On Incomplete XML Documents with Integrity Constraints

Pablo Barceló[1], Leonid Libkin[2], and Juan Reutter[2]

[1] Department of Computer Science, University of Chile
[2] School of Informatics, University of Edinburgh

Abstract. We consider incomplete specifications of XML documents in the presence of schema information and integrity constraints. We show that integrity constraints such as keys and foreign keys affect consistency of such specifications. We prove that the consistency problem for incomplete specifications with keys and foreign keys can always be solved in NP. We then show a dichotomy result, classifying the complexity of the problem as NP-complete or PTIME, depending on the precise set of features used in incomplete descriptions.

1 Introduction

While much is known about the transfer and extension of traditional relational tools to XML data, the study of incomplete information in XML has not yet received much attention. Various papers considered specific tasks related to the handling of incomplete information in XML. For example, [2] concentrated on incompleteness arising in a setting where the structure of a document is revealed by a sequence of queries, [10, 11] expressed incompleteness by means of description logic theories, and [16] showed how to deal with incompleteness in query results.

In relational theory, incompleteness of information has been studied independently of any particular application, with two seminal papers providing the foundation of the theory of databases with incomplete information. The paper [13] by Imielinski and Lipski introduced tables as a representation mechanism for incompleteness, and studied query evaluation over different types of tables. The paper [1] by Abiteboul, Kanellakis, and Grahne then answered most fundamental questions related to the complexity of computational problems associated with incompleteness.

A recent paper [5] attempted to re-do the basic results of [1, 13] in the XML context. It defined incomplete XML documents, and looked at two basic classes of problems related to them:

Representation Given an incomplete description of a document, does it represent some document (i.e., is it consistent)? And can it represent a specific complete document?

Querying How does one answer queries posed over incomplete descriptions? Specifically, how does one compute answers to queries in a way that is consistent with every document that is represented by the incomplete description, and what is the complexity?

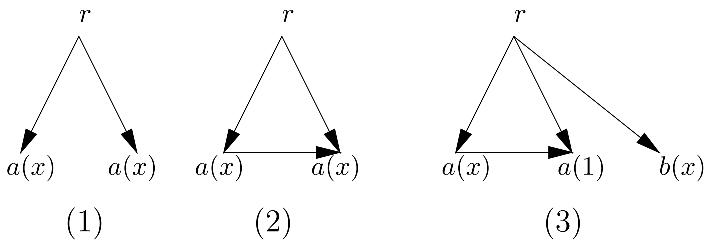
While [5] answered these questions for many types of incomplete descriptions, even in the presence of XML schemas, it did not look at the issue of documents with integrity constraints. However, integrity constraints are ubiquitous in the XML context: many documents are generated from databases that typically specify keys and inclusion constraints, and such constraints have now found their way into the standards for describing XML documents.

So we would like to see how the key tasks of handling incompleteness in XML behave when integrity constraints enter the picture. In the relational context, it is known that constraints often change the complexity of many tasks, from dependency implication to representation to querying [4, 18, 9].

In this paper we deal with the *Representation* task. As the membership problem (is a complete document represented by an incomplete description?) is not affected by the presence of constraints, we concentrate on the following:

Consistency Problem Given a schema S, an incomplete description of a document t, and a set Δ of constraints, are they consistent? That is, is there a complete document T represented by t that conforms to the schema S and satisfies all the constraints in Δ?

To see why constraints change the picture for XML with incomplete information, consider three incomplete descriptions of documents below.



Putting $a(x)$ next to a node means that it is labeled a, and the value of its attribute is not known; putting $a(1)$ means that the value of the attribute is 1. Note that variables can be re-used, i.e., we have *naïve* nulls.

Without integrity constraints, all three descriptions are consistent: we can assign any value to x, and any ordering to children that agrees with the horizontal edge we have. Now look at the tree (1) and assume that we have a constraint saying that the attribute of a-nodes is a key. Since we did not specify any relationship between the children of the a-nodes in this tree, the description is still consistent, as it is satisfied by a tree with just one a-child of the root. But tree (2) is not consistent: the horizontal edge tells us that there are two *distinct* a-nodes with the same value of their attribute, which therefore cannot be a key.

Now let us look at tree (3). Assume that we have the same key information about a. The description is consistent: one can set x to be any value other than 1. Likewise, if we say that the attribute of b is a key, the description remains consistent. However, if we add an inclusion constraint that attribute values of a-nodes are among the attribute values of b-nodes, the tree is becoming inconsistent: the key constraint tells us that $x \neq 1$ and thus the inclusion constraint is violated. In order to restore consistency, a tree that "completes" this description must add a new b-node under the root with attribute value 1.

The consistency problem for schemas and constraints (without incompleteness) was studied in [12, 3], with the main result stating that it is:

- undecidable, if non-unary constraints are used (i.e., n-attribute keys and foreign keys, for $n > 1$); and
- NP-complete for unary constraints.

Hence, in the paper we consider only unary constraints. Our main questions are:

(a) Do upper bounds on the complexity of the problem continue to hold in the presence of incomplete information?
(b) What is the precise complexity of the consistency problem?

Our main results answer these questions as follows:

(a) For DTDs, unary constraints, and incomplete information, we retain the NP upper bound.
(b) With DTDs, even very simple instances of the problem are NP-complete. Without DTDs, we prove a *dichotomy* theorem, classifying the complexity as either NP-complete or PTIME, depending on what features are allowed in incomplete descriptions.

Organization Basic notations are given in Section 2. Incomplete descriptions of XML documents are presented in Section 3. The consistency problem is described in Section 4. We state the main result and prove it. Due to space limitations, some proofs are shown in the appendix.

2 Preliminaries

XML documents and DTDs Assume that we have the following disjoint countably infinite sets:

- *Labels* of possible names of element types (that is, node labels in trees);
- *Attr* of attribute names; we precede them with an @ to distinguish them from element types;
- $\mathcal{I}$ of node ids; and
- $\mathcal{D}$ of attribute values (e.g., strings).

We formally define trees as two-sorted relational structures over node ids and attribute values.

For finite sets of labels and attributes, $\Sigma \subset \mathit{Labels}$ and $A \subset \mathit{Attr}$, define the vocabulary

$$\tau_{\Sigma,A} \;=\; (E, \mathit{NS}, (A_{@a})_{@a \in A}, (P_\ell)_{\ell \in \Sigma})$$

where all relations are binary except for the P_ℓ's, which are unary. A tree T is a 2-sorted structure of vocabulary $\tau_{\Sigma,A}$, i.e. $T = \langle V, D, \tau_{\Sigma,A} \rangle$, where $V \subset \mathcal{I}$ is a finite set of node ids, $D \subset \mathcal{D}$ is a finite set of data values, and

- E, NS are the child and the next-sibling relations, so that $\langle V, E, \mathit{NS} \rangle$ is an ordered unranked tree; we also use E^* and NS^* to denote their reflexive-transitive closures (respectively, descendant or self, and younger sibling or self).
- each $A_{@a_i}$ assigns values of attribute $@a_i$ to nodes, i.e. it is a subset of $V \times D$ such that at most one pair (s, c) is present for each $s \in V$;
- P_ℓ are labeling predicates: $s \in V$ belongs to P_ℓ iff it is labeled ℓ; as usual, we assume that the P_ℓ's are pairwise disjoint.

We refer to a node that is labeled ℓ as an ℓ-node, and to the attribute $@a$ of a node as its $@a$-attribute.

A DTD over a set $\Sigma \subset \mathit{Labels}$ of labels and $A \subset \mathit{Attr}$ of attributes is a triple $d = (r, \rho, \alpha)$, where $r \in \Sigma$, and ρ is a mapping from Σ to regular languages over $\Sigma - \{r\}$, and α is a mapping from Σ to subsets of A. As usual, r is the root, and in a tree T that conforms to d (written as $T \models d$), for each node s labeled ℓ, the set of labels of its children, read left-to-right, forms a string in the language of $\rho(\ell)$, and the set of attributes of s is precisely $\alpha(\ell)$. We assume, for complexity results, that regular languages are given by NFAs.

XML Integrity Constraints We consider keys, inclusion constraints and foreign keys as our integrity constraints. They are the most common constraints in relational databases, and are common in XML as well, as many documents are generated from databases. Moreover, these sets of constraints are similar to, but more general than XML ID/IDREF specifications, and can be used to model most of the key/keyref specifications of XML Schema used in practice [17, 15].

Let $\Sigma \subset \mathit{Labels}$ and $A \subset \mathit{Attr}$, and let T be an XML tree. Then a *constraint* φ over Σ and A is one of the following:

- *Key* $\ell.X \to \ell$, where $\ell \in \Sigma$ and X is a set of attributes from A. The XML tree T satisfies φ, denoted by $T \models \varphi$ iff for every ℓ-node s in T, X is contained in the set of attributes of s, and, in addition, T satisfies

$$\forall x \forall y \Big(P_\ell(x) \wedge P_\ell(y) \wedge \bigwedge_{@a \in X} \exists u \big(A_{@a}(x, u) \wedge A_{@a}(y, u) \big) \Big) \to x = y.$$

- *Inclusion Constraint* $\ell_1[X] \subseteq \ell_2[Y]$, where $\ell_1, \ell_2 \in \Sigma$ and $X = @a_1, \ldots, @a_n$ and $Y = @b_1, \ldots, @b_n$ are nonempty lists of attributes from A of the same length. We write $T \models \varphi$ iff for every ℓ_1-node (resp. ℓ_2-node) s in T, X (resp. Y) is contained in the set of attributes of s, and, in addition, T satisfies

$$\forall x \, \forall u_1 \cdots \forall u_n \left(\big(P_{\ell_1}(x) \wedge \bigwedge_{1 \le i \le n} A_{@a_i}(x, u_i)\big) \rightarrow \big(\exists y P_{\ell_2}(y) \wedge \bigwedge_{1 \le i \le n} A_{@b_i}(y, u_i)\big) \right).$$

- *Foreign Key*: A combination of an inclusion constraint and a key constraint, namely $\ell_1[X] \subseteq_{\mathrm{FK}} \ell_2[Y]$ if $\ell_1[X] \subseteq \ell_2[Y]$ and $\ell_2.Y \rightarrow \ell_2$. We write $T \models \varphi$ if T satisfies both the inclusion and the key constraint.

As usual, a key $\ell.X \rightarrow \ell$ indicates that two nodes labeled ℓ cannot have the same X-attribute values (i.e., X-attributes uniquely determine the node), an inclusion constraint $\ell_1[X] \subseteq \ell_2[Y]$ indicates that the list of X-attribute values of every ℓ_1 node must match the list of Y-attribute values of an ℓ_2-node, and a foreign key $\ell_1[X] \subseteq_{\mathrm{FK}} \ell_2[Y]$ indicates that X is a foreign key of ℓ_1-nodes referencing the key Y of ℓ_2-nodes.

A constraint is called *unary* if all sets of attributes involved are singletons. That is, unary keys are $\ell.@a \rightarrow \ell$ and unary inclusion constraints are $\ell_1[@a_1] \subseteq \ell_2[@a_2]$.

Consistency of constraints and DTDs The consistency problem for constraints and DTDs is as follows: given a DTD d and a set Δ of keys, inclusion constraints, and foreign keys, is there a tree T so that $T \models d$ and $T \models \Delta$? Of course by $T \models \Delta$ we mean $T \models \varphi$ for every φ in Δ.

The following is known.

Theorem 1 ([12]).

1. *The consistency problem for DTDs and constraints is undecidable, even if all sets of attributes involved in constraints have cardinality at most* 2.
2. *The consistency problem for DTDs and unary constraints is* NP*-complete.*

In view of this result, in what follows we only consider *unary constraints*.

3 XML with Incomplete Information

We follow the model of incompleteness in XML proposed in [5]. That model extends, in a natural way, the notion of tables [13] to XML documents. We shall not use every single feature of the model of [5], trying to keep the description reasonable, but the features we consider are sufficient for studying the interaction between incompleteness and constraints.

Roughly speaking, incomplete XML trees can occur as a result of missing some of the following information:

(a) attribute values (they can be replaced with variables)
(b) node labels (they can be replaced by wildcards _);
(c) precise vertical relationship between nodes (we can use descendant edges in addition to child edges);
(d) precise horizontal relationship between nodes (using younger-sibling edges instead of next-sibling).

All these types of incompleteness are represented by means of tree/forest descriptions. An ℓ-node with m attributes will be described as $\beta = \ell[@a_1 = z_1, \ldots, @a_m = z_m]$, where

- $\ell \in \Sigma \cup \{\_\}$ (label or wildcard);
- $@a_1, \ldots, @a_m$ are attribute names, and each z_i is a variable, or a constant from $\mathcal{D}$.

Incomplete documents are given by means of incomplete tree descriptions (t) and incomplete forest descriptions (f):

$$\begin{array}{rl} t := & \beta\langle f\rangle\langle\!\langle f'\rangle\!\rangle \\ f, f' := & \varepsilon \;\mid\; t_1\theta_1 t_2\theta_2 \ldots \theta_k t_{k+1} \;\mid\; f, f' \end{array} \tag{1}$$

where each t_i is a tree, and each θ_i is either $\rightarrow$ or $\rightarrow^*$. Informally, a tree $\beta\langle f\rangle\langle\!\langle f'\rangle\!\rangle$ has the node denoted by β as the root, a forest f of children, and a forest f' of descendants. A forest is either empty, or a sequence of trees with specified $\rightarrow$ and $\rightarrow^*$ relationships between their roots, or a union of forests.

For example, the tree (3) from the example in the introduction is given as follows:

$$r\langle\beta_{a(x)} \rightarrow \beta_{a(1)}, \beta_{b(x)}\rangle,$$

where $\beta_{a(x)} = a[@a = x]$, $\beta_{a(1)} = a[@a = 1]$, and $\beta_{b(x)} = b[@b = x]$, assuming that the attributes of a- and b-nodes are called $@a$ and $@b$, respectively.

There are two ways to give the semantics: by satisfaction of incomplete descriptions in trees, and by homomorphisms between relational representations. We use the former here; both are used, and shown to be equivalent, in [5].

Let $\bar{z}$ be the set of all variables (nulls) used in t. Given a valuation $\nu : \bar{z} \rightarrow \mathcal{D}$, and a node s of T, we use the semantic notion $(T, \nu, s) \models t$: intuitively, it means that a complete tree T matches t at node s, if nulls are interpreted according to ν. Then we define

$$Rep(t) = \{T \;\mid\; (T, \nu, s) \models t \text{ for some node } s \text{ and valuation } \nu\}.$$

We now define $(T, \nu, s) \models t$, as well as $(T, \nu, S) \models f$ (which means that T matches f at a set S of roots of subtrees in T). We assume that ν is the identity when applied to data values from $\mathcal{D}$.

- $(T, \nu, s) \models \ell[@a_1 = z_1, \ldots, @a_m = z_m]$ iff s is labeled ℓ (if $\ell \neq \_$) and the value of each attribute $@a_i$ of s is $\nu(z_i)$ (i.e., $(s, \nu(z_i)) \in A_{@a_i}$).
- $(T, \nu, s) \models \beta\langle f\rangle\langle\!\langle f'\rangle\!\rangle$ iff $(T, \nu, s) \models \beta$ and there is a set S of children of s such that $(T, \nu, S) \models f$ and a set S' of descendants of s such that $(T, \nu, S') \models f'$.
- $(T, \nu, \emptyset) \models \varepsilon$;
- $(T, \nu, S) \models t_1\theta_1 t_2\theta_2 \ldots \theta_k t_{k+1}$ iff there exists a sequence $s_1, \ldots, s_{k+1}$ of elements from S, in which every element from S appears at least once, and such that $(T, \nu, s_i) \models t_i$ for each $i \leq k+1$, and (s_i, s_{i+1}) is in NS whenever θ_i is $\rightarrow$, and in NS^* whenever θ_i is $\rightarrow^*$, for each $i \leq k$.
- $(T, \nu, S) \models f_1, f_2$ iff $S = S_1 \cup S_2$ such that $(T, \nu, S_i) \models f_i$, for $i = 1, 2$.

The minimum we need to describe a tree structure is the child edges and the union of forests, hence we assume that those are always present. In other words, the minimal grammar we consider is $t{:=}\beta\langle f\rangle,\ f{:=}\varepsilon \mid t \mid f, f$, with β using only labels from Σ. We refer to incomplete descriptions given by this grammar as *basic incomplete trees.*

Additional features are:

- next sibling $\rightarrow$;
- younger sibling $\rightarrow^*$;
- descendant $\langle\!\langle f\rangle\!\rangle$ (which is also represented by $\downarrow^*$); and
- wildcard $\_$ in place of labels.

Depending on which of these 4 features are used, we have 16 classes of trees. For example, $(\rightarrow, \_)$-trees refers to the situation when we allow wildcard and only $\rightarrow$ as θ_i's, and $(\downarrow^*, \rightarrow, \rightarrow^*, \_)$-trees refer to the full grammar (1).

4 Consistency problem

As already described in the introduction, we consider the consistency problem for XML incomplete descriptions in the presence of integrity constraints (keys and inclusion dependencies). More formally, let Δ be a set of unary XML integrity constraints. We consider the following problem:

Problem: Consistency(Δ)
Input: an incomplete description t
Question: is there a tree $T \in \mathit{Rep}(t)$ so that $T \models \Delta$?

We also look at the version with DTDs d, namely:

Problem: Consistency(Δ, d)
Input: an incomplete description t
Question: is there a tree $T \in \mathit{Rep}(t)$ so that $T \models \Delta$ and $T \models d$?

We classify the complexity of the problem depending on the structure of incomplete trees, ranging from basic incomplete trees (that do not use any of the $\downarrow^*, \rightarrow, \rightarrow^*, \_$ features) to incomplete trees described by the full grammar (1). Since for each of the version of Consistency we have 4 parameters that can be set, we have a total of 32 cases to consider: 16 without DTDs, and 16 with DTDs.

For a class of incomplete trees we say that the consistency problem (or the consistency problem with DTDs) is

- in PTIME, if it can be solved in PTIME given an input tree from the class;
- NP-complete, if it can be solved in NP given an input tree from the class, and, for some fixed set of unary constraints Δ (and a DTD d for the case of consistency with DTDs), then problem Consistency(Δ) (respectively, Consistency(Δ, d)) is NP-complete.

Our main result is as follows.

Theorem 2 (Dichotomy).

1. *The consistency problem (without DTDs) is in* PTIME *for basic incomplete trees, $\rightarrow$-incomplete trees and $\rightarrow^*$-incomplete trees; for the remaining 13 classes of incomplete trees it is* NP*-complete.*
2. *The consistency problem with DTDs is* NP*-complete for every class, from basic incomplete trees to $(\downarrow^*, \rightarrow, \rightarrow^*, \_)$-incomplete trees.*

The proof of the theorem is organized as follows. In section 4.1 we explore the general NP upper bound for the consistency problem, and explore the tractable fragment of $(\rightarrow)$-incomplete trees. Afterwards, in section 4.2, we provide tight lower bounds to show that Consistency becomes intractable under the addition of DTDs, wildcards or transitive closures. It should be noticed that all the lower bounds presented in this paper hold if one considers schema definitions that are more expressive than DTDs, such as XSD or Relax NG [14]. Whether the upper bounds still hold is an open question, that will be part of our future work.

4.1 Upper bounds

In [5], the authors show a general NP upper bound for the Consistency(d) problem (that is, considering only a fixed DTD and no integrity constraints). Interestingly, we show that the presence of unary integrity constraints in our problem does not alter the complexity of the consistency problem.

Theorem 3. Consistency(Δ, d) *is in* NP*, for each fixed DTD d and set Δ of unary XML integrity constraints.*

Proof sketch: One can prove this by combining two previously known results. The first is the one already mentioned that, for each fixed DTD d, the problem Consistency(d) is in NP [5]. The proof in [5] uses the following fact: If t is an incomplete description and the set $Rep_d(t) = Rep(t) \cap \{T \mid T \models d\}$ is nonempty (i.e. Consistency(d) is true for t), then $Rep_d(t)$ contains a tree of polynomial size.

The second result that we use is the following:

Theorem 4 ([12]). *There is a polynomial time algorithm that given a DTD d and a set of unary XML integrity constraints Δ, constructs an integer matrix A and an integer vector $\boldsymbol{b}$ such that there exists an XML tree T that conforms to d and satisfies Δ if and only if $A\boldsymbol{x} = \boldsymbol{b}$ has an integer solution.*

Intuitively, the solution for $A\boldsymbol{x} = \boldsymbol{b}$ represents the number of nodes in the tree that satisfies d and Δ that are labeled with each label ℓ in the alphabet. Moreover, it was shown in [12] that the solution of the system $A\boldsymbol{x} = \boldsymbol{b}$ provides an algorithm for constructing a tree that conforms to d and satisfies Δ.

One can prove Theorem 3 by combining the two results mentioned above. Intuitively, an NP algorithm for solving Consistency(Δ, d) should do the following on input t:

1. Construct from Δ and d a set of equations $A\boldsymbol{x} = \boldsymbol{b}$ as shown in Theorem 4.
2. Guess a polynomial size tree T that belongs to $Rep_d(t)$.
3. Construct in polynomial time a set of linear equations Γ_T that represent the shape of T, and augment the set of equations $A\boldsymbol{x} = \boldsymbol{b}$ with Γ_T. Let E be the augmented set of equations.
4. Check whether there is an integer solution for E.

Clearly, the whole process can be done in nondeterministic polynomial time. Intuitively, the solutions of the sets of equations E will represent, for every $\ell \in \Sigma$, the number of ℓ labeled nodes that must be introduced when extending T into a tree T' that conforms to d and satisfies Δ. As usual, the technical details behind this proof are far more complicated that the intuition provided above. We comment more about those technical details in the appendix. □

Tractable cases. The only tractable cases are obtained when we do not allow DTDs as inputs nor wildcards in the incomplete descriptions, and by severely limiting the features that may allow two formulas in an incomplete description to be witnessed by a same node in a repair: this is the case for $(\rightarrow)$-incomplete trees and $(\rightarrow^*)$-incomplete trees. But before proving this result, we make a crucial observation about the interaction of inclusion constraints in the consistency problem when no DTD is considered. The following proposition shows that the inclusion constraints can be ignored when checking CONSISTENCY w.r.t. incomplete trees without DTD's:

Proposition 1. *For every incomplete description t, and every set Δ of unary XML constraints, let Δ^K be the set of key constraints of Δ (notice that a foreign key is defined as a union of a key and an integrity constraint). Then* CONSISTENCY*(Δ) is true for t if and only if* CONSISTENCY*(Δ^K) is true for t.*

Proof. The direction from left to right is obvious (the same tree will suffice). For the other direction, let t be an incomplete description and Δ be a set of unary constraints, such that CONSISTENCY(Δ^K) is true for t, where Δ^K is as previously defined. Select a tree $T \in Rep(t) \cap \{T \mid T \models \Delta^K\}$, and consider a tree T' built as follows: for every inclusion constraint φ of the form $\ell_1[@a_1] \subseteq \ell_2[@a_2]$ in Δ, and for every ℓ_1-node s of T such that there is no ℓ_2-node s' in T with the value of its $@a_2$-attribute equal to the value of the $@a_1$-attribute of s, add to T' as a child of the root a node s' that satisfies this property. It is easy to see that $T' \models \Delta^K$. Further, by the construction of T', it is also the case that $T' \models \Delta$. Since $T' \in Rep(t)$, this finishes the proof of our claim. □

We now prove the tractable upper bound

Proposition 2. *There exists a* PTIME *algorithm for solving* CONSISTENCY*(Δ) for $(\rightarrow)$-incomplete trees and $(\rightarrow^*)$-incomplete trees.*

Proof. Notice that, just as with complete XML documents, it is possible to define a *relational* representation of an incomplete description t. Roughly speaking, given an incomplete description t over an alphabet Σ of labels and A of attributes, the relational representation of t, denoted as $\underline{re\ell}(t)$, is a structure over the vocabulary $\tau_{\Sigma,A}$ that is defined as expected, in a way that resembles the shredding of a tree under the well known edge relational representation (see [5] for a precise definition).

Thus, given a $(\rightarrow)$-incomplete description t and a set of keys Δ (due to Proposition 1 we do not consider inclusion dependencies), it is possible to define a *chase* procedure over $\underline{re\ell}(t)$ so that t is consistent under Δ if and only if there exists an accepting *chase* sequence on $\underline{re\ell}(t)$.

The chase sequence will intuitively *collapse* all formulas in t that are forced by Δ to represent only one node in every tree $T \in \mathit{Rep}(t)$. Thus, for example, if t contains two formulas $\alpha_1 = \ell[@a = a, @b = b]$ and $\alpha_2 = \ell[@a = a, @b = x]$ and Δ contains the key $\ell.@a \rightarrow \ell$, the chase will intuitively *collapse* α_1 and α_2 so that they now represent only one node, since every tree T that satisfies Δ cannot have two ℓ-nodes with the same value for their $@a$-attribute.

We omit the formal definition of this procedure and the proof of its correctness and soundness, since they can easily be obtained from the chase procedure for incomplete trees defined in [5].

The proof for the case of $(\rightarrow^*)$-incomplete descriptions is analogous, albeit in this case the procedure must take into account the fact that formulas of the form $f_1 \rightarrow^* f_2$ may be collapsed as well by the chase procedure. □

4.2 Lower Bounds

As the following theorem [5] shows, the consistency problem is already difficult when considering only DTDs (no integrity constraints).

Proposition 3 ([5]). *There exists a DTD d such that* CONSISTENCY(d) *is* NP-*complete for basic incomplete trees.*

Thus, under the presence of DTDs one cannot obtain tractability even for the most basic incomplete descriptions. On the other hand, it is easy to see that the consistency problem is tractable if we do not consider neither DTDs nor schema constraints. In fact, it can be proved that every $(\downarrow^*, \rightarrow, \rightarrow^*, \_)$-incomplete tree is consistent [5]. However, as we shall study, adding integrity constraints to the consistency problem easily yields to intractability. To begin with, the presence of wildcards is surprisingly problematic.

Proposition 4. *There exists a set of unary keys Δ such that* CONSISTENCY*(Δ) is* NP-*hard for* $(\_)$*-incomplete trees.*

The proof of this result can be found in the appendix. Notably, the incomplete trees constructed in that proof do not make use of the union operator for forests. It follows then that there exists a set of unary keys Δ such that CONSISTENCY(Δ) is NP-hard for $(\_)$-incomplete trees in which the union operator is not used.

Continuing with the lower bounds, the transitive closure operators prove also to be a problematic feature, even in the absence of the union operator for forests.

Proposition 5. *There exist sets of unary keys Δ_1 and Δ_2 such that the problems*

- Consistency*(Δ_1) for ($\downarrow^*$)-incomplete trees, and*
- Consistency*(Δ_2) for ($\rightarrow, \rightarrow^*$)-incomplete trees*

are NP*-hard, even if incomplete trees are not allowed to use the union operator.*

Proof sketch: We only have to prove hardness. Further, we only show the reduction for $(\rightarrow, \rightarrow^*)$-incomplete trees; the reduction for $(\downarrow^*)$-incomplete trees is similar. We use a reduction from the shortest common superstring problem. Given a set $S = \{s_1, \ldots, s_n\}$ of strings over a fixed alphabet Σ and a positive integer K, the shortest common superstring problem is the problem of deciding whether there exists a string $w \in \Sigma^*$, with $|w| \leq K$, such that each string $s \in S$ is a substring of w, i.e. $w = w_0 s w_1$ for some $w_0, w_1 \in \Sigma^*$.

First, define Σ' to be the alphabet $\{st, mid, end, r\}$, and let A be the set of attributes $\{@id, @e\}$. We fix Δ to be the following set of unary keys: $\{st.@id \rightarrow st, end.@id \rightarrow end\}$.

From S we construct a $(\rightarrow, \rightarrow^*)$-incomplete tree t as follows. The incomplete description t is defined as $r\langle t_K \rightarrow^* t_{s_1} \rightarrow^* \ldots \rightarrow^* t_{s_n}\rangle$. Here, t_K refers to the tree $st\langle st[@id = 1] \rightarrow mid \rightarrow mid \ldots \rightarrow mid \rightarrow end[@id = 1]\rangle$, that contains exactly K nodes labeled mid (we assume that 1 is a data value different from each letter in Σ). Further, for each string $s_i = a_1 \cdots a_m$ of S, the tree t_{s_i} is defined as

$$t_{s_i} := st\langle st[@id = 1] \rightarrow^* mid[@e = a_1] \rightarrow \cdots \rightarrow mid[@e = a_m] \rightarrow^* end[@id = 1]\rangle.$$

Intuitively, in order to restore the consistency of t with respect to Δ, one must collapse each tree of the form t_{s_i} into t_K. Since for each node s of the tree there is at most one data value c such that (s, c) belongs to $A_{@e}$, the fact that this collapse is possible implies that there exists a common superstring of the elements of S of length K. It is not hard to prove then that $Rep(t) \cap \{T \mid T \models \Delta\} \neq \emptyset$ if and only if there is a superstring of S of length at most K. Details can be found in the appendix. □

5 Future work

Our next goal is to understand the complexity of query answering in the setting of incomplete information and integrity constraints. It was shown in [5] that certain answers can be computed by naïve evaluation for trees whose structure is fully specified, but whose attributes can be assigned null values. This result can be extended to trees without any information on the sibling order, as long as queries do not mention it. Outside of these classes, computing certain answers is intractable. Hence, we plan to understand how the complexity of query evaluation is affected for these classes of incomplete trees if keys and/or foreign keys are used.

Acknowledgments The first author is supported FONDECYT grant 11080011, the second and the third author by EPSRC grant G049165 and FET-Open project FoX (Foundation of XML).

References

1. S. Abiteboul, P. Kanellakis, G. Grahne. On the representation and querying of sets of possible worlds. *TCS* 78 (1991), 158–187.
2. S. Abiteboul, L. Segoufin, V. Vianu. Representing and querying XML with incomplete information. In *PODS'01*, pages 150–161.
3. M. Arenas, W. Fan, L. Libkin. On the complexity of verifying consistency of XML specifications. *SIAM J. Comput.* 38 (2008), 841–880.
4. P. Atzeni, N. Morfuni. Functional dependencies and constraints on null values in database relations. *Information and Control* 70(1): 1–31 (1986).
5. P. Barceló, L. Libkin, A. Poggi, C. Sirangelo. XML With incomplete information: Models, Properties and Query Answering. In *PODS09* 237-246 (2009).
6. M. Benedikt, W. Fan, F. Geerts. XPath satisfiability in the presence of DTDs. *J. ACM* 55(2): (2008).
7. H. Björklund, W. Martens, T. Schwentick. Conjunctive query containment over trees. *DBPL'07*, pages 66–80.
8. H. Björklund, W. Martens, T. Schwentick. Optimizing conjunctive queries over trees using schema information. *MFCS'08*, pages 132–143.
9. A. Calì, D. Lembo, R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. *PODS'03*, pages 260-271.
10. D. Calvanese, G. De Giacomo, M. Lenzerini. Semi-structured data with constraints and incomplete information. In *Description Logics*, 1998.
11. D. Calvanese, G. De Giacomo, M. Lenzerini. Representing and reasoning on XML documents: a description logic approach. *J. Log. Comput.* 9 (1999), 295–318.
12. W. Fan and L. Libkin. On XML integrity constraints in the presence of DTDs. *J. ACM* 49 (2002), 368–406.
13. T. Imielinski, W. Lipski. Incomplete information in relational databases. *J. ACM* 31 (1984), 761–791.
14. G. Jan Bex, W. Martens, F. Neven, T. Schwentick. Expressiveness of XSDs: from Practice to Theory, There and Back Again. *WWW 2005*, pages 712-721 (2005).
15. G. Jan Bex, F. Neven, J. Van den Bussche. DTD versus XML Schema: A Practical Study. *WEBDB04*, pages 79–84 (2004).
16. Y. Kanza, W. Nutt, Y. Sagiv. Querying incomplete information in semistructured data. *JCSS* 64 (2002), 655–693.
17. A. Laender, M. Moro, C. Nascimento, P. Martins. An X-Ray on Web-Available XML Schemas. *SIGMOD Record* 38(1) (2009), 37-42.
18. M. Levene, G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science* 206 (1998), 283–300.
19. C.A. Tovey. A simplified satisfiability problem. Discrete Appl. Math. 8 (1984), pp. 8589.

A Proofs and intermediate results

A.1 Proof Sketch of Theorem 3

One can prove this by combining two previously known results. The first one is that, for each fixed DTD d, the problem CONSISTENCY(d) is in NP [5]. The proof of this fact goes as follows. Let d be a fixed DTD d. It is shown in [5] that if t is an incomplete description and the set $Rep_d(t) = Rep(t) \cap \{T \mid T \models d\}$ is nonempty (i.e. CONSISTENCY(d) is true for t), then $Rep_d(t)$ contains a tree of polynomial size.

The second result that we use is the following:

Theorem 5 ([12]). *There is a polynomial time algorithm that given a DTD d and a set of unary XML integrity constraints Δ, constructs an integer matrix A and an integer vector $\boldsymbol{b}$ such that there exists an XML tree T that conforms to d and satisfies Δ if and only if $A\boldsymbol{x} = \boldsymbol{b}$ has an integer solution.*

Intuitively, the solution for $A\boldsymbol{x} = \boldsymbol{b}$ represents the number of nodes of the conforming tree labeled with each label ℓ from an alphabet Σ of labels. Moreover, it was shown in [12] that the solution of the system $A\boldsymbol{x} = \boldsymbol{b}$ provides an algorithm for constructing the conforming tree.

One can prove Theorem 3 by combining the two results mentioned above. Intuitively, an NP algorithm for solving CONSISTENCY(Δ, d) should do the following on input t:

1. Construct from Δ and d a set of equations $A\boldsymbol{x} = \boldsymbol{b}$ as shown in Theorem 5.
2. Guess a polynomial size tree T that belongs to $Rep_d(t)$.
3. Construct in polynomial time a set of linear equations Γ_T that represent the shape of T, and augment the set of equations $A\boldsymbol{x} = \boldsymbol{b}$ with Γ_T. Let E be the augmented set of equations.
4. Check whether there is an integer solution for E.

Clearly, the whole process can be done in nondeterministic polynomial time.

Intuitively, the solutions of the sets of equations E will represent, for every $\ell \in \Sigma$, the number of ℓ labeled nodes that must be introduced when extending T into a tree T' that conforms to d and satisfies Δ.

As usual, the technical details behind this proof are far more complicated that the intuition provided above. To begin with, there is no guarantee that T may be extended into a tree T' that satisfies Δ. Thus, we do not guess a tree that conforms to d, but guess instead a tree $\hat{T}$ that, although not guaranteed to conform to d, can be extended into a tree T' that conforms to d and satisfy Δ. Moreover, we also guess which parts of $\hat{T}$ may be extended. Let us denote by $E_{\hat{T}}$ the edges of $\hat{T}$ that can be extended. Then, for each one of this edges $e \in E_{\hat{T}}$, we construct a set of linear equations Γ_e accordingly, and augment $A\boldsymbol{x} = \boldsymbol{b}$ with the set $\{\Gamma_e \mid e \in E_{\hat{T}}\}$. Intuitively, the solution to each of the equations Γ_e will represent, for every $\ell \in \Sigma$, the number of ℓ labelled nodes that must be introduced when extending $\hat{T}$ in the edge e. Moreover, since these solutions also satisfy $A\boldsymbol{x} = \boldsymbol{b}$, we are assured that there will be an extension T' of $\hat{T}$ such that T' satisfies Δ and is valid w.r.t d.

Thus, given an incomplete tree t, it can be proved that there exists an integer solution for the set of equations $\{\Gamma_e \mid e \in E_{\hat{T}}\}$ and $A\boldsymbol{x} = \boldsymbol{b}$ built from t if and only if CONSISTENCY(Δ, d) is true for t.

A.2 Proof of Proposition 4

We use a reduction from 3-SAT$_4$, a restriction of the well known 3-SAT boolean satisfiability problem in which every literal in the formula appears at most 4 times [19]. Fix an alphabet $\Sigma = \{\mathtt{true}, \mathtt{false}, N\}$ of labels, and $A = \{@tv, @a1, @a2, @a3, @p1, @p2, @p3, @p4\}$ of attributes, and let Δ be the set of keys $\{\ell.@a \to \ell \mid \ell \in \Sigma, @a \in A\}$.

Let φ be a 3-SAT$_4$ formula using variables $x_1, \ldots, x_n$, and let $c_1, \ldots, c_k$ be an enumeration of the clauses in φ. Based on φ, we give an incomplete description t over Σ and A.

Our incomplete description only uses the ↓ operator (that is, no union of forest will be ussed). Thus, essentially, the description will be no more than a vertical line of formulas connected by $\langle\rangle$ (that is, a sequence of formulas $\alpha_1\langle \cdots \langle \alpha_m\rangle\rangle$, and where some of them may use the wildcard operator. Instead of defining all formulas of t independently, we will define the forests $f_{x_1}, \ldots, f_{x_n}$ and $f_{c_1}, \ldots, f_{c_k}$, and let

$$t = f_{x_1}\langle \cdots \langle f_{x_n}\langle f_{c_1}\langle \cdots \langle f_{c_k}\rangle\rangle\rangle\rangle\rangle$$

.

To define the f_{x_i}, let $1, \ldots, n$ be a set of attribute values. For every variable x_i in φ, let f_{x_i} be defined as $\beta_{x_i}\langle\ \beta_{\hat{x}_i}\langle\ \beta_{n_i}$, where:

$$\beta_{x_i} = \_[@tv = i, @p1 = \mathit{clause}(1, x_i), @p2 = \mathit{clause}(2, x_i),\\ @p3 = \mathit{clause}(3, x_i), @p4 = \mathit{clause}(4, x_i)],$$

$$\beta_{\hat{x}_i} = \_[@tv = i, @p1 = \mathit{clause}(1, \hat{x}_i), @p2 = \mathit{clause}(2, \hat{x}_i),\\ @p3 = \mathit{clause}(3, \hat{x}_i), @p4 = \mathit{clause}(4, \hat{x}_i)],$$

$$\beta_{n_i} = N[@tv = i]$$

Notice that, since all three formulas have its $@tv$ attribute valued i, and Δ contains the dependencies $\mathtt{true}.@tv \to \mathtt{true}$, $\mathtt{false}.@tv \to \mathtt{false}$ and $N.@tv \to N$, it must be that every tree T that belongs to $\mathit{Rep}(f_{x_i})$ must assign the label `true` to the node witnessing β_{x_i} and `false` to the node witnessing $\beta_{\hat{x}_i}$, or the labels `false` and `true`, respectively (any other label assignment would result in a violation of the dependencies in Δ). Thus, intuitively, with this assignment of labels we represent the truth value that is given to variable x_i.

The function $\mathit{clause}(x, y)$ is defined as follows. Fix $3k$ elements $c_{11}, c_{12}, c_{13}, \ldots, c_{k1}, c_{k2}, c_{k3}$. Then, for every $i \in [1, n]$ and $m \in [1, 4]$, define $\mathit{clause}(m, x_i) = c_{pq}$ if the m-th occurrence of variable x_i in φ is as the q-th variable of the p-th clause.

Next, we define a tree f_{c_j} for each $j \in [1, k]$. To that extent, fix k new elements $b_1, \ldots, b_k$ Then, f_{c_j} is defined as a parent child sequence of formulas $\alpha_{j1}\langle\ \cdots \langle \alpha_{j6}$, where

$\alpha_{j1} = \_[@a1 = b_j, @p1 = c_{j1}@p2 = c_{j1}@p3 = c_{j1}@p4 = c_{j1}]$
$\alpha_{j2} = \_[@a1 = b_j, @p1 = c_{j2}@p2 = c_{j2}@p3 = c_{j2}@p4 = c_{j2}]$
$\alpha_{j3} = \_[@a1 = b_j, @a2 = b_j]$
$\alpha_{j4} = \_[@a2 = b_j, @a3 = b_j]$
$\alpha_{j5} = \_[@a3 = b_j, @p1 = c_{j3}@p2 = c_{j3}@p3 = c_{j3}@p4 = c_{j3}]$
$\alpha_{j6} = \mathtt{true}[@a3 = b_j]$.

We claim that $Rep(t) \cap \{T \mid T \models \Delta\} \neq \emptyset$ if and only if φ is satisfiable.

In order to prove our claim, it is important to notice that the structure of t is a chain of child edges. Thus, no satisfying valuation ν that verifies that a tree T belongs to $Rep(t)$ is such that ν assigns two distinct formulas of t to the same node in T.

We now continue with the proof. Assume first that t is consistent, and let $T \in Rep(t)$, $T \models \Delta$. The proof that φ is satisfiable follows from the following observation:

Claim. If ν is a valuation such that $(T, \nu, p) \models t$ for a node p of T, then for every clause $(a \vee b \vee c)$ of φ, with a, b, c representing literals or negated literals, and nodes s_a, s_b and s_c such that $(T, \nu, s_\ell) \models \beta_\ell$ for each $\ell \in [a, b, c]$, it must be the case that at least one of s_a, s_b or s_c is labelled `true`.

To prove this statement, assume that for the j-th clause of φ, of form $(a \vee b \vee c)$, the labels of the nodes s_a, s_b and s_c in T is `false` (we know it cannot be N). From the construction of t, for each β_a, β_b and β_c there attributes $@p_a, @p_b, @p_c$ in $\{@p1, @p2, @p3, @p4\}$ such that the $@p_a$,$@p_b$ and $@p_c$ attributes of s_a, s_b and s_c is valued c_{j1}, c_{j2} and c_{j3}, respectively. Let now $g_1, \dots, g_6$ be the nodes in T such that that $(T, \nu, g_m) \models \alpha_{jm}$, for each $m \in [1, 6]$. It then follows that the nodes g_1, g_2 and g_5 cannot be labeled `false`: for example, if g_1 is labelled `false`, then both s_a and g_1 are labelled `false` and have their $@p_a$-attribute valued c_{j1}. This in turn entails that $T \not\models \Delta$, since we know that s_a and g_1 cannot be the same node in T.

It is also easy to see that g_5 must be labeled N. Moreover, the labels of g_1, g_2, g_3 must be all different from each other (they all have the same value for their $a1$ attribute, and this attribute is key for every label in Σ). Assume without loss of generality that g_1 is labeled `true` and g_2 is labeled N. Then, g_3 is labeled `false`. By repeatedly using this argument, it is straightforward to derive a contradiction.

The proof follows from easily from the previous claim, as the labeling of the nodes witnessing each α_{x_i} indicate a satisfying valuation for the variables in φ.

Next, assume that φ is satisfiable via a valuation ρ. Then, consider the tree T built from t as follows:

- T consists of 1 node s_α or s_β for each node formula α or β in t. The labeling of the node and the value of it's attributes is the same as the formula. (We permit, for the moment, trees labelled with wildcards)
- Organize the nodes of T in a parent-child sequence according to the ordering of the formulas in t.
- Each node $s_{\beta_{x_i}}$ and $s_{\beta_{\bar{x}_i}}$ of t are given the label `true` and `false` if ρ assigns a 1 to the variable x_i, and `false` and `true` otherwise.
- Each of the nodes $s_{\alpha_{j1}}, \dots, s_{\alpha_{j5}}$ is given a label by correctly propagating the values of the literals of the j-th clause of φ. For example, if the j-th clause is of form $(a \vee b \vee c)$, for a, b, c literals or negated literals, and the nodes s_{β_a}, s_{β_b} and s_{β_c} are given the label `true`, then the labels `false`, N and `false` are given to nodes $s_{\alpha_{j1}}, s_{\alpha_{j2}}, s_{\alpha_{j5}}$ respectively, the label `true` is given to $s_{\alpha_{j3}}$, and the label N is given to $s_{\alpha_{j4}}$.

It is then straightforward to prove that $T \in Rep(t)$ and T is consistent w.r.t. Δ

A.3 Proof of Proposition 5

CONSISTENCY*(Δ) for* $(\rightarrow, \rightarrow^*)$*-incomplete trees:* We use a reduction from the shortest common superstring problem. Given a set $S = \{s_1, \dots, s_n\}$ of strings over a fixed alphabet Σ and a positive integer K, the shortest common superstring problem is the problem of deciding whether there exists a string $w \in \Sigma^*$, with $|w| \leq K$, such that each string $s \in S$ is a substring of w, i.e. $w = w_0 s w_1$ for some $w_0, w_1 \in \Sigma^*$.

First, define Σ' to be the alphabet $\{st, mid, end, r\}$, and let A be the set of attributes $\{@id, @e\}$. We fix Δ to be the following set of unary keys: $\{st.@id \to st, end.@id \to end\}$.

From S we construct a $(\to, \to^*)$-incomplete tree t as follows. The incomplete description t is defined as $r\langle t_K \to^* t_{s_1} \to^* \ldots \to^* t_{s_n}\rangle$. Here, t_K refers to the tree $st\langle st[@id = 1] \to mid \to mid \ldots \to mid \to end[@id = 1]\rangle$, that contains exactly K nodes labeled mid (we assume that 1 is a data value different from each letter in Σ). Further, for each string $s_i = a_1 \cdots a_m$ of S, the tree t_{s_i} is defined as

$$t_{s_i} := st\langle st[@id = 1] \to^* mid[@e = a_1] \to \cdots \to mid[@e = a_m] \to^* end[@id = 1]\rangle.$$

Intuitively, in order to restore the consistency of t with respect to Δ, one must collapse each tree of the form t_{s_i} into t_K. Since for each node s of the tree there is at most one data value c such that (s, c) belongs to $A_{@e}$, the fact that this collapse is possible implies that there exists a common superstring of the elements of S of length K. We prove next that $Rep(t) \cap \{T \mid T \models \Delta\} \neq \emptyset$ if and only if there is a superstring of S of length at most K.

Assume first that $Rep(t) \cap \{T \mid T \models \Delta\} \neq \emptyset$. Then, there exists a tree T, a node p of T and a valuation ν of the variables of t, such that $(T, \nu, p) \models t$ and $T \models \Delta$. Clearly, node p in T has label r. Let p' be a child of p such that $(T, \nu, p') \models t_K$. This means that there exist children $p_0, p_1, \ldots, p_K, p_{K+1}$ of p' in T, such that p_0 is labeled st and the value of its $@id$-attribute is 1, p_{K+1} is labeled end and the value of its $@id$-attribute is 1, each p_j $(1 \leq j \leq K)$ is labeled mid, and $(p_j, p_{j+1}) \in NS$, for each $0 \leq j \leq K$. Further, since $T \models \Delta$ and Δ contains the key $st.@id \to st$, it must be the case that $(T, \nu, p') \models t_{s_i}$, for each $1 \leq i \leq n$ (otherwise, T would have two nodes labeled st with the same value of its $@id$-attribute).

Assume that for $1 \leq i \leq n$, $s_i = a_1 \cdots a_m \in \Sigma^*$. Since $(T, \nu, p') \models T_{s_i}$ and Δ contains the key $end.@id \to @id$, it must be the case that there exists a sequence $p_{j+1}, p_{j+2}, \ldots, p_{j+m}$, $0 \leq j \leq K - m$, such that $(T, \mu, p_{j+i}) \models mid[@e = a_i]$, for each $1 \leq i \leq m$. Let a be an arbitrary symbol from Σ and let $\bar{c} = c_1 \cdots c_K$ the string from Σ^* such that, for each $1 \leq i \leq K$, $c_i = b$ if the value of the $@e$-attribute of p_i is $b \in \Sigma$, and $c_i = a$ otherwise. It is easy to see that $\bar{c}$ is a superstring of S of length K.

For the converse, assume without loss of generality that $\bar{b} = b_1 \cdots b_K$ is a supersequence of S of length K. We claim that the tree

$$T = r\langle st\langle st[@id = 1] \to mid[@c = b_1] \to \cdots \to mid[@c = b_K] \to end[@id = 1]\rangle\rangle$$

is in $Rep(t)$. Notice that trivially T satisfies Δ. The proof is rather simple and left to the interested reader.

CONSISTENCY*(Δ) for ($\downarrow^*$)-incomplete trees:* This proof is a modification of the previous reduction. We use the following construction: Let $s = a_1 \cdots a_m$ be a string from Σ^*. We define t_s^* to be the following incomplete description:

$$mid\langle\!\langle mid[@e = a_1]\langle\!\langle \cdots \langle\!\langle mid[@e = a_m]\langle\!\langle end[@id = 1]\rangle\!\rangle\rangle\!\rangle\rangle\!\rangle\rangle\!\rangle.$$

Let Σ', A and Δ be as in the previous proof. From S we now define an incomplete description t of the form $r\langle t_K\rangle\langle\!\langle t_{s_1}\rangle\!\rangle$, where t_K is now defined as

$$st[@id = 1]\langle mid\langle mid \ldots \langle mid\langle end[@id = 1]\rangle\rangle\rangle\rangle,$$

and each t_{s_i} is recursively defined as follows:

$$t_{s_i} = st[@id = 1]\langle t^*_{s_i}\rangle\langle\!\langle t_{s_{i+1}}\rangle\!\rangle,$$

if $1 \leq i \leq n - 1$, and t_{s_n} is defined as $st[@id = 1]\langle t^*_{s_n}\rangle$.

Notice that, once again, the constraints in Δ will "force" every node labeled st in a tree in $Rep(t)$ to collapse to a single node. With this consideration, the proof that $Rep(t) \cap \{T \models \Delta\}$ is nonempty if and only if there is a superstring of S of length at most K can be easily adapted from the previous reduction.