On XML Integrity Constraints in the Presence of DTDs

Wenfei Fan
Bell Laboratories
and
Leonid Libkin
University of Toronto

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.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Data models; F.4.3 [**Mathematical Logic and Formal Languages**]: Decision problems; I.7.2 [**Document Preparation**]: Markup languages

General Terms: Algorithms, Design, Languages, Theory

Additional Key Words and Phrases: Integrity Constraints, DTDs, XML, Consistency, Implication

---

Name: Wenfei Fan
Address: 600 Mountain Avenue, Murray Hill, NJ 07974, USA. Email: wenfei@research.bell-labs.com. On leave from Temple University. Supported in part by NSF Career Award IIS-0093168.
Name: Leonid Libkin
Affiliation: Department of Computer Science
Address: Department of Computer Science, University of Toronto, Toronto, Ontario, M5S 3H5, Canada. Email: libkin@cs.toronto.edu. Supported in part by National Sciences and Engineering Research Council of Canada. Research affiliation: Bell Laboratories.

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 [30]. 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; 37].

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 [27] and XML Schema [36]. Keys and foreign keys for XML are important in, among other things, query optimization [34], data integration [21], and in data transformations between XML and database formats [28].

XML data usually comes with a DTD[1] that specifies how a document is organized. Thus, a specification of an XML document may consist of both a DTD and a set of integrity constraints, such as keys and foreign keys. A legitimate question then is whether such a specification is *consistent*, or meaningful: that is, whether there exists a (finite) XML document that both satisfies the constraints and conforms to the DTD.

In the relational database setting, such a question would have a trivial answer: one can write arbitrary (`primary`) `key` and `foreign key` specifications in SQL, without worrying about consistency. However, DTDs (and other schema specifications for XML) are more complex than relational schema: in fact, XML documents are typically modeled as node-labeled trees, e.g., in XSL [15], XQL [35], XML Schema [36], XPath [16] 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:

<u>XML SPECIFICATION CONSISTENCY ($\mathcal{C}$)</u>

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

In other words, we want to validate XML specifications statically. The main reason is twofold:

[1]Throughout the paper, by a DTD we mean its type specification; we ignore its ID/IDREF constraints since their limitations have been well recognized [7; 19]. We shall only consider *finite* XML documents (trees).

first, complex interactions between DTDs and constraints are likely to result in inconsistent specifications, and second, an alternative dynamic approach to validation (simply check a document to see if it conforms to the DTD and satisfies the constraints) would not tell us whether repeated failures are due to a bad specification, or problems with the documents.

The concept of consistency of specifications was studied for other data models, such as object-oriented [12; 13] and extended relational (e.g., with support for cardinality constraints [26]).

We shall study the following four classes of constraints defined in terms of XML attributes:

—$\mathcal{C}_{K,FK}$: a class of keys and foreign keys;
—$\mathcal{C}^{Unary}_{K,FK}$: 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.

Keys and foreign keys of $\mathcal{C}_{K,FK}$ are a natural generalization of their relational counterpart, and are capable of capturing those relational constraints. A foreign key is a combination of two constraints: an inclusion constraint and a key. The $\mathcal{C}^{Unary}_{K,FK}$ constraints are a special case of $\mathcal{C}_{K,FK}$ constraints, which involve a single attribute. These unary keys and foreign keys are similar to but more general than XML ID and IDREF specifications. The study on simple constraints defined with XML attributes is a first step towards understanding the interaction between integrity constraints and schema specifications for XML. As will be seen shortly, the analyses of these simple constraints are already very intricate in the presence of DTDs.

As generalizations of $\mathcal{C}^{Unary}_{K,FK}$ constraints, $\mathcal{C}^{Unary}_{K^{\neg},IC}$ and $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$ both allow the presence of unary inclusion constraints independent of keys. In addition, $\mathcal{C}^{Unary}_{K^{\neg},IC}$ includes negations of unary keys, and $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$ further permits negations of unary inclusion constraints. Negation is considered mainly for the study of *implication* of $\mathcal{C}^{Unary}_{K,FK}$ constraints, which is the complement of a special case of the consistency problem for $\mathcal{C}^{Unary}_{K^{\neg},IC}$ (resp. $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$): given any DTD D and any finite set Σ of unary keys and inclusion constraints, is it the case that all XML trees satisfying Σ and conforming to D 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; 33]. 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 [21].

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

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

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

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

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

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

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

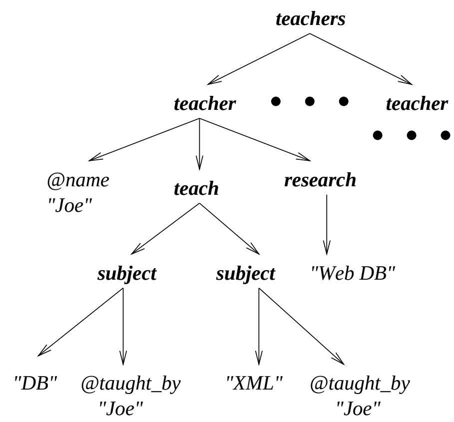

Fig. 1. An XML tree conforming to D_1

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 also enforce 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).

As another example, consider the DTD D_2 given below:

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

Observe that there exists no finite XML tree conforming to D_2. This demonstrates that there is need for studying consistency of XML specifications even in the absence of integrity constraints.

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 $\mathcal{C}_{K,FK}^{Unary}$ 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 $\mathcal{C}_{K,FK}^{Unary}$ (even under the restriction to primary keys, i.e., at most one key for each element type) is NP-complete. We further show that the problem is still in NP for an extension $\mathcal{C}_{K^{\neg},IC}^{Unary}$, which also allows negations of key constraints.
(3) Using a different coding of constraints, we show that the consistency problem remains in NP for $\mathcal{C}_{K^{\neg},IC^{\neg}}^{Unary}$, 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 as XML Schema and XML Data both subsume DTDs and they support keys and foreign keys which are more general than those considered here, the undecidability and NP-hardness results carry over to these schema specifications and constraint languages for XML.

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 [17]. In contrast, we shall show that the XML counterpart of this problem is coNP-complete.

The interaction between cardinality constraints and database schemas has been studied for object-oriented [12; 13] and extended relational data models [26]. These interactions are quite different from what we explore in this paper because XML DTDs are defined in terms of extended context free grammars and they yield cardinality constraints more complex than those studied for databases.

Key and foreign key specifications for XML have been proposed in the XML standard [6], XML Data [27], XML Schema [36] and in a recent proposal for XML keys [7]. The need for studying XML constraints has also been advocated in [38]. 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 [19], but in the absence of DTDs (in a graph model for XML), which trivializes the consistency analysis. For keys of [7], the implication problem was studied [8] in the tree model for XML, but DTDs were not considered there. 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). This paper is a full version of [18], providing the details and the proofs omitted there.

A variety of constraints have been studied for semistructured data [2; 10; 20]. In particular, [20] also studies the consistency problem; the special form of constraints used there makes it possible to encode consistency as an instance of conjunctive query containment. The interaction between path constraints and database schemas was investigated in [9]. These constraints typically 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 [23; 25]. But these generalizations were investigated in database settings, which are quite different from the tree model for XML data. Moreover, they cannot express foreign keys. Application of constraints in data transformations was studied in [28]; usefulness of keys and foreign keys in query optimization has also been recognized [34].

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 integer constraints, 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; 11; 31]) 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*: for each* $\tau \in E$*,* $P(\tau)$ *is a regular expression* α *defined as follows:*

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

where S *denotes* string *type,* $\tau' \in E$*,* ϵ *is the empty word, 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* $E \setminus \{r\}$ *is* connected to r*, i.e., either* τ *occurs in* $P(r)$*, or it appears in* $P(\tau')$ *for some* τ' *that is connected to* r*.*

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

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

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

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

An XML document is typically modeled as a node-labeled ordered tree. Given a DTD, we define the notion of its valid documents 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* nodes *(*vertices*);*
—*lab is a function that maps each node in V to a label in $E \cup A \cup \{\mathtt{S}\}$; a node $v \in V$ is called an* element of τ *if $lab(v) = \tau$ and $\tau \in E$, an* attribute *if $lab(v) \in A$, and a* text node *if $lab(v) = \mathtt{S}$;*
—*ele is a partial function defined on elements in V; for any $\tau \in E$, it maps each element v of type τ to a (possibly empty) list $[v_1, ..., v_n]$ of elements and text nodes in V such that $lab(v_1) \ldots lab(v_n)$ is in the regular language defined by $P(\tau)$;*
—*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 defined iff $lab(v) = \mathtt{S}$ or $lab(v) \in A$;*
—*$root$ is the unique node in V such that $lab(root) = r$, called* the root of T.

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

Intuitively, V is the set of nodes of the tree T. The mapping *lab* labels every node of V with a symbol from $E \cup A \cup \{\mathtt{S}\}$. Text nodes and attributes are leaves. For an element x of type τ, 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 the DTD D. The subelements of x are ordered and their labels satisfy 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 text nodes. We consider single-valued attributes. That is, if $l \in R(\tau)$ then every element of type τ has a unique l attribute with a string value. Since T has a tree structure, sharing of nodes is not allowed in T.

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

Our model is simpler than the models of XQuery [14] and XML Schema [36] as DTDs support only one basic type (PCDATA or string) and do not have complex type constructs. Furthermore, we do not have nodes representing namespaces, processing instructions and references. These simplifications allow us to concentrate on the essence of the DTD/constraint interaction. It should further be noticed that they do not affect the lower bounds results in the paper.

We need the following notations throughout the paper: 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 list $X = [l_1, \ldots, l_n]$ of attributes in $R(\tau)$, we use $x[X]$ to denote the list 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 consider three types of constraints. Let $D = (E, A, P, R, r)$ be a DTD, and T be an XML tree valid w.r.t. D. A *constraint* φ *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 $R(\tau)$. The XML tree T *satisfies* φ, denoted by $T \models \varphi$, iff in T,

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

—*Inclusion constraint:* $\tau_1[X] \subseteq \tau_2[Y]$, where $\tau_1, \tau_2 \in E$, and X, Y are nonempty lists of attributes in $R(\tau_1), R(\tau_2)$ of the same length. We write $T \models \varphi$ iff in T,

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

—*Foreign key:* a combination of two constraints, namely, an inclusion constraint $\tau_1[X] \subseteq \tau_2[Y]$ and a key $\tau_2[Y] \to \tau_2$. We write $T \models \varphi$ iff T satisfies both the key and the inclusion constraint.

That is, a key $\tau[X] \to \tau$ indicates that the set X of attributes is a key of elements of τ, i.e., two distinct τ nodes in T cannot have the same X-attribute values; an inclusion constraint $\tau_1[X] \subseteq \tau_2[Y]$ says that the list of X-attribute values of every τ_1 node in T must match the list of Y-attribute values of some τ_2 node in T; and an foreign key $\tau_1[X] \subseteq \tau_2[Y]$, $\tau_2[Y] \to \tau_2$ indicates that X is a foreign key of τ_1 elements referencing key Y of τ_2 elements.

Over a DTD D, the class $\mathcal{C}_{K,FK}$ of constraints consists of all the keys and foreign keys over D. They are called *multi-attribute* keys and foreign keys as they may be defined in terms of multiple attributes.

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

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

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

(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) $enroll[dept, course\_no] \subseteq course[dept, course\_no]$.

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

It is worth mentioning that 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. Note that a foreign key requires the presence of a key in addition to an inclusion constraint.

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

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

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

We shall also consider the following types of unary constraints over D:

—*inclusion constraint*: $\tau_1.l_1 \subseteq \tau_2.l_2$; unlike a foreign key, it does not require the presence of a key;
—the *negation of an inclusion constraint:* $\phi = \tau_1.l_1 \not\subseteq \tau_2.l_2$; for an XML tree T, $T \models \phi$ iff there is a τ_1 element x in T such that for all τ_2 element y in T, $x.l_1 \neq y.l_2$;
—the *negation of a key:* $\varphi = \tau.l \not\to \tau$; $T \models \varphi$ iff there are τ elements x_1, x_2 in T such that $x_1.l = x_2.l$, i.e., the value of the l attribute of a τ element cannot uniquely identify it in $ext(\tau)$.

With these we define two extensions of $\mathcal{C}^{Unary}_{K,FK}$ as follows. One is $\mathcal{C}^{Unary}_{K^{\neg},IC}$, the class consisting of unary keys, unary inclusion constraints and negations of unary keys. The other, $\mathcal{C}^{Unary}_{K^{\neg},IC^{\neg}}$, consists of unary keys, unary inclusion constraints and their negations. As mentioned earlier, we consider these classes mostly for the study of the implication problem for $\mathcal{C}^{Unary}_{K,FK}$ constraints.

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

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

The *implication problem* for $\mathcal{C}$ is to determine, given any DTD D, any set Σ and φ of $\mathcal{C}$ constraints 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 (cf. [1]). Let $\mathbf{R} = (R_1, \ldots, R_n)$ be a relational schema. For each relation (schema) R_i in $\mathbf{R}$, we write $Att(R_i)$ for the set of all attributes of R_i, and $Inst(R_i)$ for the set of finite instances of R_i. By database instances we mean *finite* instances. An 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:* $R[l_1, ..., l_k] \subseteq R'[l'_1, ..., l'_k]$ and $R'[l'_1, ..., l'_k] \to R'$, where R, R' are in $\mathbf{R}$, $[l_1, ..., l_k]$ and $[l'_1, ..., l'_k]$ are lists of attributes in $Att(R)$ and in $Att(R')$, respectively. In addition, the set consisting of $l'_1, ..., l'_k$ is a key of R'. We write $\mathbf{I} \models \varphi$ 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 *implies* 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 $\mathbf{R}$, any set Σ and φ 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 [19] 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. 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 list of attributes in $Att(R)$, and $[l'_1, ..., l'_k]$ is a list of attributes in $Att(R')$. In contrast to foreign keys, the set consisting of $l'_1, ..., l'_k$ is not necessarily a key of 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, I' are the instances of R, 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 *implies* 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 (see, e.g., [1] for a proof).

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

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

Note that every relation R has a key. In particular, $Att(R)$, the set of all attributes of R, is a key of R. Let Z be a key for R, i.e., $R[Z] \to R$. We define a new (fresh) relation schema R_{new} such that $Att(R_{new}) = XYZ$, i.e., the union of X, Y and Z. Intuitively, given an instance I of R, an instance I_{new} of R_{new} is to be constructed as a subset of $\Pi_{XYZ}(I)$ such that $\Pi_{XY}(I) = \Pi_{XY}(I_{new})$ and I_{new} satisfies the key $R_{new}[XY] \to R_{new}$, where $\Pi_W(I)$ denotes the projection of I on attributes W. That is, we eliminate tuples in $\Pi_{XYZ}(I)$ that violate the key. Observe that XYZ is a key for both R_{new} and R since it is the set of all attributes of R_{new}, and it contains the key Z of R (i.e., it is a *superkey* of R). Thus 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 $Att(R_{new}) = YZ$. Intuitively, given an instance I_2 of R_2, an instance I_{new} of R_{new} is to be constructed as a subset of $\Pi_{YZ}(I_2)$ by eliminating tuples that violate the key $R_{new}[Y] \to R_{new}$, such that $\Pi_Y(I_2) = \Pi_Y(I_{new})$ and I_{new} satisfies the key. Observe that YZ is a key for R_2 since it contains the key Z of R_2, i.e., it is a superkey of R_2. Thus 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 $\Sigma \vdash \theta$ iff $\Sigma' \vdash \phi_1$.

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

(1) Suppose that there is an instance $\mathbf{I}$ of $\mathbf{R}$ such that $\mathbf{I} \models \bigwedge \Sigma \wedge \neg\theta$. 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 $\mathbf{R}$, the instance of R in **I'** is the same as the instance of R in $\mathbf{I}$. We populate instances of new relations R_{new} created in the encoding as mentioned above. (a) If R_{new} is introduced in the encoding of a FD $R : X \to Y$ 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 $\mathbf{I}$. (b) If R_{new} is introduced in the encoding of an ID $R_1[X] \subseteq R_2[Y]$ 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 $\mathbf{I}$. 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 $\mathbf{I}$ of $\mathbf{R}$ by removing from **I'** all instances of new relations introduced in the encoding. It is easy to verify that $\mathbf{I} \models \bigwedge \Sigma \wedge \neg\theta$.

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 that the complement of the implication problem for keys by keys and foreign keys is also undecidable. That is to determine, given a relational schema $\mathbf{R}$, a set Σ of keys and foreign keys over $\mathbf{R}$ and a key φ over $\mathbf{R}$, whether there is an instance of $\mathbf{R}$ 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 $\mathbf{R}$, and $\varphi = R[X] \to R$ be a key over $\mathbf{R}$. Let $Y = Att(R) \setminus X$. We encode $\mathbf{R}$, Θ and φ in terms of a DTD D and a set Σ of $\mathcal{C}_{K,FK}$ constraints over D as follows. Let $D = (E, A, P, R_A, r)$, where

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

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:

—Σ_Θ includes $t_i[Z] \to t_i$ if Θ includes a key $R_i[Z] \to R_i$;
—Σ_Θ includes $t_i[Z] \subseteq t_j[Z']$, $t_j[Z'] \to t_j$ if Θ has a foreign key $R_i[Z] \subseteq R_j[Z']$, $R_j[Z'] \to R_j$.

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.

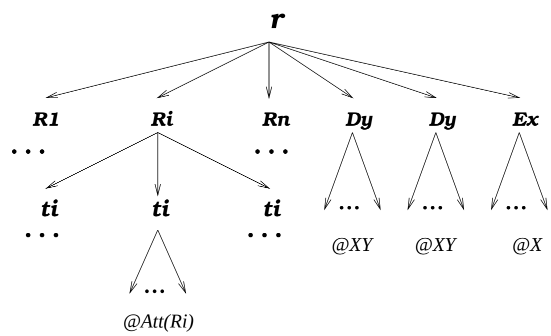

Fig. 2. A tree used in the proof of Theorem 3.1

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 having all attributes in X. If $T \models \Sigma_\varphi$, then (1) $d_1[X] = d_2[X]$ by $D_Y[X] \subseteq E_X[X]$ and the fact $|ext(E_X)| = 1$; and (2) $d_1[Y] \neq d_2[Y]$ 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 $T \models D$. 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 $Att(R) = X \cup Y$ and the set of all attributes of a relation is a key of the relation, 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 D has a valid XML tree T that satisfies 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. □

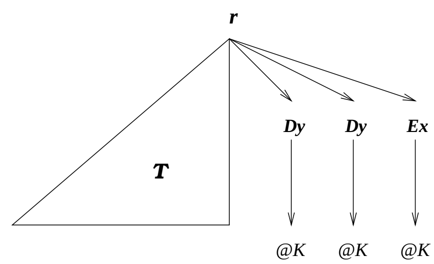

Fig. 3. A tree used in the proof of Lemma 3.3

3.2 Undecidability of implication

We next consider the implication problem.

LEMMA 3.3. *The following problems are undecidable: given any DTD D, any set Σ of $\mathcal{C}_{K,FK}$ constraints over D, any unary key φ_1 and unary inclusion constraint φ_2 over D, whether (1) $(D, \Sigma) \vdash \varphi_1$; (2) $(D, \Sigma) \vdash \varphi_2$.*

Proof: It suffices to establish a reduction from the consistency problem for $\mathcal{C}_{K,FK}$ to the complement of the implication problem for $\mathcal{C}_{K,FK}$. Let the DTD D be (E, A, P, R, r). 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 ϕ over D' as follows:

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

Clearly, Σ is also a set of $\mathcal{C}_{K,FK}$ constraints over D'. We next 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.

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 $T' \models \phi \wedge \varphi_1 \wedge \neg\varphi_2$. □

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 a valid XML tree. 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 involves keys only. Let $\mathcal{C}_K$ denote the set of all keys in $\mathcal{C}_{K,FK}$. The *consistency problem for $\mathcal{C}_K$* is to determine, given any DTD D and any set Σ of keys in $\mathcal{C}_K$ over D, whether there exists an XML tree valid w.r.t. D and satisfying Σ. Similarly, we consider the *implication problem for $\mathcal{C}_{K,FK}$*: given any DTD D, any set Σ and φ of keys in $\mathcal{C}_K$ 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 $\mathcal{C}_K$.
(3) The implication problem for $\mathcal{C}_K$.

Proof: (1) The first problem of the theorem can be reduced to the emptiness problem for a context free grammar (CFG). Observe that a DTD $D = (E, A, P, R, r)$ can be viewed as an extended CFG G_D with r as its start symbol, $\texttt{S}$ as a nonterminal with a production rule, say, $\texttt{S} \rightarrow 0$, and with attributes (A and R) ignored. It is easy to verify that D has a valid XML tree if and only if G_D is nonempty, i.e., it generates a terminal string (equivalently, a parse tree). Indeed, given an XML tree T valid w.r.t. D, one can construct a parse tree of G_D by modifying T, i.e., by removing attributes from T and modifying its text nodes. Conversely, given a parse tree T' of G_D one can construct a valid XML tree of D by modifying T', i.e., by adding attributes to T' and removing children of S nodes from T'. It is straightforward to convert the extended CFG G_D to a CFG G in linear time, by introducing new nonterminals and their (recursive) production rules to represent Kleene closures. Moreover, G_D is nonempty if and only if G is nonempty. It is well known that the emptiness problem for a CFG can be determined in linear time (cf. [24]). Putting everything together, a linear algorithm for checking the validity of D works as follows: it first generates in linear time the CFG G from D, and then checks in linear time whether G is empty; it concludes that D has a valid XML tree if and only if G is nonempty. Thus the validity of DTDs can be decided in linear time.

(2) We next prove the second statement of Theorem 3.5. That is, the consistency problem for $\mathcal{C}_K$ is decidable in linear time. Given any DTD D and any set Σ of keys in $\mathcal{C}_K$ over D, it suffices to show that Σ can be satisfied by an XML tree valid w.r.t. D if and only if D has a valid XML tree. 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 $T_1 = (V, lab, ele, att, val, root)$ valid w.r.t. D. We construct another XML tree T_2 by modifying the val function in T_1 such that for any key $\tau[X] \to \tau$ in Σ, $|ext(\tau)| = |ext(\tau.l)|$ in T_2 for every $l \in X$. That is, $T_2 \models \tau.l \to \tau$ for all $l \in X$. More specifically, let $T_2 = (V, lab, ele, att, val', root)$. 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, $T_2 \models \tau[X] \to \tau$ since for any $x, y \in ext(\tau)$, $x[X] \neq y[X]$. The other direction is immediate.

(3) Finally, we prove the last statement of Theorem 3.5. That is, the implication problem for $\mathcal{C}_K$ 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: As in the proof of the first statement of the theorem, it is easy to show that given a DTD D, one can find in linear time a CFG G such that D has a valid XML tree in which $|ext(\tau)| > 1$ if and only if the start symbol r of G derives a terminal string w whose parse tree has at least two τ nodes. This can be transformed in linear time to the problem of checking if a given CFG derives a string with at least two occurrences of a given terminal symbol, which in turn can be solved in linear time by a minor modification of the emptiness test for CFG from [24]. □

Let Σ be a set of keys in $\mathcal{C}_K$ over D, and $\varphi = \tau[X] \to \tau$ be another key in $\mathcal{C}_K$ over D. We say that Σ *subsumes* φ if there is $\phi = \tau[Y] \to \tau$ in Σ such that $Y \subseteq X$, i.e., φ is a *superkey* of ϕ. 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 $\mathcal{C}_K$ over D, and $\varphi = \tau[X] \to \tau$ another key in $\mathcal{C}_K$ 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 subsume 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 subsume 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 $\tau[Y] \to \tau$ with $Y \subseteq X$, 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'. We construct a new tree T by modifying the string values associated with the attributes of T', while leaving the other functions of T' unchanged. More specifically, we let $d_1[X] = d_2[X]$ in T but all other attributes have different string values. It is easy to verify that $T \models D$ and $T \models \neg\varphi$ by the definition of T. To show $T \models \Sigma$, suppose by contradiction that there were $\phi \in \Sigma$ such that $T \models \neg\phi$. Then ϕ must be of the form $\tau[Y] \to \tau$ where $Y \subseteq X$, i.e., φ is a superkey of ϕ, since except $d_1[X] = d_2[X]$, distinct nodes in T have the different attribute values by the definition of T. This contradicts the assumption that Σ does not subsume φ. Thus the first statement of the lemma holds.

To show that this can be done 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 φ is a superkey of some key in Σ (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 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 latter can be decided in linear time.

This completes the proof of Theorem 3.5. □

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 simplify the discussion and to establish a (slightly) stronger result, we consider a larger class of constraints, namely, $\mathcal{C}^{Unary}_{K,IC}$, the class of unary keys and unary inclusion constraints. In contrast to $\mathcal{C}^{Unary}_{K,FK}$, $\mathcal{C}^{Unary}_{K,IC}$ allows the presence of unary inclusion constraints independent of keys. 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 $\mathcal{C}^{Unary}_{K,IC}$ (and thus 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 consistency problem remains in NP when negations of keys are allowed, i.e., the problem for $\mathcal{C}^{Unary}_{K^{\neg},IC}$ constraints is also in NP. 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 on the integers, 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:

<u>LINEAR INTEGER PROGRAMMING (LIP)</u>

INPUT: An $m \times n$ matrix A of integers and a column vector $\vec{b}$ of m integers.

QUESTION: 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 [22]. In particular, when nonnegative integer solutions are considered, [32] 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 ($O(s^2 \cdot \log s)$) time 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, denoted by $|D|$ and $|\Sigma|$, respectively) the following:

—another DTD D_N, referred to as the *simplified DTD of D*, in which regular expressions are restricted to have at most one operator: either "|" (union) or "," (concatenation)[2]; we reduce the consistency of D and Σ to that of D_N and Σ, i.e., 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_N and satisfying Σ;

—a set C_Σ of linear integer constraints such that there is an XML tree valid w.r.t. D_N and satisfying Σ if and only if there is an XML tree valid w.r.t. D_N and satisfying C_Σ;

—a system Ψ_{D_N} of linear integer constraints such that there exists an XML tree valid w.r.t. D_N if and only if Ψ_{D_N} admits an integer solution; the cardinality constraints in Ψ_{D_N} are more complex than those studied in the context of object-oriented and relational databases [12; 13; 26];

—finally, a system of integer constraints $\Psi(D, \Sigma)$ from C_Σ and Ψ_{D_N} 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.

Putting everything together, we reduce the consistency problem for $\mathcal{C}^{Unary}_{K,IC}$ to the existence of a solution of an instance of LIP, and thus obtain the NP bound.

[2]We are grateful to one of the referees for suggesting this simplification of DTDs.

Proof of Theorem 4.1: We start by describing the process of simplifying DTDs. We shall then present an encoding of unary constraints and DTDs. Finally, we develop a characterization of XML specifications with both DTDs and unary constraints in terms of linear integer constraints.

Simplifying DTDs. We first explain how to reduce the consistency problem for $\mathcal{C}^{Unary}_{K,IC}$ to that over simple DTDs. Intuitively, we replace long regular expressions in $P(\tau)$ by shorter ones. Formally, consider a DTD $D = (E, A, P, R, r)$. For each $\tau \in E$, $P(\tau)$ is a regular expression α. A DTD is basically an extended regular grammar (cf. [11; 31]); thus $\tau \to \alpha$ can be viewed as the production rule for τ. We rewrite the regular expression α by introducing a set N of *new* element types (nonterminals) such that the production rules of the new DTD have one of the following forms:

$$\tau \to \tau_1,\ \tau_2 \qquad \tau \to \tau_1 \mid \tau_2 \qquad \tau \to \tau_1 \qquad \tau \to \ \mathtt{S} \qquad \tau \to \epsilon$$

where τ, τ_1, τ_2 are element types in $E \cup N$, S is the string type and ϵ denotes the empty word. More specifically, we conduct the following "simplifying" process on the production rule $\tau \to \alpha$:

(1) If $\alpha = (\alpha_1, \alpha_2)$, then we introduce two new element types τ_1, τ_2 and replace $\tau \to \alpha$ with a new rule $\tau \to \tau_1, \tau_2$. We proceed to process $\tau_1 \to \alpha_1$ and $\tau_2 \to \alpha_2$ in the same way.
(2) If $\alpha = (\alpha_1 | \alpha_2)$, then we introduce two new element types τ_1, τ_2 and replace $\tau \to \alpha$ with a new rule $\tau \to \tau_1 \mid \tau_2$. We proceed to process $\tau_1 \to \alpha_1$ and $\tau_2 \to \alpha_2$ in the same way.
(3) If $\alpha = \alpha_1^*$, then we introduce a new element type τ_1 and replace $\tau \to \alpha$ with $\tau \to \tau_1$. We proceed to process $\tau_1 \to \epsilon \mid \alpha_1, \tau_1$ in the same way.
(4) If α is one of $\tau' \in E$, S or ϵ, then the rule for τ remains unchanged.

To avoid introducing unnecessary new element types, in the first two cases above, if α_1 (resp. α_2) is a symbol of $E \cup \{\mathtt{S}\}$, we do not introduce a new element type for α_1 (resp. α_2).

We refer to the set of new element types introduced when processing $\tau \to P(\tau)$ as N_τ and the set of production rules generated/revised as P_τ. Note that $N_\tau \cap E = \emptyset$ for any $\tau \in E$.

We define a new DTD $D_N = (E_N, A, P_N, R_N, r)$, referred to as the *simplified DTD of D* (or just a *simple* DTD if D is clear from the context), where

—$E_N = E \cup \bigcup_{\tau \in E} N_\tau$, i.e., E plus those new element types introduced in the simplifying process;

—$P_N = \bigcup_{\tau \in E} P_\tau$, i.e., production rules generated/revised in the simplifying process;

—$R_N(\tau) = R(\tau)$ for each $\tau \in E$, and $R_N(\tau) = \emptyset$ for each $\tau \in E_N \setminus E$.

Note that the root element type r and the set A of attributes remain unchanged. Moreover, elements of any type in $E_N \setminus E$ do not have any attribute. Note that D_N does not contain the Kleene star "$*$".

For example, the simplified DTD of D_1 given in Section 1 is $D_1^N = (E_1^N, A_1, P_1^N, R_1^N, r)$, where

$$\begin{array}{ll}
E_1^N = \{teachers, teacher, teach, research, subject, \tau_t^1, \tau_t^2, \tau_\epsilon\} & \\
A_1 = \{name,\ taught_by\} & \\
P_1^N(teachers) & = teacher,\ \tau_t^1 \\
P_1^N(\tau_t^1) & = \tau_\epsilon \mid \tau_t^2 \\
P_1^N(\tau_\epsilon) & = \epsilon \\
P_1^N(\tau_t^2) & = teacher,\ \tau_t^1 \\
P_1^N(teacher) & = teach,\ research \\
P_1^N(teach) & = subject,\ subject \\
P_1^N(subject) & = P_1^N(research) = \mathtt{S} \\
R_1^N(teacher) & = \{name\} \\
R_1^N(subject) & = \{taught_by\} \\
R_1^N(teachers) & = R_1^N(teach) = R_1^N(research) = R_1^N(\tau_t^1) = R_1^N(\tau_t^2) = R_1^N(\tau_\epsilon) = \emptyset \\
r_1 = teachers &
\end{array}$$

Here $\tau_t^1, \tau_t^2, \tau_\epsilon$ are the new element types introduced.

The simplified DTD D_2^N of D_2 in Section 1 is the same as D_2 itself.

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 simplified DTD D_N of D. The next lemma establishes the connection between D and D_N, which allows us to consider only simple DTDs from now on.

LEMMA 4.3. *Let D be a DTD, D_N be the simplified DTD of D and Σ be a set of $\mathcal{C}_{K,IC}^{Unary}$ constraints over D. Then there exists an XML tree T_1 such that $T_1 \models D$ and $T_1 \models \Sigma$ iff there exists an XML tree T_2 such that $T_2 \models D_N$ and $T_2 \models \Sigma$.*

Proof: It suffices to show the following claim. For any XML tree $T_1 \models D$ one can construct an XML tree $T_2 \models D_N$, and for any $T_2 \models D_N$ one can construct $T_1 \models D$, such that for any element type τ in D and $l \in R(\tau)$, $|ext(\tau)|$ in T_2 equals $|ext(\tau)|$ in T_1, and $ext(\tau.l)$ in T_2 equals $ext(\tau.l)$ in T_1.

We first prove the lemma assuming that the claim is true. Assume that there exists an XML tree T_1 such that $T_1 \models D$ and $T_1 \models \Sigma$. Find the tree $T_2 \models D_N$ as in the claim. Suppose that there is $\varphi \in \Sigma$ such that $T_2 \not\models \varphi$. If φ is a key $\tau.l \to \beta.\tau$, then there are two distinct nodes $x, y \in ext(\tau)$ in T_1 such that $x.l = y.l$. Thus $|ext(\tau.l)| < |ext(\tau)|$ in T_2 since every τ element has a single l attribute. Since $T_1 \models \varphi$, it must be the case that $|ext(\tau.l)| = |ext(\tau)|$ in T_1 since the value $x.l$ of each $x \in ext(\tau)$ uniquely identifies x among all the nodes in $ext(\tau)$. This contradicts the claim that $|ext(\tau)|$ in T_2 equals $|ext(\tau)|$ in T_1 and $ext(\tau.l)$ in T_2 equals $ext(\tau.l)$ in T_1. If φ is an inclusion constraint $\tau_1.l_1 \subseteq \tau_2.l_2$, then there is $x \in ext(\tau_1)$ such that for all $y \in ext(\tau_2)$ in T_2, $x.l_1 \neq y.l_2$. That is, $x.l_1 \notin ext(\tau_2.l_2)$. By the claim, $x.l_1 \in ext(\tau_1.l_1)$ in T_1. Since $T_1 \models \varphi$, we have $x.l_1 \in ext(\tau_2.l_2)$ in T_1. Again by the claim, we have $x.l_1 \in ext(\tau_2.l_2)$ in T_2, which contradicts the assumption. The proof for the other direction is similar.

We next verify the claim. Given 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 by modifying T_1 such that $T_2 \models D_N$. Consider a τ element v in T_1. Let $ele_1(v) = [v_1, ..., v_n]$ and $w = lab_1(v_1) \ldots lab_1(v_n)$. Recall N_τ and P_τ, the set of nonterminals and the set of production rules generated when simplifying $\tau \to P(\tau)$. Let Q_τ be the set of E symbols that appear in P_τ plus S. We can view $G = (Q_\tau, N_\tau \cup \{\tau\}, P_\tau, \tau)$ as a context free grammar, where Q_τ is the set of terminals, $N_\tau \cup \{\tau\}$ the set of nonterminals, P_τ the set of production rules and τ the start symbol. Since $T_1 \models D$, we have $w \in P(\tau)$. By a straightforward induction on the structure of $P_N(\tau)$ it can be verified that w is in the language defined by G. Thus there is a parse tree $T(w)$ of the grammar G for w, and w is the frontier (the list of leaves from left to right) of $T(w)$. Without loss of generality, assume that the root of $T(w)$ is v, and the leaves are $v_1, \ldots, v_n$. Intuitively, we construct T_2 by replacing each element v in T_1 by such a parse tree. More specifically, let $T_2 = (V_2, lab_2, ele_2, att, val, root)$. Here V_2 consists of nodes in V_1 and the internal nodes introduced in the parse trees. For each x in V_2, let $lab_2(x) = lab_1(x)$ if $x \in V_1$, and otherwise let $lab_2(x)$ be the node label of x in the parse tree where x belongs. Note that nodes in $V_2 \setminus V_1$ are elements of some type in $E_N \setminus E$. If $lab_2(x)$ is an element type, let $ele_2(x)$ be the list of its children in the parse tree. Note that att and val remain unchanged. By the construction of T_2 it can be verified that $T_2 \models D_N$. Moreover, for any $\tau \in E$ and $l \in R(\tau)$, $|ext(\tau)|$ in T_2 equals $|ext(\tau)|$ in T_1 and $ext(\tau.l)$ in T_2 equals $ext(\tau.l)$ in T_1 because none of the new nodes, i.e., nodes in $V_2 \setminus V_1$, is labeled with an E type, and the function att remains unchanged.

Conversely, assume that there is $T_2 = (V_2, lab_2, ele_2, att, val, root)$ such that $T_2 \models D_N$. We construct T_1 by modifying T_2 such that $T_1 \models D$. For any node $v \in V_2$ with $lab(v) = \tau$ and $\tau \in E_N \setminus E$, we substitute the subelements of v for v in $ele(v')$, where v' is the parent of v. In addition, we remove v from V_2, $lab_2(v)$ from lab_2, and $ele_2(v)$ from ele_2. Observe that by the definition of D_N, no attributes are defined for elements of any type in $E_N \setminus E$. We repeat the process until there is no node labeled with element type in $E_N \setminus E$. Now let $T_1 = (V_1, lab_1, ele_1, att, val, root)$, where V_1, lab_1 and ele_1 are V_2, lab_2 and ele_2 at the end of the process, respectively. Observe that att, val and $root$ remain unchanged. By the definition of T_1 it can be verified that $T_1 \models D$; and in addition, for any $\tau \in E$ and $l \in R(\tau)$, $|ext(\tau)|$

in T_1 equals $|ext(\tau)|$ in T_2, and $ext(\tau.l)$ in T_1 equals $ext(\tau.l)$ in T_2, because none of the nodes removed is labeled with a type of E and the functions att and val are unchanged. □

It is easy to see that D_N is computable in linear time in the size of D.

Encoding unary constraints. We now give a coding of $\mathcal{C}^{Unary}_{K,IC}$ constraints. Let Σ be a set of $\mathcal{C}^{Unary}_{K,IC}$ constraints over DTD D and D_N be simplified DTD of D. Referring to an arbitrary XML tree T valid w.r.t. D, we derive from Σ a class of linear integer constraints on T, denoted by C_Σ and referred to *the cardinality constraints determined by* 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.
—$|ext(\tau.l)| \leq |ext(\tau)|$ and $0 \leq |ext(\tau.l)|$ are in C_Σ for any $\tau \in E$ and $l \in R(\tau)$.

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:

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

It is worth mentioning that $|ext(\tau.l)| = |ext(\tau)|$ characterizes a key $\tau.l \to \tau$. Indeed, for any XML tree T valid w.r.t. D_N, $T \models |ext(\tau.l)| = |ext(\tau)|$ iff $T \models \tau.l \to \tau$. However, things can go wrong when it comes to inclusion constraints. Although $T \models \tau_1.l_1 \subseteq \tau_2.l_2$ implies $T \models |ext(\tau_1.l_1)| \leq |ext(\tau_2.l_2)|$, the other direction does not necessarily hold. This does not lose generality as we do not intend to capture negations of inclusion constraints with this coding. Indeed, the lemma below shows that we are able to consider C_Σ instead of Σ when studying the consistency of Σ.

LEMMA 4.4. *Let D_N be a simplified DTD of D, 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 there exists an XML tree T_1 such that $T_1 \models D_N$ and $T_1 \models \Sigma$ if and only if there exists an XML tree T_2 such that $T_2 \models D_N$ and $T_2 \models C_\Sigma$. In addition, any XML tree valid w.r.t. D_N and satisfying Σ also satisfies C_Σ.*

Proof: It is easy to see that for any XML tree T_1 that satisfies Σ, it must be the case that $T_1 \models C_\Sigma$. Conversely, we show that if there exists an XML tree $T_2 = (V, lab, ele, att, val, root)$

such that $T_2 \models D_N$ and $T_2 \models C_\Sigma$, then we can construct an XML tree T_1 such that $T_1 \models D_N$ and $T_1 \models \Sigma$.

We construct T_1 from T_2 by modifying the function *val* while leaving V, *lab*, *ele*, *att* and *root* unchanged. As cardinality constraints of C_Σ do not involve text nodes, we change *val* for attributes only. More specifically, we modify $val(v)$ if $lab(v) \in A$, i.e., if v is an attribute, and leave $val(v)$ unchanged otherwise. Let $S = \{\tau.l \mid \tau \in E, l \in R(\tau)\}$. To define the new function, denoted by val', we first associate a set $V_{\tau.l}$ of string values with each $\tau.l$ in S. Let N be the maximum cardinality of $ext(\tau.l)$ in T_2, i.e., $N \geq |ext(\tau.l)|$ in T_2 for all $\tau.l \in S$. Let $V_S = \{a_i \mid i \in [1, N]\}$ be a set of distinct string values. For each $\tau.l \in S$, let $V_{\tau.l} = \{a_i \mid i \in [1, |ext(\tau.l)|]\}$, and for each $x \in ext(\tau)$, let $val'(att(x,l))$ be a string value in $V_{\tau.l}$ such that in T_1, $ext(\tau.l) = V_{\tau.l}$. In addition, for each key $\tau.l \to \tau$ in Σ, let $x.l$ be a distinct string value in $V_{\tau.l}$. This is possible because by the definition of T_1, (1) $ext(\tau)$ in T_1 equals $ext(\tau)$ in T_2; (2) $|ext(\tau.l)|$ in T_1 equals $|ext(\tau.l)|$ in T_2; and (3) $T_2 \models C_\Sigma$ and $|ext(\tau)| = |ext(\tau.l)|$ is in C_Σ. We next show that T_1 is indeed what we want. It is easy to verify that $T_1 \models D_N$ given the construction of T_1 from T_2 and the assumption that $T_2 \models D_N$. To show that $T_1 \models \Sigma$, we consider $\varphi \in \Sigma$ in the following cases. (1) If φ is a key $\tau.l \to \tau$, it is immediate from the definition of T_1 that $T_1 \models \varphi$ since for any $x \in ext(\tau)$, $x.l$ is a distinct string value in $V_{\tau.l}$. (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 $T_2 \models C_\Sigma$. Recall that by the definition of val', for $i \in [1, 2]$, $V_{\tau_i.l_i} = \{a_i \mid i \in [1, |ext(\tau_i.l_i)|]\}$ and in T_1, $ext(\tau_i.l_i) = V_{\tau_i.l_i}$. Thus $ext(\tau_1.l_1) \subseteq ext(\tau_2.l_2)$ in T_1. That is, $T_1 \models \varphi$. Therefore, $T_1 \models D_N$ and $T_1 \models \Sigma$. $\Box$

Observe that in the construction of T_1 above, it is possible that $ext(\tau_1.l_1) \subseteq ext(\tau_2.l_2)$ even if Σ does not imply $\tau_1.l_1 \subseteq \tau_2.l_2$. This does not have an impact on the consistency analysis, as negations of inclusion constraints are not involved in the analysis.

It is straightforward to verify that 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 $|\Sigma|$ and $|D|$.

Encoding DTDs. We next move to a coding of DTDs. By Lemma 4.3 we can consider simple DTDs only. Given any simple DTD $D = (E, A, P, R, r)$, we encode it in linear time with a system Ψ_D of linear integer constraints such that D has a valid XML tree if and only if Ψ_D has an integer solution.

We first describe the variables used in the system Ψ_D. For each symbol $\tau \in E \cup \{\mathtt{S}\}$, $|ext(\tau)|$ is a distinct variable. Intuitively, in an XML tree T conforming to D, $|ext(\tau)|$ keeps track of the number of all τ elements. In addition, for each occurrence of τ in the definition $P(\tau')$ of some element type τ', we also create a distinct variable. More specifically, we create such variables as follows: if $P(\tau') = \tau_1$ for $\tau_1 \in E \cup \{\mathtt{S}\}$, then we create a distinct variable $x^1_{\tau_1,\tau'}$; if $P(\tau') = (\tau_1, \tau_2)$ or $P(\tau') = (\tau_1|\tau_2)$, then we create two distinct variables $x^1_{\tau_1,\tau'}$ and $x^2_{\tau_1,\tau'}$.

Intuitively, for $i \in [1,2]$, $x^i_{\tau_1,\tau'}$ keeps track of the number of τ_i subelements at position i under all τ' elements in T. For example, given an element type definition `P(teach) = subject, subject`, we create two distinct variables $x^1_{(subject,\ teach)}$ and $x^2_{(subject,\ teach)}$. Let X_τ be the set of all variables of the form $x^i_{\tau,\tau'}$.

Using these variables, for each $\tau \in E$, we define a set ψ_τ of linear integer constraints that characterizes $P(\tau)$ quantitatively, as follows:

—If $P(\tau) = \tau_1$ for $\tau_1 \in E \cup \{\mathtt{S}\}$, then ψ_τ includes $|ext(\tau)| = x^1_{\tau_1,\tau}$. Referring to the XML tree T, this assures that each τ element has a unique τ_1 subelement.

—If $P(\tau') = (\tau_1, \tau_2)$, then ψ_τ includes $|ext(\tau)| = x^1_{\tau_1,\tau}$ and $|ext(\tau)| = x^2_{\tau_2,\tau}$. These assure that each τ element in T must have a unique τ_1 subelement and a unique τ_2 subelement.

—If $P(\tau') = (\tau_1|\tau_2)$, then ψ_τ includes $|ext(\tau)| = x^1_{\tau_1,\tau} + x^2_{\tau_2,\tau}$. These assure that each τ element in T must have either a τ_1 subelement or a τ_2 subelement, and thus the sum of the number of these τ_1 subelements and the number of τ_2 subelements equals the number of τ elements in T.

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

—$|ext(r)| = 1$; i.e., there is a unique root in any XML tree valid w.r.t. D;

—constraints of ψ_τ for each $\tau \in E$; these assure that $P(\tau)$ is satisfied;

—$|ext(\tau)| = \sum_{x^i_{\tau,\tau'} \in X_\tau} x^i_{\tau,\tau'}$ for each $\tau \in (E \setminus \{r\}) \cup \{\mathtt{S}\}$; this indicates that the set $ext(\tau)$ includes all τ elements no matter where they occur in an XML tree;

—$x \geq 0$ for any variable x used above; i.e., the number of elements (subelements) is nonnegative.

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 the variables of Ψ_D such that all the linear integer constraints in Ψ_D are satisfied.

As an example, let us consider the simple DTDs D^N_1 and D^N_2 given above. The cardinality constraints determined by these DTDs are given below:

$\Psi_{D^N_1}$:

$\psi_{teachers}$:	$\lvert ext(teachers)\rvert = x^1_{(teacher,\ teachers)}$	$\lvert ext(teachers)\rvert = x^2_{(\tau^1_t,\ teachers)}$
$\psi_{\tau^1_t}$:	$\lvert ext(\tau^1_t)\rvert = x^1_{(\tau_\epsilon,\ \tau^1_t)} + x^2_{(\tau^2_t,\ \tau^1_t)}$	
$\psi_{\tau^2_t}$:	$\lvert ext(\tau^2_t)\rvert = x^1_{(teacher,\ \tau^2_t)}$	$\lvert ext(\tau^2_t)\rvert = x^2_{(\tau^1_t,\ \tau^2_t)}$
$\psi_{teacher}$:	$\lvert ext(teacher)\rvert = x^1_{(teach,\ teacher)}$	$\lvert ext(teacher)\rvert = x^2_{(research,\ teacher)}$

ψ_{teach}: $|ext(teach)| = x^1_{(subject,\ teach)}$ $|ext(teach)| = x^2_{(subject,\ teach)}$
$\psi_{subject}$: $|ext(subject)| = x^1_{(\mathtt{S},\ subject)}$
$\psi_{research}$: $|ext(research)| = x^1_{(\mathtt{S},\ research)}$
moreover,
$|ext(teachers)| = 1$ $|ext(teacher)| = x^1_{(teacher,\ teachers)} + x^1_{(teacher,\ \tau_t^2)}$
$|ext(\tau_t^1)| = x^2_{(\tau_t^1,\ teachers)} + x^2_{(\tau_t^1,\ \tau_t^2)}$ $|ext(\tau_t^2)| = x^2_{(\tau_t^2,\ \tau_t^1)}$
$|ext(\tau_\epsilon)| = x^1_{(\tau_\epsilon,\ \tau_t^1)}$ $|ext(teach)| = x^1_{(teach,\ teacher)}$
$|ext(subject)| = x^1_{(subject,\ teach)} + x^2_{(subject,\ teach)}$
$|ext(research)| = x^2_{(research,\ teacher)}$ $|ext(S)| = x^1_{(\mathtt{S},\ subject)} + x^1_{(\mathtt{S},\ research)}$
all variables ≥ 0.

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

Ψ_{D_2}:
ψ_{db}: $|ext(db)| = x^1_{(foo,\ db)}$
ψ_{foo}: $|ext(foo)| = x^1_{(foo,\ foo)}$
moreover, $|ext(db)| = 1$ $|ext(foo)| = x^1_{(foo,\ db)} + x^1_{(foo,\ foo)}$ all variables ≥ 0.

It is easy to check that $\Psi_{D_1^N}$ is consistent, whereas $\Psi_{D_2^N}$ is not.

We next show that Ψ_D indeed characterizes the DTD D.

LEMMA 4.5. *Let D be a simple 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 T such that $T \models D$. In addition, for each $\tau \in E$, $|ext(\tau)|$ in T equals the value of the variable $|ext(\tau)|$ given by the solution of Ψ_D.*

Proof: First, assume that there is an XML tree T valid w.r.t. D. We define an integer solution of Ψ_D as follows. For each $\tau \in E \cup \{\mathtt{S}\}$, let the value of the variable $|ext(\tau)|$ be the number of τ nodes in T. We proceed to assign integer values (number of certain subelements) to other variables by considering the structure of $P(\tau)$ for each $\tau \in E$. (1) If $P(\tau) = \tau_1$ for some $\tau_1 \in E \cup \{\mathtt{S}\}$, then let the value of the variable $x^1_{\tau_1,\tau}$ be the number of τ_1 subelements of all τ elements in T. (2) If $P(\tau') = (\tau_1, \tau_2)$, then let the value of the variable $x^1_{\tau_1,\tau}$ (resp. $x^2_{\tau_2,\tau}$) be the number of the τ_1 (resp. τ_2) subelements of all τ elements. In particular, if $\tau_1 = \tau_2$, then

$x^1_{\tau_1,\tau}$ (resp. $x^2_{\tau_2,\tau}$) has the number of the first (resp. second) subelements of all τ elements. (3) If $P(\tau') = (\tau_1|\tau_2)$, then let the value of the variable $x^1_{\tau_1,\tau}$ (resp. $x^2_{\tau_2,\tau}$) be the number of τ_1 (resp. τ_2) subelements. If $\tau_1 = \tau_2$, then $x^1_{\tau_1,\tau}$ and $x^2_{\tau_2,\tau}$ may have any value as long as $|ext(\tau)| = x^1_{\tau_1,\tau} + x^2_{\tau_2,\tau}$. We next show that this assignment is an integer solution of Ψ_D. First, the value of any variable is nonnegative, as it is the number of certain elements (subelements) in T. Second, $|ext(r)| = 1$ as T has a unique root. Third, for each $\tau \in E$, by induction on the structure of $P(\tau)$, it can be verified that the assignment satisfies ψ_τ since $T \models D$ and ψ_τ describes $P(\tau)$ quantitatively. Finally, the value of the variable $|ext(\tau)|$ is equal to the sum of all variables of the form $x^i_{\tau,\ \tau'}$ ($i \in [1,2]$) since it counts all the τ elements in T no matter where they are. This can be easily verified by contradiction. Thus the assignment is indeed a solution of Ψ_D. Note that by the definition of the solution, the value of the variable $|ext(\tau)|$ given by the solution equals $|ext(\tau)|$ in T.

Conversely, assume that Ψ_D admits an integer solution. Observe that all these variables have nonnegative integer values because of the 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 nodes and label them with τ. We refer to this set of nodes as $ext(\tau)$. In addition, for each $v \in ext(\tau)$ and $l \in R(\tau)$, we create a distinct node, referred to as v_l, and label it with l. Let

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

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

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

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

It is easy to verify that these functions are well defined. Let $root$ be the node labeled r, which is unique by $|ext(r)| = 1$ in Ψ_D. Finally, to define the function ele, we first mark nodes in $ext(\tau)$ with variables in X_τ so that they can be grouped as subelements of certain elements. For each variable $x^i_{\tau,\ \tau'}$ in X_τ, we choose $x^i_{\tau,\ \tau'}$ many distinct nodes labeled τ and mark them with $x^i_{\tau,\ \tau'}$. Note that for each $\tau \in E \cup \{\mathtt{S}\}$, every τ node in $V \setminus \{root\}$ can be marked once and only once by $|ext(\tau)| = \sum_{x^i_{\tau,\tau'} \in X_\tau} x^i_{\tau,\tau'}$ in Ψ_D. Given these marked elements, starting at $root$, for each $\tau \in E$ and each τ node v, we define $ele(v)$ as follows. If $P(\tau)$ is $\tau_1 \in E \cup \{\mathtt{S}\}$, then we choose a distinct τ_1 node y marked with $x^1_{\tau_1,\tau}$ and let $ele(v) = [y]$. If $P(\tau) = (\tau_1, \tau_2)$, then we choose a τ_1 node y_1 marked with $x^1_{\tau_1,\tau}$ and a τ_2 node y_2 marked with $x^2_{\tau_2,\tau}$, and let $ele(v) = [y_1, y_2]$. If

$P(\tau) = (\tau_1|\tau_2)$, then we choose a node y marked with either $x^1_{\tau_1,\tau}$ or $x^2_{\tau_2,\tau}$ and let $ele(y) = [y]$. By Ψ_D constraints, each element or text node in $V \setminus \{root\}$ can be chosen once and only once as a subelement of some other element. By induction on the structure of $P(\tau)$, one can verify that T defined in this way is indeed an XML tree and $T \models D$. Finally, by the definition of T, $|ext(\tau)|$ in T equals the value of the variable $|ext(\tau)|$ given by the solution of Ψ_D. □

It is straightforward to show that given any simple DTD D, the set Ψ_D of cardinality constraints determined by D can be computed in linear time. As a result, the size of Ψ_D is linear in $|D|$.

Characterizing DTDs and unary constraints. To complete our characterization, given a DTD $D = (E, A, P, R, r)$ and a finite set Σ of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, we define a system $\Psi(D, \Sigma)$ of integer constraints. The system $\Psi(D, \Sigma)$, referred to as *the set of cardinality constraints determined by D and* Σ, is defined to be:

$$\Psi_{D_N} \ \cup \ C_\Sigma \ \cup \ \{(|ext(\tau)| > 0) \rightarrow (|ext(\tau.l)| > 0) \mid \tau \in E, \ l \in R(\tau)\},$$

where D_N is the simplified DTD of D, Ψ_{D_N} and C_Σ are the sets of cardinality constraints determined by D_N and Σ, respectively. In $\Psi(D, \Sigma)$ we treat $|ext(\tau.l)|$ as a variable.

We say that $\Psi(D, \Sigma)$ is *consistent* if and only if $\Psi(D, \Sigma)$ admits an integer solution.

For example, recall the DTDs D_1 and D_2, and the constraint sets Σ_1 and Σ_2 (the empty set) given 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.

Observe that $\Psi(D, \Sigma)$ can be partitioned into two sets: $\Psi(D, \Sigma) = \Psi^l(D, \Sigma) \cup \Psi^c(D, \Sigma)$, where $\Psi^l(D, \Sigma)$ consists of linear integer constraints, and $\Psi^c(D, \Sigma)$ consists of constraints of the form $(|ext(\tau)| > 0 \rightarrow |ext(\tau.l)| > 0)$, which are to ensure that every τ element has an l attribute. Note that $|ext(\tau.l)| \leq |ext(\tau)|$ is already in C_Σ.

It is easy to verify that $\Psi(D, \Sigma)$ can be computed in linear time in $|D|$ and $|\Sigma|$, and thus its size is also linear in $|D|$ and $|\Sigma|$.

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

LEMMA 4.6. *Let D be a DTD, Σ be a finite set of $\mathcal{C}^{Unary}_{K,IC}$ constraints over D, and $\Psi(D, \Sigma)$ be the set of cardinality constraints determined by D and Σ. Then $\Psi(D, \Sigma)$ is consistent if and only if there exists an XML tree T such that $T \models D$ and $T \models \Sigma$.*

Proof: Let D_N be the simplified DTD of D. By Lemma 4.3, it suffices to show that $\Psi(D, \Sigma)$ is consistent if and only if there is an XML tree T such that $T \models D_N$ and $T \models \Sigma$.

Suppose that there exists an XML tree T such that $T \models D_N$ and $T \models \Sigma$. We show that

$\Psi(D, \Sigma)$ admits an integer solution. By Lemma 4.4, we have $T \models C_\Sigma$, where C_Σ is the set of cardinality constraints determined by Σ. By Lemma 4.5, one can define an integer solution of Ψ_{D_N}. The assignment assures that for each $\tau \in E$, the value of the variable $|ext(\tau)|$ equals the number of all the τ nodes in T. We extend the assignment as follows: for each $\tau \in E$ and $l \in R(\tau)$, let the value of the variable $|ext(\tau.l)|$ be the number of distinct l attribute values of all the τ nodes in T. Thus by $T \models C_\Sigma$, this extended assignment satisfies C_Σ. In addition, if $|ext(\tau)| > 0$ then $|ext(\tau.l)| > 0$ as every τ element in T has an l attribute. Hence the assignment is indeed a solution to $\Psi(D, \Sigma)$. Thus $\Psi(D, \Sigma)$ is consistent.

Conversely, suppose that $\Psi(D, \Sigma)$ admits an integer solution. We show that there is an XML tree T such that $T \models D_N$ and $T \models \Sigma$. Observe that an integer solution to $\Psi(D, \Sigma)$ is also a solution to Ψ_{D_N}. Thus by Lemma 4.5, there is $T' = (V, lab, ele, att, val, root)$ such that $T' \models D_N$. Moreover, for each $\tau \in E$, $|ext(\tau)|$ in T' is equal to the value of the variable $|ext(\tau)|$ given by the assignment. We construct another XML tree T'' by modifying the definition of the function *val* of T' such that for each $\tau \in E$ and $l \in R(\tau)$, $|ext(\tau.l)|$ in T'' equals the value assigned to the variable $|ext(\tau.l)|$ 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. Moreover, by $(|ext(\tau)| > 0 \rightarrow |ext(\tau.l)| > 0)$ in $\Psi(D, \Sigma)$, every τ element in T'' can have an l attribute. It is straightforward to verify that $T'' \models C_\Sigma$ and $T'' \models D_N$. Hence by Lemma 4.4, there exists an XML tree T such that $T \models D_N$ and $T \models \Sigma$. □

Given these lemmas, we proceed to prove Theorem 4.1.

Proof of Theorem 4.1 (continued): We encode an instance (D, Σ) of the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ as an instance of LIP. By Lemma 4.6, it suffices to encode $\Psi(D, \Sigma)$ as an instance of LIP. Recall that $\Psi(D, \Sigma)$ can be partitioned into two sets: $\Psi^l(D, \Sigma)$ of linear integer constraints, and $\Psi^c(D, \Sigma)$ of constraints of the form $(x > 0 \rightarrow y > 0)$. We first encode $\Psi(D, \Sigma)$ with a set of linear integer constraints. Let S be the set of all the pairs (x, y) for each constraint $(x > 0 \rightarrow y > 0)$ in $\Psi^c(D, \Sigma)$. For each subset X of S, we define Ψ_X to be

$$\Psi^l(D, \Sigma) \cup \{x = 0,\ y = 0 \mid (x, y) \in X\} \cup \{x \geq 1,\ y \geq 1 \mid (x, y) \in S \setminus X\}.$$

It is easy to see that $\Psi(D, \Sigma)$ admits an integer solution if and only if there is some Ψ_X that has an integer solution. Observe that Ψ_X can be represented as an instance of LIP since an equality $F_1 = F_2$ is equivalent to inequalities $F_1 \geq F_2$ and $F_2 \geq F_1$. In addition, for all variables x in $\Psi(D, \Sigma)$, we have $x \geq 0$ in $\Psi(D, \Sigma)$. Thus any solution of Ψ_X is nonnegative. Hence we can apply the result of [32] here, which says that if Ψ_X has an integer solution, then it has one in which the values of all variables are no larger than $n\,(m\,a)^{2m+1}$, where a is the largest absolute value of the constants in Ψ_X. In other words, Ψ_X has an integer solution in which the value of each variable has a length in binary of at most $1 + \lceil \log n + (2m+1) \cdot \log(ma) \rceil$ many bits, and the bounds on solutions for all Ψ_X's are the same. 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 c can be computed in $O(s\ logs)$ time. Thus we define a new system Φ of linear integer constraints that is the same as $\Psi^l(D, \Sigma)$ except it also includes $c\,y \geq x$ for all $(x > 0) \rightarrow (y > 0)$ in $\Psi^c(D, \Sigma)$. It is easy to verify that $\Psi(D, \Sigma)$ has an integer solution iff Φ has an integer solution. Indeed, if $\Psi(D, \Sigma)$ has an integer solution then it has one bounded by c. Thus the solution satisfies $c\,y \geq x$, i.e., it is an integer solution to Φ. Conversely, if Φ has an integer solution, then it is also an integer solution of $\Psi^l(D, \Sigma)$ and moreover, if $x > 0$ then $y > 0$ by $c\,y \geq x$ in Φ; that is, it is an integer solution to $\Psi(D, \Sigma)$. As Φ can be represented as an instance of LIP, we can define an matrix A_Ψ and a vector $\vec{b}_\Psi$ of integers such that $\Psi(D, \Sigma)$ has an integer solution if and only if $A_\Psi\,\vec{x} \geq \vec{b}_\Psi$ has an integer solution. Recall that $\Psi(D, \Sigma)$ can be computed in linear time and its size, denoted by s, is linear in $|D|$ and $|\Sigma|$. Thus the instance of LIP can be computed in $O(s^2 \cdot \log s)$ time in $|D|$ and $|\Sigma|$.

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

We next establish the precise complexity bound on the consistency problem for unary keys and foreign keys:

THEOREM 4.7. *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 [22].

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

$$E = \{r\} \cup \{F_i \mid i \in [1, m]\} \cup \{b_i \mid i \in [1, m]\} \cup \{VF_i \mid i \in [1, m]\}$$

$$\cup \{X_{ij} \mid i \in [1,m],\, j \in [1,n]\} \;\cup\; \{Z_{ij} \mid i \in [1,m],\, j \in [1,n]\}$$

$$A = \{v\} \cup \{A_{ij} \mid i \in [1,m],\, j \in [1,n]\}$$

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

An XML tree valid w.r.t. D has the form shown in Figure 4. Intuitively, X_{ij} encodes x_j in F_i, and Z_{ij} encodes the value of X_{ij}: X_{ij} has value 1 if and only if X_{ij} has a Z_{ij} child. The element type VF_i is to code the value of F_i. Observe that $A\,\vec{x} = \vec{b}$ has a solution if and only if for each row $i \in [1,m]$ there is exactly one column $j \in [1,n]$ such that $a_{ij} = 1$ and $x_j = 1$. In the XML tree T representing the instance, this means that for every i there is exactly one X_{ij} element with a Z_{ij} child. This is achieved by restricting F_i to have a unique VF_i descendant, and thus to have value 1, by means of the attribute v of VF_i and constraints. More specifically, we include the following in the set Σ:

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

These ensure that $F_i = b_i = 1$ as T has a unique b_i node. In addition, to ensure that all occurrences of x_j have the same value, the following are in Σ: for $j \in [1,n]$ and $i, l \in [1,m]$,

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

These assert that X_{ij} has value 1 if and only if X_{lj} equals 1. It is easy to see 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 D has a valid XML tree satisfying Σ. Thus this is indeed a PTIME reduction from the variant of LIP. □

Recall that 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 one can specify at most one key. 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 a class $\mathcal{C}$ of XML constraints is to determine, given any DTD D and

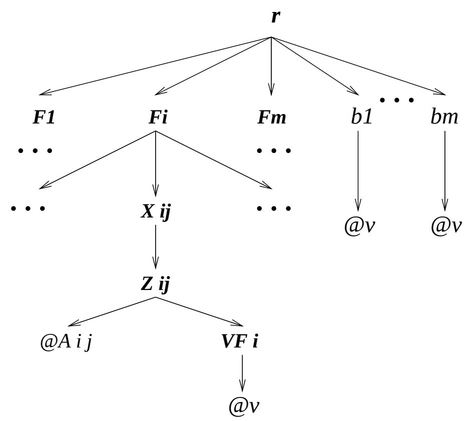

Fig. 4. A tree used in the proof of Theorem 4.7

finite set Σ of $\mathcal{C}$ 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; similarly for implication.

One might think that the primary key restriction would simplify the consistency analysis of $\mathcal{C}^{Unary}_{K,FK}$ constraints. However, it is not the case.

COROLLARY 4.8. *Under the primary key restriction, the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ remains NP-complete.*

Proof: The reduction from LIP given in the proof of Theorem 4.7 defines at most one key for each element type. □

A mild generalization of the encoding above can establish the complexity of the consistency problem for $\mathcal{C}^{Unary}_{K^\neg,IC}$, the class of unary keys, inclusion constraints and negations of keys. As we shall see shortly, the result for $\mathcal{C}^{Unary}_{K^\neg,IC}$ helps us study implication of $\mathcal{C}^{Unary}_{K,FK}$ constraints.

COROLLARY 4.9. *The consistency problem for $\mathcal{C}^{Unary}_{K^\neg,IC}$ constraints is NP-complete.*

Proof: Since $\mathcal{C}^{Unary}_{K,FK}$ is a sub-language of $\mathcal{C}^{Unary}_{K^\neg,IC}$, from Theorem 4.7 follows immediately that the consistency problem for $\mathcal{C}^{Unary}_{K^\neg,IC}$ is NP-hard. We next show that the problem remains in NP. Let D be a DTD and Σ be a set of $\mathcal{C}^{Unary}_{K^\neg,IC}$ 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 $T \models D$. 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

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

As $\Psi(D, \Sigma)$ can be computed in PTIME, 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 Lemmas 4.5 and 4.6 and Theorem 4.1, it can be verified that 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 $\tau.l \not\to \tau$, 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. Thus by $T \models \Sigma_2$, 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 $\Psi(D, \Sigma_1)$, again by Lemma 4.5 and 4.6 and Theorem 4.1, it can be verified that 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 $\tau.l \not\to \tau$ 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, $T \models \tau.l \not\to \tau$. Hence $T \models D$ and $T \models \Sigma$. □

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 4.8.

Corollary 4.9 also tells us the complexity of a special case of the implication problem for $\mathcal{C}^{Unary}_{K,FK}$, referred to as *implication problem for unary keys by $\mathcal{C}^{Unary}_{K,FK}$ constraints*:

THEOREM 4.10. *The following is coNP-complete, even under the primary key restriction: given any DTD D, any set Σ of $\mathcal{C}^{Unary}_{K,FK}$ constraints and any unary key φ over D, whether $(D, \Sigma) \vdash \varphi$.*

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 $T \models \neg\varphi$. Since $\Sigma \cup \{\neg\varphi\}$ is a set of $\mathcal{C}^{Unary}_{K\neg,IC}$ constraints, the implication problem for unary keys by $\mathcal{C}^{Unary}_{K,FK}$ constraints is in coNP by Corollary 4.9. To see that the problem is coNP-hard, recall the encoding given in the proof of Lemma 3.3. If the set Σ of constraints given is a set of $\mathcal{C}^{Unary}_{K,FK}$ constraints, then that encoding also serves as a reduction from the consistency problem for $\mathcal{C}^{Unary}_{K,FK}$ to the complement of $(D, \Sigma) \vdash \varphi$. Thus from Theorem 4.1 follows that the implication problem for unary keys by $\mathcal{C}^{Unary}_{K,FK}$ constraints is coNP-hard. Observe that the reduction in the proof of Lemma 3.3 defines at most one key for each element type. Thus given a set Σ of constraints, if Σ satisfies the primary key restriction, then so does the set of all constraints used in the reduction. Hence it remains coNP-hard even under the primary key restriction. □

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. The motivation for considering a fixed DTD is because in practice, one often defines the DTD of a specification at one time, but writes constraints in stages: constraints are added incrementally when new requirements are discovered.

Corollary 4.11. *For a* fixed *DTD, the following problems are decidable in PTIME:*

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

Proof: By Theorems 4.1, 4.10 and Corollary 4.9, an instance (D, Σ) of these problems can be encoded as a system Φ of linear integer constraints. That is, these problems can be reduced to checking whether Φ admits an integer solution. The system Φ consists of constraints of C_Σ (derived from Σ) and Ψ_{D_N} (derived from the simplified DTD D_N of D), and can be computed in PTIME in $|D|$. Given a fixed DTD D, the number of variables in C_Σ is bounded by the size of D ($O(|D|^2)$), and the number of variables in Ψ_{D_N} is also fixed. Thus the number of variables in Φ is bounded. It is known that when the number of variables in a system of linear integer constraints is bounded, checking whether the system admits an integer solution can be done in PTIME [29]. Putting these together, we have Corollary 4.11. □

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 4.7, 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 4.9. 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

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

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 $|\, ext(\tau_i.l_i) \setminus ext(\tau_j.l_j)\,|$ for v_{ij}. Thus $v_{ij} > 0$ in $\Psi(D, \Sigma)$ says that $ext(\tau_i.l_i) \not\subseteq ext(\tau_j.l_j)$ 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 4.9 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 $|\, ext(\tau_i.l_i) \setminus ext(\tau_j.l_j)\,|$. 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

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

Again as in the proof of Corollary 4.9, 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 $ext(\tau_i.l_i) = A_i$ 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 $|\, ext(\tau_i.l_i) \setminus ext(\tau_j.l_j)\,| > 0$. 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

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

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 [32] 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 an 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

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

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., [22]).

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

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

By the assumption, all entries in $\mathbf{U}, \mathbf{V}$ are bounded by M, and hence we may assume that 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 $i \leq n$, and $U \setminus A_{i-n}$ 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 that 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 $Y_{n+i} = U \setminus Y_i$ for all $i \leq n$. We have $w_{i,n+i} = v_{ii} = 0$, and thus $Y_{n+i} \subseteq U \setminus Y_i$. 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 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 $U \setminus Y_i$ 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 a 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. □

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.*

Proof: The implication 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 constraint), 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. One can show that it is coNP-hard under the primary key restriction using an argument similar to the proof of Theorem 4.10. Similarly for the implication problem for $\mathcal{C}^{Unary}_{K,FK}$. □

Finally, along the same lines as Corollary 4.11, 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* $\mathcal{C}^{Unary}_{K^\neg,IC^\neg}$.

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 4.11, 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 [29]. 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. □

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 par-

	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

Fig. 5. The main results of the paper

ticular, 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 [27], XML Schema [36] and the generalization of DTDs proposed in [33]. It remains open, however, whether the upper bounds (i.e., the decidability and NP membership results) are still intact in these settings.

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 extend 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 as normalization of relational schemas. Coding with linear integer constraints gives us decidability for some implication problems for XML constraints, which is a first step towards a design theory for XML specifications.

Acknowledgments. We thank Michael Benedikt, Alberto Mendelzon, Frank Neven and Dan Suciu for helpful discussions. We are grateful to the referees for valuable suggestions on simplifying the proofs and on improving the paper.

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. *JCSS*, 58(4):428–452, 1999.

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

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

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

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

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

[8] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan. Reasoning about keys for XML. In *DBPL'01*, 2001.

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

[10] P. Buneman, W. Fan, and S. Weinstein. Path constraints in semistructured databases. *JCSS*, 61(2):146–193.

[11] 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.

[12] D. Calvanese and M. Lenzerini. Making object-oriented schemas more expressive. In *PODS'94*, pages 243–254.

[13] D. Calvanese and M. Lenzerini. On the interaction between ISA and cardinality constraints. In *ICDE'94*, pages 204–213.

[14] D. Chamberlin et al. XQuery 1.0: An XML Query Language. W3C Working Draft, June 2001. `http://www.w3.org/TR/xquery`.

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

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

[17] 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.

[18] W. Fan and L. Libkin. On XML integrity constraints in the presence of DTDs. In *PODS'01*, pages 114–125.

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

[20] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Verifying integrity constraints on Web sites. In *IJCAI'99*, pages 614–619.

[21] 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.

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

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

[24] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation* (2nd Edition). Addison Wesley, 2000.

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

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

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

[28] D. Lee and W. W. Chu. Constraint-preserving transformation from XML document type to relational schema. In *ER'00*, pages 323–338, 2000.

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

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

[31] F. Neven. Extensions of attribute grammars for structured document queries. In *DBPL'99*, pages 99–116, 1999.

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

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

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

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

[36] H. S. Thompson et al. XML Schema. W3C Working Draft, May 2001. `http://www.w3.org/XML/Schema`.

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

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