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 [21]. 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, 28].

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 [19] and XML Schema [27]. Keys and foreign keys for XML are important in, among other things, query optimization [25], data integration [14], and in data exchange for converting databases to an XML encoding.

*Research affiliation: Bell Laboratories.

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 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 [10, 29], XQL [26], XML Schema [27], XPath [11] 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?

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, 24]. 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 [14].

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:

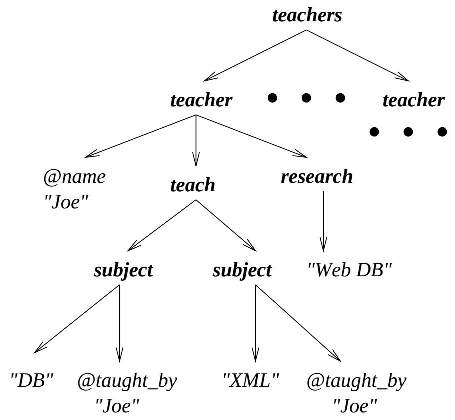

Figure 1: An XML tree conforming to D_1

```
<!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 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:

$$|ext(teacher.name)| = |ext(teacher)|,$$

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

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 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 specifications and constraint languages for XML, e.g., XML Data and 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 [12]. 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 [19] and XML Schema [27]. The need for studying XML constraints has also been advocated in [30]. 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 [13], 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, 8]. The interaction between path constraints and database schemas was investigated in [7]. 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, [16, 17]. 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.

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, 9, 22]) 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:

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

  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. 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

$$\begin{array}{l}
E_1 = \{teachers,\ teacher,\ teach,\ research,\ subject\} \\
A_1 = \{name,\ taught_by\} \\
P_1(teachers) = teacher,\ teacher^* \\
P_1(teacher) \ \ = teach,\ research \\
P_1(teach) \quad\ \ = subject,\ subject \\
P_1(subject) \ \ = P_1(research) = \mathtt{S} \\
R_1(teacher) \ \ = \{name\} \\
R_1(subject) \ \ = \{taught_by\} \\
R_1(teachers) = R_1(teach) = R_1(research) = \emptyset \\
r_1 = teachers
\end{array}$$

Similarly, we represent the DTD D_2 given in Section 1 as $(E_2, A_2, P_2, R_2, r_2)$, where

$$\begin{array}{l}
E_2 = \{db,\ foo\} \\
A_2 = \emptyset \\
P_2(db) \quad = P_2(foo) = foo \\
R_2(db) \quad = R_2(foo) = \emptyset \\
r_2 = db
\end{array}$$

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 $ele(v) = [v_1, ..., v_n]$ (a list), 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 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 $\mathtt{S}$. 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

$$E_3 = \{school,\ student,\ course,\ enroll,\ name,\ subject\}$$

$$A_3 = \{student\_id,\ course\_no,\ dept\}$$

$$\begin{aligned}
P_3(school) &= course^*,\ student^*,\ enroll^* \\
P_3(course) &= subject \\
P_3(student) &= name \\
P_3(enroll) &= P_3(name) = P_3(subject) = \mathtt{S} \\
R_3(course) &= \{dept,\ course_no\} \\
R_3(student) &= \{student_id\} \\
R_3(enroll) &= \{student_id,\ dept,\ course_no\} \\
R_3(school) &= R_3(name) = R_3(subject) = \emptyset
\end{aligned}$$

$$r_3 = school$$

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

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

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 $\tau_2[Y] \rightarrow \tau_2$, then $T \models \tau_2[Y] \rightarrow \tau_2$ 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 $\mathcal{C}_{K,FK}^{Unary}$, is a sublanguage of $\mathcal{C}_{K,FK}$. A $\mathcal{C}_{K,FK}^{Unary}$ constraint is a $\mathcal{C}_{K,FK}$ constraint defined with a single attribute. More specifically, a *constraint φ of $\mathcal{C}_{K,FK}^{Unary}$* over DTD D is either

- *key:* $\tau.l \rightarrow \tau$, where $\tau \in E$ and $l \in R(\tau)$; or
- *foreign key:* $\tau_1.l_1 \subseteq \tau_2.l_2$ and $\tau_2.l_2 \rightarrow \tau_2$, 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 $\mathcal{C}_{K,FK}^{Unary}$ 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 $\mathcal{C}_{K,FK}^{Unary}$ as follows. One is $\mathcal{C}_{K^\neg,IC}^{Unary}$, the class consisting of unary keys, unary inclusion constraints and negations of unary keys. The other, $\mathcal{C}_{K^\neg,IC^\neg}^{Unary}$, 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 $\mathcal{C}_{K,FK}$, $\mathcal{C}_{K,FK}^{Unary}$, $\mathcal{C}_{K^\neg,IC}^{Unary}$ or $\mathcal{C}_{K^\neg,IC^\neg}^{Unary}$, 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. We show that the consistency and implication problems for $\mathcal{C}_{K,FK}$ are undecidable, but we identify several special cases of the problems and show that these cases are decidable in PTIME.

3.1 Undecidability of consistency analysis

Our main result is negative:

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

Proof: We first show that an implication problem associated with keys and foreign keys in relational databases is undecidable, and then present a reduction from (the complement of) the implication problem to the consistency problem for $\mathcal{C}_{K,FK}$ constraints.

Let us first review keys, foreign keys and their associated implication problems in relational databases (see [1] for detailed discussions). Let $\mathbf{R} = (R_1, \ldots, R_n)$ be a relational schema. For each relation (schema) R_i in $\mathbf{R}$, we use $Att(R_i)$ to denote the set of all attributes of R_i, and $Inst(R_i)$ to denote the set of finite instances of R_i. It should be mentioned that by database instances in this paper we mean *finite* instances. Observe that a database instance $\mathbf{I}$ of $\mathbf{R}$ has the form $(I_1, \ldots, I_n)$, where $I_i \in Inst(R_i)$ for all $i \in [1, n]$. For an instance $I_i \in Inst(R_i)$, a tuple $t \in I_i$ and an attribute $l \in Att(R_i)$, we use $t.l$ to denote the l attribute value of t. Keys and foreign keys over $\mathbf{R}$ are defined as follows:

- *key:* $R[l_1, ..., l_k] \to R$, where $R \in \mathbf{R}$, and for any $i \in [1, k]$, $l_i \in Att(R)$. An instance $\mathbf{I}$ of $\mathbf{R}$ *satisfies* the key constraint φ, denoted by $\mathbf{I} \models \varphi$, if

$$\forall\, t_1\, t_2 \in I\ (\bigwedge_{1 \le i \le k} (t_1.l_i = t_2.l_i) \to \bigwedge_{l \in Att(R)} (t_1.l = t_2.l)),$$

  where I is the instance of R in $\mathbf{I}$;

- *foreign key:* $R[l_1, ..., l_k] \subseteq R'[l'_1, ..., l'_k]$, where R, R' are in $\mathbf{R}$, $[l_1, ..., l_k]$ is a sequence of attributes in $Att(R)$, and $[l'_1, ..., l'_k]$ is a sequence of attributes in $Att(R')$. In addition, $R'[l'_1, ..., l'_k] \to R'$, i.e., the set consisting of $l'_1, ..., l'_k$ is a key of R'. An instance $\mathbf{I}$ of $\mathbf{R}$ *satisfies* the foreign key constraint φ, denoted by $\mathbf{I} \models \varphi$, if $\mathbf{I} \models R'[l'_1, ..., l'_k] \to R'$ and moreover,

$$\forall\, t_1 \in I\ \exists\, t_2 \in I'\ (\bigwedge_{1 \le j \le k} t_1.l_j = t_2.l'_j),$$

  where I and I' are the instances of R and R' in $\mathbf{I}$, respectively.

Let $\Sigma \cup \{\varphi\}$ be a set of keys and foreign keys over $\mathbf{R}$. We use $\Sigma \vdash \varphi$ to denote that Σ *satisfies* φ, i.e., for any instance $\mathbf{I}$ of $\mathbf{R}$, if $\mathbf{I} \models \Sigma$, then $\mathbf{I} \models \varphi$.

In relational databases, the *implication problem for keys and foreign keys* is the problem of determining, given a relational schema $\mathbf{R}$ and a set $\Sigma \cup \{\varphi\}$ of keys and foreign keys over $\mathbf{R}$, whether $\Sigma \vdash \varphi$. A special case of the problem is the *implication problem for keys by keys and foreign keys*, which is to determine whether $\Sigma \vdash \varphi$ where φ is a key and Σ is a set of keys and foreign keys over $\mathbf{R}$.

It was shown in [13] that the implication problem for keys and foreign keys in relational databases is undecidable. The lemma below shows a stronger result.

Lemma 3.2: *In relational databases, the implication problem for keys by keys and foreign keys is undecidable.* □

Proof: We prove this by reduction from the implication problem for functional dependencies by functional and inclusion dependencies, which is undecidable (see, e.g., [1] for a proof). Before we give the reduction, we first review functional and inclusion dependencies in relational databases. Let $\mathbf{R}$ be a relational schema. Functional dependencies (FDs) and inclusion dependencies (IDs) over $\mathbf{R}$ are defined as follows.

- *FD.* $R : X \to Y$, where $R \in \mathbf{R}$, and X and Y are subsets of attributes in $Att(R)$. An instance $\mathbf{I}$ of $\mathbf{R}$ *satisfies* the FD θ, denoted by $\mathbf{I} \models \theta$, if $\forall t_1\, t_2 \in I\ (\bigwedge_{l \in X} (t_1.l = t_2.l) \to \bigwedge_{l' \in Y} (t_1.l' = t_2.l'))$, where I is the instance of R in $\mathbf{I}$. Observe that keys are a special case of FDs in which $Y = Att(R)$.
- *ID.* $R[l_1, ..., l_k] \subseteq R'[l'_1, ..., l'_k]$, where $R, R' \in \mathbf{R}$, $[l_1, ..., l_k]$ is a sequence of attributes in $Att(R)$, and similarly, $[l'_1, ..., l'_k]$ is a sequence of attributes in $Att(R')$. It should be mentioned that in contrast to foreign keys, the set consisting of $l'_1, ..., l'_k$ is not necessarily a key for R'. An instance $\mathbf{I}$ of $\mathbf{R}$ *satisfies* the ID θ, denoted by $\mathbf{I} \models \theta$, if $\forall t_1 \in I\ \exists t_2 \in I'\ (\bigwedge_{1 \le j \le k} t_1.l_j = t_2.l'_j)$, where I and I' are the instances of R and R' in $\mathbf{I}$, respectively.

Let $\Sigma \cup \{\theta\}$ be a set of FDs and IDs over $\mathbf{R}$. We use $\Sigma \vdash \theta$ to denote that Σ *satisfies* θ as for keys and foreign keys. The *implication problem for FDs by FDs and IDs* is the problem to determine, given any relational schema $\mathbf{R}$, any set Σ of FDs and IDs over $\mathbf{R}$ and a FD θ over $\mathbf{R}$, whether $\Sigma \vdash \theta$. This is a well-known undecidable problem [1].

We encode FDs and IDs in terms of keys and foreign keys as follows.

(1) FD $\psi = R : X \to Y$.

Let Z be a key for R, i.e., $R[Z] \to R$ (note that every relation schema has a key). We define a new (fresh) relation schema R_{new} such that $Att(R_{new}) = XYZ$, i.e., the union of X, Y and Z. Observe that XYZ is a key for both R_{new} and R. We encode ψ with:

$$\phi_1 = R_{new}[X] \to R_{new}, \qquad \phi_2 = R[XY] \subseteq R_{new}[XY],$$
$$\phi_3 = R_{new}[XYZ] \subseteq R[XYZ], \quad \phi_4 = R_{new}[XY] \to R_{new}.$$

(2) ID $\psi = R_1[X] \subseteq R_2[Y]$.

Let Z be a key for R_2, i.e., $R_2[Z] \to R_2$. We define a new schema R_{new} such that $Att(R_{new}) = YZ$. Observe that YZ is a key for R_2. We encode ψ with:

$$\phi_1 = R_{new}[Y] \to R_{new}, \quad \phi_2 = R_1[X] \subseteq R_{new}[Y], \quad \phi_3 = R_{new}[YZ] \subseteq R_2[YZ].$$

We next show that the encoding is indeed a reduction from the implication problem for FDs by FDs and IDs to the implication problem for keys by keys and foreign keys. Given a relational schema $\mathbf{R}$, a set Σ of FDs and IDs over $\mathbf{R}$, and a FD $\theta = R_\theta : X \to Y$ over $\mathbf{R}$, as described above we encode Σ with a set Σ_1 of keys and foreign keys, and encode θ with

$$\phi_1 = R^\theta_{new}[X] \to R^\theta_{new}, \qquad \phi_2 = R_\theta[XY] \subseteq R^\theta_{new}[XY],$$
$$\phi_3 = R^\theta_{new}[XYZ] \subseteq R_\theta[XYZ], \quad \phi_4 = R^\theta_{new}[XY] \to R^\theta_{new}.$$

Let $\Sigma' = \Sigma_1 \cup \{\phi_2, \phi_3, \phi_4\}$. It suffices to show that

$$\Sigma \vdash \theta \quad \text{iff} \quad \Sigma' \vdash \phi_1.$$

Let **R'** be the relational schema that includes all relation schemas in **R** as well as new relations created in the encoding. We show the claim as follows.

(1) Suppose that there is an instance **I** of **R** such that **I** $\models \bigwedge \Sigma \wedge \neg\theta$. We show that there is an instance **I'** of **R'** such that **I'** $\models \bigwedge \Sigma' \wedge \neg\phi_1$ We construct **I'** such that for any R in **R**, the instance of R in **I'** is the same as the instance of R in **I**. We populate instances of new relations R_{new} created in the encoding as follows.

- If R_{new} is introduced in the encoding of a FD $R : X \to Y$ as described above, then we let the instance I_{new} of R_{new} in **I'** be a subset of $\Pi_{XYZ}(I)$ such that $\Pi_{XY}(I) = \Pi_{XY}(I_{new})$ and $I_{new} \models R_{new}[XY] \to R_{new}$, where I is the instance of R in **I** and $\Pi_W(I)$ is the projection of I on attributes W.
- If R_{new} is introduced in the encoding of an ID $R_1[X] \subseteq R_2[Y]$ as described above, then let the instance I_{new} of R_{new} in **I'** be a subset of $\Pi_{YZ}(I_2)$ such that $\Pi_Y(I_2) = \Pi_Y(I_{new})$ and $I_{new} \models R_{new}[Y] \to R_{new}$, where I_2 is the instance of R_2 in **I**.

It is easy to verify that **I'** $\models \bigwedge \Sigma' \wedge \neg\phi_1$.

(2) Suppose that there is an instance **I'** of **R'** such that **I'** $\models \bigwedge \Sigma' \wedge \neg\phi_1$. We construct an instance **I** of **R** by removing from **I'** all instances of new relations introduced in the encoding. It is easy to verify that **I** $\models \bigwedge \Sigma \wedge \neg\theta$.

Therefore, the encoding is indeed a reduction from the implication problem for FDs by FDs and IDs. This shows that the implication problem for keys by keys and foreign keys is undecidable. □

From Lemma 3.2 follows immediately that the complement of the implication problem for keys by keys and foreign keys is also undecidable. That is the problem to determine, given a relational schema **R**, a set Σ of keys and foreign keys over **R** and a key φ over **R**, whether there is an instance of **R** satisfying $\bigwedge \Sigma \wedge \neg\varphi$.

We now continue with the proof of Theorem 3.1, i.e., the consistency problem for $\mathcal{C}_{K,FK}$ constraints is undecidable. Given Lemma 3.2, it suffices to give a reduction from the complement of the implication problem for keys by keys and foreign keys. Let $\mathbf{R} = (R_1, \ldots, R_n)$ be a relational schema, Θ be a set of keys and foreign keys over **R**, and $\varphi = R[X] \to R$ be a key over **R**. Let $Y = Att(R) \setminus X$. We encode **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

$$
\begin{aligned}
E &= \{R_i \mid i \in [1,n]\} \cup \{t_i \mid i \in [1,n]\} \cup \{r,\ D_Y,\ E_X\} \\
A &= \bigcup_{i \in [1,n]} Att(R_i)
\end{aligned}
$$

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

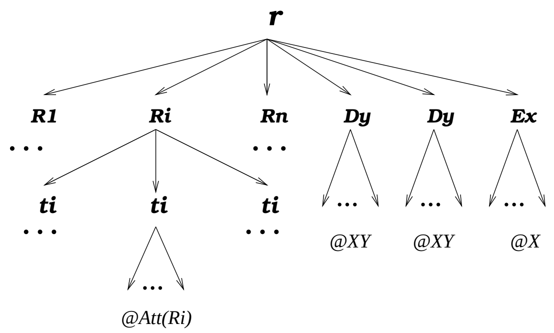

Figure 2: A tree used in the proof of Theorem 3.1

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:

- For every key $R_i[Z] \to R_i$ in Θ, $t_i[Z] \to t_i$ is in Σ_Θ.
- For every 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:

$$D_Y[Y] \to D_Y, \quad E_X[X] \to E_X, \quad D_Y[X] \subseteq E_X[X], \quad D_Y[X,Y] \subseteq t_\varphi[X,Y], \quad t_\varphi[XY] \to t_\varphi,$$

where $[X, Y]$ stands for the concatenation of list X and list Y, and t_φ is the grammar symbol in $P(R) = t_\varphi^*$. Observe 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 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. Assume that there is an instance $\mathbf{I}$ of $\mathbf{R}$ satisfying $\bigwedge \Theta \wedge \neg\varphi$. We construct an XML tree T from $\mathbf{I}$ as follows. Let T have a root node r and a R_i node for each R_i in $\mathbf{R}$. For any $R_i \in \mathbf{R}$ and each tuple p in the instance of R_i in $\mathbf{I}$, we create a distinct t_i node x such that $p.l = x.l$ for all $l \in Att(R_i)$. Since $\mathbf{I} \models \neg\varphi$, there are two tuples p and p' in the instance of R in $\mathbf{I}$ such that $p[X] = p'[X]$ and $p[Y] \neq p'[Y]$. We create two distinct D_Y nodes d_1 and d_2 such that $d_1.l = p.l$ and $d_2.l = p'.l$ for all $l \in Att(R)$. In addition, we create a single E_X node e such that $e.l = p.l$ for all $l \in X$. We define the edge relation of T such that T has the form shown in Figure 2. It is easy to verify that T is a tree valid w.r.t. D. By $\mathbf{I} \models \Theta$ it is easy to verify that $T \models \Sigma_\Theta$. By the definition of T, it is also easy to see that $T \models \Sigma_\varphi$. In particular, since $Att(R) = X \cup Y$, we have $T \models t_\varphi[XY] \to t_\varphi$, where t_φ is the symbol in $P(R) = t_\varphi^*$. Therefore, $T \models \Sigma$. Conversely, suppose that there is an XML tree T valid w.r.t. D and satisfying Σ. We define an instance $\mathbf{I}$ of schema $\mathbf{R}$ as follows. For each t_i node x, let $(l_1 = x.l_1, \ldots, l_m = x.l_m)$ be a tuple in the instance of R_i in $\mathbf{I}$, where $l_1, \ldots, l_m$ are an enumeration of $Att(R_i)$. Obviously $\mathbf{I}$ is an instance of $\mathbf{R}$. By $T \models \Sigma_\Theta$, it is easy to verify that $\mathbf{I} \models \Theta$. Moreover, by $T \models \Sigma_\varphi$ and the definition of $\mathbf{I}$, we have $\mathbf{I} \models \neg\varphi$ since there must be two tuples d_1 and d_2 in the instance of R in $\mathbf{I}$ such that $d_1[X] = d_2[X]$ but $d_1[Y] \neq d_2[Y]$. Thus the encoding is indeed a reduction from the complement of the implication problem for keys by keys and foreign keys.

This completes the proof of Theorem 3.1. □

3.2 Undecidability of implication

We next consider the implication problem.

Lemma 3.3: *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: It suffices to establish the undecidability of the complements of the implication problems. This can be done by reduction from the consistency problem for $\mathcal{C}_{K,FK}$. Let $D = (E, A, P, R, r)$ be a DTD, and Σ a set of $\mathcal{C}_{K,FK}$ constraints over D. We define another DTD $D' = (E', A', P', R', r)$, where

$E' = E \cup \{D_Y, E_X\}$		where D_Y, E_X are fresh element types
$A' = A \cup \{K\}$		where K is a fresh attribute
$P'(r)$	$= P(r), D_Y, D_Y, E_X$	i.e., $P(r)$ followed by two D_Y elements and an E_X element
$P'(\tau)$	$= P(\tau)$	for all $\tau \in E \setminus \{r\}$
$P'(D_Y)$	$= P'(E_X) = \epsilon$	
$R'(D_Y)$	$= \{K\}$	
$R'(E_X)$	$= \{K\}$	
$R'(\tau)$	$= R(\tau)$	for all $\tau \in E$

We define a unary key φ_1, a unary inclusion constraint φ_2 and another key over D' as follows:

$$\varphi_1 = D_Y.K \to D_Y, \qquad \varphi_2 = D_Y.K \subseteq E_X.K, \qquad \phi = E_X.K \to E_X.$$

Obviously, every $\mathcal{C}_{K,FK}$ constraint over D is also a $\mathcal{C}_{K,FK}$ constraint over D'. Thus Σ is also a set of $\mathcal{C}_{K,FK}$ constraints over D'. We show that

1. Σ is satisfiable over D iff $\bigwedge \Sigma \wedge \phi \wedge \varphi_2 \wedge \neg\varphi_1$ is satisfiable over D';

2. Σ is satisfiable over D iff $\bigwedge \Sigma \wedge \phi \wedge \varphi_1 \wedge \neg\varphi_2$ is satisfiable over D'.

For if these hold, then the encoding is a reduction from the consistency problem for $\mathcal{C}_{K,FK}$ to the complements of the implication problems described in Lemma 3.3. Lemma 3.3 holds if the complements of the implication problems are undecidable.

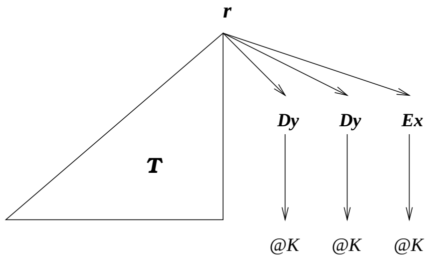

Figure 3: A tree used in the proof of Lemma 3.3

We prove (1) as follows. If there exists a tree $T \models D'$ and $T \models \bigwedge \Sigma \wedge \phi \wedge \varphi_2 \wedge \neg\varphi_1$, then we construct another tree T' by removing D_Y, E_X elements from T. Obviously, $T' \models D$ and $T' \models \Sigma$. Conversely, suppose that there is a tree $T \models D$ and $T \models \Sigma$. We construct another tree T' from T as shown in Figure 3. Let us refer to the two D_Y elements in T' as d_1, d_2, and the E_X element as e. Let $d_1.K = d_2.K = e.K$. Then it is easy to see that $T' \models D'$, $T' \models \Sigma$ and $T' \models \phi \wedge \varphi_2 \wedge \neg\varphi_1$.

We now prove (2). As above, we can show that if there is a tree $T \models D'$ and $T \models \bigwedge \Sigma \wedge \phi \wedge \varphi_1 \wedge \neg\varphi_2$, then there exists another tree T' such that $T' \models D$ and $T' \models \Sigma$. Conversely, suppose that there is a tree $T \models D$ and $T \models \Sigma$. We construct a tree T' from T as shown in Figure 3. Again we refer to the two D_Y elements in T' as d_1, d_2, and the E_X element as e. Now let $d_1.K \neq d_2.K$. Then it is easy to see that $T' \models D'$, $T' \models \Sigma$ and $T' \models \phi \wedge \varphi_1 \wedge \neg\varphi_2$.

This completes the proof of Lemma 3.3. □

From Lemma 3.3 we immediately obtain:

Corollary 3.4: *For $\mathcal{C}_{K,FK}$ constraints, the implication problem is undecidable.* □

3.3 PTIME decidable cases

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.

Theorem 3.5: *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 $\mathcal{C}_{K,FK}$.*

□

Proof: To prove statement (1) of this theorem, we first define the following notion. It should be noted here that by (XML) trees, we mean *finite* XML trees. Let $D = (E, A, P, R, r)$ be a DTD and α be a regular expression over $E \cup \{\mathtt{S}\}$. We say that α *yields a tree* if one of the following is satisfied:

- α is one of ϵ, $\mathtt{S}$ or α_1^*;
- α is some τ in E and $P(\tau)$ yields a tree;
- $\alpha = (\alpha_1, \alpha_2)$ and both α_1 and α_2 yield trees;
- $\alpha = (\alpha_1 \mid \alpha_2)$ and either α_1 or α_2 yields a tree.

It suffices to verify:

Claim 1. There exists an XML tree valid w.r.t. $D = (E, A, P, R, r)$ iff $P(r)$ yields a tree.

Claim 2. Given an arbitrary DTD $D = (E, A, P, R, r)$, it can be decided in linear time whether $P(r)$ yields a tree.

We first show Claim 1. If $P(r)$ yields a tree, one can easily construct an XML tree valid w.r.t. D. Conversely, suppose that there exists an XML tree T valid w.r.t. D. We define the *depth* of T to be the number of nodes in the longest path from the root to a leaf of T. Since T is finite, its depth is some natural number n. We show by induction on n that $P(r)$ yields a tree.

Basis: $n = 1$. Since T is valid w.r.t. D, ϵ must be in the regular language $L(P(r))$ defined by $P(r)$. By induction on the structure of a regular expression α, we can verify that α yields a tree if $\epsilon \in L(\alpha)$.

Inductive step: Assume the statement when $n = m$. We show that it also holds for $n = m + 1$. Let $[v_1, ..., v_l]$ be the subelements of the root of T. Since T is valid w.r.t. D, $lab(v_1) ... lab(v_l)$ must be in $L(P(r))$. Observe that for each $i \in [1, l]$, the subtree of T rooted at v_i has a depth at most m. Given these, by induction on structure of $P(r)$ and induction hypothesis, we can verify that $P(r)$ yields a tree.

We next show Claim 2. We use a boolean recursive function *finite-tree*(α) to check whether a regular expression α over $E \cup \{\mathtt{S}\}$ yields a tree. For each $\tau \in E$, we use two global boolean variables: $checked(\tau)$ to indicate whether τ has been checked by the function, and $finite(\tau)$ to indicate whether τ yields a tree. These are used to ensure that each τ in E is checked at most once by the function. Initially, $checked(\tau)$ and $finite(\tau)$ have value $false$ for each $\tau \in E$.

```
function finite-tree(α)
    if α is one of ε, S or α1*
        then return true
    else if α = (α1, α2)
        then return (finite-tree(α1) and finite-tree(α2))
    else if α = (α1 | α2)
        then return (finite-tree(α1) or finite-tree(α2))
    else if α = τ in E
        then (1) if checked(τ)
                    then return finite(τ);
             (2) checked(τ) := true;
```

(3) $finite(\tau) \quad := \textit{finite-tree}(P(\tau));$
(4) return $finite(\tau);$
else return $false;$

It is easy to verify that $P(r)$ yields a tree if and only if $\textit{finite-tree}(P(r))$ returns true. In addition, the function call $\textit{finite-tree}(P(r))$ takes $O(p)$ time, where p is the size of the function P. This is because for each τ in E, $P(\tau)$ is checked at most once by the function.

From these follows immediately the first statement of Theorem 3.5.

We next prove the second statement of Theorem 3.5. That is, the consistency problem for keys in $\mathcal{C}_{K,FK}$ is decidable in linear time. Given any DTD D and any set Σ of keys in $\mathcal{C}_{K,FK}$ over D, it suffices to show that Σ can be satisfied by an XML tree valid w.r.t. D if and only if there exists an XML tree valid w.r.t. D. For if it holds, then the second statement follows immediately from the first statement of Theorem 3.5.

We now show the claim. Suppose that there exists an XML tree $T_1 = (V, \textit{lab}, \textit{ele}, \textit{att}, \textit{val}, \textit{root})$ valid w.r.t. D. We construct another XML tree T_2 by modifying the val function in T_1 such that for any constraint $\tau[X] \rightarrow \tau$ in Σ, $|ext(\tau)| = |ext(\tau.l)|$ for every $l \in X$. That is, $T_2 \models \tau.l \rightarrow \tau$ for all $l \in X$. More specifically, let $T_2 = (V, \textit{lab}, \textit{ele}, \textit{att}, \textit{val}', \textit{root})$. Observe that the only difference between T_1 and T_2 is the definition of the function val'. For any v_1, v_2 in V with $lab(v_1) = \tau$ and $lab(v_2) = \tau$, we can make $val'(att(v_1, l)) \neq val'(att(v_2, l))$ for any $l \in X$. Let $val'(v) = val(v)$ for all other vertices in V. It is easy to verify that T_2 is valid w.r.t. D since T_1 is valid w.r.t. D. In addition, $T_2 \models \tau[X] \rightarrow \tau$ since for any $x, y \in ext(\tau)$, $x[X] \neq y[X]$. The other direction is immediate.

Finally, we prove the last statement of Theorem 3.5. That is, the implication problem for keys in $\mathcal{C}_{K,FK}$ is decidable in linear time. To show this, we need the following lemma.

Lemma 3.6: *For any DTD D and element type τ in D, it is decidable in linear time whether there is an XML tree T such that $T \models D$ and moreover, $|ext(\tau)| > 1$ in T.* □

Proof: The proof is a generalization of the proof of the first statement of Theorem 3.5 given above. Let $D = (E, A, P, R, r)$ be a DTD and $\tau \in E$. To show the lemma, we first introduce some notations. Let $M = \{0, 1, x\}$ be a linear-ordered set with $0 < 1 < x$, and $\oplus, \otimes$ be two binary operators on M defined by:

$$v_1 \oplus v_2 = \begin{cases} 0 & \text{if } v_1 = v_2 = 0 \\ 1 & \text{if } v_1 = 0, v_2 = 1 \text{ or } v_1 = 1, v_2 = 0 \\ x & \text{otherwise} \end{cases}$$

$$v_1 \otimes v_2 = \begin{cases} 0 & \text{if } v_1 = 0 \text{ or } v_2 = 0 \\ 1 & \text{if } v_1 = v_2 = 1 \\ x & \text{otherwise} \end{cases}$$

Given a regular expression α over $E \cup \{\mathtt{S}\}$, we define *the maximum number of τ elements* in α to be a value of M, denoted by $num(\alpha, \tau)$, as follows:

- if α is either ϵ or $\mathtt{S}$, then let $num(\alpha, \tau) = 0$;
- suppose α is $\tau' \in E$ and $P(\tau')$ yields a tree; if $\tau' \neq \tau$, let $num(\alpha, \tau) = num(P(\tau'), \tau)$; otherwise (i.e., if $\tau' = \tau$) let $num(\alpha, \tau) = num(P(\tau), \tau) \oplus 1$;

- if $\alpha = (\alpha_1, \alpha_2)$ and both α_1 and α_2 yield trees, then let $num(\alpha, \tau) = num(\alpha_1, \tau) \oplus num(\alpha_2, \tau)$;
- suppose $\alpha = (\alpha_1 \,|\, \alpha_2)$ and either α_1 or α_2 yields a tree; for $i = 1, 2$, let $f(\alpha_i, \tau) = num(\alpha_i, \tau)$ if α_i yields a tree, and $f(\alpha_i, \tau) = 0$ otherwise; let $num(\alpha, \tau) = max(f(\alpha_1, \tau), f(\alpha_2, \tau))$;
- if α is α_1^*, then let $num(\alpha, \tau) = num(\alpha_1, \tau) \otimes x$ if α_1 yields a tree, and let $num(\alpha, \tau) = 0$ otherwise;
- $num(\alpha, \tau)$ is 0 otherwise.

To prove the lemma, it suffices to show the following:

Claim 1. There exists an XML tree valid w.r.t. D with $|ext(\tau)| > 1$ iff $P(r)$ yields a tree and $num(P(r), \tau) = x$.

Claim 2. It is decidable in linear time whether $P(r)$ yields a tree and $num(P(r), \tau) = x$.

We first show Claim 1. If $P(r)$ yields a tree and $num(P(r), \tau) = x$, then we can easily construct a tree T such that $T \models D$ and $|ext(\tau)| > 1$, following the definition of the function num given above. Conversely, if there exists a tree T such that $T \models D$ and $|ext(\tau)| > 1$, we can verify by induction on the depth of T that $P(r)$ yields a tree and $num(P(r), \tau) = x$, where the *depth* of T is the number of nodes in the longest path from the root to a leaf of T.

We next show Claim 2. We use a recursive function $ftree(\alpha)$ to check whether a regular expression α over $E \cup \{\mathtt{S}\}$ yields a tree and if so, we compute $num(\alpha, \tau)$. The function returns a pair (b, m), where b is a boolean indicating whether α yields a tree, and m is the value of $num(\alpha, \tau)$ if b is true. For each $\tau' \in E$, we are to use two global boolean variables: $checked(\tau')$ to indicate whether τ' has been checked by the function, and $finite(\tau')$ to indicate whether τ' yields a tree. These are used to ensure that each τ' in E is checked at most once by the function. Initially, $checked(\tau')$ and $finite(\tau')$ have value $false$ for each $\tau' \in E$. We also use a global variable $num(\tau')$ for each $\tau' \in E$ to record $num(P(\tau'), \tau)$. Initially, $num(\tau') = 0$.

```
function ftree(α)
    1. case α of
          either ϵ or S                                    |
               return (true, 0);
          τ' ∈ E                                           |
               (1) if checked(τ') and finite(τ') and τ' = τ
                     then return (true, num(τ) ⊕ 1)
                     else if checked(τ')
                     then return (finite(τ'), num(τ'));
               (2) checked(τ') := true;
               (3) (b, m)      := ftree(P(τ'));
               (4) finite(τ')  := b;
               (5) num(τ')     := m;
               (6) if b and τ = τ'
                     then return (true, m ⊕ 1);
                     else if b
                     then return (true, m);                |
```

(α_1, α_2)

(1) $(b_1, m_1) := \mathit{ftree}(\alpha_1)$;
(2) $(b_2, m_2) := \mathit{ftree}(\alpha_2)$;
(3) if b_1 and b_2
then return $(\mathit{true},\, m_1 \oplus m_2)$; |

$(\alpha_1 \,|\, \alpha_2)$

(1) $(b_1, m_1) := \mathit{ftree}(\alpha_1)$;
(2) $(b_2, m_2) := \mathit{ftree}(\alpha_2)$;
(3) if $b_1 = \mathit{false}$
then $m_1 := 0$;
(4) if $b_2 = \mathit{false}$
then $m_2 := 0$;
(5) if b_1 or b_2
then return $(\mathit{true},\, \mathit{max}(m_1, m_2))$; |

α_1^*

(1) $(b_1, m_1) := \mathit{ftree}(\alpha_1)$;
(2) if b_1
then return $(b_1, m_1 \otimes x)$
else return $(\mathit{true}, 0)$;

2. return $(\mathit{false}, 0)$;

It is easy to verify that $P(r)$ yields a tree with $\mathit{num}(P(r), \tau) = x$ if and only if $\mathit{ftree}(P(r))$ returns (true, x). In addition, the function *ftree* is called at most $O(p)$ time, where p is the size of the function P. This is because for each τ' in E, $P(\tau')$ is checked at most once by the function. □

Let Σ be a set of keys in $\mathcal{C}_{K,FK}$ over D, and $\varphi = \tau[X] \rightarrow \tau$ be another key in $\mathcal{C}_{K,FK}$ over D. We say that Σ *contains a superkey of* φ if there is $\tau[Y] \rightarrow \tau$ in Σ such that $X \subseteq Y$, i.e., X is a subset of Y. Using this and Lemma 3.6 we can prove the following:

Lemma 3.7: *Let D be a DTD, Σ a set of keys in $\mathcal{C}_{K,FK}$ over D, and $\varphi = \tau[X] \rightarrow \tau$ another key in $\mathcal{C}_{K,FK}$ over D. There is an XML tree T such that $T \models D$, $T \models \Sigma$ but $T \models \neg\varphi$ if and only if Σ does not contain a superkey of φ and moreover, there is an XML tree T' such that $T' \models D$ and $|ext(\tau)| > 1$ in T'. In addition, this is decidable in linear time in the sizes of D and $\Sigma \cup \{\varphi\}$.* □

Proof: We first show that there is an XML tree T such that $T \models D$, $T \models \Sigma$ but $T \models \neg\varphi$ iff Σ does not contain a superkey of φ and moreover, there is an XML tree T' such that $T' \models D$ and $|ext(\tau)| > 1$ in T'. Suppose that there is an XML tree T such that $T \models D$, $T \models \Sigma$ and $T \models \neg\varphi$. Then obviously, T is valid w.r.t. D, and moreover, there must be at least two τ elements d_1, d_2 in T such that $d_1[X] = d_2[X]$ but $d_1 \neq d_2$ since $T \models \neg\varphi$. Thus there must be $|ext(\tau)| > 1$ in T. In addition, Σ cannot contain $\tau[Y] \rightarrow \tau$ with $X \subseteq Y$, since otherwise it would contradict $T \models \neg\varphi$ and $T \models \Sigma$. Conversely, let T' be a tree such that $T' \models D$ and $|ext(\tau)| > 1$ in T'. Thus there are at least two τ elements d_1, d_2 in T'. Let $d_1[X] = d_2[X]$ but assign different string values to all other attributes in T'. Let us refer to the tree constructed this way as T. It is easy to verify that $T \models D$ and $T \models \neg\varphi$ by the definition of T. Moreover, $T \models \Sigma$ by the definition of T and the fact that $d_1[X] = d_2[X]$ does not violate any keys in Σ, given the assumption that Σ does not contain a superkey of φ. Thus the first statement of the lemma holds.

To show that this can be determined in linear time, observe that by Lemma 3.6, it can be decided in linear time in the size of D whether there is a tree T such that $T \models D$ and $|ext(\tau)| > 1$ in T. In addition, it is decidable in linear time in the size of $\Sigma \cup \{\varphi\}$ whether Σ contains a superkey of φ (see e.g., [1] for discussions about a linear time algorithm for checking implication of functional dependencies). Thus it is decidable in linear time in the sizes of D and $\Sigma \cup \{\varphi\}$ whether these conditions hold. This completes the proof of the lemma. $\Box$

Lemma 3.7 suffices to prove the third statement of Theorem 3.5 because $(D, \Sigma) \vdash \varphi$ iff there is no XML tree T such that $T \models D$, $T \models \Sigma$ but $T \models \neg\varphi$. By Lemma 3.7, the later can be decided in linear time.

This completes the proof of Theorem 3.5. $\Box$

Given Theorem 3.5, 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 [15]. In particular, when nonnegative integer solutions are considered, [23] 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. A road map of the proof is as follows. 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 [18];
- 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.

Proof of Theorem 4.1: We first present an encoding of unary constraints and DTDs, and then develop a characterization of XML specifications with both DTDs and unary constraints in terms of linear equalities and inequalities.

Encoding unary constraints

Consider a DTD $D = (E,\, A,\, P,\, R,\, r)$ and a set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D. Referring to an arbitrary XML tree T valid w.r.t. D, we derive from Σ a class of cardinality constraints on T, denoted by C_Σ, as follows. For any $\varphi \in \Sigma$,

- if φ is a key constraint $\tau.l \to \tau$, then $|ext(\tau.l)| = |ext(\tau)|$ is in C_Σ;
- if φ is an inclusion constraint $\tau_1.l_1 \subseteq \tau_2.l_2$, then $|ext(\tau_1.l_1)| \leq |ext(\tau_2.l_2)|$ is in C_Σ.

Moreover, for any $\tau \in E$ and $l \in R(\tau)$, $|ext(\tau.l)| \leq |ext(\tau)|$ and $0 \leq |ext(\tau.l)|$ are in C_Σ. We refer to C_Σ as *the cardinality constraints determined by* Σ. We use $T \models C_\Sigma$ to denote that T satisfies all constraints of C_Σ.

For example, recall the set Σ_1 of $\mathcal{C}^{Unary}_{K,FK}$ constraints over the DTD D_1 given in Section 1. The set of cardinality constraints determined by Σ_1, denoted by C_{Σ_1}, consists of:

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

The lemma below shows that C_Σ characterizes consistency of Σ. It allows us to consider C_Σ instead of Σ when studying consistency of Σ.

Lemma 4.3: *Let D be a DTD, Σ be a set of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, and C_Σ be the set of cardinality constraints determined by Σ. Then the following two statements are equivalent.*

1. *There exists an XML tree T_1 such that $T_1 \models D$ and $T_1 \models \Sigma$.*
2. *There exists an XML tree T_2 such that $T_2 \models D$ and $T_2 \models C_\Sigma$.*

In addition, any XML tree valid w.r.t. D and satisfying Σ also satisfies C_Σ. □

Proof: We show (1) $\Leftrightarrow$ (2).

(1) $\Rightarrow$ (2). This direction is immediate. Given an XML tree T_1 such that $T_1 \models D$ and $T_1 \models \Sigma$, it is easy to verify that $T_1 \models C_\Sigma$. Thus (2) holds.

(2) $\Rightarrow$ (1). Let $T_2 = (V, lab, ele, att, val, root)$ be an XML tree valid w.r.t. D and satisfying C_Σ. We construct an XML tree $T_1 = (V, lab, ele, att, val', root)$ such that $T_1 \models \Sigma$. Observe that the only difference between T_1 and T_2 is the definition of the function val'. The function val' has the following property: for any $v \in V$,

$$
val'(v) = \begin{cases} val(v) & \text{if } lab(v) \notin A \\ \text{a string} & \text{otherwise} \end{cases}
$$

This suffices given $T_1 \models \Sigma$ since it is easy to verify that T_1 is an XML tree valid w.r.t. D by the assumption about T_2 and by Definition 2.2.

We next define the function val' such that $T_1 \models \Sigma$. To do so, let $S = \{\tau.l \mid \tau \in E, l \in R(\tau)\}$. We define a partial order $\preceq$ on S as follows:

- $\tau.l \preceq \tau.l$,
- $\tau_1.l_1 \preceq \tau_2.l_2$ if $T_2 \models |ext(\tau_1.l_1)| \leq |ext(\tau_2.l_2)|$,
- $\tau_1.l_1 \preceq \tau_3.l_3$ if $\tau_1.l_1 \preceq \tau_2.l_2$ and $\tau_2.l_2 \preceq \tau_3.l_3$.

It is easy to verify that $\preceq$ is indeed a partial order on S. Using $\preceq$ we define an equivalence relation $\sim$ on S as follows:

$$
\tau_1.l_1 \sim \tau_2.l_2 \quad \text{iff} \quad \tau_1.l_1 \preceq \tau_2.l_2 \text{ and } \tau_2.l_2 \preceq \tau_1.l_1
$$

Let $[\tau.l]$ denote the equivalence class of $\tau.l$ with respect to $\sim$, and let $S_{[\,]} = \{[\tau.l] \mid \tau.l \in S\}$. We now extend $\preceq$ to $S_{[\,]}$ such that $[\tau_1.l_1] \preceq [\tau_2.l_2]$ iff $\tau_1.l_1 \preceq \tau_2.l_2$. We then topologically sort $S_{[\,]}$ based on $\preceq$. For each $[\tau.l]$, let $s[\tau.l]$ be a distinct set. Based on the topological order, we assign a set of string values to each $s[\tau.l]$ using the algorithm below:

1. for each $s[\tau.l] \in S_{[\,]}$ do
   $s[\tau.l] := \emptyset$;
2. for each $s[\tau.l] \in S_{[\,]}$ in the topological order do
   (1) add distinct string values to $s[\tau.l]$ such that $|s[\tau.l]|$ is equal to $|ext(\tau.l)|$ in T_2;
   (2) for each $s[\tau'.l'] \in S_{[\,]}$ such that $[\tau.l] \preceq [\tau'.l']$ do
   $s[\tau'.l'] := s[\tau'.l'] \cup s[\tau.l]$;

Now for each $\tau \in E$, $l \in R(\tau)$ and $x \in ext(\tau)$, let $val'(att(x, l))$ be a string value in $s[\tau.l]$ such that in T_1, $|s[\tau.l]| = |ext(\tau.l)|$. This is possible since $T_2 \models C_\Sigma$ and moreover, $ext(\tau)$ in T_1 is the same as $ext(\tau)$ in T_2 by the definition of T_1. Given the definitions of T_1 and val' above, it is easy to verify that $T_1 \models C_\Sigma$. In addition, it can be verified that $T_1 \models \Sigma$. More specifically, for any $\varphi \in \Sigma$, (1) if φ is $\tau.l \to \tau$, then for any distinct $x, y \in ext(\tau)$, $x.l \neq y.l$ by the cardinality constraint $|ext(\tau.l)| = |ext(\tau)|$ and the definition of val'; (2) if φ is $\tau_1.l_1 \subseteq \tau_2.l_2$, then $T_2 \models |ext(\tau_1.l_1)| \leq |ext(\tau_2.l_2)|$ by $T_2 \models C_\Sigma$. Thus for any $x \in ext(\tau_1)$, we have that $x.l_1 \in ext(\tau_2.l_2)$ by the definition of val'. Therefore, T_1 is indeed an XML tree such that $T_1 \models D$ and $T_1 \models \Sigma$. Thus (1) holds. $\Box$

It is straightforward to verify the following.

Lemma 4.4: *Given any set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints over a DTD D, the set C_Σ of cardinality constraints determined by Σ can be computed in linear time in terms of the sizes of Σ and D.* $\Box$

Encoding DTDs

For any DTD, we encode it in PTIME with a set of cardinality constraints such that the DTD has an XML tree valid w.r.t. it if and only if the set of cardinality constraints is consistent. We encode DTDs in two steps.

Eliminating Kleene closures. Element type definitions in a DTD are defined in terms of regular expressions. To simplify the discussion when it comes to the study of cardinality constraints imposed by DTDs, we want to rewrite element type definitions to eliminate occurrences of the Kleene star. Moreover, the rewriting should not affect consistency analysis of $\mathcal{C}^{Unary}_{K,IC}$ constraints. More specifically, given a DTD D, we define another DTD D^- such that D^- does not contain the Kleene star and in addition, for any set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, there exists an XML tree valid w.r.t. D and satisfying Σ if and only if there exists an XML tree valid w.r.t. D^- and satisfying Σ.

We next present a rewriting that preserves consistency of $\mathcal{C}^{Unary}_{K,IC}$ constraints. Let $D = (E, A, P, R, r)$ be a DTD. For each element type $\tau \in E$, the definition of τ, $P(\tau)$, is a regular expression α. We define another element type definition function P^-: $P^-(\tau) = f(\alpha)$, where $f(\alpha)$ is defined by:

$$f(\alpha) = \begin{cases} \alpha & \text{if } \alpha \text{ is } \mathtt{S},\ \tau' \text{ or } \epsilon \\ f(\alpha_1), f(\alpha_2) & \text{if } \alpha = \alpha_1, \alpha_2 \\ f(\alpha_1) \mid f(\alpha_2) & \text{if } \alpha = \alpha_1 \mid \alpha_2 \\ \tau_\alpha & \text{if } \alpha = \alpha_1^* \end{cases}$$

Here τ_α is a new element type and

$$P^-(\tau_\alpha) \;=\; \epsilon \mid \alpha_1,\, \tau_\alpha.$$

That is, we eliminate occurrences of the Kleene star by introducing new element types and by rewriting Kleene closures in terms of direct recursion. We refer to the function f as *the transition function.*

Formally, we define *the $*$-free DTD of D*, denoted by D^-, to be $D^- \;=\; (E^-, A, P^-, R^-, r)$, where

- $E^- = E \cup E_a$, and E_a is the set of new element types τ_α's introduced in the process of eliminating occurrences of the Kleene star, as described above;
- P^- is defined in terms of P and the transition function f as described above;
- $R^-(\tau) = R(\tau)$ for all $\tau \in E$ and $R^-(\tau) = \emptyset$ for all $\tau \in E_a$.

Observe that the element type r of the root and the set A of attributes remain unchanged in D^-.

For example, the $*$-free DTD of D_1 given in Section 1 is $D_1^- \;=\; (E_1^-, A_1, P_1^-, R_1^-, r)$, where

$$\begin{array}{ll}
E_1^- = \{teachers, teacher, teach, research, subject, \tau_t\} & \\
A_1 = \{name,\, taught_by\} & \\
P_1^-(teachers) & = teacher,\, \tau_t \\
P_1^-(\tau_t) & = \epsilon \mid teacher,\, \tau_t \\
P_1^-(teacher) & = teach,\, research \\
P_1^-(teach) & = subject,\, subject \\
P_1^-(subject) & = P_1^-(research) = \mathtt{S} \\
R_1^-(teacher) & = \{name\} \\
R_1^-(subject) & = \{taught_by\} \\
R_1^-(teachers) & = R_1^-(teach) = R_1^-(research) = R_1^-(\tau_t) = \emptyset \\
r_1 = teachers &
\end{array}$$

Here τ_t is the only new element type introduced.

The $*$-free DTD of D_2 given in Section 1 is D_2 itself, because D_2 does not contain any Kleene star.

Obviously, any set Σ of $\mathcal{C}_{K,IC}^{Unary}$ constraints over D is also a set of $\mathcal{C}_{K,IC}^{Unary}$ constraints over the $*$-free DTD D^- of D. The next lemma shows that D^- preserves consistency of Σ. It allows us to consider $*$-free DTDs only, without loss of generality.

Lemma 4.5: *Let D be a DTD, D^- the $*$-free DTD of D, and Σ a set of $\mathcal{C}_{K,IC}^{Unary}$ constraints over D. Then the following two statements are equivalent.*

1. *There exists an XML tree T_1 such that $T_1 \models D$ and $T_1 \models \Sigma$.*
2. *There exists an XML tree T_2 such that $T_2 \models D^-$ and $T_2 \models \Sigma$.*

$\square$

Proof: We first show the following:

Claim: Let D be a DTD and D^- be the $*$-free DTD of D. Then there exists an XML tree $T_1 \models D$ iff there exists an XML tree $T_2 \models D^-$.

To verify the claim, first assume that there exists an XML tree $T_1 = (V_1, lab_1, ele_1, att, val, root)$ such that $T_1 \models D$. We construct an XML tree

$$T_2 = (V_2, lab_2, ele_2, att, val, root)$$

by modifying T_1 such that $T_2 \models D^-$. We proceed to construct T_2 as follows. Initially, let $V_2 = V_1$, $lab_2 = lab_1$ and $ele_2 = ele_1$. Note that att, val and $root$ are the same in T_1 and T_2. We traverse T_2 starting from $root$. For each $v \in V_2$ such that $lab(v) = \tau$ and $\tau \in E$, we increment V_2, lab_2 and ele_2 as follows. Assume $P(\tau) = \alpha$ and $ele(v) = [v_1, ..., v_n]$. Let $w = lab(v_1) ... lab(v_n)$. Since $T_1 \models D$, by Definition 2.2, w must be in the regular language defined by α. By the definitions of the transition function f and the function P^- for element type definitions in D^-, it is easy to verify that w can be derived from the rule $P^-(\tau) = f(\alpha)$ and the related rules for new element types in E_a introduced by $f(\alpha)$. Let t be a derivation tree of w. Without loss of generality, assume the root of t is v, the leaves of t are v_1, ..., v_n, and internal nodes of t are x_1, ..., x_m. Observe that these internal nodes are labeled with element types in E_a, i.e., $E^- \setminus E$. We modify T_2 by substituting t for v. More specifically, we (i) increment V_2 by including x_1, ..., x_m; (ii) increment lab_2 by defining $lab_2(x_i)$ to be the label of x_i in t, for each $i \in [1, m]$; and (iii) increment ele_2 according to the structure of t, i.e., by substituting the children of v in t for $[v_1, ..., v_n]$ and by defining $ele_2(x_i)$ to be the children of x_i in t, for $i \in [1, m]$. Let T_2 be the tree obtained at the end of the traversal. It is easy to verify that T_2 is finite and in addition, $T_2 \models D^-$.

Conversely, let $T_2 = (V_2, lab_2, ele_2, att, val, root)$ such that $T_2 \models D^-$. We construct an XML tree

$$T_1 = (V_1, lab_1, ele_1, att, val, root)$$

by modifying T_2 such that $T_1 \models D$. We construct T_1 as follows. For any node $v \in V_2$ with $lab(v) = \tau$ and $\tau \in E_a$, i.e., $E^- \setminus E$, we substitute the subelements of v for v in $ele(v')$, where v' is the parent of v. In addition, we delete (i) v from V_2, (ii) $lab_2(v)$ from lab_2, and (iii) $ele_2(v)$ from ele_2. We repeat the process until there is no node labeled with element type in $E^- \setminus E$. Now let V_1, lab_1 and ele_1 be V_2, lab_2 and ele_2 obtained at the end of the process, respectively. Observe that att, val and $root$ remain unchanged. It is easy to verify that T_1 is indeed an XML tree valid w.r.t. D.

Using the claim, we proceed to show Lemma 4.5.

(1) $\Rightarrow$ (2). Assume that there exists an XML tree T_1 such that $T_1 \models D$ and $T_1 \models \Sigma$. We show that there exists an XML tree T_2 such that $T_2 \models D^-$ and $T_2 \models \Sigma$. By Lemma 4.3, there is an XML tree T_1' such that $T_1' \models D$ and $T_1' \models C_\Sigma$, where C_Σ is the set of cardinality constraints determined by Σ. By the claim, we can construct an XML tree T_2' such that $T_2' \models D^-$. Moreover, observe that the construction given in the proof of the claim does not alter $ext(\tau)$ and $ext(\tau.l)$ for any $\tau \in E$ and $l \in R(\tau)$. Therefore, by $T_1' \models C_\Sigma$ we have $T_2' \models C_\Sigma$. Thus again by Lemma 4.3, there is an XML tree T_2 such that $T_2 \models D^-$ and $T_2 \models \Sigma$. That is, (2) holds.

(2) $\Rightarrow$ (1). Suppose that there exists an XML tree T_2 such that $T_2 \models D^-$ and $T_2 \models \Sigma$. We show that there exists an XML tree T such that $T \models D$ and $T \models \Sigma$. The proof is similar to the one given above. Consider the construction of T_1 from T_2 given in the proof of the claim. This construction does not alter $ext(\tau)$ and $ext(\tau.l)$ for any $\tau \in E$ and $l \in R(\tau)$. Thus $T_1 \models C_\Sigma$ iff $T_2 \models C_\Sigma$. Hence by Lemma 4.3, there exists an XML tree T such that $T \models D$ and $T \models \Sigma$. Therefore, (1) holds if (2) holds. $\Box$

Finally, it is easy to verify the following.

Lemma 4.6: *Given any DTD D, its $*$-free DTD D^- can be computed in linear time.* □

Characterizing DTDs. Next, we show that cardinality constraints can also be used to characterize DTDs. Without loss of generality by Lemma 4.5, we need to consider $*$-free DTDs only.

Consider a DTD $D = (E, A, P, R, r)$. To define cardinality constraints to characterize D, we first describe unknowns to be used in these constraints. For each symbol $\tau \in E \cup \{\mathtt{S}\}$, we assume a set of unknowns $X_\tau = \{x^i_\tau \mid i \in \mathbb{N}\}$, where $\mathbb{N}$ is the set of natural numbers. For each $\tau \in E$, consider the regular expression $\alpha = P(\tau)$. We define *the set of regular expressions in* $P(\tau)$, denoted by S_τ, to be the set consisting of (1) α, (2) regular expressions in α_1 or α_2, if $\alpha = (\alpha_1, \alpha_2)$ or $\alpha = (\alpha_1 \mid \alpha_2)$. For the regular expressions in S_τ, we also assume a set of unknowns $Y_\tau = \{y^\tau_{\beta,k} \mid \beta \in S_\tau,\, k \in \mathbb{N}\}$.

Using these unknowns, we define a function g from S_τ and Y_τ to sets of cardinality constraints as follows:

$$g(\beta,\, y^\tau_{\beta,k}) \;=\; \begin{cases} \{y^\tau_{\beta,k} = x^i_{\tau'}\} & \text{if } \beta \text{ is } \tau' \\ \{y^\tau_{\beta,k} = x^i_S\} & \text{if } \beta \text{ is } \mathtt{S} \\ \{y^\tau_{\beta,k} = y^\tau_{\epsilon,j}\} & \text{if } \beta \text{ is } \epsilon \\ \{y^\tau_{\beta,k} = y^\tau_{\beta_1,k_1},\ y^\tau_{\beta,k} = y^\tau_{\beta_2,k_2}\} \,\cup\, g(\beta_1,\, y^\tau_{\beta_1,k_1}) \,\cup\, g(\beta_2,\, y^\tau_{\beta_2,k_2}) & \text{if } \beta = \beta_1, \beta_2 \\ \{y^\tau_{\beta,k} = y^\tau_{\beta_1,k_1} + y^\tau_{\beta_2,k_2}\} \,\cup\, g(\beta_1,\, y^\tau_{\beta_1,k_1}) \,\cup\, g(\beta_2,\, y^\tau_{\beta_2,k_2}) & \text{if } \beta = \beta_1 \mid \beta_2 \end{cases}$$

Here $x^i_{\tau'} \in X_{\tau'}$, $x^i_S \in X_{\mathtt{S}}$, and $y^\tau_{\epsilon,j}$, $y^\tau_{\beta_1,k_1}$, $y^\tau_{\beta_2,k_2} \in Y_\tau$ are fresh variables.

The set of cardinality constraints determined by τ, denoted by ψ_τ, consists of the following:

- $|ext(\tau)| = y^\tau_{\alpha,1}$, where $\alpha = P(\tau)$;
- constraints in $g(\alpha,\, y^\tau_{\alpha,1})$;
- $x^i_{\tau'} \geq 0$ and $y^\tau_{\beta,k} \geq 0$ for all $x^i_{\tau'}$ and $y^\tau_{\beta,k}$ in $g(\alpha,\, y^\tau_{\alpha,1})$.

Intuitively, constraints of ψ_τ describe the grammar $P(\tau)$ in a quantitative way. Referring to an XML tree T valid w.r.t. D, let us consider the subelements $\vec{v} = [v_1, ..., v_n]$ of a node in T labeled τ. The labels of the subelements, $lab(\vec{v}) = [lab(v_1), ..., lab(v_n)]$, must observe $P(\tau)$. Moreover, a subsequence of $lab(\vec{v})$ may correspond to a certain occurrence of a regular expression β in S_τ. For example, if $P(\tau) = (\beta_1, \beta_2)$, then there is $m \in [1, n]$ such that $[lab(v_1), ..., lab(v_m)]$ and $[lab(v_{m+1}), ..., lab(v_n)]$ are in the regular languages defined by β_1 and β_2, respectively. We say that $[v_1, ..., v_m]$ and $[v_{m+1}, ..., v_n]$ correspond to β_1 and β_2, respectively. In particular, if $\beta = \tau'$ for some $\tau' \in E \cup \{\mathtt{S}\}$, an occurrence of β may correspond to a node in T labeled τ'. The unknown $x^i_{\tau'}$ intends to indicate the number of all τ' nodes in T that correspond to a certain occurrence of τ' in $P(\tau)$. Similarly, $y^\tau_{\beta,k}$ keeps track of the number of subelement sequences that correspond to the kth occurrence of β in $P(\tau)$. Constraints of ψ_τ specify dependencies among these numbers. To illustrate this, let us consider the structure of $\alpha = P(\tau)$. Suppose that in T, there are $y^\tau_{\alpha,1}$ many nodes labeled τ. Suppose α is some $\tau' \in E \cup \{\mathtt{S}\}$, then each τ node has a distinct subelement labeled τ'. Thus there must be $y^\tau_{\alpha,1}$ many nodes labeled τ' that are subelements of τ nodes. This is exactly what the constraint $y^\tau_{\alpha,1} = x^i_{\tau'}$ says. Next, suppose $\alpha = (\beta_1, \beta_2)$. The subelements $\vec{v}$ of a τ node can be partitioned into $\vec{v} = \vec{v_1}\,\vec{v_2}$ such that $\vec{v_1}$ and $\vec{v_2}$ correspond to β_1 and β_2, respectively. Assume that the numbers of subelement sequences of τ nodes corresponding to β_1 and β_2 are $y^{\tau}_{\beta_1,k_1}$ and $y^{\tau}_{\beta_2,k_2}$, respectively. Then both $y^{\tau}_{\beta_1,k_1}$ and $y^{\tau}_{\beta_2,k_2}$ must equal to $y^{\tau}_{\alpha,1}$. The constraints $y^{\tau}_{\alpha,1} = y^{\tau}_{\beta_1,k_1}$ and $y^{\tau}_{\alpha,1} = y^{\tau}_{\beta_2,k_2}$ describe this quantitative relation. Finally, suppose $\alpha = (\beta_1 \,|\, \beta_2)$. The subelements of a τ node correspond to either β_1 or β_2. Assume that the numbers of subelements of τ nodes corresponding to β_1 and β_2 are $y^{\tau}_{\beta_1,k_1}$ and $y^{\tau}_{\beta_2,k_2}$, respectively. Then the sum of $y^{\tau}_{\beta_1,k_1}$ and $y^{\tau}_{\beta_2,k_2}$ must equal to $y^{\tau}_{\alpha,1}$. This is what the constraint $y^{\tau}_{\alpha,1} = y^{\tau}_{\beta_1,k_1} + y^{\tau}_{\beta_2,k_2}$ says.

The set of cardinality constraints determined by DTD D, denoted by Ψ_D, consists of the following:

- $|ext(r)| = 1$;
- constraints of ψ_τ for each $\tau \in E$;
- $|ext(\tau)| = \sum_{i \in [1, m^\tau]} x^i_\tau$ for each $\tau \in E \cup \{\mathtt{S}\}$, where without loss of generality, we assume x^1_τ, ..., $x^{m^\tau}_\tau$ are all unknowns of X_τ occurring in some $\psi_{\tau'}$.

Intuitively, referring to an XML tree T, the constraint $|ext(r)| = 1$ asserts that T has a unique root. For each $\tau \in E$, ψ_τ assures that the grammar $P(\tau)$ is observed. Consider $\tau \in E \cup \{\mathtt{S}\}$. Observe that each node of T labeled τ corresponds to a certain occurrence of τ in $P(\tau')$ for some element type τ'. Obviously, the set $ext(\tau)$ should include all τ vertices in T no matter what occurrences of τ they correspond to. This is exactly what the constraint $|ext(\tau)| = \sum_{1 \le i \le m^\tau} x^i_\tau$ says, where x^i_τ indicates the number of all τ nodes in T that correspond to a certain occurrence of τ.

By treating $|ext(\tau)|$ in Ψ_D as an unknown for each $\tau \in E \cup \{\mathtt{S}\}$, we can view Ψ_D as a system of linear equalities and inequalities. We say that Ψ_D is *consistent* if and only if Ψ_D admits an integer solution. That is, there is an integer assignment to each unknown $|ext(\tau)|$, x^i_τ and $y^{\tau}_{\beta,k}$ that satisfies all the equalities and inequalities in Ψ_D.

For example, let us consider DTDs D_1^- and D_2^- given above. The cardinality constraints determined by these DTDs are given below:

$$
\begin{array}{llll}
\Psi_{D_1}: & & & \\
\quad \psi_{teachers}: & |ext(teachers)| = x^1_{teacher} & |ext(teachers)| = x^1_{\tau_t} & \\
\quad \psi_{teacher}: & |ext(teacher)| = x^1_{teach} & |ext(teacher)| = x^1_{research} & \\
\quad \psi_{\tau_t}: & |ext(\tau_t)| = y^{\tau_t}_\epsilon + y^{\tau_t}_\beta & y^{\tau_t}_\beta = x^2_{teacher} & y^{\tau_t}_\beta = x^2_{\tau_t} \\
\quad \psi_{teach}: & |ext(teach)| = x^1_{subject} & |ext(teach)| = x^2_{subject} & \\
\quad \psi_{subject}: & |ext(subject)| = x^1_{\mathtt{S}} & & \\
\quad \psi_{research}: & |ext(research)| = x^2_{\mathtt{S}} & & \\
\quad \text{moreover,} & |ext(teachers)| = 1 & |ext(teacher)| = x^1_{teacher} + x^2_{teacher} & \\
 & |ext(\tau_t)| = x^1_{\tau_t} + x^2_{\tau_t} & |ext(teach)| = x^1_{teach} & \\
 & |ext(subject)| = x^1_{subject} + x^2_{subject} & |ext(research)| = x^1_{research} & \\
 & |ext(S)| = x^1_{\mathtt{S}} + x^2_{\mathtt{S}} & \text{all unknowns} \ge 0. &
\end{array}
$$

Observe, for example, $x^1_{teacher}$ indicates the number of `teacher` nodes that are children of `teachers` nodes, and $x^2_{teacher}$ stands for the number of `teacher` children of nodes labeled τ_t. The cardinality of $ext(teacher)$ equals to the sum of $x^1_{teacher}$ and $x^2_{teacher}$. Obviously, there is a unique node labeled `teachers`, i.e., the root. Hence we have $x^1_{teacher} = 1$ since the root has a unique `teacher` child. Thus $|ext(teacher)| = 1 + x^2_{teacher}$.

Ψ_{D_2}:

ψ_{db}: $|ext(db)| = x^1_{foo}$

ψ_{foo}: $|ext(foo)| = x^2_{foo}$

moreover, $|ext(db)| = 1$ $\quad |ext(foo)| = x^1_{foo} + x^2_{foo}$ $\quad$ all unknowns ≥ 0.

It is easy to check that Ψ_{D_1} is consistent, whereas Ψ_{D_2} is not.

We next show that Ψ_D indeed characterizes the DTD D.

Lemma 4.7: *Let D be a (*-fee) DTD and Ψ_D be the set of cardinality constraints determined by D. Then Ψ_D is consistent if and only if there is an XML tree valid w.r.t. D.* □

Proof: First, assume $T = (V, lab, ele, att, val, root)$ is an XML tree valid w.r.t. D. We show that Ψ_D admits an integer solution. It suffices to define an integer assignment to unknowns in Ψ_D that satisfies equalities and inequalities in Ψ_D. To do so, for each $\tau \in E$, let the unknown $|ext(\tau)|$ be the number of vertices in V labeled τ. Assume $P(\tau) = \alpha$. Since $T \models D$, for any vertex $v \in \tau$ with $lab(v) = \tau$ and $ele(v) = [v_1, ..., v_n]$, $lab(v_1)...lab(v_n)$ must be in the regular language defined by α. In other words, $[v_1, ..., v_n]$ observes the regular expression α. Thus for $i \in [1, n]$, we can mark v_i with $x^i_{\tau'}$ if v_i corresponds to a certain occurrence of τ' in α, following the definition of the function g. Similarly, we can mark a sequence of $[v_1, ..., v_n]$ with $y^\tau_{\beta,k}$ to indicate that the sequence corresponds to a certain occurrence of the regular expression β in α. More specifically, we mark $\vec{v} = [v_1, ..., v_n]$ with $y^\tau_{\alpha,1}$, and consider the structure of α as follows. If α is some $\tau' \in E \cup \{\mathtt{S}\}$, then n must equal to 1. We mark v_1 with $x^i_{\tau'}$, where $x^i_{\tau'}$ is the corresponding variable in $g(\alpha, y^\tau_{\alpha,1})$. If α is ϵ, then $n = 0$. We mark the empty sequence with $y^\tau_{\epsilon,k}$, where $y^\tau_{\epsilon,k}$ is the corresponding variable in $g(\alpha, y^\tau_{\alpha,1})$. If α is (β_1, β_2), then $\vec{v}$ can be partitioned into two subsequences $\vec{v_1} = [v_1, ..., v_m]$ and $\vec{v_2} = [v_{m+1}, ..., v_n]$ such that $lab(v_1) \ldots lab(v_m)$ and $lab(v_{m+1}) \ldots lab(v_n)$ are in the regular languages defined by β_1 and β_2, respectively. We mark $\vec{v_1}$ with $y^\tau_{\beta_1,k_1}$ and mark $\vec{v_2}$ with $y^\tau_{\beta_2,k_2}$, where $y^\tau_{\beta_1,k_1}$ and $y^\tau_{\beta_2,k_2}$ are the corresponding variables in $g(\alpha, y^\tau_{\alpha,1})$. We then proceed to mark subsequences of $\vec{v_1}$ and $\vec{v_2}$ in the same way. If α is $(\beta_1 \mid \beta_2)$, then $\vec{v}$ corresponds to either β_1 or β_2. We mark the sequence with the corresponding variable in $g(\alpha, y^\tau_{\alpha,1})$ and proceed to mark its subsequences as above. These marks provide a finer labeling that keeps track of how $[v_1, ..., v_n]$ corresponds to regular expressions in $P(\tau)$. Let the unknown $x^i_{\tau'}$ be assigned as its value the number of all vertices in V marked $x^i_{\tau'}$, and similarly, let $y^\tau_{\beta,k}$ be assigned the number of all vertex sequences marked $y^\tau_{\beta,k}$. This is indeed an integer assignment since T is finite. This assignment must satisfy ψ_τ for each $\tau \in E$, since $T \models D$ and ψ_τ is determined by $P(\tau)$ in D. Formally, this can be verified by induction on the structure of $P(\tau)$. In addition, it is easy to verify that the assignment satisfies the other constraints in Ψ_D. More specifically, $|ext(r)| = 1$, all unknowns ≥ 0, and $ext(\tau)$ is the collection of all vertices corresponding to some occurrences of τ in element type definitions of D. Therefore, the assignment is indeed an integer solution to Ψ_D. That is, Ψ_D is consistent.

Conversely, assume that Ψ_D admits an integer solution. That is, there is an integer assignment to unknowns $|ext(\tau)|$, x^i_τ and $y^\tau_{\beta,k}$ satisfying equalities and inequalities in Ψ_D. Observe that all these unknowns are equal to or greater than 0 because of inequalities in Ψ_D. We show that there is an XML tree $T = (V, lab, ele, att, val, root)$ valid w.r.t. D. To do so, for each $\tau \in E \cup \{\mathtt{S}\}$, we create $|ext(\tau)|$ many distinct vertices and label them with τ. We refer to this set of vertices as $ext(\tau)$. In addition, for each $\tau \in E$, each $v \in ext(\tau)$ and $l \in R(\tau)$, we create a distinct vertex and label it with l. We refer to this vertex as v_l. Let

$$V = \bigcup_{\tau \in E} ext(\tau) \ \cup \ ext(\mathtt{S}) \ \cup \ \bigcup_{\tau \in E} \{v_l \mid v \in ext(\tau), \, l \in R(\tau)\}.$$

The functions lab, att and val are defined as follows: for each $v \in V$ and $l \in A$,

$$lab(v) = \begin{cases} \tau & \text{if } v \in ext(\tau) \text{ and } \tau \in E \cup \{\mathtt{S}\} \\ l & \text{if } v = v_l' \text{ for some } v_l' \end{cases}$$

$$att(v, l) = \begin{cases} v_l & \text{if } v_l \in V \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$val(v) = \begin{cases} \text{empty string} & \text{if } lab(v) \text{ is } \mathtt{S} \text{ or } l, \text{ where } l \in A \\ \text{undefined} & \text{otherwise} \end{cases}$$

It is easy to verify that these functions are well defined. Let $root$ be the vertex labeled r. This vertex is unique since $|ext(r)| = 1$ is in Ψ_D. Finally, we define the function ele. To define ele, we do the following:

(1) For each x_τ^i in Ψ_D, we choose x_τ^i many distinct vertices labeled τ and mark them with x_τ^i. We mark a vertex at most once. Observe that every vertex in V labeled τ can be marked once and only once since the assignment to unknowns is an integer solution to Ψ_D.

(2) For each $y_{\beta,k}^\tau$ in Ψ_D, we choose $y_{\beta,k}^\tau$ many distinct sequences of vertices in V and mark them with $y_{\beta,k}^\tau$, following the definition of the function g. More specifically, if $y_{\beta,k}^\tau = x_{\tau'}^i$ is in Ψ_D for some $\tau' \in E \cup \{\mathtt{S}\}$, then we pick $x_{\tau'}^i$ many distinct vertices labeled $x_{\tau'}^i$ and mark them also with $y_{\beta,k}^\tau$. If $y_{\beta,k}^\tau = y_{\epsilon,j}^\tau$ is in Ψ_D, then we create $y_{\beta,k}^\tau$ many empty sequences and mark them with $y_{\beta,k}^\tau$. If $y_{\beta,k}^\tau = y_{\beta_1,k_1}^\tau$ and $y_{\beta,k}^\tau = y_{\beta_2,k_2}^\tau$ are in Ψ_D, then we choose $y_{\beta,k}^\tau$ many distinct sequences marked with y_{β_1,k_1}^τ and $y_{\beta,k}^\tau$ many distinct sequences marked with y_{β_2,k_2}^τ. We create $y_{\beta,k}^\tau$ many distinct sequences of the form $\vec{v} = \vec{v_1}\vec{v_2}$, where $\vec{v_1}$ and $\vec{v_2}$ are sequences chosen as above and marked with y_{β_1,k_1}^τ and y_{β_2,k_2}^τ, respectively. We mark $\vec{v}$ with $y_{\beta,k}^\tau$. If $y_{\beta,k}^\tau = y_{\beta_1,k_1}^\tau + y_{\beta_2,k_2}^\tau$ is in Ψ_D, then we choose y_{β_1,k_1}^τ many distinct sequences marked with y_{β_1,k_1}^τ and y_{β_2,k_2}^τ many distinct sequences marked with y_{β_2,k_2}^τ. We mark these sequences also with $y_{\beta,k}^\tau$. Again this is possible by the definition of constraints in Ψ_D.

For each $v \in V$, if $lab(v) = \tau$, $\tau \in E$ and $P(\tau) = \alpha$, then we choose a distinct sequence of vertices $[v_1, ..., v_n]$ marked with $y_{\alpha,1}^\tau$. This is possible by constraints of Ψ_D. We define $ele(v) = [v_1, ..., v_n]$. It can be shown that $lab(v_1)...lab(v_n)$ is indeed in the regular language defined by α, by the definition of Ψ_D. Formally, this can be verified by induction on the structure of α, i.e., considering the cases when α is one of $\mathtt{S}$, τ, ϵ, $(\beta_1 \mid \beta_2)$ and (β_1, β_2). It is also easy to verify that T defined in this way is indeed an XML tree valid w.r.t. D. Therefore, there exists an XML tree valid w.r.t. D. □

Furthermore, given an integer solution to Ψ_D, one can construct an XML tree T valid w.r.t. D such that there is certain quantitative connection between $ext(\tau)$ in T and the corresponding unknown $|ext(\tau)|$ in Ψ_D.

Corollary 4.8: *Let D be a ($*$-fee) DTD and Ψ_D be the set of cardinality constraints determined by D. If Ψ_D admits an integer solution such that $|ext(\tau)|$ is assigned a value n_τ for each $\tau \in E$, then there is an XML tree T such that $T \models D$ and in addition, $|ext(\tau)|$ in T is equal to n_τ.* □

Proof: This follows immediately from the proof of Lemma 4.7. Indeed, given an integer solution to Ψ_D, the proof of Lemma 4.7 constructs an XML tree T valid w.r.t. D. In addition, for any $\tau \in E$, the number of vertices in T labeled τ is exactly the nonnegative integer value assigned to the unknown $|ext(\tau)|$ in Ψ_D. Thus Corollary 4.8 holds. □

The proof of the next lemma is straightforward.

Lemma 4.9: *Given any ($*$-free) DTD D, the set Ψ_D of cardinality constraints determined by D can be computed in linear time.* □

Immediately from Lemma 4.9 it follows that the size of Ψ_D is linear in the size of D.

Characterizing DTDs and unary constraints

To complete our characterization, given a DTD D and a finite set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, we define a finite set $\Psi(D, \Sigma)$ of cardinality constraints. The set $\Psi(D, \Sigma)$ can be treated as a system of linear equalities and inequalities. We show that there exists an XML tree valid w.r.t. D and satisfying Σ if and only if $\Psi(D, \Sigma)$ admits a (legal) integer solution.

We proceed in two steps: we first consider $*$-free DTDs and then show that the results established for $*$-free DTDs can be extended to DTDs that are not necessarily $*$-free.

Let $D = (E, A, P, R, r)$ be a $*$-free DTD, and Σ be a finite set of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D. We define *the set of cardinality constraints determined by D and Σ*, denoted by $\Psi(D, \Sigma)$, to be

$$\Psi(D, \Sigma) \ = \ \Psi_D \cup C_\Sigma,$$

where C_Σ is the set of cardinality constraints determined by Σ, and Ψ_D is the set of cardinality constraints determined by D, as defined above.

For each $\tau \in E \cup \{\mathtt{S}\}$, we treat $|ext(\tau)|$ in $\Psi(D, \Sigma)$ as an unknown. Similarly, for each $\tau \in E$ and $l \in R(\tau)$, we treat $|ext(\tau.l)|$ also as an unknown. Thus $\Psi(D, \Sigma)$ can be viewed as a system of linear equalities and inequalities. We say that $\Psi(D, \Sigma)$ is *consistent* if and only if $\Psi(D, \Sigma)$ admits a *legal* integer solution. By a legal solution we mean a solution that satisfies the following condition: if $|ext(\tau)| > 0$ then $|ext(\tau.l)| > 0$ for each $|ext(\tau)|$ and $|ext(\tau.l)|$ appearing in $\Psi(D, \Sigma)$. This is to ensure that every τ element has an l attribute. Observe that $|ext(\tau.l)| \leq |ext(\tau)|$ is in C_Σ.

For example, recall DTDs D_1^- and D_2^- given above and constraint sets Σ_1 and Σ_2 (the empty set) in Section 1. It is easy to verify that neither $\Psi(D_1^-, \Sigma_1)$ nor $\Psi(D_2^-, \Sigma_2)$ is consistent. This is consistent with the observations made in Section 1.

We next show that $\Psi(D, \Sigma)$ characterizes D and Σ.

Lemma 4.10: *Let D be a ($*$-free) DTD, Σ a finite set of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, and $\Psi(D, \Sigma)$ the set of cardinality constraints determined by D and Σ. Then $\Psi(D, \Sigma)$ is consistent if and only if there is an XML tree T such that $T \models D$ and $T \models \Sigma$.* □

Proof: Suppose that there exists an XML tree T such that $T \models D$ and $T \models \Sigma$. We show that $\Psi(D, \Sigma)$ admits a legal integer solution. By Lemma 4.3, we have $T \models C_\Sigma$, where C_Σ is the finite set of cardinality constraints determined by Σ. As in the proof of Lemma 4.7, we can define an integer assignment to the unknowns in $\Psi(D, \Sigma)$ such that equalities and inequalities in Ψ_D are satisfied. Observe that the assignment assures that for each $\tau \in E$, the unknown $|ext(\tau)|$ equals to the number of all vertices in T labeled τ. We extend the assignment as follows: for each $\tau \in E$ and $l \in R(\tau)$, let the unknown $|ext(\tau.l)|$ be the number of distinct l attribute values of all vertices in T labeled τ. Thus by $T \models C_\Sigma$, this extended assignment satisfies all the equalities and inequalities in C_Σ. In addition, if $|ext(\tau)| > 0$ then $|ext(\tau.l)| > 0$ since every τ element in T has an l attribute. Hence the assignment is indeed a legal solution to $\Psi(D, \Sigma)$. This is, $\Psi(D, \Sigma)$ is consistent.

Conversely, suppose that $\Psi(D, \Sigma)$ admits a legal integer solution. We show that there is an XML tree T such that $T \models D$ and $T \models \Sigma$. As in the proofs of Lemma 4.7 and Corollary 4.8, given an integer solution to $\Psi(D, \Sigma)$, we can construct an XML tree $T' = (V, lab, ele, att, val, root)$ such that $T' \models D$. Moreover, for each $\tau \in E$, if the unknown $|ext(\tau)|$ is assigned n_τ by the integer assignment, then n_τ is number of all vertices in T' labeled τ. We modify the definition of the function val such that in T', for each $\tau \in E$ and $l \in R(\tau)$ that occur in Σ, the number of distinct l attribute values of all vertices labeled τ equals to the value assigned to the unknown $|ext(\tau.l)|$ by the assignment. This is possible since $|ext(\tau.l)| \leq |ext(\tau)|$ is in C_Σ, and the assignment is also a solution to C_Σ. Therefore, there is an XML tree T'' such that $T'' \models C_\Sigma$ and $T'' \models D$. Observe that since the solution is legal, we have $|ext(\tau.l)| > 0$ if $|ext(\tau)| > 0$, and thus every τ element in T'' can have an l attribute. Hence by Lemma 4.3, there exists an XML tree T such that $T \models D$ and moreover, $T \models \Sigma$. □

Next, consider DTDs that are not necessarily $*$-free. The corollary below shows that for any DTD D and finite set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, $\Psi(D^-, \Sigma)$ also characterizes whether there exists an XML tree valid w.r.t. D and satisfying Σ, where D^- is the $*$-free DTD of D as defined above.

Corollary 4.11: *Let D be a DTD, D^- the $*$-free DTD of D, Σ a finite set of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, and $\Psi(D^-, \Sigma)$ the set of cardinality constraints determined by D^- and Σ. Then $\Psi(D^-, \Sigma)$ is consistent if and only if there is an XML tree T such that $T \models D$ and $T \models \Sigma$.* □

Proof: First assume that there is an XML tree T such that $T \models D$ and $T \models \Sigma$. By Lemma 4.5, there is an XML tree T' such that $T' \models D^-$ and $T' \models \Sigma$. Thus by Lemma 4.10, $\Psi(D^-, \Sigma)$ is consistent.

Conversely, suppose that $\Psi(D^-, \Sigma)$ admits a legal integer solution. By Lemma 4.10, there exists an XML tree T' such that $T' \models D^-$ and $T' \models \Sigma$. Thus by Lemma 4.5, there exists an XML tree T such that $T \models D$ and $T \models \Sigma$. □

Given these lemmas, we proceed to prove Theorem 4.1.

Proof of Theorem 4.1 (continued): We encode an instance of the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ as an instance of LIP. By Corollary 4.11, given any DTD D and any set Σ of $\mathcal{C}^{Unary}_{K,FK}$ constraints over D, there is an XML tree valid w.r.t. D and satisfying Σ if and only if $\Psi(D^-, \Sigma)$ has a legal integer solution. Note that $\Psi(D^-, \Sigma)$ can be encoded as an instance of LIP. More specifically, observe that an equality $F_1 = F_2$ in $\Psi(D^-, \Sigma)$ is equivalent to inequalities $F_1 \geq F_2$ and $F_2 \geq F_1$. Thus without loss of generality, we assume $\Psi(D^-, \Sigma)$ consists of only inequalities. We enumerate the unknowns in $\Psi(D^-, \Sigma)$ as $x_1, ..., x_n$ and inequalities as $F_1, ..., F_n$. Thus we can define an $m \times n$ matrix A_Ψ of integers and a vector $\vec{b}_\Psi$ of m integers such that for $i \in [1, m]$, $\sum_{j \in [1,n]} a_{ij}\, x_j \geq b_i$ is F_i. Hence $\Psi(D^-, \Sigma)$ admits an integer solution if and only if there exists a vector $\vec{x}$ such that $A_\Psi\, \vec{x} \geq \vec{b}_\Psi$. It should be noted that for all unknowns x appearing in $\Psi(D^-, \Sigma)$, we have $x \geq 0$ in $\Psi(D^-, \Sigma)$. In other words, we are interested in nonnegative solutions to $A_\Psi\, \vec{x} \geq \vec{b}_\Psi$ only.

We now consider the problem of determining whether $\Psi(D^-, \Sigma)$ admits a legal integer solution. A legal solution requires that $|ext(\tau.l)| > 0$ if $|ext(\tau)| > 0$. To ensure this, consider a system Ψ of linear inequalities that includes all the inequalities in $\Psi(D^-, \Sigma)$ and moreover, $|ext(\tau.l)| \geq 1$ and $|ext(\tau)| \geq 1$ for each $|ext(\tau.l)|$ and $|ext(\tau)|$ appearing in $\Psi(D^-, \Sigma)$. Let X be some set of pairs $(|ext(\tau.l)|, |ext(\tau)|)$ that appear in $\Psi(D^-, \Sigma)$, and Ψ_X be a system of linear inequalities that is the same as Ψ except that it includes $|ext(\tau.l)| = 0$ and $|ext(\tau)| = 0$ for each $(|ext(\tau.l)|, |ext(\tau)|)$ in X, instead of $|ext(\tau.l)| \geq 1$

and $|ext(\tau)| \geq 1$. We can encode Ψ_X as an instance $A_{\Psi_X}\,\vec{x} \geq \vec{b}_{\Psi_X}$ of LIP, as described above. It is easy to see that $\Psi(D^-, \Sigma)$ admits a legal integer solution if and only if there is some $A_{\Psi_X}\,\vec{x} \geq \vec{b}_{\Psi_X}$ that admits an integer solution. More specifically, an integer solution to any $A_{\Psi_X}\,\vec{x} \geq \vec{b}_{\Psi_X}$ is a legal integer solution to $\Psi(D^-, \Sigma)$, and conversely, a legal integer solution to $\Psi(D^-, \Sigma)$ is an integer solution to some $A_{\Psi_X}\,\vec{x} \geq \vec{b}_{\Psi_X}$. By the result of [23] cited above, if Ψ_X has an integer solution, then it has one in which each unknown has a length in binary of at most $1 + \lceil \log n + (2m+1) \cdot \log(ma) \rceil$ bits, where m, n and a are determined by $\Psi(D^-, \Sigma)$ as described above. As a result, the bounds on solutions for all Ψ_X's are the same. Therefore, if the encoding $A_\Psi\,\vec{x} \geq \vec{b}_\Psi$ of $\Psi(D^-, \Sigma)$ has a legal integer solution, then it has one in which each unknown has a length in binary of at most $1 + \lceil \log n + (2m+1) \cdot \log(ma) \rceil$ bits. Let c be a number that in binary notation has $1 + \lceil \log n + (2m+1) \cdot \log(ma) \rceil$ many 1's. Observe that the number c can be computed in $O(s \cdot \log s)$ time in the size of $\Psi(D^-, \Sigma)$. In light of these, we define a new system Φ of linear inequalities that is the same as $\Psi(D^-, \Sigma)$ except it also includes:

$$c\,|ext(\tau.l)| \geq |ext(\tau)|$$

for every $|ext(\tau.l)|$ and $|ext(\tau)|$ appearing in $\Psi(D^-, \Sigma)$. It is easy to verify that $\Psi(D^-, \Sigma)$ has a legal integer solution if and only Φ has an integer solution. Indeed, if $\Psi(D^-, \Sigma)$ has a legal integer solution then it has one bounded by c. Thus the solution satisfies $c\,|ext(\tau.l)| \geq |ext(\tau)|$. In other words, it is an integer solution to Φ. Conversely, if Φ has an integer solution, then it is also an integer solution of $\Psi(D^-, \Sigma)$ and moreover, if $|ext(\tau)| > 0$ then $|ext(\tau.l)| > 0$ by $c\,|ext(\tau.l)| \geq |ext(\tau)|$ in Φ. In other words, it is a legal solution to $\Psi(D^-, \Sigma)$. Therefore, by Corollary 4.11, there exists an XML tree valid w.r.t. D and satisfying Σ iff Φ has an integer solution. By Lemmas 4.6, 4.4 and 4.9, $\Psi(D^-, \Sigma)$ can be computed in linear time and its size is linear in the sizes of D and Σ. Thus Φ can be computed in PTIME (in fact, $O(s^2 \cdot \log s)$) in the sizes of D and Σ.

In addition, it should be noted that from the discussions above follows that determining whether Φ admits an integer solution is in NP. Therefore, the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ is in NP.

This completes the proof of Theorem 4.1. □

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

Proof: 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 [15].

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

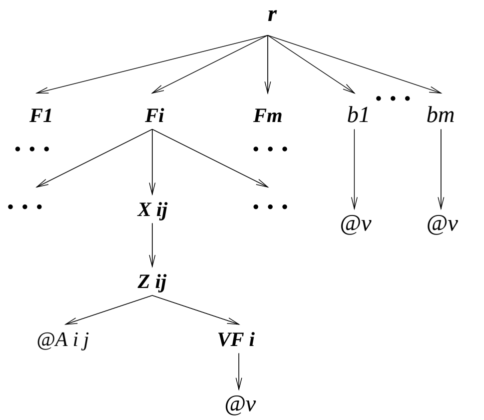

Figure 4: A tree used in the proof of Theorem 4.12

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

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

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 $\mathcal{C}_{K,FK}^{Unary}$ 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 $i, l \in [1,m]$,

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

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 $i \in [1,m]$,

$$VF_i.v \rightarrow VF_i, \qquad b_i.v \rightarrow b_i, \qquad VF_i.v \subseteq b_i.v, \qquad b_i.v \subseteq VF_i.v.$$

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 4.

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.

This completes the proof of Theorem 4.12. $\square$

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 $\tau.l \rightarrow \tau$. 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 $\mathcal{C}_{K,FK}^{Unary}$ is to determine, given any DTD D and finite set Σ of $\mathcal{C}_{K,FK}^{Unary}$ 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 $\mathcal{C}_{K,FK}^{Unary}$ constraints. However, this is not the case.

Corollary 4.13: *Under the primary key restriction, the consistency problem for $\mathcal{C}_{K,FK}^{Unary}$ remains NP-complete.* $\square$

Proof: The reduction from LIP given in the proof of Theorem 4.12 defines at most one key for each element type. $\square$

A mild generalization of the encoding given above can establish the complexity of the consistency problem for $\mathcal{C}_{K^{\neg},IC}^{Unary}$, the class of unary keys, inclusion constraints and negations of keys.

Corollary 4.14: *The consistency problem for $\mathcal{C}_{K^{\neg},IC}^{Unary}$ constraints is NP-complete.* $\square$

Proof: Since $\mathcal{C}_{K,FK}^{Unary}$ is a sub-language of $\mathcal{C}_{K^{\neg},IC}^{Unary}$, from Theorem 4.12 follows immediately that the consistency problem for $\mathcal{C}_{K^{\neg},IC}^{Unary}$ is NP-hard. We next show that the problem remains in NP. Let D be a DTD and Σ be a set of $\mathcal{C}_{K^{\neg},IC}^{Unary}$ constraints over D. We write Σ as $\Sigma_1 \cup \Sigma_2$, where Σ_1 is a set of unary keys and unary inclusion constraints over D, and Σ_2 is a set of negations of unary keys over D. Let $\Psi(D, \Sigma_1)$ be the system of linear inequalities determined by D and Σ_1, as defined in the proof of Theorem 4.1. It admits an integer solution iff there exists an XML tree T such that $T \models \Sigma_1$ and $T \models D$. Here by an integer solution we mean a legal integer solution. We define another system of linear inequalities, denoted by $\Psi(D, \Sigma)$ and referred to as *the system determined by D and Σ*, to be

$$\Psi(D, \Sigma) \;=\; \Psi(D, \Sigma_1) \;\cup\; \{|ext(\tau.l)| < |ext(\tau)| \;\mid \neg(\tau.l \rightarrow \tau) \in \Sigma_2\}.$$

Clearly, $\Psi(D, \Sigma)$ can be computed in PTIME by Lemmas 4.4, 4.6 and 4.9 and the proof of Theorem 4.1. It suffices to show the following claim.

Claim: There is an XML tree T such that $T \models \Sigma$ and $T \models D$ iff $\Psi(D, \Sigma)$ has an integer solution.

For if it holds, then the problem is in NP by reduction to LIP as in the proof of Theorem 4.1.

We show the claim as follows. Assume that there exists a tree T such that $T \models \Sigma$ and $T \models D$. Since $T \models \Sigma_1$, by Lemma 4.10, Corollary 4.11 and the proof of Theorem 4.1, there is an integer solution to $\Psi(D, \Sigma_1)$, the system of linear inequalities determined by D and Σ_1, such that the values of the variables $|ext(\tau)|$ and $|ext(\tau.l)|$ in $\Psi(D, \Sigma_1)$ given by the solution are the cardinalities $|ext(\tau)|$ and $|ext(\tau.l)|$ in T. Note that for all element type τ and attribute l of τ in D, $|ext(\tau)|$ and $|ext(\tau.l)|$ are variables in $\Psi(D, \Sigma_1)$. Thus for each $\tau.l \not\rightarrow \tau$, the solution also assigns values to $|ext(\tau)|$ and

$|ext(\tau.l)|$. We claim that it is also a solution to $\Psi(D, \Sigma)$. To see this, observe that it is always true that $|ext(\tau)| \geq |ext(\tau.l)|$ in T since every τ element in T contributes at most one distinct $\tau.l$ value. Moreover, since $T \models \Sigma_2$, there must be two distinct τ elements d_1 and d_2 in T such that $d_1.l = d_2.l$. Thus $|ext(\tau)| > |ext(\tau.l)|$. Therefore, all inequalities in $\Psi(D, \Sigma)$ are satisfied by the solution.

Conversely, assume that $\Psi(D, \Sigma)$ has an integer solution. Since it is also a solution to $\Psi(D, \Sigma_1)$, by Lemma 4.10, Corollary 4.11 and the proof of Theorem 4.1, there is a tree T such that $T \models D$, $T \models \Sigma_1$ and moreover, the cardinalities $|ext(\tau)|$ and $|ext(\tau.l)|$ in T are the values of the variables $|ext(\tau)|$ and $|ext(\tau.l)|$ in $\Psi(D, \Sigma_1)$ given by the solution. We claim that $T \models \Sigma$. Indeed, for any $\tau.l \not\rightarrow \tau$ in Σ_2, we have $|ext(\tau)| > |ext(\tau.l)|$ in T. Thus there must be two distinct τ elements d_1 and d_2 in T such that $d_1.l = d_2.l$. That is, $T \models \tau.l \not\rightarrow \tau$. Hence $T \models D$ and $T \models \Sigma$. This completes the proof of the claim. $\square$

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 4.13.

Corollary 4.14 also tells us the complexity of a special case of the implication problem for $\mathcal{C}_{K,FK}^{Unary}$, referred to as *implication problem for unary keys by* $\mathcal{C}_{K,FK}^{Unary}$ *constraints*:

Theorem 4.15: *The following is coNP-complete, even under the primary key restriction: given any DTD D, any set Σ of $\mathcal{C}_{K,FK}^{Unary}$ constraints and a unary key φ over D, whether $(D, \Sigma) \vdash \varphi$.* $\square$

Proof: 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 $\mathcal{C}_{K^\neg,IC}^{Unary}$ constraints. Thus the implication problem for unary keys by $\mathcal{C}_{K,FK}^{Unary}$ constraints is in coNP by Corollary 4.14. To show that the problem is coNP-hard under the primary key restriction, observe that in the reduction from the consistency problem for $\mathcal{C}_{K,FK}$ to the complement of the implication problem for $\mathcal{C}_{K,FK}$ given in the proof of Lemma 3.3, if the set Σ of constraints given is a set of $\mathcal{C}_{K,FK}^{Unary}$ constraints under the primary key restriction, then these and the new constraints defined in the reduction do not violate the primary key restriction. In other words, the reduction also serves as a reduction from the consistency problem for $\mathcal{C}_{K,FK}^{Unary}$ to the complement of the implication problem for $\mathcal{C}_{K,FK}^{Unary}$ under the primary key restriction. Moreover, the reduction is definable in PTIME. Therefore, from Corollary 4.13 follows immediately that the implication problem for unary keys by $\mathcal{C}_{K,FK}^{Unary}$ constraints is coNP-hard. $\square$

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 3.5. We next show that given a fixed DTD D, the consistency and implication analyses become simpler.

Corollary 4.16: *For a* fixed *DTD, the following problems are decidable in PTIME:*

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

$\square$

Proof: 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 4.14. 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 [20]. As shown by Theorem 4.1 and Corollary 4.14, 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 4.16. □

5 Unary keys, inclusion constraints and negations

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 4.12, 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.

Proof: 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. Let D be a DTD and Σ be a set of $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$ constraints over D. We partition Σ into Σ_1 and Σ_2, where Σ_1 is a set of $\mathcal{C}^{Unary}_{K^\neg,IC}$ constraints, and Σ_2 consists of negations of unary inclusion constraints over D. Let $\Psi(D, \Sigma_1)$ be the system of linear inequalities determined by D and Σ_1, as described in the proof of Corollary 4.14. Let $l_1, \ldots, l_n$ be an enumeration of all attributes in D. Without loss of generality, assume that l_i is an attribute of element type τ_i (note that τ_i's need not be distinct). Let $\mathbf{U} = (u_{ij})^n_{i,j=1}$ and $\mathbf{V} = (v_{ij})^n_{i,j=1}$ be two matrices whose elements are nonnegative integers. We say that they admit a *set representation* if there is a family of finite sets $A_1, \ldots, A_n$ such that

$$u_{ij} = |A_i \cap A_j|, \quad v_{ij} = |A_i - A_j|.$$

We extend $\Psi(D, \Sigma_1)$ with new variables u_{ij}, v_{ij}, and equalities:

- $|ext(\tau_i.l_i)| = u_{ii} = u_{ij} + v_{ij}$ for all $i, j \in [1, n]$;

- $v_{ij} = 0$ for all $\tau_i.l_i \subseteq \tau_j.l_j$ in Σ_1, and moreover, $v_{ii} = 0$;
- $v_{ij} > 0$ for all $\tau_i.l_i \not\subseteq \tau_j.l_j$ in Σ_2.

Let us denote the new system by $\Psi(D, \Sigma)$ and refer to it as *the system determined by D and Σ*. Observe that $\Psi(D, \Sigma)$ can be simply converted to a system of linear inequalities (by treating an equality as two inequalities).

The intended interpretation for the variable u_{ij} is $|\, ext(\tau_i.l_i) \cap ext(\tau_j.l_j)\,|$, and $|\, ext(\tau_i.l_i) - ext(\tau_j.l_j)\,|$ for v_{ij}. Thus, the inequalities $v_{ij} > 0$ in $\Psi(D, \Sigma)$ says that $ext(\tau_i.l_j) \not\subseteq ext(\tau_j.l_j)$ for all $\tau_i.l_i \not\subseteq \tau_j.l_j$ in Σ_2.

The lemma below reveals the connection between the encoding and the consistency problem we are investigating.

Lemma 5.2: *The linear system $\Psi(D, \Sigma)$ determined by DTD D and constraints Σ has an integer solution with $\mathbf{U}, \mathbf{V}$ having a set representation if and only if there is an XML tree T such that $T \models D$ and $T \models \Sigma$.* □

Proof: Let D be a DTD, Σ_1 be a set of $\mathcal{C}^{Unary}_{K^{\neg},IC}$ constraints over D, Σ_2 be a set of negations of unary inclusion constraints over D, $\Sigma = \Sigma_1 \cup \Sigma_2$, and $\Psi(D, \Sigma)$ be the system of linear inequalities determined by D and Σ as described above. We show that $\Psi(D, \Sigma)$ has an integer solution with $\mathbf{U}, \mathbf{V}$ having a set representation iff there is an XML tree T such that $T \models \Sigma$ and $T \models D$.

Assume that there exists an XML tree T such that $T \models \Sigma$ and $T \models D$. Since $T \models \Sigma_1$, as in the proof of Corollary 4.14 we can define an integer solution to $\Psi(D, \Sigma_1)$, the system of linear inequalities determined by D and Σ_1. We extend the solution as follows: let u_{ij} be $|\, ext(\tau_i.l_i) \cap ext(\tau_j.l_j)\,|$, and v_{ij} be $|\, ext(\tau_i.l_i) - ext(\tau_j.l_j)\,|$. It is easy to verify that this is indeed a solution to $\Psi(D, \Sigma)$ with $\mathbf{U}, \mathbf{V}$ having a set representation.

Conversely, assume that $\Psi(D, \Sigma)$ has an integer solution with $\mathbf{U}, \mathbf{V}$ having a set representation. Then there are finite sets $A_1, \ldots, A_n$ such that

$$u_{ij} = |\, A_i \cap A_j\,|, \qquad v_{ij} = |\, A_i - A_j\,|\,.$$

Again as in the proof of Corollary 4.14, we create a tree T such that $T \models \Sigma_1$ and $T \models D$. In addition, we define the *val* function in T such that $ext(\tau_i) = A_i$ for $i \in [1, n]$. This is possible since $|ext(\tau_i.l_i)| = u_{ii} = u_{ij} + v_{ij}$ is in $\Psi(D, \Sigma)$ for all $i, j \in [1, n]$. Because $v_{ij} > 0$ is in $\Psi(D, \Sigma)$ for all $\tau_i.l_i \not\subseteq \tau_j.l_j$ in Σ_2, we have $|\, ext(\tau_i.l_i) - ext(\tau_j.l_j)\,| > 0$. That is, $T \models \tau_i.l_i \not\subseteq \tau_j.l_j$. Thus $T \models \Sigma_2$. This completes the proof of the lemma. □

It remains to show that one can check in NP whether the system $\Psi(D, \Sigma)$ has an integer solution with $\mathbf{U}, \mathbf{V}$ having a set representation. We start with a lemma.

Lemma 5.3: *Given $\Psi(D, \Sigma)$, one can compute, in polynomial time, a number M such that $\Psi(D, \Sigma)$ has an integer solution with $\mathbf{U}, \mathbf{V}$ having a set representation if and only if it admits such a solution with all variables being bounded by M.* □

Proof: To prove the lemma, we need to extend $\Psi(D, \Sigma)$. Let Θ be the set of functions $\theta : \{1, \ldots, n\} \to \{0, 1\}$ which are not identically 0, where n is the number of attributes in D. For every θ, we introduce a new variable z_θ (note that the number of variables is now exponential in the size of the problem). The intended interpretation of z_θ is the cardinality of

$$\bigcap_{i:\theta(i)=1} ext(\tau_i.l_i) \quad - \quad \bigcup_{j:\theta(j)=0} ext(\tau_j.l_j).$$

We now extend $\Psi(D,\Sigma)$ to $\Psi'(D,\Sigma)$ by adding the following equalities:

$$u_{ij} \;=\; \sum_{\theta:\theta(i)=\theta(j)=1} z_\theta, \qquad v_{ij} \;=\; \sum_{\theta:\theta(i)=1,\theta(j)=0} z_\theta.$$

Clearly, $\Psi(D,\Sigma)$ has an integer solution with $\mathbf{U},\mathbf{V}$ having a set representation iff $\Psi'(D,\Sigma)$ has an integer solution, as the variables z_θ describe all possible intersections of $ext(\tau_i.l_i)$ and their complements, and the equalities above show how to reconstruct u_{ij} and v_{ij} from them. We thus must show that if $\Psi'(D,\Sigma)$ has an integer solution then it must have one with a bound on u_{ij}, v_{ij}, which is polynomial (in terms of the size of $\Psi(D,\Sigma)$). For that, recall [23] that if a system of k linear inequalities with l variables and all coefficients at most c has an integer solution, then it has an integer solution in which none of the variables exceeds $l(ck)^{2k+1}$. Thus, M can be taken to be a number that in binary notation has $1 + \lceil \log l + (2k+1)\cdot\log(ck)\rceil$ many 1's. Note that the number of variables, l, of $\Psi'(D,\Sigma)$ is at most exponential in the size of $\Psi(D,\Sigma)$, and the number of equalities, k, is at most polynomial. This shows that M can be found in polynomial time, and thus proves the lemma. □

Given Lemmas 5.2 and 5.3, let us go back to the proof of that consistency analysis of Σ over D is in NP. We present a NP algorithm for determining the consistency of Σ over D. Our nondeterministic machine computes M given by Lemma 5.3, and then guesses a solution with all the components bounded by M. It then tests if the $\mathbf{U},\mathbf{V}$ part has a set representation. To do so, we transform $\mathbf{U},\mathbf{V}$, in polynomial time, into another matrix $\mathbf{W}$, and then run a nondeterministic polynomial time machine on $\mathbf{W}$. If it returns 'yes', then $\mathbf{U},\mathbf{V}$ have a set representation, and thus by Lemma 5.2 the answer to whether Σ is consistent over D is 'yes'.

Let $K = M\cdot n$, where n is the number of all attributes in D. We now define the matrix $\mathbf{W}$. It is a $2n\times 2n$ matrix, with

$$w_{ij} \;=\; \begin{cases} u_{ij} & \text{if } i,j \le n \\ v_{i,j-n} & \text{if } i \le n, j > n \\ v_{i-n,j} & \text{if } i > n, j \le n \\ K - u_{i-n,j-n} - v_{i-n,j-n} - v_{j-n,i-n} & \text{if } i,j > n \end{cases}$$

Recall the INTERSECTION PATTERN problem: Given an $m\times m$ matrix $\mathbf{A}$, are there sets $Y_1,\ldots,Y_m$ such that $a_{ij} = |Y_i \cap Y_j|$? This problem is known to be NP-complete (see, e.g., [15]).

We now show the following: The INTERSECTION PATTERN problem returns 'yes' on input $\mathbf{W}$ iff $\mathbf{U},\mathbf{V}$ have a set representation.

First, assume $\mathbf{U},\mathbf{V}$ have a set representation. That is, there are finite sets $A_1,\ldots,A_n$ such that

$$u_{ij} \;= |A_i \cap A_j|, \qquad v_{ij} \;= |A_i - A_j|\,.$$

By the assumption, all entries in $\mathbf{U},\mathbf{V}$ are bounded by M, and hence we may assume all sets in the representation are subsets of a set U of cardinality K. Let $m = 2n$ and define Y_i to be A_i for $i \le n$, and $U - A_{i-n}$ for $i > n$. Then $\mathbf{W}$ is the intersection pattern for this family of sets, and thus the INTERSECTION PATTERN problem returns 'yes' on $\mathbf{W}$.

Next, assume the INTERSECTION PATTERN returns 'yes' on $\mathbf{W}$, so we have a family of sets $Y_1, \ldots, Y_{2n}$ for which $\mathbf{W}$ is the intersection pattern. Let U be the union of all Y_j's. We show $Y_{n+i} = U - Y_i$ for all $i \leq n$. We have $w_{i,n+i} = v_{ii} = 0$, and thus $Y_{n+i} \subseteq U - Y_i$. Moreover, we have $|Y_i \cup Y_{n+i}| = w_{ii} + w_{n+i,n+i} = K$. We next show that for every $i, j \leq n$ it is the case that $Y_i \cup Y_{n+i} = Y_j \cup Y_{n+j}$ (and thus equals to U). Note that both $Y_i \cup Y_{n+i}$ and $Y_j \cup Y_{n+j}$ are K-element sets. Furthermore,

$$(Y_i \cup Y_{n+i}) \cap (Y_j \cup Y_{n+j}) = (Y_i \cap Y_j) \cup (Y_i \cap Y_{n+j}) \;\cup\; (Y_{n+i} \cap Y_j) \cup (Y_{n+i} \cap Y_{n+j}).$$

Observe that these four sets are pairwise disjoint, and their cardinalities are $w_{ij} = u_{ij}, w_{i,j+n} = v_{ij}, w_{i+n,j} = v_{ji}$ and $w_{i+n,j+n} = K - u_{ij} - v_{ij} - v_{ji}$, respectively. Thus, the cardinality of the set $(Y_i \cup Y_{n+i}) \cap (Y_j \cup Y_{n+j})$ is K, and since the cardinality of each $Y_i \cup Y_{n+i}$ and $Y_j \cup Y_{n+j}$ is K, we conclude $Y_i \cup Y_{n+i} = Y_j \cup Y_{n+j}$. This finally shows that U has cardinality K, and thus each Y_{n+i} is $U - Y_i$ for all $i \leq n$. This immediately gives us a set representation for $\mathbf{U}, \mathbf{V}$.

To conclude, once we guessed a bounded solution to $\Psi(D, \Sigma)$ (all components are at most M), we proceed to compute in polynomial time the matrix $\mathbf{W}$ from $\mathbf{U}$ and $\mathbf{V}$, and then run a nondeterministic polynomial time algorithm on it to check if $\mathbf{W}$ is an intersection pattern. Putting everything together, we see that this nondeterministic polynomial time algorithm returns 'yes' iff there is bounded solution (and thus, there is a solution) to $\Psi(D, \Sigma)$ for which $\mathbf{U}, \mathbf{V}$ have a set representation. By Lemma 5.2, this happens if and only if there exists an XML tree T such that $T \models D$ and $T \models \Sigma$.

This completes the proof of Theorem 5.1. $\square$

We next investigate implication problems.

Theorem 5.4: *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.* $\square$

Proof: 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 4.15.

We next consider the problem $(D, \Sigma) \vdash \varphi$ for $\mathcal{C}^{Unary}_{K,FK}$. A foreign key of $\mathcal{C}^{Unary}_{K,FK}$ consists of a unary key φ_1 and a unary inclusion constraint φ_2. Thus when φ is a foreign key, the problem is a conjunction of two:

$$(1) \quad (D, \Sigma) \vdash \varphi_1; \qquad (2) \quad (D, \Sigma \cup \{\varphi_1\}) \vdash \varphi_2.$$

Observe that when φ is a key in $\mathcal{C}^{Unary}_{K,FK}$, $(D, \Sigma) \vdash \varphi$ is actually problem (1) given above. Since both problems (1) and (2) are instances of the implication problem for $\mathcal{C}^{Unary}_{K,IC}$, they are in coNP. Moreover, they can be shown to be coNP-hard as in the proof of Theorem 4.15. Therefore, the implication problem for $\mathcal{C}^{Unary}_{K,FK}$ is coNP-complete. $\square$

Finally, along the same lines as Corollary 4.16, we show the following:

Corollary 5.5: *For a fixed DTD, the following problems can be determined in PTIME:*

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

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

$\square$

Proof: Let D be a DTD and Σ be a set of $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$ constraints over D. Let $\Psi'(D,\Sigma)$ be the system of linear inequalities determined by D and Σ, as defined in the proof of Theorem 5.1. As in the proof of Corollary 4.16, one can show that the number of variables in $\Psi'(D,\Sigma)$ is bounded by a function on the size of D. Therefore, when D is fixed, the number of variables in $\Psi'(D,\Sigma)$ is bounded by a constant. It is known that when the number of variables in a system of linear inequalities is bounded, it can be determined in PTIME whether the system admits an integer solution [20]. By the proofs of Lemma 5.2 and Theorem 5.1, $\Psi'(D,\Sigma)$ admits an integer solution if and only if there is an XML tree T such that $T \models D$ and $T \models \Sigma$. Thus Corollary 5.5 follows from Theorems 5.1 and 5.4. $\square$

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 $\mathcal{C}^{Unary}_{K,FK}$, 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 $\mathcal{C}^{Unary}_{K,FK}$ is NP-complete. Furthermore, the problem remains in NP for $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$, 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 $\mathcal{C}^{Unary}_{K,FK}$ 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 5.

It is worth remarking that the undecidability and NP-hardness results also hold for other schema specifications beyond DTDs, such as XML Data [19], XML Schema [27] and the generalization of DTDs proposed in [24].

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. 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.

	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 5: The main results of the paper

Acknowledgments. We thank Michael Benedikt, Alberto Mendelzon, Frank Neven and Jérôme Siméon for helpful discussions.

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

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

[9] 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.

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

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

[12] 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.

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

[14] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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