linear constraints on the integers. This enables us to show that the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ (even under the restriction to primary keys) is NP-complete. We further show that the problem is still in NP for an extension $\mathcal{C}^{Unary}_{K^{\neg},IC}$, which also allows negations of key constraints.
3. Using a different coding of constraints, we show that the consistency problem remains in NP for $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$, the class of unary keys, unary inclusion constraints and their negations. Among other things, this shows that the implication problem for unary keys and unary foreign keys is coNP-complete.
4. We also identify several tractable cases of the consistency problem, i.e., practical situations where the consistency problem is decidable in PTIME.

The undecidability of the consistency problem contrasts sharply with its trivial counterpart in relational databases. The coding of DTDs and unary constraints with linear integer constraints reveals some insight into the interaction between DTDs and unary constraints. Moreover, it allows us to use the techniques from linear integer programming in the study of XML constraints.

It should be mentioned that the undecidability and NP-hardness results carry over to other schema and constraint specifications for XML, e.g., XML Schema.

Related work. Keys, foreign keys and the more general inclusion and functional dependencies have been well studied for relational databases (cf. [1]). In particular, the implication problem for unary inclusion and functional dependencies is in linear time [13]. In contrast, we shall show that the XML counterpart of this problem is coNP-complete.

Key and foreign key specifications for XML have been proposed in the XML standard [6], XML Data [21] and XML Schema [29]. The need for studying XML constraints has also been advocated in [32]. DTDs in the XML standard allow one to specify limited (primary) unary keys and foreign keys with ID and IDREF attributes. However, they are not scoped: one has no control over what IDREF attributes point to. XML Data and XML Schema support more expressive specifications for keys and foreign keys with, e.g., XPath expressions. However, the consistency problems associated with constraints defined in these languages have not been studied. We consider simple XML keys and foreign keys in this paper to focus on the nature of the interaction between DTDs and constraints. The implication problem for a class of keys and foreign keys was investigated in [15], but in the absence of DTDs (in a graph model for XML), which trivializes the consistency analysis. To the best of our knowledge, no previous work has considered the interaction between DTDs, and keys and foreign keys for XML (in the tree model).

A variety of path constraints have been studied for semistructured and XML data [2, 9]. The interaction between path constraints and database schemas was investigated in [8]. Path constraints specify inclusions among certain sets of objects in edge-labeled graphs, and are not capable of expressing keys. Various generalizations of functional dependencies have also been studied, see, for example, [18, 19]. But these generalizations were investigated in database settings, which are quite different from the tree model for XML data considered in this paper. Moreover, they cannot express foreign keys.

Organization. The rest of the paper is organized as follows. Section 2 defines four classes of XML constraints, namely, $\mathcal{C}_{K,FK}$, $\mathcal{C}^{Unary}_{K,FK}$, $\mathcal{C}^{Unary}_{K^{\neg},IC}$ and $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$. Section 3 establishes the undecidability of the consistency problem for $\mathcal{C}_{K,FK}$, the class of keys and foreign keys. Section 4 provides an encoding for DTDs and unary constraints with linear equalities and inequalities, and shows that the consistency problems are NP-complete for $\mathcal{C}^{Unary}_{K,FK}$ and $\mathcal{C}^{Unary}_{K^{\neg},IC}$. Section 5 further shows that the problem remains in NP for $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$, the class of unary keys, inclusion constraints and their negations. Section 6 summarizes the main results of the paper and identifies directions for further work. All the proofs are given in the full version of the paper [14].

2 DTDs, keys and foreign keys

In this section, we first present a formalism of XML DTDs [6] and the XML tree model. We then define four classes of XML constraints.

2.1 DTDs and XML trees

We extend the usual formalism of DTDs (as extended context free grammars [5, 10, 24]) by incorporating attributes.

Definition 2.1: A *DTD* (Document Type Definition) is defined to be $D = (E, A, P, R, r)$, where:

- E is a finite set of *element types*;
- A is a finite set of *attributes*, disjoint from E;
- P is a mapping from E to *element type definitions*: $P(\tau)$ is a regular expression α defined as follows:

  $$\alpha \ ::= \ \mathtt{S} \ | \ \tau' \ | \ \epsilon \ | \ \alpha|\alpha \ | \ \alpha,\alpha \ | \ \alpha^*$$

  where $\mathtt{S}$ denotes *string* type, $\tau' \in E$, ϵ is the empty sequence, and "|", "," and "*" denote union, concatenation, and the Kleene closure, respectively;
- R is a mapping from E to $\mathcal{P}(A)$, the power-set of A; if $l \in R(\tau)$ then we say l is *defined for* τ;
- $r \in E$ and is called *the element type of the root*.

We normally denote element types by τ and attributes by l. Without loss of generality, assume that r does not occur in $P(\tau)$ for any $\tau \in E$. We also assume that each τ in E is *connected to* r, i.e., either τ occurs in $P(r)$, or it occurs in $P(\tau')$ for some τ' that is connected to r. □

We consider single-valued attributes only. That is, if $l \in R(\tau)$ then every element of type τ has a unique l attribute and the value of the l attribute is a string.

As an example, let us consider the `teacher` DTD D_1 given in Section 1. In our formalism, D_1 can be represented as $(E_1, A_1, P_1, R_1, r_1)$, where

$$E_1 = \{teachers, teacher, teach, research, subject\}$$

$$A_1 = \{name, taught\_by\}$$

$$\begin{aligned}
P_1(teachers) &= teacher, teacher^* \\
P_1(teacher) &= teach, research \\
P_1(teach) &= subject, subject \\
P_1(subject) &= P_1(research) = \mathtt{S}
\end{aligned}$$

$$\begin{aligned}
R_1(teacher) &= \{name\} \\
R_1(subject) &= \{taught_by\} \\
R_1(teachers) &= R_1(teach) = R_1(research) = \emptyset
\end{aligned}$$

$$r_1 = teachers$$

An XML document is typically modeled as a node-labeled ordered tree. Given a DTD, we define the notion of documents that conform to it as follows.

Definition 2.2: Let $D = (E, A, P, R, r)$ be a DTD. An *XML tree T valid w.r.t. D* (*conforming to D*) is defined to be $T = (V, lab, ele, att, val, root)$, where

- V is a finite set of *vertices* (*nodes*);
- lab is a mapping from V to $E \cup A \cup \{\mathtt{S}\}$;
- ele is a partial function from V to sequences of V vertices such that for any $v \in V$, $ele(v)$ is defined iff $lab(v) = \tau$ and $\tau \in E$, and moreover, if $P(\tau)$ is α and $ele(v) = [v_1, ..., v_n]$, then $lab(v_1)...lab(v_n)$ must be in the regular language defined by α;
- att is a partial function from $V \times A$ to V such that for any $v \in V$ and $l \in A$, $att(v, l)$ is defined iff $lab(v) = \tau$, $\tau \in E$ and $l \in R(\tau)$;
- val is a partial function from V to string values such that for any node $v \in V$, $val(v)$ is a string iff $lab(v) = \mathtt{S}$ or $lab(v) \in A$;
- $root$ is a distinguished vertex in V and is called *the root of T*. Without loss of generality, assume $lab(root) = r$ and in addition, that there is a unique node in T labeled r.

For any node $v \in V$, if $ele(v)$ is defined then the nodes v' in $ele(v)$ are called the *subelements* of v. For any $l \in A$, if $att(v, l) = v'$ then v' is called *an attribute* of v. In either case we say that there is a *parent-child edge* from v to v'. The subelements and attributes of v are called its *children*. An XML tree has a tree structure, i.e., for each $v \in V$, there is a unique path of parent-child edges from root r to v. We write $T \models D$ when T is valid w.r.t. D. □

Intuitively, V is the set of vertices of the tree T. The mapping lab labels every node of V with a symbol from $E \cup A \cup \{\mathtt{S}\}$. Vertices labeled with element types of E are internal nodes of T, and those labeled $\mathtt{S}$ or attributes of A are leaves. If a node x is labeled τ in E, then the functions ele and att define the children of x, which are partitioned into *subelements* and *attributes* according to $P(\tau)$ and $R(\tau)$ in DTD D. The subelements of node x are ordered and their labels observe the regular expression $P(\tau)$. In contrast, its attributes are unordered and are identified by their labels (names). The function val

assigns string values to attributes and to nodes labeled S. Since T has a tree structure, sharing of nodes is not allowed in T.

In this paper, we only consider *finite* XML trees, i.e., XML trees with a finite set of vertices.

For example, Figure 1 depicts an XML tree valid w.r.t. the DTD D_1 given in Section 1.

We need the following notations: for any $\tau \in E \cup \{\mathtt{S}\}$, $ext(\tau)$ denotes the set of all the nodes in T labeled τ. For any node x in T labeled by τ and for any attribute $l \in R(\tau)$, we write $x.l$ for $val(att(x, l))$, i.e., the *value* of the attribute l of node x. We define $ext(\tau.l)$ to be $\{x.l \mid x \in ext(\tau)\}$, which is a set of strings. For each τ element x in T and a sequence $X = [l_1, \ldots, l_n]$ of attributes in $R(\tau)$, we use $x[X]$ to denote the sequence of X-attribute values of x, i.e., $x[X] = [x.l_1, \ldots, x.l_n]$. For a set S, $|S|$ denotes its cardinality.

2.2 XML constraints

We next define our constraint languages for XML. We begin with the class of multi-attribute keys and foreign keys, denoted by $\mathcal{C}_{K,FK}$.

Let $D = (E, A, P, R, r)$ be a DTD. A *constraint* φ *of* $\mathcal{C}_{K,FK}$ *over* D has one of the following forms:

- *key:* $\tau[X] \to \tau$, where $\tau \in E$ and X is a set of attributes in $R(\tau)$. It indicates that the set X of attributes is a key of elements of τ.
- *foreign key:* $\tau_1[X] \subseteq \tau_2[Y]$ and $\tau_2[Y] \to \tau_2$, where $\tau_1, \tau_2 \in E$, X, Y are nonempty sequences of attributes in $R(\tau_1)$, $R(\tau_2)$, respectively, and moreover, X and Y have the same length. This constraint indicates that X is a foreign key of τ_1 elements referencing key Y of τ_2 elements.

A constraint of the form $\tau_1[X] \subseteq \tau_2[Y]$ is called an *inclusion constraint.* Observe that a foreign key is actually a pair of constraints, namely, an inclusion constraint $\tau_1[X] \subseteq \tau_2[Y]$ and a key $\tau_2[Y] \to \tau_2$. Note that inclusion constraints do not require the presence of keys.

To illustrate keys and foreign keys of $\mathcal{C}_{K,FK}$, let us consider a DTD $D_3 = (E_3, A_3, P_3, R_3, r_3)$, where

$$
\begin{aligned}
E_3 &= \{school, student, course, enroll, name, subject\}\\
A_3 &= \{student\_id, course\_no, dept\}\\
P_3(school) &= course^*, student^*, enroll^*\\
P_3(course) &= subject\\
P_3(student) &= name\\
P_3(enroll) &= P_3(name) = P_3(subject) = \mathtt{S}\\
R_3(course) &= \{dept, course\_no\}\\
R_3(student) &= \{student\_id\}\\
R_3(enroll) &= \{student\_id, dept, course\_no\}\\
R_3(school) &= R_3(name) = R_3(subject) = \emptyset\\
r_3 &= school
\end{aligned}
$$

Typical $\mathcal{C}_{K,FK}$ constraints over D_3 include:

$$
\begin{array}{ll}
(1) & student[student\_id] \to student,\\
(2) & course[dept, course\_no] \to course,\\
(3) & enroll[student\_id, dept, course\_no] \to enroll,\\
(4) & enroll[student\_id] \subseteq student[student\_id],\\
(5) & enroll[dept, course\_no] \subseteq course[dept, course\_no].
\end{array}
$$

The first three constraints are keys in $\mathcal{C}_{K,FK}$, the last two are inclusion constraints, and the pairs (4, 1) and (5, 2) are foreign keys in $\mathcal{C}_{K,FK}$.

An XML tree T *satisfies* a $\mathcal{C}_{K,FK}$ constraint φ, denoted by $T \models \varphi$, iff

- if φ is a key $\tau[X] \to \tau$, then in T,

  $$\forall\, x\, y \in ext(\tau)\ (\bigwedge_{l \in X} (x.l = y.l) \to x = y).$$

  That is, two distinct τ nodes in T cannot have the same X-attribute values;
- if φ is a foreign key consisting of $\tau_1[X] \subseteq \tau_2[Y]$ and $\tau_2[Y] \to \tau_2$, then $T \models \tau_2[Y] \to \tau_2$ and

  $$\forall\, x \in ext(\tau_1)\ \exists\, y \in ext(\tau_2)\ (x[X] = y[Y]).$$

  That is, the sequence of X-attribute values of every τ_1 node in T must match the sequence of Y-attribute values of some τ_2 node in T. In addition, Y is a key of τ_2.

Two notions of equality are used to define keys: string value equality is assumed in $x.l = y.l$ (when comparing attribute values), and $x = y$ is true if and only if x and y are the same node (when comparing elements). This is different from the semantics of keys in relational databases.

It should be noted that given any DTD D, there are finitely many $\mathcal{C}_{K,FK}$ constraints over D.

The class of unary keys and foreign keys for XML, denoted by $\mathcal{C}^{Unary}_{K,FK}$, is a sublanguage of $\mathcal{C}_{K,FK}$. A $\mathcal{C}^{Unary}_{K,FK}$ constraint is a $\mathcal{C}_{K,FK}$ constraint defined with a single attribute. More specifically, a *constraint* φ *of* $\mathcal{C}^{Unary}_{K,FK}$ over DTD D is either

- *key:* $\tau.l \to \tau$, where $\tau \in E$ and $l \in R(\tau)$; or
- *foreign key:* $\tau_1.l_1 \subseteq \tau_2.l_2$ and $\tau_2.l_2 \to \tau_2$, where $\tau_1, \tau_2 \in E$, $l_1 \in R(\tau_1)$, and $l_2 \in R(\tau_2)$.

For example, the constraints of Σ_1 given in Section 1 are $\mathcal{C}^{Unary}_{K,FK}$ constraints over the DTD D_1.

A *unary inclusion constraint* is a constraint of the form $\tau_1.l_1 \subseteq \tau_2.l_2$. With unary inclusion constraints we define two extensions of $\mathcal{C}^{Unary}_{K,FK}$ as follows. One is $\mathcal{C}^{Unary}_{K^\neg,IC}$, the class consisting of unary keys, unary inclusion constraints and negations of unary keys. The other, $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$, consists of unary keys, unary inclusion constraints and their negations.

Finally, we describe the consistency and implication problems associated with XML constraints. Let $\mathcal{C}$ be one of $\mathcal{C}_{K,FK}$, $\mathcal{C}^{Unary}_{K,FK}$, $\mathcal{C}^{Unary}_{K^\neg,IC}$ or $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$, D a DTD, Σ a set of $\mathcal{C}$ constraints over D and T an XML tree valid w.r.t. D. We write $T \models \Sigma$ when $T \models \phi$ for all $\phi \in \Sigma$. Let φ be another $\mathcal{C}$ constraint. We say that Σ *implies* φ over D, denoted by $(D, \Sigma) \vdash \varphi$, if for any XML tree T such that $T \models D$ and $T \models \Sigma$, it must be the case that $T \models \varphi$. It should be noted when φ is a foreign key, φ consists of an inclusion constraint ϕ_1 and a key ϕ_2. In this case $(D, \Sigma) \vdash \varphi$ in fact means that $(D, \Sigma) \vdash \phi_1 \wedge \phi_2$.

The central technical problem investigated in this paper is the *consistency problem*. The consistency problem for $\mathcal{C}$ is to determine, given any DTD D and any set Σ of $\mathcal{C}$ constraints over D, whether there is an XML tree T such that $T \models \Sigma$ and $T \models D$.

The *implication problem* for $\mathcal{C}$ is to determine, given any DTD D and any set $\Sigma \cup \{\varphi\}$ of keys and foreign keys of $\mathcal{C}$ over D, whether $(D, \Sigma) \vdash \varphi$.

3 General keys and foreign keys

In this section we study $\mathcal{C}_{K,FK}$, the class of multi-attribute keys and foreign keys. Our main result is negative:

Theorem 3.1: *The consistency problem for $\mathcal{C}_{K,FK}$ constraints is undecidable.* □

Proof sketch: The proof consists of two steps. First, we show that in relational databases, the implication problem for keys by keys and foreign keys is undecidable. That is the problem to determine, given a relational schema $\mathbf{R}$, a set Σ of keys and foreign keys over $\mathbf{R}$ and a key φ, whether $\Sigma \vdash \varphi$. This can be verified by reduction from implication problem for functional and inclusion dependencies, which is undecidable (see, e.g., [1]). Second, we provide a reduction from (the complement of) the implication problem to the consistency problem for $\mathcal{C}_{K,FK}$ constraints. More specifically, let $\mathbf{R}$ = $(R_1, \ldots, R_n)$ be a relational schema, Θ be a set of keys and foreign keys over $\mathbf{R}$, and $\varphi = R[X] \to R$ be a key over $\mathbf{R}$, where R is R_s for some $s \in [1, n]$. Let $Y = Att(R) \setminus X$, where $Att(R)$ denotes the set of all attributes of R. We encode $\mathbf{R}$, Θ and φ in terms of a DTD D and a set Σ of $\mathcal{C}_{K,FK}$ constraints over D as follows. Let $D = (E,\ A,\ P,\ R_A,\ r)$, where

$$E = \{R_i \mid i \in [1,n]\} \cup \{t_i \mid i \in [1,n]\} \cup \{r,\ D_Y,\ E_X\}$$

$$A = \bigcup_{i\in[1,n]} Att(R_i)$$

$$\begin{array}{lll}
P(r) & = R_1, \ldots, R_n, D_Y, D_Y, E_X & \\
P(R_i) & = t_i^* & \text{for } i \in [1,n] \\
P(t_i) & = \epsilon & \text{for } i \in [1,n] \\
P(D_Y) & = P(E_X) = \epsilon & \\
R_A(t_i) & = Att(R_i) & \text{for } i \in [1,n] \\
R_A(D_Y) & = X \cup Y & \\
R_A(E_X) & = X & \\
R_A(r) & = R_A(R_i) = \emptyset & \text{for } i \in [1,n]
\end{array}$$

In particular, we denote $P(R) = t_\varphi^*$ for the relation R in φ. Note that $R = R_s$ and $t_\varphi = t_s$ for some $s \in [1, n]$.

We encode Θ and φ with $\Sigma = \Sigma_\Theta \cup \Sigma_\varphi$, where Σ_Θ is defined as follows:

(1) For every key $R_i[Z] \to R_i$ in Θ, $t_i[Z] \to t_i$ is in Σ_Θ.

(2) For any foreign key $R_i[Z] \subseteq R_j[Z']$ and $R_j[Z'] \to R_j$ in Θ, Σ_Θ includes $t_i[Z] \subseteq t_j[Z']$ and $t_j[Z'] \to t_j$.

The set Σ_φ consists of the following:

$$\begin{array}{lll}
D_Y[Y] \to D_Y, & E_X[X] \to E_X, & t_\varphi[XY] \to t_\varphi, \\
D_Y[X] \subseteq E_X[X], & D_Y[X,Y] \subseteq t_\varphi[X,Y], &
\end{array}$$

where $[X, Y]$ denotes the concatenation of lists X and Y, and t_φ is the grammar symbol in $P(R) = t_\varphi^*$. Note that $Att(R) = X \cup Y$ and thus XY is a key of t_φ.

As depicted in Figure 2, in any XML tree valid w.r.t. D, there are two distinct D_Y nodes d_1 and d_2 that have all the attributes in $X \cup Y$, and a single E_X node that has all attributes in X. If $T \models \Sigma_\varphi$, then

- $d_1[X] = d_2[X]$ by $D_Y[X] \subseteq E_X[X]$ and the fact $|ext(E_X)| = 1$,
- $d_1[Y] \neq d_2[Y]$ by $D_Y[Y] \to D_Y$.

These nodes will serve as a witness for $\neg\varphi$. Given these, we can show that $\bigwedge \Theta \wedge \neg\varphi$ can be satisfied by an instance of $\mathbf{R}$ if and only if Σ can be satisfied by an XML tree valid w.r.t. D. See [14] for the detailed proof. □

We next consider the implication problem.

Lemma 3.2: *Let D be a DTD, Σ be any set of $\mathcal{C}_{K,FK}$ constraints over D, φ_1 be any unary key and φ_2 be any unary inclusion constraint, then the following problems are undecidable: (1) $(D,\Sigma) \vdash \varphi_1$; (2) $(D,\Sigma) \vdash \varphi_2$.* □

Proof sketch: It suffices to establish the undecidability of the complements of the implication problems. This can be done by reduction from the consistency problem for $\mathcal{C}_{K,FK}$ (see [14]). □

From Lemma 3.2 we immediately obtain:

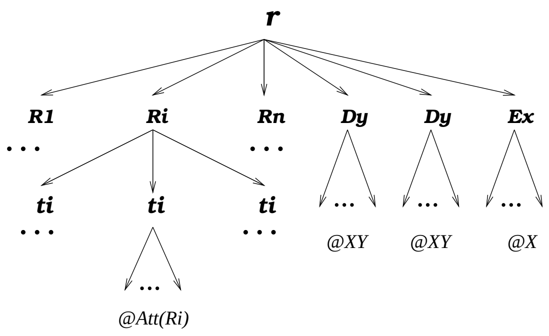

Figure 2: A tree used in the proof of Theorem 3.1

Corollary 3.3: *For $\mathcal{C}_{K,FK}$ constraints, the implication problem is undecidable.* □

While the general consistency and implication problems are undecidable, it is possible to identify some decidable cases of low complexity. The first one is checking whether a DTD has an XML tree valid w.r.t. it. This is a special case of the consistency problem, namely, when the given set of $\mathcal{C}_{K,FK}$ constraints is empty. A more interesting special case of the consistency problem is the *consistency problem for keys in $\mathcal{C}_{K,FK}$*. That is to determine, given any DTD D and any set Σ of keys in $\mathcal{C}_{K,FK}$ over D, whether there exists an XML tree valid w.r.t. D and satisfying Σ. Similarly, we consider the *implication problem for keys in $\mathcal{C}_{K,FK}$*: given any DTD D and any set $\Sigma \cup \{\varphi\}$ of keys in $\mathcal{C}_{K,FK}$ over D, whether $(D, \Sigma) \vdash \varphi$. The next theorem tells that all these cases are decidable (see [14] for the proof).

Theorem 3.4: *The following problems are decidable in linear time:*

1. *Given any DTD D, whether there exists an XML tree valid w.r.t. D.*
2. *The consistency problem for keys in $\mathcal{C}_{K,FK}$.*
3. *The implication problem for keys in $\mathcal{C}_{K,FK}$.*

□

Given Theorem 3.4, one would be tempted to think that when only foreign keys are considered, the analyses of consistency and implication could also be simpler. However, it is not the case. Recall that a foreign key of $\mathcal{C}_{K,FK}$ consists of an inclusion constraint and a key. Thus we cannot exclude keys in the presence of foreign keys. It is not hard to show that consistency and implication of foreign keys in $\mathcal{C}_{K,FK}$ remain undecidable.

4 Unary keys and foreign keys

The undecidability of the consistency problem for general keys and foreign keys motivates us to look for restricted classes of constraints. One important class is $\mathcal{C}^{Unary}_{K,FK}$, the class of unary keys and foreign keys. A cursory examination of existing XML specifications reveals that most keys and foreign keys are single-attribute constraints, i.e., unary. In particular, in XML DTDs, one can only specify unary constraints with ID and IDREF attributes.

In this section, we first investigate the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$. To do so, we consider a larger class of constraints. Let us refer to the class of unary keys and unary inclusion constraints as $\mathcal{C}^{Unary}_{K,IC}$. We develop an encoding of DTDs and $\mathcal{C}^{Unary}_{K,IC}$ constraints with linear integer constraints. This enables us to reduce the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ to the linear integer programming problem, one of the most studied NP-complete problems. We then use the same technique to show that the problem remains in NP when negations of keys are allowed. Finally, we identify several tractable cases of the consistency problems.

4.1 Coding DTDs, unary constraints

We show that $\mathcal{C}^{Unary}_{K,IC}$ constraints and DTDs can be encoded with linear equalities and inequalities, called *cardinality constraints*. The encoding allows us to reduce the consistency problem for $\mathcal{C}^{Unary}_{K,IC}$ constraints in PTIME to the *linear integer programming* (*LIP*) problem: Given an $m \times n$ matrix A of integers and a column vector $\vec{b}$ of m integers, does there exist a column vector $\vec{x}$ of n integers such that $A\,\vec{x} \geq \vec{b}$? That is, for $i \in [1, m]$,

$$\sum_{j \in [1,n]} a_{ij}\, x_j \geq b_i,$$

where a_{ij} is the jth element of the ith row of A, x_j is the jth entry of $\vec{x}$ and b_i is the ith entry of $\vec{b}$. It is known that LIP is NP-complete in the strong sense [17]. In particular, when nonnegative integer solutions are considered, [25] has shown that if the problem has a solution, then it has another solution in which for all $j \in [1, n]$, x_j is no larger than $n\,(m\,a)^{2m+1}$, where a is the largest absolute value of elements in A and $\vec{b}$.

More specifically, we show the following:

Theorem 4.1: *There is a polynomial ($O(s^2 \cdot \log s)$) algorithm that, given a DTD D and a set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints, constructs an integer matrix A and an integer vector $\vec{b}$ such that there exists an XML tree valid w.r.t. D and satisfying Σ if and only if $A\,\vec{x} \geq \vec{b}$ has an integer solution.* □

As an immediate result, we have:

Corollary 4.2: *The consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ constraints is in NP.* □

The proof of Theorem 4.1 is a bit involved, and consists of three main steps. Given a DTD D and a set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, we define in $O(s^2 \cdot \log s)$ time (in the sizes of D and Σ) the following:

- a set C_Σ of cardinality constraints such that there is an XML tree valid w.r.t. D and satisfying Σ if and only if there is an XML tree valid w.r.t. D and satisfying C_Σ; these constraints are of the forms: $|ext(\tau_1)| = |ext(\tau_1.l_1)|$, $|ext(\tau_1.l_1)| \leq |ext(\tau_2.l_2)|$, where τ_1, τ_2 are element types and l_1, l_2 are their attributes.
- a system Ψ_D of cardinality constraints such that there exists an XML tree valid w.r.t. D if and only if Ψ_D admits an integer solution; the cardinality constraints in Ψ_D are more complex than cardinality constraints studied in the context of relational databases [20];
- finally, a system of linear equalities and inequalities $\Psi(D, \Sigma)$ from C_Σ and Ψ_D such that there exists an XML tree valid w.r.t. D and satisfying Σ if and only if $\Psi(D, \Sigma)$ admits an integer solution.

All details of the encodings and the proofs of correctness can be found in [14]. Here we illustrate the encoding by an example. Consider a simplified specification for our `teacher` example given in Section 1 (τ_1, τ_2 stand for `teacher`, `subject`, and l_1, l_2 for `name`, `taught_by`, respectively).

DTD $D = (E, A, P, R, r)$, where

$$
\begin{array}{llll}
E = \{r, \tau_1, \tau_2\}, & & & \\
A = \{l_1, l_2\} & & & \\
P(r) = \tau_1, \tau_1*, & P(\tau_1) = \tau_2, \tau_2, & P(\tau_2) = \epsilon, & \\
R(\tau_1) = \{l_1\}, & R(\tau_2) = \{l_2\}, & R(r) = \emptyset. &
\end{array}
$$

Constraints Σ:

$$\tau_1.l_1 \to \tau_1, \qquad \tau_2.l_2 \to \tau_2, \qquad \tau_2.l_2 \subseteq \tau_1.l_1.$$

We encode Σ with a set C_Σ:

$$
\begin{array}{ll}
|ext(\tau_1)| = |ext(\tau_1.l_1)|, & |ext(\tau_2)| = |ext(\tau_2.l_2)|, \\
|ext(\tau_1.l_1)| \geq |ext(\tau_2.l_2)|. &
\end{array}
$$

To encode D, we first eliminate the occurrence of the Kleene star by introducing a new element type τ_t and rewriting element type definitions:

$$P(r) = \tau_1, \tau_t, \qquad P(\tau_t) = \epsilon \mid \tau_1, \tau_t.$$

It is shown that such rewriting does not affect DTD conformance and constraint satisfaction (see [14]). We then encode the modified D with a system Ψ_D:

$$
\begin{array}{lll}
\psi_r\colon & |ext(r)| = x^1_{\tau_1} = x^1_{\tau_t}, & \\
\psi_{\tau_t}\colon & |ext(\tau_t)| = x^1_\epsilon + y, & y = x^2_{\tau_1} = x^2_{\tau_t}, \\
\psi_{\tau_1}\colon & |ext(\tau_1)| = x^1_{\tau_2} = x^2_{\tau_2}, & \\
\psi_{\tau_2}\colon & |ext(\tau_2)| = x^2_\epsilon, & \\
\phi_r\colon & |ext(r)| = 1, & \\
\phi_{\tau_t}\colon & |ext(\tau_t)| = x^1_{\tau_t} + x^2_{\tau_t}, & \\
\phi_{\tau_1}\colon & |ext(\tau_1)| = x^1_{\tau_1} + x^2_{\tau_1}, & \\
\phi_{\tau_2}\colon & |ext(\tau_2)| = x^1_{\tau_2} + x^2_{\tau_2}, & \\
\text{all unknowns} \geq 0. & &
\end{array}
$$

Here we treat $|ext(\tau)|, x, y$ as unknowns of the system, and use ψ_τ to encode $P(\tau)$. Referring to an XML tree T conforming to D, recall that $|ext(\tau)|$ denotes the number of all τ nodes in T. Obviously $|ext(r)| = 1$ because T has a unique root. By $P(r) = (\tau_1, \tau_t)$, the root must have a τ_1 child and a τ_t child. Let $x^1_{\tau_1}$ and $x^1_{\tau_t}$ denote the numbers of τ_1 and τ_t children of the root, respectively. Then we must have $|ext(r)| = x^1_{\tau_1} = x^1_{\tau_t}$, which is what ψ_r says. Similarly, by $P(\tau_1) = (\tau_2, \tau_2)$, if we use $x^1_{\tau_2}, x^2_{\tau_2}$ to denote the numbers of the first and second τ_2 children of τ_1 nodes in T, respectively, then we must have $|ext(\tau_1)| = x^1_{\tau_2} = x^2_{\tau_2}$. That is exactly what ψ_{τ_1} specifies. Recall $P(\tau_t) = (\epsilon \mid \tau_1, \tau_t)$. Each τ_t node in T has either no children (ϵ) or a τ_1 child and a τ_t child. Let x^1_ϵ and y denote the numbers of occasions when τ_t nodes have empty children and nonempty children, respectively, and more specifically, let $x^2_{\tau_1}$ and $x^2_{\tau_t}$ denote the numbers of τ_1 and τ_t children of τ_t nodes in T, respectively. Then we must have that $|ext(\tau_t)|$ equals to the sum of x^1_ϵ and y, and moreover, $y = x^2_{\tau_1} = x^2_{\tau_t}$, which are what ψ_{τ_t} states. Observe that $ext(\tau)$ includes all τ nodes in T no matter where they occur. This is what $\phi_{\tau_t}, \phi_{\tau_1}$ and ϕ_{τ_2} assert.

We define $\Psi(D, \Sigma)$ to be $C_\Sigma \cup \Psi_D$, which is a system of linear constraints on nonnegative integers. Here $\Psi(D, \Sigma)$ does not admit an integer solution. More specifically, from $\Psi(D, \Sigma)$ we have that $|ext(\tau_2)| = 2\,|ext(\tau_1)|$ on the one hand, and $|ext(\tau_2)| \leq |ext(\tau_1)|$ on the other hand, while $|ext(\tau_1)| \geq 1$ (by $\psi_r, \phi_r, \phi_{\tau_1}$). Thus by Theorem 4.1, there is no XML tree T such that $T \models D$ and $T \models \Sigma$. This is consistent with the observation of Section 1.

The encoding is not only interesting in its own right, but also useful in the consistency analyses of $\mathcal{C}^{Unary}_{K,FK}$ and $\mathcal{C}^{Unary}_{K^\neg,IC}$ constraints, as well as in resolving a special case of $\mathcal{C}^{Unary}_{K,FK}$ constraint implication.

4.2 $\mathcal{C}^{Unary}_{K,FK}$ and $\mathcal{C}^{Unary}_{K^\neg,IC}$ constraints

Next, we establish the precise complexity bound on the consistency problem for unary keys and foreign keys:

Theorem 4.3: *The consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ constraints is NP-complete.* □

Proof sketch: Corollary 4.2 has shown that the problem is in NP. We show that it is NP-hard by reduction from a variant of LIP, namely,

$$A\vec{x} = \vec{b},$$

where for all $i \in [1,m]$, $j \in [1,n]$, a_{ij} coefficients are in $\{0,1\}$, all b_i elements are 1, and all x_j components are binary, i.e., in $\{0,1\}$. It is known that the variant is also NP-complete [17].

Given an instance $A\vec{x} = \vec{b}$ of the variant of LIP, we define a DTD D and a set Σ of $\mathcal{C}^{Unary}_{K,FK}$ constraints over D such that there is an XML tree valid w.r.t. D and satisfying Σ if and only if $A\vec{x} = \vec{b}$ admits a binary solution. For $i \in [1,m]$, we use F_i to denote $\sum_{j\in[1,n]} a_{ij}\, x_j$.

We define D to be (E, A, P, R, r), where

$$\begin{array}{l}
E = \{r\} \cup \{F_i \mid i \in [1,m]\} \\
\quad \cup \{b_i \mid i \in [1,m]\} \\
\quad \cup \{VF_i \mid i \in [1,m]\} \\
\quad \cup \{X_{ij} \mid i \in [1,m], j \in [1,n]\} \\
\quad \cup \{Z_{ij} \mid i \in [1,m], j \in [1,n]\} \\
A = \{v\} \cup \{A_{ij} \mid i \in [1,m], j \in [1,n]\}
\end{array}$$

$$\begin{array}{lll}
P(r) & = F_1, ..., F_m, b_1, ..., b_m & \\
P(F_i) & = X_{ij_1}, ..., X_{ij_l} & \text{for all } i \in [1,m], \\
 & \multicolumn{2}{l}{\text{where } X_{ij_1}, ..., X_{ij_l} \text{ is a subsequence of}} \\
 & \multicolumn{2}{l}{X_{i1}, ..., X_{im} \text{ such that } X_{ij} \text{ is in } P(F_i)} \\
 & \multicolumn{2}{l}{\text{iff } a_{ij} \text{ in A is 1}} \\
P(X_{ij}) & = Z_{ij} \mid \epsilon & \text{for } i \in [1,m] \text{ and } j \in [1,n] \\
P(Z_{ij}) & = VF_i & \text{for } i \in [1,m] \text{ and } j \in [1,n] \\
P(VF_i) & = P(b_i) = \epsilon & \text{for } i \in [1,m] \\
R(Z_{ij}) & = \{A_{ij}\} & \text{for } i \in [1,m] \text{ and } j \in [1,n] \\
R(VF_i) & = R(b_i) = \{v\} & \text{for } i \in [1,m] \\
R(r) & = R(F_i) = R(X_{ij}) = \emptyset &
\end{array}$$

Intuitively, X_{ij} indicates x_j in F_i, and Z_{ij} denotes the value of X_{ij}: X_{ij} has value 1 if and only if X_{ij} has a Z_{ij} child. The attribute A_{ij} of Z_{ij} is used to ensure that all occurrences of x_j have the same value. The element type VF_i indicates the value of F_i, and its attribute v is to ensure that the value of F_i is 1. More specifically, these are captured by the set Σ of $\mathcal{C}^{Unary}_{K,FK}$ constraints over D. To ensure that all occurrences of x_j have the same value, the following are in Σ: for $j \in [1,n]$ and $i, l \in [1,m]$,

$$Z_{ij}.A_{ij} \to Z_{ij}, \qquad Z_{ij}.A_{ij} \subseteq Z_{lj}.A_{lj}.$$

These assert that X_{ij} has value 1 if and only if X_{lj} equals to 1. To ensure $F_i = b_i$, we include the following in Σ: for $i \in [1,m]$,

$$\begin{array}{ll}
VF_i.v \to VF_i, & b_i.v \to b_i, \\
VF_i.v \subseteq b_i.v, & b_i.v \subseteq VF_i.v.
\end{array}$$

These assert that F_i node has a unique VF_i descendent, and thus has value 1. An XML tree valid w.r.t. D has the form shown in Figure 3.

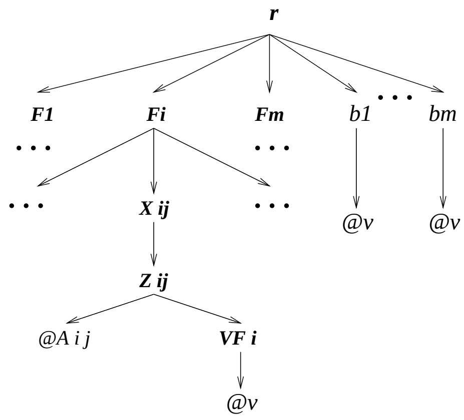

Figure 3: A tree used in the proof of Theorem 4.3

It is easy to verify that the encoding can be done in PTIME in m and n. Moreover, $A\vec{x} = \vec{b}$ admits a binary solution if and only if there is an XML tree valid w.r.t. D and satisfying Σ. Thus what given above is indeed a PTIME reduction from the variant of LIP. □

In relational databases, it is common to consider primary keys. That is, for each relation one can specify at most one key, namely, the primary key of the relation. In the XML setting, the *primary key restriction* requires that for each element type $\tau \in E$, one can specify at most one key, i.e., there is at most one $l \in R(\tau)$ such that $\tau.l \to \tau$. This is the case for "keys" specified with ID attributes, since in a DTD, at most one ID attribute can be specified for each element type. Under the primary key restriction, the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ is to determine, given any DTD D and finite set Σ of $\mathcal{C}^{Unary}_{K,FK}$ constraints in which there is at most one key for each element type (given either as keys or as part of foreign keys), whether there is an XML tree valid w.r.t. D and satisfying Σ. One might think that the primary key restriction would simplify the consistency analysis of $\mathcal{C}^{Unary}_{K,FK}$ constraints. However, this is not the case.

Corollary 4.4: *Under the primary key restriction, the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ remains NP-complete.* □

Proof sketch: The reduction from LIP given in the proof of Theorem 4.3 defines at most one key for each element type. □

A mild generalization of the encoding given above can establish the complexity of the consistency problem for $\mathcal{C}^{Unary}_{K^{\neg},IC}$, the class of unary keys, inclusion constraints and negations of keys (see [14] for the encoding and a proof for the following corollary).

Corollary 4.5: *The consistency problem for $\mathcal{C}^{Unary}_{K^{\neg},IC}$ constraints is NP-complete.* □

It should be mentioned that the problem remains NP-hard under the primary key restriction. This can be verified along the same lines as the proof of Corollary 4.4.

Corollary 4.5 also tells us the complexity of a special case of the implication problem for $\mathcal{C}^{Unary}_{K,FK}$, referred to as *implication problem for unary keys by* $\mathcal{C}^{Unary}_{K,FK}$ *constraints*:

Theorem 4.6: *The following is coNP-complete, even under the primary key restriction: given any DTD D, any set* Σ *of* $\mathcal{C}^{Unary}_{K,FK}$ *constraints and a unary key* φ *over* D*, whether* $(D, \Sigma) \vdash \varphi$. □

Proof sketch: Observe that $(D, \Sigma) \vdash \varphi$ iff $\Sigma \cup \{\neg\varphi\}$ and D are not consistent, i.e., there exists no XML tree T such that $T \models D$, $T \models \Sigma$ and $T \models \neg\varphi$. Note that $\Sigma \cup \{\neg\varphi\}$ is a set of $\mathcal{C}^{Unary}_{K^\neg,IC}$ constraints. Thus the implication problem for unary keys by $\mathcal{C}^{Unary}_{K,FK}$ constraints is the complement of a special case of the consistency problem for $\mathcal{C}^{Unary}_{K^\neg,IC}$, and hence in coNP. We show it is coNP-hard by reduction from the complement of the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$. See [14] for details. □

Finally, we identify some PTIME decidable cases of the consistency and implication problems. First, these problems for unary keys only are decidable in linear time, by Theorem 3.4. We next show that given a fixed DTD D, the consistency and implication analyses become simpler.

Corollary 4.7: *For a* fixed *DTD, the following problems are decidable in PTIME:*

- *The consistency problems for* $\mathcal{C}^{Unary}_{K,FK}$ *and* $\mathcal{C}^{Unary}_{K^\neg,IC}$.
- *Implication of unary keys by* $\mathcal{C}^{Unary}_{K,FK}$ *constraints.*

□

Proof sketch: Recall the encoding given in the proof of Theorem 4.1. Given a fixed DTD D, the number of unknowns in C_Σ is bounded by the size of D ($O(s^2)$, where s is the size of D), and the number of unknowns in Ψ_D is determined by D and fixed. Thus the number of unknowns in $\Psi(D, \Sigma)$ is bounded. In other words, the number of unknowns in the system of linear integer constraints that encodes D and Σ is bounded. This follows from the proofs of Theorem 4.1 and Corollary 4.5 (see [14]). It is known that when the number of unknowns in a system of linear constraints is bounded, checking whether the system admits an integer solution can be done in PTIME [22]. As shown by Theorem 4.1 and Corollary 4.5, Σ can be satisfied by an XML tree valid w.r.t. D if and only if their encoding system admits an integer solution. The system can be computed in PTIME in the size of D. Putting these together, we have Corollary 4.7. □

5 Incorporating negation

In Section 4, we have shown that the consistency problem for unary keys and foreign keys is NP-complete. In this section, we extend the result by showing that the problem remains in NP when negations of these unary constraints are allowed. That is, the problem is NP-complete for $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$, the class of unary keys, inclusion constraints and their negations. This helps us settle the implication problems for $\mathcal{C}^{Unary}_{K,FK}$ and the more general $\mathcal{C}^{Unary}_{K,IC}$, the class of unary keys and foreign keys, and the class of unary keys and inclusion constraints, respectively. This is one of the reasons that we are interested in the consistency problem for $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$.

Theorem 5.1: *The consistency problem for* $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$ *is NP-complete.* □

While this theorem subsumes Theorem 4.3, the reduction is quite different from the nice encoding with instances of LIP that we used for $\mathcal{C}^{Unary}_{K,FK}$. In fact, while typically NP-complete problems are easily shown to be in NP, and only the reduction from a known NP-complete problem is difficult, for the consistency problem for $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$, the opposite is the case, and the proof of membership in NP is a little involved (even assuming the encoding of keys and inclusion constraints by instances of LIP given in the previous section). We cannot reduce the problem directly to LIP as before, because there is no direct connection between $\tau_i.l_i \not\subseteq \tau_j.l_j$ and the cardinalities $|ext(\tau_i)|$, $|ext(\tau_j)|$, $|ext(\tau_i.l_i)|$ and $|ext(\tau_j.l_j)|$ in an XML tree.

We develop an NP algorithm for determining the consistency of $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$ constraints. The algorithm takes advantage of another encoding of $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$ constraints with linear integer constraints, which characterizes a set interpretation of unary inclusion constraints and their negations. The encoding and the details of the proof can be found in [14].

We next investigate implication problems.

Theorem 5.2: *For each of* $\mathcal{C}^{Unary}_{K,IC}$ *and* $\mathcal{C}^{Unary}_{K,FK}$*, the implication problem is coNP-complete, even under the primary key restriction.* □

Proof sketch: The problem for $\mathcal{C}^{Unary}_{K,IC}$ is to determine, for a DTD D, a set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints, and a constraint φ (unary key or unary inclusion), whether $(D, \Sigma) \vdash \varphi$. Note that $(D, \Sigma) \vdash \varphi$ iff there is no XML tree T with $T \models D \wedge \bigwedge \Sigma \wedge \neg\varphi$, and $\Sigma \cup \{\neg\varphi\}$ is a set of $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$ constraints. Thus by Theorem 5.1, the implication problem for $\mathcal{C}^{Unary}_{K,IC}$ is in coNP. It is shown to be coNP-hard in the same way as in the proof of

Theorem 4.6. Similarly, we show that the implication problem for $\mathcal{C}_{K,IC}^{Unary}$ is also coNP-complete (see [14]). □

Finally, along the same lines as Corollary 4.7, we show the following (see [14] for the proof):

Corollary 5.3: *For a fixed DTD, the following problems can be determined in PTIME:*

- *The implication problem for* $\mathcal{C}_{K,FK}^{Unary}$.
- *The consistency problem for* $\mathcal{C}_{K^{\neg},IC^{\neg}}^{Unary}$.

□

6 Conclusion

We have studied the consistency problems associated with four classes of integrity constraints for XML. We have shown that in contrast to its trivial counterpart in relational databases, the consistency problem is undecidable for $\mathcal{C}_{K,FK}$, the class of multi-attribute keys and foreign keys. This demonstrates that the interaction between DTDs and key/foreign key constraints is rather intricate. This negative result motivated us to study $\mathcal{C}_{K,FK}^{Unary}$, the class of unary keys and foreign keys, which are commonly used in practice. We have developed a characterization of DTDs and unary constraints in terms of linear integer constraints. This establishes a connection between DTDs, unary constraints and linear integer programming, and allows us to use techniques from combinatorial optimization in the study of XML constraints. We have shown that the consistency problem for $\mathcal{C}_{K,FK}^{Unary}$ is NP-complete. Furthermore, the problem remains in NP for $\mathcal{C}_{K^{\neg},IC^{\neg}}^{Unary}$, the class of unary keys, unary inclusion constraints and their negations.

We have also investigated the implication problems for XML keys and foreign keys. In particular, we have shown that the problem is undecidable for $\mathcal{C}_{K,FK}$ and it is coNP-complete for $\mathcal{C}_{K,FK}^{Unary}$ constraints. Several PTIME decidable cases of the implication and consistency problems have also been identified. The main results of the paper are summarized in Figure 4.

It is worth remarking that the undecidability and NP-hardness results also hold for other schema specifications beyond DTDs, such as XML Schema [29] and the generalization of DTDs proposed in [26].

This work is a first step towards understanding the interaction between DTDs and integrity constraints. A number of questions remain open. First, we have only considered keys and foreign keys defined with XML attributes. We expect to expand techniques developed here for more general schema and constraint specifications, such as those proposed in XML Schema and in a recent proposal for XML keys [7]. Second, other constraints commonly found in databases, e.g., inverse constraints, deserve further investigation. Third, a lot of work remains to be done on identifying tractable yet practical classes of constraints and on developing heuristics for consistency analysis. Finally, a related project is to use integrity constraints to distinguish good XML design (specification) from bad design, along the lines of normalization of relational schemas. Coding with linear integer constraints gives us decidability for some implication problems for XML constraints, which is a first step towards a design theory for XML specifications.

Acknowledgments. We thank Michael Benedikt, Alberto Mendelzon, Frank Neven and Jérôme Siméon for helpful discussions. Part of this work was done while the second author was visiting INRIA.

References

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

[2] S. Abiteboul and V. Vianu. Regular path queries with constraints. In *PODS'97*, pages 122–133.

[3] V. Apparao et al. Document Object Model (DOM) Level 1 Specification. W3C Recommendation, Oct. 1998.
`http://www.w3.org/TR/REC-DOM-Level-1/`.

[4] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. In *SIGMOD'99*, pages 597–599.

[5] C. Beeri and T. Milo. Schemas for integration and translation of structured and semi-structured data. In *ICDT'99*, pages 296–313.

[6] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. W3C Recommendation, Feb. 1998.
`http://www.w3.org/TR/REC-xml/`.

[7] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan. Keys for XML. In *WWW'10*, 2001.

[8] P. Buneman, W. Fan, and S. Weinstein. Interaction between path and type constraints. In *PODS'99*, pages 56–67.

[9] P. Buneman, W. Fan, and S. Weinstein. Path constraints on semistructured and structured data. In *PODS'98*, pages 129–138.

[10] D. Calvanese, G. De Giacomo, and M. Lenzerini. Representing and reasoning on XML documents: A description logic approach. *J. Logic and Computation*, 9(3):295–318, 1999.

[11] J. Clark. XSL Transformations (XSLT). W3C Recommendation, Nov. 1999.
`http://www.w3.org/TR/xslt`.

	multi-attribute keys, foreign keys	*unary keys, foreign keys*	*primary, unary keys, foreign keys*	*DTD fixed, unary keys, foreign keys*	*multi-attribute keys only*
consistency	undecidable	NP-complete	NP-complete	PTIME	linear time
implication	undecidable	coNP-complete	coNP-complete	PTIME	linear time

Figure 4: The main results of the paper

[12] J. Clark and S. DeRose. XML Path Language (XPath). W3C Recommendation, Nov. 1999. `http://www.w3.org/TR/xpath`.

[13] S. S. Cosmadakis, P. C. Kanellakis, and M. Y. Vardi. Polynomial-time implication problems for unary inclusion dependencies. *J. ACM*, 37(1):15–46, Jan. 1990.

[14] W. Fan and L. Libkin. On XML integrity constraints in the presence of DTDs. Full version of the paper: `http://www.cis.temple.edu/~fan/papers/xml/pods01-full.ps.gz`

[15] W. Fan and J. Siméon. Integrity constraints for XML. In *PODS'00*, pages 23–34.

[16] D. Florescu, L. Raschid, and P. Valduriez. A methodology for query reformulation in CIS using semantic knowledge. *Int'l J. Cooperative Information Systems (IJCIS)*, 5(4):431–468, 1996.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[18] C. Hara and S. Davidson. Reasoning about nested functional dependencies. In *PODS'99*, pages 91–100.

[19] M. Ito and G. E. Weddell. Implication problems for functional constraints on databases supporting complex objects. *JCSS*, 50(1):165–187, 1995.

[20] P. C. Kanellakis. On the computational complexity of cardinality constraints in relational databases. *Information Processing Letters*, 11(2):98–101, Oct. 1980.

[21] A. Layman et al. XML-Data. W3C Note, Jan. 1998. `http://www.w3.org/TR/1998/NOTE-XML-data`.

[22] H. W. Lenstra. Integer programming in a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[23] J. Melton and A. Simon. *Understanding the New SQL: A Complete Guide*. Morgan Kaufman, 1993.

[24] F. Neven. Extensions of attribute grammars for structured document queries. In *DBPL'99*.

[25] C. H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.

[26] Y. Papakonstantinou and V. Vianu. Type inference for views of semistructured data. In *PODS'00*, pages 35–46.

[27] L. Popa. *Object/Relational Query Optimization with Chase and Backchase*. PhD thesis, University of Pennsylvania, 2000.

[28] J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). Workshop on XML Query Languages, Dec. 1998.

[29] H. S. Thompson et al. XML Schema Part 1: Structures. W3C Working Draft, Apr. 2000. `http://www.w3.org/TR/xmlschema-1/`.

[30] J. D. Ullman. *Database and Knowledge Base Systems*. Computer Science Press, 1988.

[31] P. Wadler. A formal semantics for patterns in XSL. Technical report, Bell Labs, 2000.

[32] J. Widom. Data management for XML: Research directions. In *IEEE Data Engineering Bulletin*, 22(3): 44-52, 1999.