XML with Incomplete Information

Pablo Barceló* Leonid Libkin§ Antonella Poggi† Cristina Sirangelo‡

Abstract

We study models of incomplete information for XML, their computational properties, and query answering. While our approach is motivated by the study of relational incompleteness, incomplete information in XML documents may appear not only as null values but also as missing structural information. Our goal is to provide a classification of incomplete descriptions of XML documents, and separate features - or groups of features - that lead to hard computational problems from those that admit efficient algorithms. Our classification of incomplete information is based on the combination of null values with partial structural descriptions of documents. The key computational problems we consider are consistency of partial descriptions, representability of complete documents by incomplete ones, and query answering. We show how factors such as schema information, the presence of node ids, and missing structural information affect the complexity of these main computational problems, and find robust classes of incomplete XML descriptions that permit tractable query evaluation.

1 Introduction

The transfer and extension of relational tools to deal with XML data has been a central theme in database research over the past decade. One area that has not witnessed much activity is the handling of incomplete information in XML. And yet incomplete information is ubiquitous in XML applications, especially in exchanging and integrating web data – the key applications XML was designed for.

In the research literature, there are some papers that address the problem of incompleteness in XML, but this typically happens in some specific scenarios. For example, the paper [4] concentrated on handling incompleteness arising in a dynamic setting in which the structure of a tree is revealed by a sequence of queries; graph and tree data models expressed as description logic theories that could incorporate incompleteness were dealt with in [13, 14]; incompleteness in query results but not inputs was studied in [27]; and incorporating probabilities into XML was looked at in [36, 16]. In practice incomplete information needs to be modeled as well, most commonly by optional attributes, or tricks such as `minOccurs="0"` to introduce nulls at the level of elements.

Our goal is to provide a systematic study of incomplete information in XML that is independent of any particular application. We would like to address the same problems as the fundamental study of relational incompleteness [3, 26], namely:

1. study models of incompleteness in XML and their semantics; and
2. study the key computational tasks associated with such models (e.g., query answering) with the main goal of separating features that lead to good algorithmic solutions from those that lead to intractability. We would like to find robust classes of models and queries (such as naïve tables and unions of conjunctive queries for relations) for which query evaluation is tractable.

*Department of Computer Science, University of Chile, pbarcelo@dcc.uchile.cl.
§School of Informatics, University of Edinburgh, libkin@inf.ed.ac.uk.
†DIS, Sapienza Università di Roma, poggi@dis.uniroma1.it.
‡LSV, ENS-Cachan, CNRS and INRIA, cristina.sirangelo@lsv.ens-cachan.fr.

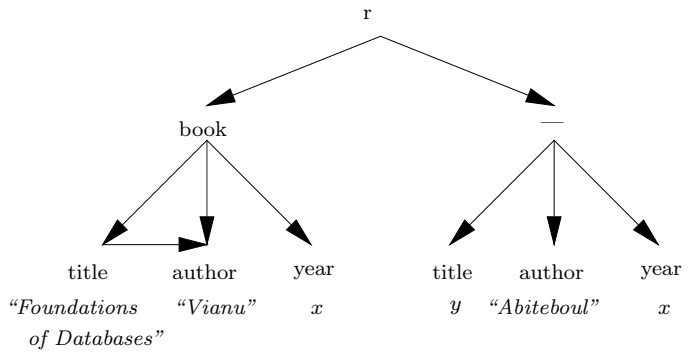


Figure 1: An incomplete XML document

The results we obtain can be used in any application scenario, as they say for which classes of problems and models efficient solutions cannot be found, and for which classes such solutions exist.

The inspiration for such a general study comes from the study of incompleteness in relational databases. There, incompleteness arises when some attribute values are unknown for a variety of reasons and are represented as nulls. The design of SQL adopted a single type of null and the (often criticized [17]) reasoning model based on the 3-valued logic. Theoretical investigations of nulls culminated in two papers that are the foundation of the theory of relational incompleteness. The paper by Imielinski and Lipski [26] introduced the notion of tables as a representation mechanism for incomplete information, and looked at types of tables that are suitable for evaluating queries from various sublanguages of relational algebra. The paper by Abiteboul, Kanellakis, and Grahne [3] studied the complexity of computational problems associated with incompleteness, and provided a clear separation between tractable and intractable cases. These results continue to be very influential. For example, the fact that unions of conjunctive queries can be evaluated in polynomial time over naïve tables (in which nulls can be repeated) is used heavily in data integration and exchange [1, 21, 29], where, in particular, it influences the choice of queries and solution instances in data exchange.

The structure of XML documents is much more complicated than that of relational databases, and missing information may appear not only among attribute values, but also in the *structure* itself. In addition, the way we view XML documents may lead to different representations of incomplete information.

To see how incompleteness can be represented in XML, consider a document that describes books and papers, by giving their titles, authors, and years of publication. An incomplete description of such a document is presented in Figure 1. The left subtree talks about the *Foundations of Databases* book; it tells us that one of the authors is Vianu, but it does not give us precise information about the publication date (year is null, given by a variable x). The second subtree says that there is some publication by Abiteboul (we do not know if it is a book or an article since wildcard is used as a label); all we know about it is that it was published in the same year x. We also know that the author node for Vianu is an immediate successor of the book title, but no other information about sibling ordering is available.

This document can represent many complete trees: one example is a description of *Foundations of Databases*. In that case we assume that the root has just one child (which is consistent with the description, since _ matches every label), with one title node, a year node with the value '1995', and three author nodes for Abiteboul, Hull, and Vianu. We are making the open world assumption and allow addition of nodes; in particular the incomplete document above does not have the knowledge that Hull is one of the authors.

We now turn to a slightly different way of modeling XML, which corresponds to the DOM interface

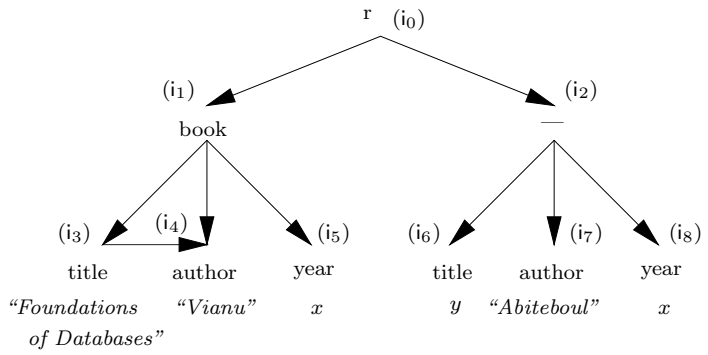


Figure 2: An incomplete XML document under DOM representation

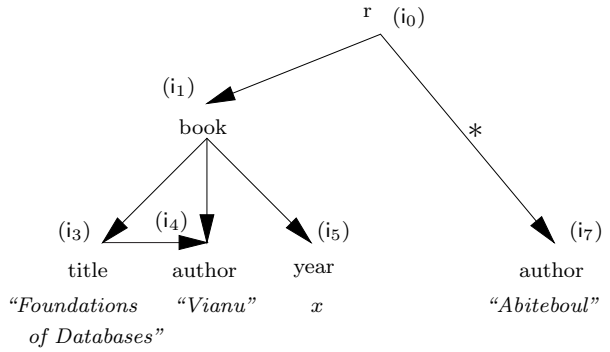


Figure 3: An incomplete XML document with missing structural information

[20]. In that case, we can access each node in a document by its id, and apply various methods that produce its parent, left and right siblings, first child, all children, etc. The key point is that a node is uniquely identified by its id. Consider now Figure 2, that looks like almost the same incomplete document.

The small change – we gave ids to all nodes, shown in parentheses as (i_k) – makes a big impact on the semantics. For example, it is no longer possible that the document represents a single book, as before. Indeed, we know that the two children of the root are different, since $\mathsf{i}_1 \neq \mathsf{i}_2$.

But one can still have an incomplete document description that is consistent with the document representing only information about *Foundations of Databases*, even with unique ids associated with each node. Assume that we lose *structural* information that the author-node i_7 is a grandchild of the root, and instead we only know that it is a descendant of the root, as shown in Figure 3. Then it is still consistent with an incomplete description that i_7 is a child of i_1 and thus describes an author of *Foundations of Databases*.

These examples start giving us an indication of the nature of incomplete information in XML, and how various choices of parameters affect the semantics of incompleteness. In addition to the standard missing information – attribute values – we may have missing structure information such as labels (replaced by wildcards) or information about edges (in the above examples, we miss some next-sibling information or replace a precise path to a node by a single descendant edge). Furthermore, there is a choice of having node ids, which affects the semantics of incompleteness.

Note that incomplete descriptions provided above may arise in several contexts, for instance in a data integration setting. In our running example, suppose that we wish to maintain data about books

and papers, together with their title, authors, and year of publication. Specifically, suppose that we look for such data on the Web and we find two documents, known to provide publications occurred the same year (which is unknown). One document tells that *Foundations of Databases* is a book and one of the authors is Vianu, while the other document tells that Abiteboul is the author of another publication. Depending on the rationale of integrating these documents, we would resort to one or another incomplete description. Thus, depending on whether we want to integrate them by allowing to possibly merge the book by Vianu with the publication by Abiteboul or not, we would respectively represent the integrated document respectively by the first (or the third) incomplete description or the second one. Also, depending on whether we aim at integrating, besides information content, nodes identity, we would opt for an incomplete tree or an incomplete DOM-tree.

In comparison with relational databases, there are many more parameters to consider when we classify incomplete descriptions of XML trees. They include the nature of nulls for attributes, the exact set of axes used in descriptions, the presence of node ids. A full classification of those will give us a large number of cases, and studying all of them is certainly not our goal.

What we want to understand in this paper is the interplay between features, or groups of features, that leads to efficient algorithms (or intractability) for various computational problems associated with incomplete information. We want to find robust and naturally definable classes of incomplete descriptions that lead to efficient algorithmic solutions.

Summary of the main results

We start by reviewing relational incompleteness in Section 2. Then, in Section 3, we describe XML documents in a way that makes it easy to introduce models of incompleteness, by eliminating some of the features of complete documents. After that we do the following.

1. We introduce models of incomplete XML documents (in Section 4). Incompleteness may occur at the data level (missing attribute values), structure level (missing structural information), and node level (missing node ids). We primarily concentrate on two types of models with respect to the node level: in one (called *incomplete trees*), all node ids are (distinct) variables. In the other, called *incomplete DOM-trees* (by analogy with the DOM interface for XML), all ids are present, i.e., every node can be identified by its id.
2. In Section 5 we study the *consistency problem* for incomplete XML documents: that is, given an incomplete (DOM-)tree, and perhaps some schema information, is there a document that conforms to both? The key results are as follows:
   - The consistency problem is always in NP. Without the schema information, it is trivially solvable for incomplete trees if there is no "marking" of nodes (i.e., saying that a node is a first child, or a leaf, etc.). With markings, we give a full dichotomy classification into PTIME and NP-complete cases. The tractable cases work by an adaptation of chase.
   - With the schema information, given by (very simple) DTDs, the consistency problem for incomplete trees is NP-complete.
   - With DOM-trees, the situation is very different: without DTDs, the problem is always in PTIME (although the algorithm is much more involved), and it remains in PTIME even with DTDs, under some mild restrictions.
3. In Section 6, we study the *membership problem*: given an incomplete description of an XML document and a complete XML tree, can the former represent the latter? The problem is in NP for incomplete trees, and could be NP-hard. For DOM-trees, it is in PTIME, as well as for incomplete trees in which nulls for attribute values cannot be repeated (an analog of relational Codd tables).

4. In Section 7 we study query answering, more precisely, the complexity of computing certain answers. To define certain answers properly, we look at queries that output sets of tuples of attribute values. Our goal is to find a class that behaves similarly to unions of conjunctive queries over naïve relational tables. We do the following.

   - We show that query answering is in coNP.
   - Then we identify features of incomplete trees that easily lead to coNP-hardness. We prove a series of results showing that these include: the presence of schema information, the presence of transitive closures of axes (e.g., descendant), and the lack of information about the sibling ordering.
   - Excluding the features that lead to coNP-hardness, we get a class of *rigid* incomplete trees: their structure is fully described by means of child and next-sibling edges, but labels and attribute values may be unknown. For them, we have an analog of the relational naïve evaluation that correctly computes certain answers in polynomial time.

Then, in Section 8, we give an overview of restrictions that lead to tractability of various computational tasks.

Some of the proofs have been put into the appendix, due to space requirements. This paper is the full version of [8].

2 Incompleteness in relational databases

We now briefly recall the basics of incomplete information in relational databases [5, 3, 26]. Incompleteness is represented by means of *tables* in which both values and variables (for nulls) can be used. For example, $T = \{(1, x), (y, 2), (x, 1)\}$ is a table. Such a table can represent complete relations, i.e. relations without nulls, that contain all the tuples in T under some valuation of nulls. Formally, a relation R is represented by T if there is a *valuation* ν (i.e. a mapping from nulls to constants) such that $\nu(T) \subseteq R$. The set of such relations is usually denoted by $Rep(T)$. This definition naturally extends to databases with multiple relations. Note that we are making the open world assumption here; under the closed world assumption, $Rep(T)$ would consist only of relations $\nu(T)$.

There are different types of tables: in *Codd tables*, all variable occurrences are distinct; in *naïve* tables, the same variable can occur more than once (as in the table T above), and in conditional tables one can impose more complex conditions than just equality on variables [26].

The key computational problems related to incompleteness are membership and query answering (there are several others considered, e.g., in [3], but they are variations on these two themes). The membership problem is to check if a complete database is represented by an incomplete one, that is, whether $R \in Rep(T)$. For query answering, typically we deal with *certain answers* [26], defined as

$$certain(Q, T) \;=\; \bigcap \{Q(R) \mid R \in Rep(T)\}.$$

Key results from [26] tell us where the tractability boundary for these problems are. For example, membership is PTIME for Codd tables but NP-complete for naïve tables. Query answering over naïve tables is tractable for unions of conjunctive queries. This is done by the *naïve evaluation*. Under it, nulls are viewed as values, with two nulls being equal if they are syntactically the same, but only null-free tuples are kept in the output. For instance, suppose we have naïve tables $T_1 = \{(1, x), (2, y)\}$ over attributes A, B and $T_2 = \{(x, 2), (y, y)\}$ over attributes B, C. Then naïve evaluation of the query $T_1 \bowtie T_2$ produces the empty set: we perform $T_1 \bowtie T_2$ as if x and y were values, and get tuples $(1, x, 2)$ and $(2, y, y)$, both containing nulls. However, naïve evaluation of $\pi_{A,C}(T_1 \bowtie T_2)$ results in a single

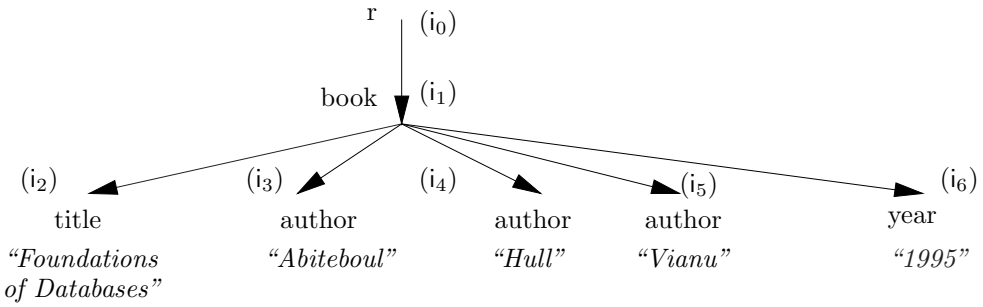
tuple $(1, 2)$: after applying the projection to $T_1 \bowtie T_2$, we get tuples $(1, 2)$ and $(2, y)$, one of which contains no nulls, and thus belongs to certain answers.

For relational algebra, the complexity ranges from coNP-complete under the closed world assumption to undecidable under the open world assumption [3, 38]. This evaluation strategy has found multiple applications in data integration and exchange [29, 21].

3 XML documents

Before introducing models of incompleteness in XML, we define complete XML trees. We describe them in an exhaustive way – including information about child and next-sibling axes, their transitive closures, labels, and attributes - so that later we introduce models of incompleteness by removing features of complete documents.

We first explain this representation by means of an example. Consider the document below. We have not shown the next-sibling edges but we assume the order of the children of the book node to be from left-to-right as shown in the picture.



This XML document will be described as a relational structure over two domains: of node ids $V = \{\mathsf{i}_0, \mathsf{i}_1, \mathsf{i}_2, \mathsf{i}_3, \mathsf{i}_4, \mathsf{i}_5, \mathsf{i}_6\}$, and of values $D = \{$*"Foundations of Databases", "Abiteboul", "Hull", "Vianu", "1995"*$\}$. On domain V, we define the following predicates:

- Edge relation E: $(\mathsf{i}_0, \mathsf{i}_1), (\mathsf{i}_1, \mathsf{i}_2), (\mathsf{i}_1, \mathsf{i}_3)$, etc.
- Descendant[1] relation E^*, which is the transitive-reflexive closure of E (for example, $(\mathsf{i}_0, \mathsf{i}_j) \in E^*$ for $0 \leq j \leq 6$).
- Next-sibling relation NS: $(\mathsf{i}_2, \mathsf{i}_3), (\mathsf{i}_3, \mathsf{i}_4)$, etc.
- Its reflexive-transitive closure NS^* (that includes all $(\mathsf{i}_\ell, \mathsf{i}_k)$ for $2 \leq \ell \leq k \leq 6$).
- Labeling predicates for each label; e.g, the set $P_{author} = \{\mathsf{i}_3, \mathsf{i}_4, \mathsf{i}_5\}$ and the set $P_{book} = \{\mathsf{i}_1\}$.
- Markings for leaves, root, first and last children: $Root = \{\mathsf{i}_0\}, Leaf = \{\mathsf{i}_2, \mathsf{i}_3, \mathsf{i}_4, \mathsf{i}_5, \mathsf{i}_6\}$, $FC = \{\mathsf{i}_1, \mathsf{i}_2\}$, and $LC = \{\mathsf{i}_1, \mathsf{i}_6\}$.
- Assignment of attribute values to nodes. Let us assume that we have attributes $@author, @title$, and $@year$. Then we have relations $A_{@author} = \{(\mathsf{i}_3,$ *"Abiteboul"*$), (\mathsf{i}_4,$ *"Hull"*$), (\mathsf{i}_3,$ *"Vianu"*$)$ showing assignment of values of the $@author$ attribute to nodes, as well as $A_{@title} = \{(\mathsf{i}_2,$ *"Foundations of Databases"*$)\}$ and $A_{@year} = \{(\mathsf{i}_6,$ *"1995"*$)\}$.

We now give a formal definition. Assume the following disjoint countably infinite sets:

- *Labels* of possible names of element types (that is, node labels in trees);

[1]Technically, this is the descendant-or-self relation, as we use the reflexive-transitive closure. However, since we always use this relation, we shall be using the term descendant throughout, omitting 'or-self'.

- *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. In fact we define them to be structures of a very large vocabulary; the reason is that we want complete descriptions to contain all the information about trees, and in incomplete descriptions we shall be restricting the vocabulary.

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

$$\tau_{\Sigma,A} \;=\; \left(\begin{array}{l} E, NS, E^*, NS^*, (A_{@a})_{@a \in A} \\ (P_\ell)_{\ell \in \Sigma}, \mathit{Root}, \mathit{Leaf}, FC, LC \end{array} \right)$$

where all relations in the first line are binary and all relations in the second line are unary. A tree is a 2-sorted structure of vocabulary $\tau_{\Sigma,A}$, i.e. $\langle V, D, \tau_{\Sigma,A} \rangle$, 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 $\langle V, E, NS \rangle$ is an ordered unranked tree; E^* and NS^* are 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 (i, c) is present for each $\mathsf{i} \in V$;
- P_ℓ are labeling predicates: $\mathsf{i} \in V$ belongs to P_ℓ if and only if it is labeled ℓ; as usual, we assume that the P_ℓ's are pairwise disjoint;
- Sets *Root*, *Leaf*, FC, LC contain the root, the leaves, first (oldest) and last (youngest) children of nodes.

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.

We now show how to produce complete descriptions of XML trees by means of a grammar that will guide us when we develop incomplete descriptions of trees. Trees (t) and forests (f) can be given by the following syntax:

$$t := \beta\langle f \rangle \qquad f := \varepsilon \mid tf \tag{1}$$

where β ranges over descriptions of nodes.

A node description β of a node whose label is $\ell \in \mathit{Labels}$, whose id is $\mathsf{i} \in \mathcal{I}$ and whose attributes $@a_1, \ldots, @a_m$ have values $v_1, \ldots, v_m \in \mathcal{D}$ is $\beta \;=\; \ell(\mathsf{i})[@a_1 = v_1, \ldots, @a_m = v_m]$. Each tree $\beta\langle f \rangle$ is given by a description of its root node β and the forest f of its children, and each forest f is either empty or a sequence of trees. Trees are *ordered*: for the tree $\beta\langle t_1 \ldots t_k \rangle$ we assume that the tree t_1 is rooted at the first child of the node given by β, the tree t_2 at the second child, and so on.

4 Models of incompleteness in XML

We start with complete tree descriptions (1) and see how missing information can be incorporated into them. As the result, we get descriptions of incomplete trees and forests.

A first thing that can be missing is attribute values. In addition to them, the following structural information can be missing too:

(a) node ids (they can be replaced by node 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).

In both (c) and (d), we may allow partial information to be recovered: for example, we may know that a node is a leaf, without knowing its parent, or that it is a first child, without knowing its next sibling.

We now represent all these types of incompleteness by means of more expressive tree/forest descriptions than those in (1). Since we deal with two-sorted structures (over nodes and attribute values), we shall need variables of two kinds to represent unknown values of those. That is, we assume that we have disjoint sets of variables $\mathcal{V}_{\text{node}}$ (for node variables) and $\mathcal{V}_{\text{attr}}$ (for nulls that correspond to attribute values).

Node descriptions These are of the form

$$\beta \;=\; \ell^{\mu}(x)[@a_1 = z_1, \ldots, @a_m = z_m],$$

where

- $\ell \in \Sigma \cup \{\_\}$ (label or wildcard);
- μ is a *marking*: a subset (possibly empty) of *root, leaf, fc, lc.*
- $x \in \mathcal{V}_{\text{node}} \cup \mathcal{I}$ is a node variable or a constant node id.
- $@a_1, \ldots, @a_m$ are attribute names, and each z_i is a variable from $\mathcal{V}_{\text{attr}}$ or a constant from $\mathcal{D}$.

Incomplete descriptions We define incomplete tree descriptions (t) and incomplete forest descriptions (f) by

$$\begin{array}{rcl} t & := & \beta\langle f\rangle\langle\langle f'\rangle\rangle \\ f, f' & := & \varepsilon \;\mid\; t_1\;\theta_1\;t_2\;\theta_2\;\ldots\theta_{k-1}\;t_k \;\mid\; f \| f' \end{array} \tag{2}$$

where each θ_i is either $\rightarrow$ or $\rightarrow^*$, each t_i is an incomplete tree description and β is a node description.

Before giving a formal definition of the semantics (actually, two equivalent definitions), we provide an intuitive explanation of the semantics of incomplete descriptions. A node description $\ell^{\mu}(x)[@a_1 = z_1, \ldots, @a_m = z_m]$ introduces a node whose id is x, with m attributes $@a_i$'s whose values are z_i's. In addition, we may have extra information provided by the markings; for example, if $\mu = \{leaf, fc\}$, then we know that the node is a leaf, and the first child of its parent.

A tree description $\beta\langle f\rangle\langle\langle f'\rangle\rangle$ indicates a tree with a root node described by β so that it has a forest f of children and a forest f' of descendants. Forests could be empty (ε), or unions of forests ($f \| f'$), or forests of sibling trees (e.g., $t_1 \rightarrow t_2 \rightarrow^* t_3$ says that we have a forest consisting of two or three trees, so that the root of t_2 is the next sibling after the root of t_1, and the root of t_3 is a younger sibling than those two roots).

As an example, we describe the tree in Figure 3 from the introduction in our syntax. The 6 nodes are described by:

$$\begin{array}{lcl}\beta_0 & = & r^{\{root\}}(\mathsf{i}_0)\\ \beta_1 & = & book(\mathsf{i}_1)\\ \beta_3 & = & title(\mathsf{i}_3)[@title = \text{“}Found\ of\ DB\text{”}]\\ \beta_4 & = & author(\mathsf{i}_4)[@author = \text{“}Vianu\text{”}]\\ \beta_5 & = & year(\mathsf{i}_5)[@year = x]\\ \beta_7 & = & author(\mathsf{i}_7)[@author = \text{“}Abiteboul\text{”}]\end{array}$$

Then the whole tree is described by

$$\beta_0\langle\ \beta_1\langle\beta_3 \rightarrow \beta_4 \parallel \beta_5\rangle\ \rangle\ \langle\langle\beta_7\rangle\rangle.$$

(Strictly speaking, one should write $\beta_3\langle\varepsilon\rangle \rightarrow \beta_4\langle\varepsilon\rangle \parallel \beta_5\langle\varepsilon\rangle$ instead of $\beta_3 \rightarrow \beta_4 \parallel \beta_5$, but we shall omit empty forests ε for notational convenience and write just β instead of the more formal $\beta\langle\varepsilon\rangle$).

4.1 Classification of incomplete descriptions

There are three different groups of parameters that can vary as we define incomplete tree descriptions.

Node ids One possibility is to disregard them, as often done in the work on tree patterns [7, 10, 11, 18], i.e., assume that each node has a distinct variable for node id. In that case, we shall speak of *incomplete trees*. The incomplete tree description essentially enforces a tree structure for such incomplete descriptions (except possibly markings conflicting with the rest of the description).

At the opposite end, we have a model that corresponds to the DOM interface to XML, which assigns a constant id to each node [20, 24]. Such incomplete descriptions will be referred to as *incomplete DOM-trees*.

We formalize this in the following definition.

Definition 4.1. *Incomplete descriptions in which all node ids are variables (i.e. from $\mathcal{V}_{node}$), and no variable node id can be reused, are called* incomplete trees. *Incomplete descriptions in which all node ids are constants (i.e. from $\mathcal{I}$) are called* incomplete DOM-trees.

As in incomplete trees all node variables are distinct, we may in fact just omit them, writing, for example, $r\langle a \rightarrow b \| c\rangle$ instead of the more formal $r(x_1)\langle a(x_2) \rightarrow b(x_3) \| c(x_4)\rangle$. In incomplete DOM-trees, on the other hand, non-tree-shaped descriptions are possible, due to the reuse of ids. For example, $a(\mathsf{i}_0)\langle b(\mathsf{i}_1)\langle a(\mathsf{i}_0)\rangle\rangle$ says that a node with label b and id i_1 is a child of a node with label a and id i_0 which in turn is a child of a node with id i_1, i.e., the same node. This generates a cycle of length 2 and hence the description cannot represent any tree.

We now look at other parameters of incomplete descriptions.

Structure Another parameter refers to how much of the structure of a document can be described: that is, the set of axes used (among $\downarrow, \downarrow^*, \rightarrow, \rightarrow^*$, where $\downarrow^*$ and $\rightarrow^*$ are the reflexive-transitive closures of $\downarrow$ and $\rightarrow$), whether the union operation $\|$ on forests is allowed and whether markings μ can be used in descriptions. More precisely, we always assume that the child axis is allowed. The $\downarrow^*$ axis is allowed when we have the $\langle\langle f\rangle\rangle$ construct. The $\rightarrow, \rightarrow^*$, and $\|$ constructs occur in the description of forests. Finally if we have nodes with markings (among *root, leaf, fc, lc*), we indicate their presence by putting μ in the structure. Hence, the structural description is a subset of

$$\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|, \mu.$$

We shall always precede the definition of a class of trees with this structural information. For example, $(\downarrow, \rightarrow, \|, \mu)$-incomplete trees refers to incomplete trees that only use child, next-sibling, union of forests, and markings of nodes, and $(\downarrow, \downarrow^*, \|)$-DOM trees refers to DOM-trees that only use child, descendant, and union of forests (and do not use markings, sibling and younger-sibling).

Data values The third parameter refers to the treatment of attribute values. Normally, we allow both constants and variables, i.e., an analog of naïve tables. But in some cases we look at purely structural information, with no data values. Then we talk about trees *without attributes*.

To summarize, classes of incomplete descriptions will be referred to as

$$\textit{(structure)}\text{-incomplete} \left\{ \begin{array}{l} \text{tree} \\ \text{DOM-tree} \end{array} \right\}$$

(possibly *without attributes*), where structure is a subset of $\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|, \mu$.

Even assuming that we always have the child axis in descriptions, these parameters give rise to 2^7 cases. Of course we shall not be attempting to classify them all; rather, our goal is to understand which combinations of parameters give us good algorithms, and which naturally lead to intractability.

We now give a few remarks comparing these classes of tree descriptions with incomplete patterns considered in [4, 10, 11].

In general, the treatment of node ids need not be limited to the two extremes: all distinct variable ids, or all constant ids. One could use a model in which all ids are variables but some could be the same. Such a model would subsume tree patterns/conjunctive queries of [10, 11]. However, this does not give us proofs for free, as most proofs of hardness results in [10, 11] are based on the assumption that variables can be repeated and thus they apply to neither incomplete trees, in which we do not repeat variables, nor to incomplete DOM-trees, in which we do not use variables.

The model of [4], introduced in the context of active documents, is incomparable with ours. Indeed, on one hand, it considers only unordered trees, in which at most one attribute per node is permitted. On the other hand, it handles types of incompleteness that we do not deal with. Specifically, it assumes that a prefix of the document is completely known, while the rest is coded by a restricted form of DTDs. As more queries are posed, both portions of the documents are refined, on the basis of the answers. The model of [4] can be potentially captured by an extension of our model by an analog of conditional tables, but this is beyond the scope of this work.

4.2 Semantics

We provide two equivalent semantics: one views incomplete descriptions as formulae with free variables and gives a Tarskian satisfaction relation for them in complete trees. The other defines a relational representation of incomplete descriptions and then uses the standard relational incompleteness semantics via homomorphisms. Both give us the notion of $Rep(t)$ as a set of complete trees represented by the incomplete description t.

Let $\bar{x}$ be the set of all node variables used in t and $\bar{z}$ the set of all nulls used in t. Given a valuation $\nu = (\nu_{node}, \nu_{attr})$ with $\nu_{node} : \bar{x} \to \mathcal{I}$ and $\nu_{attr} : \bar{z} \to \mathcal{D}$, 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 node variables and 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 further define $Rep_{\Sigma,A}(t)$ as the restrictions of $Rep(t)$ to $\tau_{\Sigma,A}$-trees, for $\Sigma \subset Labels$ and $A \subset Attr$.

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_{node} and ν_{attr} are the identity when applied to node ids from $\mathcal{I}$ and data values from $\mathcal{D}$.

- $(T, \nu, s) \models \ell^{\mu}(x)[@a_1 = z_1, \ldots, @a_m = z_m]$ if and only if $\nu_{node}(x) = s$, node s is labeled ℓ (if $\ell \neq \_$), all the μ-markings are correct in s, and the value of each attribute $@a_i$ of s is $\nu_{attr}(z_i)$ (i.e., $(s, \nu_{attr}(z_i)) \in A_{@a_i}$).
- $(T, \nu, s) \models \beta\langle f \rangle\langle\langle f' \rangle\rangle$ if and only if $(T, \nu, s) \models \beta$ 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 $(T, \nu, S') \models f'$.
- $(T, \nu, \emptyset) \models \varepsilon$;
- $(T, \nu, \{s_1, \ldots, s_k\}) \models t_1\theta_1 t_2\theta_2 \ldots \theta_{k-1}t_k$ if and only if (s_i, s_{i+1}) is in NS whenever θ_i is $\rightarrow$, and in NS^* whenever θ_i is $\rightarrow^*$, for each $i < k$, and $(T, \nu, s_i) \models t_i$ for all i.
- $(T, \nu, S) \models f_1 \| f_2$ if and only if $S = S_1 \cup S_2$ such that $(T, \nu, S_i) \models f_i$, for $i = 1, 2$.

Remark. Note that the node s in the definition of $(T, \nu, s) \models t$ is superfluous since $s = \nu_{node}(x)$ for $t = \ell(x)[\ldots]\langle f \rangle\langle\langle f' \rangle\rangle$, but we prefer to make it explicit for notational convenience.

Relational representations Just as complete XML trees, incomplete trees have a natural relational representation. We shall present it now, and show that the semantics of incompleteness can be described in terms of homomorphisms between relational representations of incomplete and complete trees.

With each incomplete tree description t with labels from $\Sigma \subset \mathit{Labels}$ and attributes from $A \subset \mathit{Attr}$, we associate a relational structure $\underline{rel}(t)$ of vocabulary $\tau_{\Sigma,A}$. These will be two-sorted structures, whose active domains are subsets of $\mathcal{I} \cup \mathcal{V}_{\text{node}}$ and of $\mathcal{D} \cup \mathcal{V}_{\text{attr}}$, defined as unions of active domains of all node descriptions. For a node description $\beta = \ell^{\mu}(x)[@a_1 = z_1, \ldots, @a_m = z_m]$, we let $adom_{node}(\beta) = \{x\}$ and $adom_{attr}(\beta) = \{z_1, \ldots, z_m\}$.

For a tree (t) or forest (f) description, $\underline{rel}(t)$ or $\underline{rel}(f)$ is a two-sorted structure over domains $adom_{node}(t)$ and $adom_{attr}(t)$ (or f), defined inductively (together with the notion of root nodes) as follows:

1. If $t = \beta\langle f \rangle\langle\langle f' \rangle\rangle$, where $\beta = \ell^{\mu}(x)[(@a_i = z_i)_{i=1}^m]$, then $\underline{rel}(t)$ includes the union of $\underline{rel}(f)$ and $\underline{rel}(f')$ and in addition it has the following: all tuples $A_{@a_i}(x, z_i)$, all tuples $E(x, y)$, where y is a root node of f, all tuples $E^*(x, y')$, where y' is a root node of f'. Furthermore, x is added to P_ℓ if $\ell \neq \_$ and to unary relations $\mathit{Root}, \mathit{Leaf}, FC, LC$ according to the markings μ. The root node of t is x.
2. For $f = \varepsilon$, all the relations are empty;
3. For $f = t_1\ \theta_1\ \ldots\ \theta_{k-1}\ t_k$, where $x_1, \ldots, x_k$ are the root nodes of $t_1, \ldots, t_k$, we let $\underline{rel}(f)$ be the union of all $\underline{rel}(t_i)$s, and in addition we put (x_i, x_{i+1}) in NS or NS^*, depending on whether θ_i is $\rightarrow$ or $\rightarrow^*$. We call the x_i's the root nodes of f.
4. $\underline{rel}(f \| f')$ is the union of $\underline{rel}(f)$ and $\underline{rel}(f')$. We also define the root nodes of $f \| f'$ as the union of the root nodes of f and f'.

Let $h_1 : \mathcal{V}_{\text{node}} \cup \mathcal{I} \rightarrow \mathcal{V}_{\text{node}} \cup \mathcal{I}$ and $h_2 : \mathcal{V}_{\text{attr}} \cup \mathcal{D} \rightarrow \mathcal{V}_{\text{attr}} \cup \mathcal{D}$ be mappings that are constant on $\mathcal{I}$ and $\mathcal{D}$. Then $\bar{h} = (h_1, h_2)$ is a *homomorphism* of two relational structures T_1 and T_2 of vocabularies τ_{Σ_1,A_1} and τ_{Σ_2,A_2}, with $\Sigma_1 \subseteq \Sigma_2$ and $A_1 \subseteq A_2$, if for every tuple $\bar{x}$ in a relation R of τ_{Σ_1,A_1} in T_1, the tuple $\bar{h}(\bar{x})$ is in the relation R in T_2. Here, $\bar{h}(x)$ refers to $h_1(x)$ if $x \in \mathcal{V}_{\text{node}} \cup \mathcal{I}$ and to $h_2(x)$ if $x \in \mathcal{V}_{\text{attr}} \cup \mathcal{D}$.

We can alternatively define the semantics of t by the existence of a homomorphism from t into a complete tree T. This is equivalent to the first definition:

Proposition 4.2. *$T \in Rep(t)$ if and only if there is a homomorphism $\bar{h} : \underline{rel}(t) \rightarrow T$.*

Proof. Let t and f be incomplete descriptions, T be a complete tree, and $\nu = (\nu_{node}, \nu_{attr})$ be a valuation. We next show, by induction on the structure of t and f, the following two statements (recall the definition of *root* of an incomplete description):

$$(T, \nu, s) \models t \iff \nu \text{ is a homomorphism from } \underline{rel}(t) \text{ to } T \text{ and } s = \nu(x), \text{ where } x \text{ is the root of } t.$$

$$(T, \nu, S) \models f \iff \nu \text{ is a homomorphism from } \underline{rel}(f) \text{ to } T \text{ and } S = \{\nu(x) |\ x \text{ is a root of } f\}.$$

These statements imply that for every incomplete description t and every tree T, there exists a valuation ν and a node s in T such that $(T, \nu, s) \models t$ if and only if there exists a homomorphism from $\underline{rel}(t)$ to T (viewed as a 2-sorted structure of vocabulary $\tau_{\Sigma,A}$), and thus conclude the proof of the proposition.

We now prove the two statements above.

- Suppose that $f = \varepsilon$. Then $\underline{rel}(f)$ is empty and the statement trivially holds for $S = \emptyset$.

- Suppose that $t = \ell^{\mu}(x)[@a_1 = z_1, \ldots, @a_m = z_m]$. Assume first that $\ell \in \mathit{Labels}$. By the semantics of incomplete descriptions, $(T, \nu, s) \models t$ if and only if $s = \nu(x)$ and T contains atoms $P_\ell(s)$, $\mu(s)$, $A_{@a_i}(s, d_i)$ for $i \in [1, m]$, with $d_i = \nu_{attr}(z_i)$. Now, by construction, $\underline{rel}(t)$ consists of the set of atoms $\{P_\ell(x), \mu(x), A_{@a_i}(x, z_i) \mid i \in [1, m]\}$. Hence, it is easy to see that $(T, \nu, s) \models t$ if and only if ν is a homomorphism from $\underline{rel}(t)$ to T and $s = \nu(x)$. In the case that $\ell = \_$ the same argument works by removing atom $P_\ell(x)$ from $\underline{rel}(t)$, and by ignoring label predicates on node s of T.

- Suppose that $t = \beta\langle f' \rangle \langle\langle f'' \rangle\rangle$, and let x be the node variable of β. Then x is also the root node of t, as well as the root node of β. By the semantics of incomplete descriptions, $(T, \nu, s) \models t$ if and only if:

  (i) $(T, \nu, s) \models \beta$,

  (ii) there exists a set S' of children of s such that $(T, \nu, S') \models f'$ and

  (iii) there exists a set S'' of descendants of s, such that $(T, \nu, S'') \models f''$

  Now, by the induction hypothesis, (i), (ii) and (iii) above are equivalent respectively to the following statements:

  (vi) ν is a homomorphism from $\underline{rel}(\beta)$ to T and $s = \nu(x)$;

  (v) ν is a homomorphism from $\underline{rel}(f')$ to T and the nodes $\{\nu(y') |\ y' \text{ is a root of } f'\}$ are children of s (i.e. tuples $E(s, \nu(y'))$ are in T, for each root node y' of f');

  (vi) ν is a homomorphism from $\underline{rel}(f'')$ to T and the nodes $\{\nu(y'') |\ y'' \text{ is a root of } f''\}$ are descendants of s (i.e. tuples $E^*(s, \nu(y''))$ are in T, for each root node y'' of f'');

  Moreover, by construction, $\underline{rel}(t)$ is the union of $\underline{rel}(\beta)$, $\underline{rel}(f')$ and $\underline{rel}(f'')$ with the set of atoms $\{E(x, y') \mid y' \text{ is a root of } f'\}$, and the set of atoms $\{E^*(x, y'') \mid y'' \text{ is a root of } f''\}$.

  It is now immediate to check that the conjunction of (iv), (v) and (vi) is equivalent to stating that ν is a homomorphism from $\underline{rel}(t)$ to T and $s = \nu(x)$. Hence $(T, \nu, s) \models t$ if and only if ν is a homomorphism from $\underline{rel}(t)$ to T and $s = \nu(x)$.

- The cases $f = [t_1\ \theta_1\ t_2\ \theta_2\ \ldots \theta_{k-1}\ t_k]$ and $f = f_1 \| f_2$ can be handled similarly. □

Since $\underline{rel}(t)$ is an incomplete database over $\tau_{\Sigma,A}$, we can look at the set of complete databases $Rep(\underline{rel}(t))$ that it represents. Then one can ask how $Rep_{\Sigma,A}(t)$ and $Rep(\underline{rel}(t))$ are related. It turns out that they are the same when we restrict our attention to trees (note $Rep(\underline{rel}(t))$ need not contain only trees). We say that a database D of $\tau_{\Sigma,A}$ *represents* an incomplete tree description t if and only if $Rep_{\Sigma,A}(t) = Rep(D) \cap$ TREES, where TREES refers to all databases of our relational vocabularies that represent complete XML trees. The proof of the following result is in the appendix.

Proposition 4.3. *a)* $\underline{rel}(t)$ *represents* t*;*
b) for every structure D *of vocabulary* $\tau_{\Sigma,A}$*, there is an incomplete tree description* t_D *so that* D *represents* t_D*.*

Summary: incomplete trees vs DOM-trees For the convenience of the reader, we provide a quick summary of the main differences between the two key objects of our study.

In DOM-trees nodes come with explicit ids, hence we always know which nodes of complete trees they map into. Given any two nodes in an incomplete DOM-tree, we know whether they refer to the same node of a complete tree, or to different ones. In particular, if a complete tree is given, then the node-homomorphism from the DOM-tree into it is already implicit.

On the other hand, incomplete trees leave this open to arbitrary interpretations. They cannot require that two nodes of an incomplete tree be equal (i.e., mapped into the same node of a complete tree), nor can they require two (unordered) siblings to be different. This is achieved by using all distinct variables as node ids.

5 The consistency problem

The standard computational problems studied in connection with incomplete information in relational databases are membership (whether a complete database can be represented by an incomplete description) and query answering. Others are variations of these two (e.g., containment $Rep(R) \subseteq Rep(R')$ can be viewed as a special case of query answering). In the case of XML we have an additional problem that needs to be addressed – consistency. Due to complicated descriptions of XML documents, it is possible to provide inconsistent specifications. This is a well-recognized phenomenon, and there are many results on consistency and satisfiability for XML schemas, constraints, patterns, and queries [6, 9, 10, 11, 15, 22, 23, 31, 32, 37]. We already saw some examples of inconsistent descriptions: for example, under the DOM model, we can say that nodes with ids i_1 and i_2 are connected by the child edge in both directions, which is inconsistent with any tree description. With markings too inconsistency is possible, e.g., $a\langle b^{root}\rangle$ saying that a child node is marked *root*.

The presence of a DTD also may lead to inconsistency. In the next example we actually use data values and nulls. Consider a DTD $\rho(r) = bb;\ \rho(b) = \varepsilon$, where b has an attribute $@a$, and a description $r\langle b[@a = c_1] \rightarrow b[@a = c_2] \parallel b[@a = z] \rightarrow b[@a = z]\rangle$, where $c_1 \neq c_2$ are two constants from $\mathcal{D}$. This is inconsistent with the DTD.

We consider the following problem:

PROBLEM:	CONSISTENCY
INPUT:	an incomplete description t
QUESTION:	is $Rep(t) \neq \emptyset$?

We also look at a variation with a fixed DTD d: the problem CONSISTENCY(d) asks whether $Rep_d(t) = Rep(t) \cap \{T \mid T \models d\}$ is nonempty.

5.1 An upper bound

First, we get an upper bound on the complexity of the problem of consistency of incomplete tree descriptions.

Theorem 5.1. *Both* Consistency *and* Consistency*(d) are in* NP *for incomplete descriptions. In fact, even if both t and d are given as inputs, checking whether $Rep_d(t) \neq \emptyset$ can be done in* NP.

Proof. Let $d = (r, \rho, \alpha)$ be a DTD over Σ and A, and let $\Sigma_d \subseteq \Sigma$ be the set of all those labels ℓ that are "useful" in d; that is, there exists a tree T_ℓ with a node labeled ℓ and such that T_ℓ conforms to d. As the following proposition states, Σ_d can be constructed in polynomial time from d.

Proposition 5.2. *[2] There exists a polynomial time algorithm that, given a DTD d, computes Σ_d.*

We say that the tree T over vocabulary $\tau_{\Sigma,A}$ *weakly* conforms to the DTD d, if for each node s labeled ℓ in T it is the case that (1) $\ell \in \Sigma_d$, (2) if s is the root of T then $\ell = r$, (3) the set of attributes of s is precisely $\alpha(\ell)$, and (4) if s is not a leaf of T, then it is the case that the labels of its children, read from left-to-right, forms a string in the regular language $\rho(\ell)$. Intuitively, T weakly conforms to d if every label used in T is useful in d, the tree obtained from T by considering all nodes besides the leaves conforms to d, and the leaves of T conform to d with respect to α.

It follows from the next claim, that Consistency(d) is in NP even if both t and d are given as inputs. Indeed, the claim proves that in order to show that $Rep_d(t) \neq \emptyset$, a nondeterministic algorithm only needs to guess a tree T (of vocabulary $\tau_{\Sigma,A}$), of polynomial size in t and d, and a mapping $\bar{h} : \underline{rel}(t) \to T$, of size polynomial in t, and then verify that T weakly conforms to d and that $\bar{h} : \underline{rel}(t) \to T$ is a homomorphism. This can be easily done in nondeterministic polynomial time.

Claim 5.3. *Let t be an incomplete tree description. There exists a polynomial $\varphi(x, y)$ that depends neither on t nor d, such that $Rep_d(t) \neq \emptyset$ if and only if there exists a tree T and a mapping $\bar{h} : \underline{rel}(t) \to T$, such that T weakly conforms to d, $\bar{h} : \underline{rel}(t) \to T$ is a homomorphism, and the size of T is at most $\varphi(|t|, |d|)$.*

We next prove Claim 5.3. Define $\varphi(x, y)$ to be $k_2+1+k_2\cdot(k_1-2)\cdot x^2$, where $k_1 = (y+3)(x^2+1)+1$ and $k_2 = yx^2$. Assume first that there exists a tree T and a mapping $\bar{h} : \underline{rel}(t) \to T$, such that T weakly conforms to d, $\bar{h} : \underline{rel}(t) \to T$ is a homomorphism, and the size of T is at most $\varphi(|t|, |d|)$. We prove next that $Rep_d(t) \neq \emptyset$.

For each $\ell \in \Sigma_d$, let T_ℓ be an arbitrary tree such that ℓ appears in T_ℓ and i_ℓ be an arbitrary node of T_ℓ that is labeled ℓ. We denote by $T_\ell^{\downarrow}$ the subtree of T_ℓ induced by all descendants of i_ℓ, including i_ℓ. Let $s_1, \ldots, s_m$ be an enumeration of the leaves in T, and assume that for each $1 \leq i \leq m$, s_i is labeled $\ell_i \in \Sigma_d$ in T. Then recursively construct a sequence $T_0, T_1, \ldots, T_m$ as follows: $T_0 = T$ and for each $1 \leq i \leq m$, T_i is the tree obtained from T_{i-1} by replacing s_i with a copy of $T_{\ell_i}^{\downarrow}$ whose set of node ids is disjoint from the set of node ids in T_{i-1}. It is not hard to see that T_m conforms to d, and that there is a homomorphism from $\underline{rel}(t)$ to T_m. It follows from Proposition 4.2 that $Rep_d(t) \neq \emptyset$.

Assume on the other hand that $Rep_d(t) \neq \emptyset$. We prove next that there exists a tree T and a mapping $\bar{h} : \underline{rel}(t) \to T$, such that T weakly conforms to d, $\bar{h} : \underline{rel}(t) \to T$ is a homomorphism, and the size of T is at most $\varphi(|t|, |d|)$.

Since $Rep_d(t) \neq \emptyset$, it follows from Proposition 4.2 that there exists a tree T_0 that conforms to d and a homomorphism $\bar{h}_0 : \underline{rel}(t) \to T_0$. What we do first is to construct, from T_0, another tree in $Rep_{\Sigma,A}(t)$, such that this tree weakly conforms to d and all of its vertical paths are of polynomial length. In order to do that we define below the notion of vertical shortcuts.

Define the *skeleton* of T_0, denoted by $sk(T_0)$, recursively as follows: (1) If a node s is the root of T_0 or belongs to the image of $\bar{h}_0$, then s belongs to $sk(T_0)$; and (2) if the nodes s_1 and s_2 of T_0 belong to $sk(T_0)$, then so it does its least common ancestor. It is easy to see that the size of $sk(T_0)$ is at most quadratic in the size of t.

Vertical shortcuts: Let $|\Sigma| = q$ and consider an arbitrary vertical path $s_1 \dots s_{q+4}$ in T_0, such that none of nodes $s_1, \dots, s_{q+3}$ belongs to $sk(T_0)$ and s_{q+4} has a descendant in $sk(T_0)$. Because the length of this path is bigger than $q+3$, there exist two indexes $1 < j_1 < j_2 < q+4$, such that s_{j_1} and s_{j_2} have the same label in T_0. Let $T_0(s_{j_1} \uparrow s_{j_2})$ be the tree obtained from T_0 by replacing the tree rooted at s_{j_1} with the tree rooted at s_{j_2}. We say that $T_0(s_{j_1} \uparrow s_{j_2})$ is a *vertical shortcut* of T_0. It is not hard to see that $T_0(s_{j_1} \uparrow s_{j_2})$ still conforms to d. It is also possible to prove that every element in $sk(T_0)$ belongs to $T_0(s_{j_1} \uparrow s_{j_2})$. Indeed, assume for the sake of contradiction, that there exists an element s in the image of $sk(T_0)$ that does not belong to $T_0(s_{j_1} \uparrow s_{j_2})$. Then s belongs to the subtree rooted at s_k, for some $k \in [j_1, j_2 - 1]$. But then s_k is the least common ancestor of s and any descendant s' of s_{j_2} that belongs to $sk(T_0)$. It follows that s_k belongs to $sk(T_0)$, which is a contradiction. In addition, it is not hard to see that $\bar{h}_0 : \underline{rel}(t) \to T_0(s_{j_1} \uparrow s_{j_2})$ is a homomorphism.

Applying the process of vertical short-cutting inductively, we obtain a tree T_1 that conforms to d, and such that the mapping $\bar{h}_0 : \underline{rel}(t) \to T_1$ is a homomorphism. We define $sk(T_1) = sk(T_0)$. Notice that it may still be the case that some vertical paths in T_1 are not of polynomial length. This may happen, for instance, if there is a subtree rooted at a node s in T_0 that does not contain a node in $sk(T_0)$, but that has a vertical path that is not of polynomial length. In order to prune the long vertical paths of T_1, we construct from T_1 a new tree T_2 as follows: The tree T_2 is obtained from T_1 by removing all proper descendants of each node s in T_1, such that s does not have a proper descendant in $sk(T_1)$. Clearly, every element in $sk(T_1)$ belongs to T_2, and $\bar{h}_0 : \underline{rel}(t) \to T_2$ is a homomorphism. We define $sk(T_2) = sk(T_1)$. Further, it is easy to see that T_2 weakly conforms to d.

We claim that the length of each vertical path in T_2 is at most $(q+3) \cdot (|sk(T_0)| + 1) + 1$, i.e. each path of T_2 is of polynomial length. Indeed, assume for the sake of contradiction that there exists a vertical path $s_1 \dots s_n$ in T_2 such that $n > (q+3) \cdot (|sk(T_0)| + 1) + 1$. We assume without loss of generality that both s_1 and s_n belong to $sk(T_2)$. Let $s_1 = s_{i_1} < s_{i_2} < \dots < s_{i_m} = s_n$ be the elements of this path that belong to $sk(T_2)$. Then, since T_1 is obtained from T_0 by applying all possible vertical shortcuts, it must be the case that $i_{j+1} - i_j \leq q+2$, for each $1 \leq j \leq m-1$. Since $m \leq |sk(T_2)|$ and $|sk(T_2)| = |sk(T_0)|$, it must be the case that $n \leq (q+3) \cdot |sk(T_0)| + 1$, which is a contradiction.

From T_2 we now construct a new tree, such that this tree belongs to $Rep_{\Sigma,A}(t)$, it weakly conforms to d, and the number of children of each one of its nodes is polynomially bounded.

Horizontal shortcuts: Let p be the maximum number of states of an NFA of the form $\rho(\ell)$, for $\ell \in \Sigma$. Let $s_1 \dots s_{p+1}$ be a horizontal path in T_2, such that no subtree rooted at a node of the form s_j, for $j \in [1, p]$, has an element in $sk(T_2)$. Further, assume that the parent s of the elements in this path is labeled ℓ. Choose an arbitrary accepting run π of the NFA $\rho(\ell)$ over the children of s. Since the length of the path is strictly bigger than p, there exist two indexes $1 \leq j_1 < j_2 \leq p+1$, such that $\pi(s_{j_1}) = \pi(s_{j_2})$. Thus, removing the subtrees of T_2 rooted at $s_{j_1}, \dots, s_{j_2-1}$ yields a tree $T_2(s_{j_1} \leftarrow s_{j_2})$ that weakly conforms to d, and such that every element of $sk(T_2)$ belongs to $T_2(s_{i_1} \leftarrow s_{i_2})$ and $\bar{h}_0 : \underline{rel}(t) \to T_2(s_{i_1} \leftarrow s_{i_2})$ is a homomorphism. By inductively applying the horizontal short-cutting technique, we obtain a tree T_3 that weakly conforms to d, every element of $sk(T_2)$ belongs to T_3 and $\bar{h}_0 : \underline{rel}(t) \to T_3$ is a homomorphism, the length of each path in T_3 is polynomial in the size of t and d, and for each node s in T_3 the number of children of s is at most $p \cdot (|sk(T_3)| + 1) = p \cdot (|sk(T_0)| + 1)$, i.e. polynomial in the size of t and d.

Let $k'_1 = (q+3) \cdot (|sk(T_0)| + 1) + 1$ and $k'_2 = p \cdot |sk(T_0)|$. It is not hard to see that for each $i \leq k'_1$,

the number of nodes of T_3 of depth $\leq i$ is bounded by u_i, where:

$$\begin{aligned} u_1 &= 1 \\ u_2 &= u_1 + k_2' \\ u_3 &= u_2 + |sk(T_0)| \cdot k_2' \\ &\cdots \\ u_i &= u_{i-1} + |sk(T_0)| \cdot k_2' \end{aligned}$$

Thus, the size of T_3 is bounded by $u = k_2' + 1 + k_2' \cdot (k_1' - 2) \cdot |sk(T_0)|$. Clearly, $u \leq \varphi(|t|, |d|)$, It follows that the size of T_3 is bounded by $\varphi(|t|, |d|)$, which concludes the proof of both the claim and Theorem 5.1. □

Notice that a slight variation of this proof also shows the following. Given an incomplete description t and a nondeterministic tree automaton $\mathcal{A}$, the problem of whether there exists a tree T that belongs to $Rep(t)$ and to the language defined by $\mathcal{A}$ is also in NP.

We want to understand which features lead to NP-hardness, and which ones allow efficient algorithms. Before we embark on this study, we make a few remarks.

The consistency problem appears related to several well-studied problems – chase-based tools, constraint satisfaction, automata on trees – but techniques from those areas do not seem to provide us with a way of getting efficient algorithms. For example, some of the algorithmic techniques for checking consistency have a feel of a chase procedure that completes the relational representation $\underline{rel}(t)$. But we cannot apply chase 'as is'. The main constraint – that the resulting structure be a tree – is not even first-order expressible. Also, some constraints are disjunctive in nature: e.g., for two children s and s' of the same node, either $s \rightarrow^* s'$ or $s' \rightarrow^* s$ holds. While chase with disjunctive constraints has been considered [19], it generally yields intractable upper bounds, which we already have from Theorem 5.1.

By Proposition 4.2, consistency can be viewed as the existence of a homomorphism from $\underline{rel}(t)$ into some structure T. This suggests applicability of constraint satisfaction tools, since tractable restrictions are very well understood (cf. [28]). But Theorem 5.1 only provides an upper bound on the size of T. In particular, it is possible for T to have both long branches and high branching degree, and hence Theorem 5.1 does not give a construction for a polysize T to reduce consistency to constraint satisfaction.

The problem with using automata is that data values come from an infinite domain. While some automata models have been developed for them [34, 35], they do not lead to efficient algorithms for expressive problems such as those we consider here. Furthermore, even without data values, not all incomplete descriptions can be represented by automata of polynomial size.

5.2 Consistency of incomplete trees

5.2.1 Consistency without DTDs

The first result is about the consistency problem without DTDs. For incomplete trees, only markings can lead to inconsistency. For descriptions with markings we provide a full classification: we prove a dichotomy that classifies all the cases as either PTIME or NP-complete.

Theorem 5.4. *Each $(\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|)$-incomplete tree (i.e., an incomplete tree without markings) is consistent.*

With markings, the complexity of Consistency *is*

- NP*-complete for the fragments $(\downarrow, \rightarrow, \star, fc, lc)$ and $(\downarrow, \downarrow^*, \star, fc, lc, leaf)$, where $\star$ is $\rightarrow^*$ or $\|$;*

- PTIME *for all other fragments containing* $\downarrow$.

Proof. We first handle the no-markings case, and then present samples of cases for the dichotomy result, with remaining cases in the appendix.

CONSISTENCY ***for incomplete trees without markings.*** Given a $(\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|)$- incomplete tree t, we define a function δ which is intended to map each $(\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|)$- incomplete tree t into a tree $T \in Rep(t)$.

We fix an arbitrarily chosen mapping $h = (h_1, h_2)$ such that $h_2 : \mathcal{V}_{\text{attr}} \cup \mathcal{D} \rightarrow \mathcal{V}_{\text{attr}} \cup \mathcal{D}$ is the identity on $\mathcal{D}$, and $h_1 : \mathcal{V}_{\text{node}} \cup \mathcal{I} \rightarrow \mathcal{V}_{\text{node}} \cup \mathcal{I}$ is the identity on $\mathcal{I}$ and is injective on $\mathcal{V}_{\text{node}}$.

The function δ is defined inductively on the structure of t:

- If $t = \beta\langle f_1 \rangle\langle\langle f_2 \rangle\rangle$, with $\beta = \ell(x)[@a_1 = z_1, \ldots, @a_m = z_m]$ then $\delta(t) = B\langle \delta(f_1)\ \delta(f_2)\rangle$, where $B = l(h(x))[@a_1 = h(z_1), \ldots, @a_m = h(z_m)]$, and $l = \ell$ if $\ell \in$ *Labels*, otherwise l is an arbitrary label of *Labels*.
- $\delta(\varepsilon) = \varepsilon$
- $\delta(f \| f') = \delta(f)\ \delta(f')$
- $\delta(t_1\theta_1 t_2 \ldots \theta_{k-1} t_k) = \delta(t_1)\ \delta(t_2) \cdots \delta(t_k)$

A routine induction argument proves the following lemma:

Lemma 5.5. *For each $(\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|)$-incomplete tree t and each $(\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|)$-incomplete forest f, by letting s the root node of $\delta(t)$ and $s_1, \ldots, s_k$ the set of root nodes of the sequence of trees $\delta(f)$:*

- *$(\delta(t), h, s) \models t$;*
- *for each complete tree $T = B\langle f_1\ \delta(f)\ f_2\rangle$, where B is an arbitrary complete node description and f_1 and f_2 two arbitrary sequences of complete trees, $(T, h, s_1, \ldots, s_k) \models f$.*

As a corollary of Lemma 5.5, for each $(\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \|)$-incomplete tree t we have that $\delta(t) \in Rep(t)$, therefore t is consistent.

CONSISTENCY ***of*** $(\downarrow, \rightarrow, \|, fc, lc)$ ***and*** $(\downarrow, \rightarrow, \rightarrow^*, fc, lc)$***-incomplete trees.*** We consider next the case when incomplete trees contain markings. In particular, we prove NP-hardness of CONSISTENCY for $(\downarrow, \rightarrow, \|, fc, lc)$-incomplete trees without attributes. We reduce 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 $w_0, w_1 \in \Sigma^*$.

We define a $(\downarrow, \rightarrow, \|, fc, lc)$-incomplete tree t without attributes over alphabet $\Sigma \cup \{R\}$ with $R \notin \Sigma$

$$t = R(x)\langle f_K \| f_{s_1} \| \ldots \| f_{s_n} \rangle$$

where f_K is the incomplete forest:

$$f_K = \_^{fc}(x_1) \rightarrow \_ \rightarrow \ldots \rightarrow \_^{lc}(x_K)$$

having exactly K wildcard nodes. For each string $s = a_1 a_2 \cdots a_m \in S$, the incomplete forest f_s is defined as:

$$f_s = a_1 \rightarrow a_2 \ldots \rightarrow a_m$$

(where node variables are omitted for the sake of clarity).

We claim that $Rep(t) \neq \emptyset$ if and only if there exists a common superstring of S of length not greater than K. Indeed, assume there exists such a superstring w; if $|w| < K$ then we pad w with an arbitrary suffix $w_1 \in \Sigma^*$ such that $|ww_1| = K$. Let $w' = ww_1 = b_1 \cdots b_K$. We now show that the complete tree:

$$T = R(\mathsf{i}_0)\langle b_1(\mathsf{i}_1) \dots b_K(\mathsf{i}_K)\rangle$$

is in $Rep(t)$. In fact:

- since f_K has size K, there exists a valuation ν_0 with $\nu_0(x_i) = \mathsf{i}_i$ for each $i \in [1, n]$ and such that $(T, \nu_0, \mathsf{i}_1, \dots, \mathsf{i}_K) \models f_K$;
- For each $s \in S$ (because s is a substring of $b_1 \cdots b_K$), there exist children $\mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}$ of i_0 in T and a valuation ν_s of node variables of f_s such that $(T, \nu_s, \mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}) \models f_s$.

Now it is enough to take a valuation ν mapping x (the root of t) into i_0, such that ν coincides with ν_0 on node variables of f_K, and ν coincides with ν_s on node variables of f_s, for each $s \in S$. We have $(T, \nu, \mathsf{i}_0) \models t$.

Conversely assume that $Rep(t) \neq \emptyset$, then there exists a tree T over some alphabet $\Sigma' \subseteq \mathit{Labels}$, a node p of T and a valuation ν of node variables of t such that $(T, \nu, p) \models t$. The node p in T has label R; let $b_1(\mathsf{i}_1) \dots b_l(\mathsf{i}_l)$ be the sequence of children of p, with $b_1 \cdots b_l \in \Sigma'^*$. Since node x_1 of t is labeled as fc, we have that $\nu(x_1) = \mathsf{i}_1$ and therefore $\nu(x_j) = \mathsf{i}_j$, for each $j \in [1, K]$. But x_K is labeled with lc, therefore we have $\nu(x_K) = \mathsf{i}_l$, hence $l = K$. We also know that for each $s \in S$, there must exist nodes $\mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}$ among the children of p such that $(T, \nu, \mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}) \models f_s$.

It follows that $b_1 \cdots b_K$ is a superstring of s, for each $s \in S$. However $b_1 \cdots b_K$ is a string of Σ'^*, so it is not yet a solution for the shortest superstring problem. But if we replace each symbol $b_i \notin \Sigma$ with an arbitrary symbol of Σ the resulting string of Σ^* has length K and is still a superstring of strings s, for each $s \in S$. This completes the reduction.

The same reduction can be slightly modified to prove that Consistency of $(\downarrow, \rightarrow, \rightarrow^*, fc, lc)$-incomplete trees is NP-hard: we construct incomplete trees $t_0 = R(y_0)^{fc,lc}\langle f_K\rangle$ and $t_i = R(y_i)\langle f_{s_i}\rangle$ for $i \in [1, n]$ and

$$t = R(x)\langle t_0 \rightarrow^* t_1 \dots \rightarrow^* t_n\rangle$$

It is straightforward to adapt the previous proof and show that $Rep(t) \neq \emptyset$ if and only if S has a superstring of length at most K.

The cases of $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$ and $(\downarrow, \downarrow^*, \rightarrow^*, fc, lc, leaf)$-incomplete trees are presented in the appendix.

Now we move to ***polynomial time cases***.

Given an arbitrary incomplete tree t, we define a *chase* on its relational representation $\underline{rel}(t)$. We prove that the chase may either fail or result, in polynomial time, in a new structure denoted by $chase(t)$. We finally prove that, in any fragment of incomplete trees including neither $(\downarrow, \rightarrow, \|, fc, lc)$ nor $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$ nor $(\downarrow, \rightarrow, \rightarrow^*, fc, lc)$ nor $(\downarrow, \downarrow^*, \rightarrow^*, fc, lc, leaf)$, the chase succeeds if and only if t is consistent. Moreover we show how a tree in $Rep(t)$ can be constructed from $chase(t)$.

We now define the chase on an arbitrary incomplete tree t with labels from a set $\Sigma \subset \mathit{Labels}$ and attributes from $A \subset \mathit{Attr}$. Intuitively, the objective of the chase is to move all markings of t into the right place (i.e. *root* markings only on the root, *leaf* markings only on leaves and *fc* and *lc* markings only on first and last children).

A chase step applies to an incomplete relational structure having a *tree-shape*. Intuitively a tree-shaped structure is a generalization of an incomplete tree where the *NS* and *NS** relations over children (or descendants) of the same node define an arbitrary graph, instead of being restricted to a union of simple paths.

More formally, an incomplete relational structure D in the vocabulary $\tau_{\Sigma,A}$ has a *tree-shape* if it satisfies all of the following properties (with a little abuse of notation, in what follows we denote by $adom_{node}(D)$ and $adom_{attr}(D)$ the node and the attribute sort of D):

- the structure obtained from D by replacing the NS and NS^* relations with empty ones, and removing possible tuples of the form (x, x) from relation E^*, is the relational representation of an incomplete tree; this incomplete tree will be denoted by $t(D)$.
- $adom_{node}(D) = adom_{node}(t(D))$, that is, the instances of NS and NS^* relations in D are over domain $adom_{node}(t(D))$. We'll denote by $G_{NS}(D)$ the graph whose nodes are the variables $adom_{node}(D)$ and whose edges are of two types: NS-edges defined by the relation NS in D and NS^*-edges defined by $NS^* \setminus \{(x, x)|x \in \mathcal{V}_{\text{node}}\}$.
- if C is a non-singleton connected component of $G_{NS}(D)$, there exists $x \in adom_{node}(D) \setminus C$ such that:
  - either $E(x, y)$ holds for all $y \in C$
  - or $E^*(x, y)$ holds for all $y \in C$.

  In both cases x will be referred to as the *parent* of C. We will say that x is the *E-parent* of C in the first case and the *E^*-parent* of C in the second case.

Notice that $\underline{re\ell}(t)$ has a tree-shape, according to the above definition.

In order to describe the application of chase steps we need to define the following operation on node descriptions:

Definition 5.6. *Let $\{\beta_1, \ldots, \beta_n\}$ be a set of node descriptions with $\beta_i = \ell_i^{\mu_i}(x_i)[@a_{i1} = z_{i1}, \ldots, @a_{im_i} = z_{im_i}]$, where all variables x_i are distinct, for $i \in [1, n]$.*

A merging mapping *for $\{\beta_1, \ldots, \beta_n\}$ is a mapping $h_{null} : \mathcal{V}_{attr} \cup \mathcal{D} \to \mathcal{V}_{attr} \cup \mathcal{D}$ such that:*

- *h_{null} is the identity on constants and*
- *whenever $@a_{ik} = @a_{jl}$ for some $i, j \in [1, n]$ and $k \in [1, m_i]$ and $l \in [1, m_j]$, then $h_{null}(z_{ik}) = h_{null}(z_{jl})$*

Definition 5.7. *If $\{\beta_1, \ldots, \beta_n\}$ is a set of node descriptions having all distinct node variables we say that $\beta_1, \ldots, \beta_n$ can be* merged *if both the following conditions hold:*

- *there exist no two descriptions β_i and β_j with labels ℓ_i and ℓ_j, such that $\ell_i, \ell_j \in$ Labels and $\ell_i \neq \ell_j$;*
- *there exists a* merging mapping *for $\beta_1, \ldots, \beta_n$.*

A merging mapping *h_m for node descriptions $\beta_1, \ldots, \beta_n$ is* minimal *if all merging mappings h of $\beta_1, \ldots, \beta_n$ can be written as:*

$$h = h' \circ h_m$$

for some mapping $h' : \mathcal{V}_{attr} \cup \mathcal{D} \to \mathcal{V}_{attr} \cup \mathcal{D}$.

If $\beta_1, \ldots, \beta_m$ can be merged and have node variables $x_1, \ldots, x_m$ respectively, we denote by $h_{\beta_1 \ldots \beta_n}$ a mapping (h_{null}, h_{node}), where h_{null} is a minimal merging mapping for $\beta_1, \ldots, \beta_n$, and $h_{node} : \mathcal{V}_{\text{node}} \cup \mathcal{I} \to \mathcal{V}_{\text{node}} \cup \mathcal{I}$ is the mapping sending each node variable x_i into x_1, for each $1 \leq i \leq m$ (and being the identity elsewhere).

The existence of a merging mapping for $\beta_1, \ldots, \beta_n$ can be easily checked by solving the system of equalities $\{z_{ik} = z_{jl} | @a_{ik} = @a_{jl}\}$ by successive replacement. If the replacement procedure succeeds

without ever generating an equality $c = c'$ for some $c, c' \in \mathcal{D}$ with $c \neq c'$, then it results in a mapping $h : \mathcal{V}_{\text{attr}} \cup \mathcal{D} \to \mathcal{V}_{\text{attr}} \cup \mathcal{D}$ which is the identity on constants. The mapping h is a merging mapping since it satisfies the equalities, but is also minimal because all other solutions of the system (that is, all other merging mappings) can be obtained by assigning arbitrary values in $\mathcal{V}_{\text{attr}} \cup \mathcal{D}$ to variables in the image of h.

We are now ready to describe the chase steps and their application.

Chase steps. Assume D is a relational structure having a tree-shape. We now describe when chase steps are applicable on nodes of D. If a chase step is applicable on some node x of D, the application of the step on x may either fail or result in a new structure D' with tree-shape. This is described next.

leaf step. A *leaf step* is *applicable* on node $x \in adom_{node}(D)$ if x occurs in the *Leaf* relation and is not a leaf of $t(D)$.

If a *leaf step* is applicable on node x of D, it applies as follows:

- If there exists $y \in adom_{node}(D)$ such that $E(x, y)$ holds in D, the application of the step on x fails.
- If there exists no such y, then the subtree of $t(D)$ whose root variable is x is of the form $\beta\langle\langle f \rangle\rangle$ (we know f is not empty since x is not a leaf). Let $x_1, \ldots, x_n$ be the root variables of the forest f. If there exist x_i, x_j with $1 \leq i, j \leq n$ such that $NS(x_i, x_j)$ holds, then the application of the step fails.
- Otherwise let $\beta_1, \ldots, \beta_n$ be the node descriptions of roots of f with node variables $x_1, \ldots, x_n$ respectively. If $\beta_1, \ldots, \beta_n, \beta$ cannot be merged, the application of the step fails.
- Otherwise let $h = h_{\beta, \beta_1, \ldots, \beta_m}$, The application of the step results in a new structure $D' = h(D)$.

We next show that the structure D' has a tree-shape, by proving that it satisfies the three properties of a tree-shaped structure.

1) The incomplete tree $t(D')$ can be obtained from $t(D)$ by making node x collapse with its children. The fact that node descriptions of x and its children can be *merged* guarantees that the variable x appears in only one relation P_ℓ in D'. Moreover the application of h to data variables of D guarantees that collapsed nodes agree on common attributes, and thus the attribute relations of D' still code functions (that is, each attribute relation $A_{@a}$ in D' associates at most one attribute value to each node).

2) $adom_{node}(D') = h(adom_{node}(D))$ and, since D has a tree-shape, $adom_{node}(D) = adom_{node}(t(D))$. Therefore $adom_{node}(D') = h(adom_{node}(t(D))) = adom_{node}(t(D'))$.

3) It remains to show that each non-singleton connected component of $G_{NS}(D')$ has a *parent* in D'. Indeed, up to self-NS^* loops, $G_{NS}(D') = h(G_{NS}(D))$. Then $G_{NS}(D')$ is obtained by collapsing nodes $x, x_1, \ldots, x_n$ in the graph $G_{NS}(D)$. Since the set of nodes $\{x_1, \ldots, x_n\}$ must coincide with a set of connected components of $G_{NS}(D)$ (due to the tree-shape of D), their collapsing does not affect any other connected component. (Notice that the collapsing of $x, x_1, \ldots, x_n$ may introduce a self-NS^* loop in node x of D', but self-NS^* loops are not part of the G_{NS} graphs, so also the connected component of x is not affected by the collapsing.) As a consequence, each connected component of $G_{NS}(D')$ is of the form $h(C)$, where C is a connected component of $G_{NS}(D)$, and C contains no x_i, for $i \in [1, n]$.

Now let $h(C)$ be a non-singleton connected component of $G_{NS}(D')$, where C is a connected component of $G_{NS}(D)$ containing no x_i. Let z be the *parent* of C in D. We prove that $h(z)$

is the *parent* of $h(C)$ in D'. Note that one only needs to prove that $h(z) \notin h(C)$, because the E and E^* relations are preserved by h. Indeed, by definition of *parent* , $z \notin C$ and z is the parent of nodes of C in $t(D)$. Hence if we assume $h(z) \in h(C)$, then h collapses z with one of its children in $t(D)$. This implies, by definition of h, that $z = x$, and hence C is contained in $\{x_1, \ldots, x_n\}$. This is a contradiction. Then $h(z)$ is the *parent* of $h(C)$.

From the three properties above, it follows that D' has a tree-shape.

root step. A *root step* is *applicable* on node $x \in adom_{node}(D)$ if x occurs in the *Root* relation, but x is not the root of $t(D)$.

If a *root step* is applicable on node x of D, it applies as follows:

- If there exists $y \in adom_{node}(D)$ such that $E(y, x)$ holds in D, the application of the step on x fails.
- If there exists no such y, then let $z \neq x$ be the node variable such that $E^*(z, x)$ holds in D (we know z exists since the step is applicable), and let $\beta\langle f \rangle\langle\langle f \rangle\rangle$ be the subtree of $t(D)$ whose root variable is z. Also let $C = \{x_1, \ldots, x_n\}$ be the connected component of the graph $G_{NS}(D)$ containing x. If there exist x_i, x_j in C such that $NS(x_i, x_j)$ holds, then the application of the step fails.
- Otherwise let $\beta_1, \ldots \beta_n$ be the descriptions of nodes of $t(D)$ with variables $x_1, \ldots x_n$, respectively. If $\beta_1, \ldots, \beta_n, \beta$ cannot be merged then the application of the step fails.
- Otherwise let $h = h_{\beta, \beta_1, \ldots, \beta_n}$. The application of the step results in the structure $D' = h(D)$.

The structure D' has still a tree-shape, using the same argument as in the previous case.

push-fc step and push-lc step. A *push-fc step* [a *push-lc step*, respectively] is *applicable* on node $x \in adom_{node}(D)$ if x occurs in the *FC* relation [*LC* relation, resp.], and x has an incoming edge [outgoing edge , resp.] in the graph $G_{NS}(D)$.

If a *push-fc step* [a *push-lc step*, respectively] is applicable on x, let (y, x) [(x, y) resp.] be an edge of $G_{NS}(D)$.

- if (y, x) [(x, y), resp.] is an *NS*-edge then the step fails.
- Otherwise $y \neq x$ and $NS^*(y, x)$ holds in D [$NS^*(x, y)$ resp.]. In this case, let β and β' be the node descriptions at nodes x and y respectively. If β and β' cannot be merged then the step fails.
- Otherwise let $h = h_{\beta, \beta'}$. The result of the application of the step is $D' = h(D)$.

We now prove that the structure D' has a tree-shape.

1) When replacing *NS* and NS^* relations of D' with empty ones, and removing possible tuples of the form (w, w) from E^*, we obtain the image of $t(D)$ by h. This is still the relational representation of an incomplete tree, since only sibling nodes x and y of $t(D)$ are collapsed by h, and their descriptions can be merged. Therefore the incomplete tree $t(D')$ is obtained from $t(D)$ by identifying the sibling nodes x and y.

2) $adom_{node}(D') = adom_{node}(t(D'))$ using the same argument as in the previous chase steps.

3) We now prove that each non-singleton connected component of $G_{NS}(D')$ has a *parent* in D'. Because h only collapses nodes belonging to the same connected component of $G_{NS}(D)$, each connected component of $G_{NS}(D')$ coincides with $h(C)$ for some connected component C of $G_{NS}(D)$. Thus let $h(C)$ be a non-singleton connected component of $G_{NS}(D')$, and let z be the *parent* of C in D. We now show that $h(z)$ is the *parent* of $h(C)$ in D'. Again one only needs to

prove that $h(z) \notin h(C)$. Indeed, by definition of *parent* , $z \notin C$, and z is not a sibling of nodes of C in $t(D)$. Therefore $h(z) \in h(C)$ would imply that h collapses two distinct non-sibling nodes in D. Since this is not the case, $h(z) \notin h(C)$. Therefore $h(z)$ is the *parent* of $h(C)$ in D'.

This completes the proof that D' has a tree-shape.

merge-fc step and merge-lc step. A *merge-fc step* [a *merge-lc step*, respectively] is *applicable* on nodes $x_1, x_2 \in adom_{node}(D)$, with $x_1 \neq x_2$, if both x_1 and x_2 occur in the FC relation [LC relation, resp.], both belong to the same connected component of $G_{NS}(D)$, and do not have incoming edges [outgoing edges, resp.] in $G_{NS}(D)$.

If a *merge-fc step* [a *merge-lc step*, respectively] is applicable on x_1 and x_2 in D, then let β_1 and β_2 be the descriptions of nodes of $t(D)$ having node variables x_1 and x_2 respectively.

- If β_1 and β_2 cannot be merged then the step fails.
- Otherwise let $h = h_{\beta_1,\beta_2}$. The result of the application of the step is $D' = h(D)$.

The structure D' has a tree-shape: the same argument as the previous chase step works, since x_1 and x_2 belong to the same connected component of $G_{NS}(D)$.

union-fc step and union-lc step. A *union-fc step* [a *union-lc step*, respectively] is *applicable* on nodes $x_1, x_2 \in adom_{node}(D)$, if both occur in the FC relation [LC relation, resp.] and:

- x_1 and x_2 occur in two distinct connected components of $G_{NS}(D)$ and
- for some node $y \in adom_{node}(D)$, both $E(y, x_1)$ and $E(y, x_2)$ hold in D.

If a *union-fc step* [a *union-lc step*, respectively] is applicable on x_1 and x_2 in D, then let β_1 and β_2 be the descriptions of nodes of $t(D)$ having node variables x_1 and x_2 respectively.

- If β_1 and β_2 cannot be merged then the step fails.
- Otherwise let $h = h_{\beta_1,\beta_2}$. The result of the application of the step is $D' = h(D)$.

We next show that the structure D' has a tree-shape. The first two properties of a tree-shaped structure are proved as in the case of *push-fc* and *push-lc* steps.

It remains to show that each non-singleton connected component of $G_{NS}(D')$ has a *parent* in D'. Let C_1 and C_2 be the connected components of nodes x_1 and x_2 in the graph $G_{NS}(D)$. Then connected components of $G_{NS}(D')$ are $\{h(C_1 \cup C_2)\} \cup \{h(C) | C \text{ is a connected component of } G_{NS}(D) \wedge C \neq C_1 \wedge C \neq C_2\}$. Now consider a non-singleton connected component of $G_{NS}(D')$. It must be of the form $h(S)$ where S is either a connected component C of $G_{NS}(D)$ or $C_1 \cup C_2$. In the first case let z be the *parent* of C in D, in the second case let z be the *parent* of C_1 and C_2 in D (we know that C_1 and C_2 have the same *parent* thanks to the applicability of the *union-fc* or *union-lc* step). We prove that $h(z)$ is the *parent* of $h(S)$ in D'. Again one only needs to prove that $h(z) \notin h(S)$. Indeed, by definition of *parent* , $z \notin S$ and z is not a sibling of nodes of S in $t(D)$. hence $h(z) \in h(S)$ would imply that h collapses two distinct non-sibling nodes of $t(D)$. Since this is not the case, we must have $h(z) \notin h(S)$. Then $h(z)$ is the *parent* of $h(S)$.

It follows that D' has a tree-shape.

fc/lc step. An *fc/lc step* is *applicable* on node $x \in adom_{node}(D)$, if

- x occurs both in the FC and the LC relation and

- there exists $x' \in adom_{node}(D)$ with $x' \neq x$ and some node $y \in adom_{node}(D)$ such that both $E(y, x)$ and $E(y, x')$ hold in D.

If an *fc/lc step* is applicable on x in D, let y be the node such that $E(x, y)$ holds in D. The subtree of $t(D)$ whose root variable is y will be of the general form $\beta\langle f\rangle\langle\langle f'\rangle\rangle$. Let $x_1, \ldots x_n$ be the root variables of the forest f; this set contains x and some other node $x' \neq x$.

- If there exist x_i, x_j with $1 \leq i, j \leq n$ such that $NS(x_i, x_j)$ holds in D, then the application of the step fails.
- Otherwise let $\beta_1, \ldots \beta_n$ be the node descriptions of roots of f with node variables $x_1, \ldots, x_n$ respectively. If $\beta_1, \ldots, \beta_n$ cannot be merged then the application of the step fails.
- Otherwise let $h = h_{\beta_1,\ldots,\beta_n}$. The application of the step results in $D' = h(D)$.

If the step succeeds, D' can be shown to preserve a tree-shape using a similar argument as in the case of *union-fc* and *union-lc* steps.

in-sibling step and out-sibling step. An *in-sibling step* [*out-sibling step*, respectively] is *applicable* on node $x \in adom_{node}(D)$ if x has two distinct incoming [outgoing, resp.] NS-edges in $G_{NS}(D)$ and x is not in the FC relation [LC relation, resp.] of D.

If an *in-sibling step* [*out-sibling step*, resp.] is applicable on node x in D, let y_1, y_2 be two distinct nodes of $adom_{node}(D)$ such that $NS(y_1, x)$ and $NS(y_2, x)$ [$NS(x, y_1)$ and $NS(x, y_2)$, resp.] hold in D. Let also β_1 and β_2 the node descriptions having node variables y_1 and y_2, respectively.

- If β_1 and β_2 cannot be merged, the application of the step fails.
- Otherwise let $h = h_{\beta_1,\beta_2}$. The application of the step gives $D' = h(D)$.

If the step succeeds, we can show that D' has a tree-shape exactly as in the case of a *push-fc* step (in fact, as in the case of a *push-fc* step, y_1 and y_2 belong to the same connected component).

root-child step. A *root-child* step is applicable on node $x \in adom_{node}(D)$ if x occurs both in the *Root* relation and in one of the child marking relations (i.e., either the FC or LC relation).

If a *root-child* step is applicable then the application of the step always fails.

This completes the definition of the chase steps. In the sequel we will say that a chase step is applicable if one of the above steps is applicable on some node.

We now define a chase sequence.

Definition 5.8. *A* chase sequence *for the incomplete tree t is a sequence of tree-shaped structures*

$$D_0 D_1 \ldots D_i \ldots$$

such that $D_0 = rel(t)$ and each D_j in the sequence, with $j > 0$, results from the successful application of some chase step to D_{j-1}.

Then we prove some properties of chase sequences that will be used in the sequel.

Lemma 5.9. *Given an incomplete tree t, every chase sequence for t is finite. Moreover if $D_0, \ldots D_k$ is a chase sequence for t, then $k < |adom_{node}(t)|$.*

Proof. Assume $D_0 D_1 \dots D_i \dots$ is a chase sequence for t. Each element D_j of the sequence, with $j > 0$, is obtained from D_{j-1} by successful application of some chase step. Thus $D_j = h(D_{j-1})$, where h is the mapping applied in the chase step. By the definitions of the chase steps, h is such that there exist at least two distinct node variables x and y in $adom_{node}(D_{j-1})$ with $h(x) = h(y)$. Hence $|adom_{node}(D_j)| < |adom_{node}(D_{j-1})|$. As a consequence, if $|adom_{node}(D_0)| = n$, then each structure D_j in the chase sequence has $|adom_{node}(D_j)| \leq n - j$. It follows that, for each structure D_j in the chase sequence, $j \leq n - 1$. Then the length of the sequence has an upper bound $n = |adom_{node}(t)|$. □

Definition 5.10. *A* valid *chase sequence for t is a chase sequence $D_0, \dots D_k$ for t such that in D_k:*

1. *either no chase step is applicable,*
2. *or there exists an applicable chase step that fails.*

In the first case the valid sequence is called successful*, and in the second case it is called* failing*.*

Lemma 5.11. *For each incomplete tree t a valid chase sequence for t can be computed in polynomial time in the size of t.*

Proof. Given a chase sequence $D_0 \dots D_i$, a chase sequence $D_0 \dots D_{i+1}$ (if it exists) can be computed in polynomial time in the size of D_0. In fact one needs to look for an applicable step in D_i and, if it exists and its application does not fail, find the associated mapping h and compute $h(D_i)$.

Checking whether there exists an applicable step in D_i only requires to perform a constant number of joins of relations in D_i and therefore can be done in polynomial time in the size of D_i. If no applicable step is found, one can conclude that $D_0 \dots D_i$ is a successful chase sequence. If an applicable step has been found, checking whether the application of the step succeeds requires (for all types of chase step):

- computing the connected components of $G_{NS}(D_i)$;
- at most a linear scan of relations NS and E of D_i to look for possible edges that make the step fail.
- checking for the existence of a merging mapping of a set of node descriptions $\beta_1, \dots \beta_n$ where n is bounded by $|D_i|$.

All these tasks can be performed in polynomial time in the size of D_i. In fact checking the existence of a merging mapping of $\beta_1 \dots \beta_n$ requires to solve a system of $O(|D_i|^2)$ equalities of the form $z = z'$ with $z, z' \in \mathcal{V}_{\text{attr}} \cup \mathcal{D}$. As already observed, if the successive replacement of variables in this system succeeds, it results in a minimal merging mapping of $\beta_1, \dots \beta_n$, and therefore it computes the mapping h.

If the application of the step fails, one can conclude that $D_0 \dots D_i$ is a failing chase sequence. Otherwise one can compute $D_{i+1} = h(D_i)$ in polynomial time in the size of D_i.

Now since D_i is a homomorphic image of D_0, the size of D_i is bounded by the size of D_0. As a consequence there exists a fixed polynomial p and a procedure that, given a chase sequence $\sigma = D_0 \dots D_i$, in time $O(p(|D_0|))$ either concludes that σ is valid or computes an augmented chase sequence $D_0 \dots D_i D_{i+1}$.

A valid chase sequence for t can be computed by first computing the initial chase sequence $\sigma = \underline{rel}(t)$ and then repeatedly applying the above procedure to augment σ. After at most $|adom_{node}(t)|$ steps, σ has to be recognized valid, otherwise the above procedure would compute a chase sequence of length greater than $|adom_{node}(t)|$.

We conclude that a valid chase sequence for t can be computed in time $O(|adom_{node}(t)| \times p(|rel(t)|))$, that is, in polynomial time in the size of t. □

The following lemma proves that structures forming a chase sequence are equivalent over trees:

Lemma 5.12. *If $D_0, \dots D_k$ is a chase sequence for the incomplete tree t then for each $0 < i \leq k$, and for each complete tree T, there exists a homomorphism $h_i : D_i \to T$ if and only if there exists a homomorphism $h_{i-1} : D_{i-1} \to T$.*

Proof. Given a complete tree T and an index $0 < i \leq k$, let h be the homomorphism from D_{i-1} to D_i. If there exists a homomorphism $h_i : D_i \to T$, then clearly $h_i \circ h$ is a homomorphism from D_{i-1} to T.

Conversely assume that there exists a homomorphism $h_{i-1} : D_{i-1} \to T$. Since h is the application of some successful chase step on D_{i-1}, then $h = (h_{node}, h_{null})$ where:

- h_{null} is a minimal merging mapping for some set of node descriptions $\beta_0 \dots \beta_n$ in $t(D_{i-1})$ (depending on the type of chase step) and
- h_{node} maps node variables of $\beta_0 \dots \beta_n$ into one of them, and is the identity in any other element of $\mathcal{V}_{\text{node}} \cup \mathcal{I}$.

Claim 5.13. *If we let $h_{i-1} = (h_{i-1}^{node}, h_{i-1}^{null})$, then:*

1. *h_{i-1}^{null} is a merging mapping for $\beta_0 \dots \beta_n$ and*
2. *for each $i, j \in [0, n]$, if x_i and x_j are node variables of β_i and β_j resp., then $h_{i-1}^{node}(x_i) = h_{i-1}^{node}(x_j)$.*

Proof of the claim We show here the case that a *leaf step* is applied from D_{i-1} to D_i , but a similar argument holds for all other types of chase steps.

In the case of a leaf step, the merged node descriptions $\beta_0, \dots, \beta_n$ have node variables $x_0, x_1, \dots x_n \in adom_{node}(D_{i-1})$, respectively, where:

- the subtree description of $t(D_{i-1})$ with root variable x_0 is $\beta_0 \langle\!\langle f \rangle\!\rangle$,
- root node descriptions of f are $\beta_1, \dots \beta_n$ and
- x_0 belongs to the *Leaf* relation in D_{i-1}.

This implies (using the fact that h_{i-1} is a homomorphism) that $h_{i-1}(x_0)$ is a leaf node of T. Moreover, for each $j \in [1, n]$, the fact that $E^*(x_0, x_j)$ holds in D_{i-1} implies $E^*(h_{i-1}(x_0), h_{i-1}(x_j))$ in T. Therefore, since $h_{i-1}(x_0)$ is a leaf of T, we have $h_{i-1}(x_j) = h_{i-1}(x_0)$. This proves 2.

Now take β_k and β_l for some arbitrary $k, l \in [0, n]$. If an attribute formula $@a = z$ occurs in β_k and $@a = z'$ occurs in β_l, then the tree T must contain tuples $A_{@a}(h_{i-1}(x_k), h_{i-1}(z))$ and $A_{@a}(h_{i-1}(x_l), h_{i-1}(z'))$. Since $h_{i-1}(x_k) = h_{i-1}(x_l)$ and relation $A_{@a}$ codes a function (that is, it associates at most one attribute value to each node), $h_{i-1}(z) = h_{i-1}(z')$. This proves 1.

The easy extension of this argument to all chase steps concludes the proof of the claim. □

The claim implies that h_{i-1} can be rewritten as follows:

- Because h_{i-1}^{null} is a merging mapping for $\beta_0, \dots \beta_n$ and h_{null} is a minimal merging mapping for $\beta_0, \dots \beta_n$, we have that $h_{i-1}^{null} = h'_{null} \circ h_{null}$ for some mapping $h'_{null} : \mathcal{V}_{\text{attr}} \cup \mathcal{D} \to \mathcal{V}_{\text{attr}} \cup \mathcal{D}$ preserving constants.
- By definition of h_{node} and by point 2 of the claim, we can rewrite $h_{i-1}^{node} = h_{i-1}^{node} \circ h_{node}$.

It follows that $h_{i-1} = h' \circ h$ where $h' = (h_{i-1}^{node}, h'_{null})$. Hence the fact that h_{i-1} is a homomorphism from D_{i-1} to T (i.e. $h_{i-1}(D_{i-1}) \subseteq T$) implies $h'(h(D_{i-1})) \subseteq T$, and thus $h'(D_i) \subseteq T$. Then h' is a homomorphism from D_i to T. This concludes the proof of Lemma 5.12. □

If $\sigma = D_0, \ldots, D_k$ is a valid chase sequence for the incomplete tree t, then the structure D_k will be denoted as $chase_\sigma(t)$. As a corollary of Lemma 5.12 and Proposition 4.2, a complete tree T is in $Rep(t)$ if and only if there exists a homomorphism from $chase_\sigma(t)$ to T. Therefore the following corollary:

Corollary 5.14. *Given an incomplete tree t, and a valid chase sequence σ for it, t is consistent if and only if there exists a complete tree T and a homomorphism from $chase_\sigma(t)$ to T.*

We are now ready to characterize consistency of incomplete trees using the chase.

Lemma 5.15. *Given an incomplete tree t and a valid chase sequence σ for t, if t is consistent then σ is successful.*

Proof. We prove that if t is consistent, every applicable step in $chase_\sigma(t)$ succeeds, thus σ cannot be failing.

Assume that t is consistent and there is an applicable leaf step in $chase_\sigma(t)$, we prove that the application of the step must succeed.

Let $D = chase_\sigma(t)$. Since t is consistent, by Corollary 5.14 there exists a complete tree T and a homomorphism $h : D \to T$. On the other hand there exists an applicable leaf step in D, therefore there exists a node $x \in adom_{node}(D)$ which occurs in the *Leaf* relation and is not a leaf of $t(D)$. Since h is a homomorphism, $h(x)$ is a leaf node of T, then the following holds:

a. There does not exist $y \in adom_{node}(D)$ such that $E(x, y)$ holds in D, otherwise we would have $E(h(x), h(y))$ in T, therefore $h(x)$ would not be a leaf. Then the subtree of $t(D)$ whose root variable is x is of the form $\beta\langle\langle f \rangle\rangle$ with f not empty. Let $x_1, \ldots x_n$ be the root variables of the forest f.

b. $E^*(h(x), h(x_i))$ holds in T, for each $i \in [1, n]$; but $h(x)$ is a leaf of T, thus $h(x_i) = h(x)$. Hence there cannot exist nodes x_i, x_j with $1 \leq i, j \leq n$ such that $NS(x_i, x_j)$ holds in D, otherwise $NS(h(x), h(x))$ would hold in T.

c. Let $\beta_1, \ldots \beta_n$ be the root node descriptions of f with node variables $x_1, \ldots, x_n$ respectively. We now show that it must be possible to merge $\beta_1, \ldots, \beta_n, \beta$.

   Indeed assume that $l \in Labels$ is the label of $h(x)$, then $h(x) = h(x_1) = \cdots = h(x_n)$ occur in relation P_l of T. Thus if a node variable in $\{x, x_1, \ldots x_n\}$ occurs in a labeling relation of D, this relation must be P_l. As a consequence no two node descriptions in $\{\beta, \beta_1, \ldots \beta_n\}$ have distinct labels in *Labels*. Moreover h itself is a merging mapping of $\beta, \beta_1, \ldots, \beta_n$. Then $\beta, \beta_1, \ldots, \beta_n$ can be merged according to Def. 5.7.

Items a., b. and c. above prove that the application of the leaf step on x succeeds. Similarly one also proves that all other types of chase steps applicable in $chase_\sigma(t)$ succeed. In particular we have to prove that no *root-child* step is applicable. In fact if there is an applicable *root-child* step then there exists a node $x \in adom_{node}(D)$ occurring both in the *Root* relation and the FC (or LC) relation of D. On the other hand, as in the previous case, there exists a homomorphism $h : D \to T$ for some complete tree T. Then $h(x)$ is both the root of T and a first child (or last child) of some node of T, which is a contradiction. We omit the detailed proof for the other chase steps, since it follows the same lines, and conclude the proof of the lemma. □

We show next that the converse of Lemma 5.15 holds for all fragments of incomplete trees containing neither $(\downarrow, \rightarrow, \|, fc, lc)$ nor $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$ nor $(\downarrow, \rightarrow, \rightarrow^*, fc, lc)$ nor $(\downarrow, \downarrow^*, \rightarrow^*, fc, lc, leaf)$.

The idea of the proof is as follows. Intuitively the chase enforces constraints on the nodes of t due to the presence of markings (for instance if a node is marked as fc and has a preceding sibling then

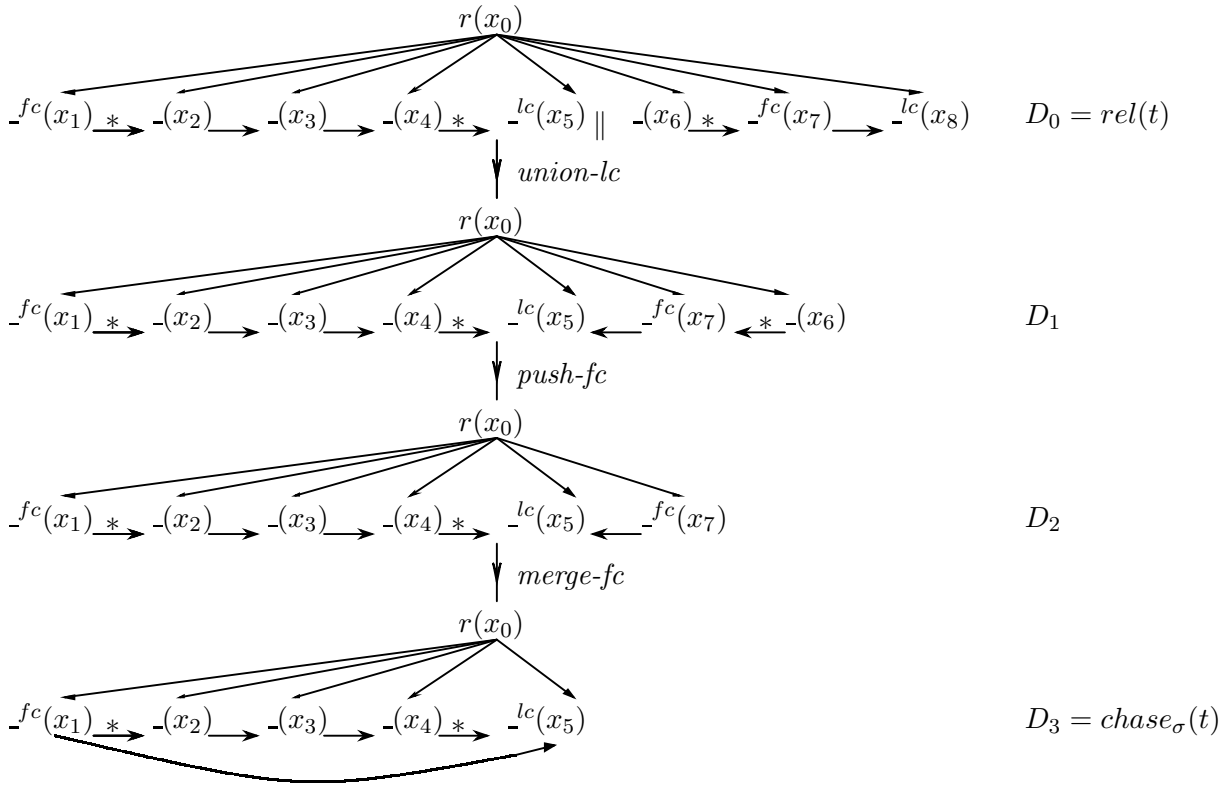


Figure 4: A successful chase sequence for t

the two nodes must coincide). The chase also enforces some other constraints, imposed by the fact that t must represent trees (for instance if, after collapsing some nodes, a node has two distinct next siblings, they must coincide). If σ is a successful chase sequence for t, then $chase_\sigma(t)$ satisfies all these constraints. Intuitively this means that in $chase_\sigma(t)$ all markings are in the right place, each node can have at most one next sibling, and can be the next sibling of at most one other node. Nevertheless this does not ensure that $chase_\sigma(t)$ represents some tree. To see why consider the following example.

Figure 4 shows a successful chase sequence for an incomplete tree t. First a *union-lc* step collapses nodes x_8 and x_5, which occur in two distinct connected components of $G_{NS}(D_0)$ and are both marked as lc. Then a *push-fc* step is applicable in D_1, since node x_7 is marked as fc but has an incoming edge in $G_{NS}(D_1)$. The application of this step results in D_2 where a *merge-fc* step is applicable, and collapses nodes x_1 and x_7. In the resulting tree-shaped structure D_3 no chase step is applicable. Clearly D_3 does not represent any tree.

Remark that in this example t contains $(\downarrow, \rightarrow, \|, fc, lc)$. We next show that this situation cannot occur if t contains neither $(\downarrow, \rightarrow, \|, fc, lc)$ nor $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$ nor $(\downarrow, \rightarrow, \rightarrow^*, fc, lc)$ nor $(\downarrow, \downarrow^*, \rightarrow^*, fc, lc, leaf)$. To prove this we show that, given the restricted fragment of t, in any successful chase sequence of t the graphs $G_{NS}(D_i)$ satisfy some properties with the fc-marked and lc-marked nodes. These properties rule out cases such as the one presented in the previous example. This will show that for any successful chase sequence σ, the structure $chase_\sigma(t)$ (and therefore t) is consistent.

The properties of $G_{NS}(D_i)$ that we will be interested in (regardless of the fragment of incomplete trees we consider) are listed next.

For a tree-shaped structure D we consider the following eight properties:

1. : If a node $x \in adom_{node}(D)$ has two distinct incoming edges in $G_{NS}(D)$, then x is either in FC

or LC.

2. : If a node $x \in adom_{node}(D)$ has two distinct outgoing edges in $G_{NS}(D)$, then x is either in FC or LC.

3. : If C is a directed cycle of $G_{NS}(D)$ then C contains a node x belonging to either FC or LC relation.

4. : Each connected component of $G_{NS}(D)$ is a *simple* directed path of NS-edges (where *simple* means never going through the same node) and there exist no two distinct connected components of $G_{NS}(D)$ having the same *parent* .

5. : If a node $x \in adom_{node}(D)$ has two distinct incoming edges in $G_{NS}(D)$, then $x \in FC$.

6. : If a node $x \in adom_{node}(D)$ has two distinct outgoing edges in $G_{NS}(D)$, then $x \in LC$.

7. : If C is a directed cycle of $G_{NS}(D)$ then C contains a node $x \in FC$.

8. : If C is a directed cycle of $G_{NS}(D)$ then C contains a node $x \in LC$.

For each of the above properties $\mathcal{P}$ we say that $\mathcal{P}$ is *preserved* by a class S of chase steps if the following holds:

If D is a tree-shaped structure satisfying $\mathcal{P}$ and there exists an applicable step of class $\mathcal{S}$ in D whose application is successful and results in D', then also D' satisfies $\mathcal{P}$.

Claim 5.16.
- *Properties from 1 to 3 are preserved by every type of chase steps different from* in-sibling *and* out-sibling.
- *Property 4 is preserved by all types of chase steps.*
- *Property 5 is preserved by every type of chase steps different from* merge-lc*,* union-lc.
- *Property 6 is preserved by every type of chase steps different from* merge-fc *and* union-fc.
- *Property 7 is preserved by every type of chase steps different from* merge-lc*,* push-lc*,* in-sibling *and* out-sibling.
- *Property 8 is preserved by every type of chase steps different from* merge-fc*,* push-fc*,* in-sibling *and* out-sibling.

The proof of the above lemma is a routine case-analysis on the different step types and is omitted. Also some conjunctions of properties are preserved by chase steps, as shown by the following claim.

Claim 5.17.
- *The conjunction of property 5 and property 7 is preserved by every type of chase steps different from* merge-lc*,* union-lc *and* push-lc.
- *The conjunction of property 6 and property 8 is preserved by every type of chase steps different from* merge-fc*,* union-fc *and* push-fc.

Proof. By Claim 5.16, every step of type different from *merge-lc*, *union-lc*, *push-lc* and *in-sibling* and *out-sibling* preserves the conjunction of properties 5 and 7. We need to prove that also *in-sibling* and *out-sibling* steps preserve the conjunction of properties 5 and 7. Given a databases D with tree shape, if D satisfies the conjunction of properties 5 and 7, then no *in-sibling* step is applicable in D (because of property 5). Therefore the conjunction is trivially preserved by *in-sibling* steps.

Now assume that D satisfies the conjunction of properties 5 and 7, and D' results from the application of some *out-sibling* step to D, then by Claim 5.16, D' satisfies property 5. We need to show that it also satisfies property 7. Since *out-sibling* is applicable, we know that there are nodes $x, y_1, y_2 \in adom_{node}(D)$ such that $NS(x, y_1)$ and $NS(x, y_2)$ hold. Moreover $D' = h(D)$ for some homomorphism h which is the identity on all $adom_{node}(D)$ except y_2, and $h(y_2) = y_1$.

Now assume there is a directed cycle c' in $G_{NS}(D')$; then c' is a sequence of edges $e'_1 \dots e'_k$ of $G_{NS}(D')$. Furthermore, each edge $e'_i = h(e_i)$ for some edge e_i of $G_{NS}(D)$. Then there are two cases. In the first case, $e_1 \dots e_k$ contains a directed cycle of $G_{NS}(D)$, then there exists a node z traversed by $e_1 \dots e_k$ which is in relation FC of D. Hence $h(z)$ is a node of c' belonging to relation FC of D'. In the case that $e_1 \dots e_k$ does not contain a directed cycle of $G_{NS}(D)$, it easy to check that it must contain a directed path p either from y_1 to y_2 or from y_2 to y_1 (this is because h merges y_1 and y_2).

Assume p is the sequence of vertices $y_1 z_1 \dots z_k y_2$. There are two cases:

1. If $z_k = x$, then $p' = y_1 z_1 \dots z_k y_1$ is a directed cycle in $G_{NS}(D)$. Therefore it must contain a node in the FC relation of D. Consequently also $h(p)$ (because it coincides with $h(p')$) contains a node in the FC relation of D'.

2. Otherwise $z_k \neq x$. Therefore in $G_{NS}(D)$ there are two distinct incoming edges in node y_2. Hence, by property 5, the node y_2 must be in relation FC of D. Then also in this case $h(p)$ must go through a node occurring in relation FC of D'.

In both cases $h(p)$ is contained in c', therefore c' traverses a node in the FC relation of D'. The case that p goes from y_2 to y_1 is symmetric. This proves property 7 for D'.

The proof for the conjunction of property 6 and property 8 is dual. This proves the claim. □

In the sequel we will also make use of the following lemma whose proof is straightforward from Proposition 4.2 and the definition of tree-shaped structure:

Lemma 5.18. *If D is a tree-shaped structure, T a complete tree, and there exists a valuation ν of $adom_{node}(D)$ and variables from $adom_{attr}(D)$ such that:*

- *$(T, \nu, s) \models t(D)$ for some node s of T and*
- *for each NS-edge (x, y) of $G_{NS}(D)$, we have $NS(\nu(x), \nu(y))$ in T, and*
- *for each NS*-edge (x, y) of $G_{NS}(D)$, we have $NS^*(\nu(x), \nu(y))$ in T.*

then there exists a homomorphism from D to T.

We are now ready to study the properties of the chase in individual fragments of incomplete trees. As an example, we show the proof for incomplete trees without *lc* markings. Other cases are shown in the appendix.

For incomplete trees without *lc* markings the converse of Lemma 5.15 holds:

Lemma 5.19. *Given an* str*-incomplete tree t, where* str *does not contain* lc*, and given a valid chase sequence σ for t, if σ is successful, then t is consistent.*

Proof. Let $\sigma = D_0, \dots D_k$, where $D_0 = rel(t)$ and $D_k = chase_\sigma(t)$. For each $i \in [0, k]$ the relation LC is empty in D_i. Therefore neither *merge-lc* nor *union-lc* nor *push-lc* steps are applicable in D_i, for all $i \in [0, k]$.

Moreover D_0 satisfies properties 5 and 7. In fact D_0 is the relational representation of an incomplete tree, therefore connected components of the graph $G_{NS}(D_0)$ are simple paths. Thus no node in $adom_{node}(D_0)$ can have two distinct incoming edges or two distinct outgoing edges in $G_{NS}(D_0)$. Therefore by Claim 5.17, each D_i in the chase sequence, satisfies properties 5 and 7. Furthermore in D_k no chase step is applicable. This implies:

- By property 5 and the fact that no *push-fc* step is applicable, each node of $G_{NS}(D_k)$ has at most one incoming edge. In particular, if a node x of $G_{NS}(D_k)$ is in relation FC of D_k, then x has no incoming edges in $G_{NS}(D_k)$.
- By property 7 there are no directed cycles in $G_{NS}(D_k)$.
- By the fact that no *out-sibling* step is applicable in D_k, each vertex of $G_{NS}(D_k)$ has at most one outgoing NS-edge.
- By the fact that no *union-fc* step is applicable in D_k, for each $x \in adom_{node}(D_k)$ there exists at most one connected component of $G_{NS}(D_k)$ containing a node in FC and having x as *E-parent* .

By the first two items above, we conclude that each connected component of $G_{NS}(D_k)$ is a directed tree (with edges departing from the root). Moreover this directed tree has the following properties:

- each vertex of the directed tree has at most one outgoing NS-edge;
- only the root of the directed tree is possibly in the FC relation of D_k.

Also $t(D_k)$ has the following properties:

- By the fact that no *root* step is applicable in D_k, only the root variable of $t(D_k)$ is possibly in the *Root* relation of D_k.
- By the fact that no *leaf* step is applicable in D_k, only leaf variables of $t(D_k)$ (that is, node variables of subtrees $\beta\langle\varepsilon\rangle\langle\langle\varepsilon\rangle\rangle$ of $t(D_k)$) can be in the *Leaf* relation.
- By the fact that no *root-child* step is applicable in D_k, no node is both in the *Root* and the FC relation of D_k.

These properties of D_k allow us to construct a complete tree T having a homomorphism from D_k as follows.

We choose an arbitrary mapping $h_{null} : \mathcal{V}_{\text{attr}} \to \mathcal{D}$ and let h_0 be a mapping coinciding with h_{null} on $\mathcal{V}_{\text{attr}}$ and with the identity on $\mathcal{I}$, $\mathcal{V}_{\text{node}}$ and $\mathcal{D}$. We then let $D = h_0(D_k)$. Clearly D has still a tree shape: $t(D)$ can be obtained from $t(D_k)$ by applying h_0 on its variables, and $G_{NS}(D) = G_{NS}(D_k)$. Moreover also D satisfies the same properties as D_k listed above.

For an incomplete tree t, we will let t^* be the incomplete tree obtained from t by removing possible fc and lc markings from the root of t.

For each subtree t' of $t(D)$ we show how to construct a tree T and a mapping $\nu : adom_{node}(t') \to \mathcal{I}$, sending the root node variable of t' into the root s of T and satisfying:

- $(T, \nu, s) \models t'^*$
- for each $x, y \in adom_{node}(t')$, if (x, y) is an NS-edge (resp., NS^*-edge) of $G_{NS}(D)$, then $NS(\nu(x), \nu(y))$ (resp., $NS^*(\nu(x), \nu(y))$) holds in T

We proceed by induction on the structure of t'. Recall that $t(D)$ and therefore t' has empty NS and NS^* relations.

If $t' = \ell^{\mu}(x)[@a_1 = v_1, \ldots, @a_m = v_m]\langle\varepsilon\rangle\langle\langle\varepsilon\rangle\rangle$ then we construct the tree $T = B\langle\varepsilon\rangle$, where $B = \bar{\ell}(\mathsf{i})[@a_1 = v_1, \ldots, @a_m = v_m]$, the id i is arbitrarily chosen from $\mathcal{I}$ and $\bar{\ell} = \ell$ if $\ell \in$ *Labels*, otherwise ℓ is an arbitrary label of *Labels*. Clearly the valuation ν mapping x into i is such that $(T, \nu, \mathsf{i}) \models t'^*$, and preserves edge relations of $G_{NS}(D)$ (because $adom_{node}(t')$ contains only one node).

Now assume $t' = \beta\langle t_1 \| \ldots \| t_n\rangle\langle\langle t_{n+1} \| \ldots \| t_m\rangle\rangle$, where $\beta = \ell^{\mu}(x)[@a_1 = v_1, \ldots, @a_p = v_p]$. Assume also that $x_1, \ldots x_m$ are the root node variables of $t_1, \ldots t_m$ respectively.

Assume we have constructed

- trees T_i with root ids i_i, for all $i \in [1, m]$
- valuations $\nu_i : adom_{node}(t_i) \to \mathcal{I}$ preserving edges of $G_{NS}(D)$ such that $(T_i, \nu_i, \mathsf{i}_i) \models t_i^*$ for each $i \in [1, m]$.

We now construct a tree T from subtrees $T_1, \dots T_m$ as follows. We know each connected component of $G_{NS}(D)$ is a directed tree. Assume w.l.o.g that this tree is ordered and that if there exists and *NS*-edge (x', y') in a connected component, then y' is the left most child of x' in the directed tree.

Now let $C_1, \dots C_l$ be all connected components of $G_{NS}(D)$ having *E-parent* x (components $C_1, \dots C_l$ partition $\{x_i | i \in [1, n]\}$). Similarly let $C_{l+1}, \dots C_k$ be all connected components of $G_{NS}(D)$ having *E*-parent* x (components $C_{l+1}, \dots C_k$ partition $\{x_i | i \in [n+1, m]\}$). Assume w.l.o.g. that C_1 is the (only) connected component of $G_{NS}(D)$ having *E-parent* x and containing a node in *FC* (if such component exists).

For each component $C \in \{C_1, \dots C_k\}$, let $\bar{C}$ be a permutation of vertices in C corresponding to a prefix left-to-right depth-first traversal of the directed tree connecting C. If $\bar{C} = x_{j_1} x_{j_2} \dots x_{j_l}$ with $j_1, \dots, j_l$ in $[1, m]$, we let $f_{\bar{C}}$ be the forest $T_{j_1} T_{j_2} \dots T_{j_l}$.

For each connected component $C \in \{C_{l+1} \dots C_k\}$ we construct a tree T_C having a new fresh root id i_C and an arbitrary root label $d \in$ *Labels*, defined as $T_C = d(\mathsf{i}_C)\langle f_{\bar{C}} \rangle$. Then we construct $T = B\langle f_{\bar{C}_1} \cdots f_{\bar{C}_l} T_{C_{l+1}} \cdots T_{C_k} \rangle$ where the node description B is constructed from β as in the base case, but its id i is chosen so as to be distinct from all other ids in T.

The valuation ν sending x into i and coinciding with ν_i on $adom_{node}(t_i)$ preserves edges of $G_{NS}(D)$. In fact for each *NS*-edge (*NS**-edge, resp.) e of $G_{NS}(D)$, where nodes of e are in $adom_{node}(t_i)$ for some $1 \leq i \leq m$, we have $NS(\nu_i(e))$ (resp. $NS^*(\nu_i(e))$) in T_i, by induction hypothesis. Otherwise e is an edge in C_j for some $1 \leq j \leq k$, then $e = (x_p, x_q)$ for some $1 \leq p, q \leq m$. It is easy to verify, by construction of $f_{\bar{C}_j}$, that for each *NS*-edge (*NS**-edge, resp.) (x_p, x_q) of C_j, we have $NS(\mathsf{i}_p, \mathsf{i}_q)$ (resp. $NS^*(\mathsf{i}_p, \mathsf{i}_q)$) in T.

It remains to verify that $(T, \nu, \mathsf{i}) \models t'^*$. For each incomplete tree t_i, with $1 \leq i \leq m$, no node of t_i is marked as *root* (because x_i is not the root of t). Then the fact that $(T_i, \nu_i, \mathsf{i}_i) \models t_i^*$ implies that also $(T, \nu, \mathsf{i}_i) \models t_i^*$. Moreover if the root node description of t_i does not contain *fc* markings, then also $(T, \nu, \mathsf{i}_i) \models t_i$ holds. If instead the root node description of t_i contains a *fc* marking and x_i belongs to a connected component $C \in \{C_1, \dots C_k\}$, then x_i is the root of the directed tree connecting C, therefore T_i is the left-most subtree in $f_{\bar{C}}$. This implies, by construction of T that:

- if $C \in \{C_{l+1}, \dots C_k\}$ then i_i is the first child of node i_C;
- otherwise C must be C_1 and x_i must be the root node of C_1; hence, by construction of T, node i_i is the first child of node i.

Then also in this case $(T, \nu, \mathsf{i}_i) \models t_i$. Moreover $(T, \nu, \mathsf{i}) \models \beta^*$ by construction of B and thanks to the fact that the root node description of t'^* cannot contain neither *fc* nor *leaf* markings. Finally, by construction of T, nodes $\mathsf{i}_1 \dots \mathsf{i}_n$ are children of i and $\mathsf{i}_{n+1} \dots \mathsf{i}_m$ are descendants of i. On the whole this implies that $(T, \nu, \mathsf{i}) \models t'^*$. This completes the induction.

So we have proved that there exists a tree T and a mapping $\nu : adom_{node}(t(D)) \to \mathcal{I}$, sending the root node variable of $t(D)$ into the root i of T and preserving edges of $G_{NS}(D)$ such that $(T, \nu, \mathsf{i}) \models t(D)^*$

Now there are two cases; if the root of $t(D)$ is not marked with *fc* then $t(D) = t(D)^*$ and therefore $(T, \nu, \mathsf{i}) \models t(D)$. If on the contrary the root of $t(D)$ is marked with *fc*, then it cannot be marked as *root*. In this case we modify T by adding an extra root having i as the only child. Then we have $(T, \nu, \mathsf{i}) \models t(D)$.

In both cases we have constructed a tree T and a mapping ν such that:

- $(T, \nu, \mathrm{i}) \models t(D)$;
- for each $x, y \in adom_{node}(t(D))$, if (x, y) is an NS-edge (resp., NS^*-edge) of $G_{NS}(D)$, then $NS(\nu(x), \nu(y))$ (resp., $NS^*(\nu(x), \nu(y))$) holds in T.

We conclude using Lemma 5.18 that there exists a homomorphism h from D to T. Thus $h \circ h_0$ is a homomorphism from D_k (that is, $chase_\sigma(t)$) to T.

Corollary 5.14 then implies that t is consistent and concludes the proof of Lemma 5.19. □

The converse of Lemma 5.15 holds for all other fragments of incomplete trees containing neither $(\downarrow, \rightarrow, \|, fc, lc)$ nor $(\downarrow, \rightarrow, \rightarrow^*, fc, lc)$, nor $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$ nor $(\downarrow, \downarrow^*, \rightarrow^*, fc, lc, leaf)$. The remaining cases are shown in the appendix.

To conclude, in all the above fragments, consistency of an incomplete tree t can be checked by the following procedure, in polynomial time in the size of t:

- compute a valid chase sequence for t (according to Lemma 5.11);
- if the chase sequence is failing, conclude that t is not consistent (Lemma 5.15);
- otherwise, if the chase sequence is successful, conclude that t is consistent (Lemmas 5.19, A.1, A.2 and A.3).

This concludes the proof of Theorem 5.4. □

5.2.2 Consistency with DTDs

Next, we look at consistency in the presence of schema information, given by DTDs. Then we have intractability already for simple descriptions of incomplete trees.

Theorem 5.20. *There exist DTDs d_1, d_2, d_3 such that:*

- CONSISTENCY(d_1) *is* NP*-complete for $(\downarrow, \|)$-incomplete trees.*
- CONSISTENCY(d_2) *is* NP*-complete for $(\downarrow, \rightarrow, \|)$-incomplete trees, even without attributes.*
- CONSISTENCY(d_3) *is* NP*-complete for $(\downarrow, \downarrow^*, \|)$-incomplete trees, even without attributes.*

Proof. We show the reduction for the case of $(\downarrow, \|)$-incomplete trees. The other two cases are in the appendix.

We reduce the 3-coloring problem to CONSISTENCY(d_1) where d_1 is the following DTD:

$$\begin{aligned} R &\to CCC \\ C &\to DD \\ D &\to \varepsilon \end{aligned}$$

where labels C and D have an attribute *color*.

Let $G = \langle V, E \rangle$ be a graph, where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots e_m\}$. We now give a $(\downarrow, \|)$-incomplete tree t such that $Rep_{d_1}(t) \neq \emptyset$ if and only if G is 3-colorable:

$$t = R\langle t_r \| t_g \| t_b \| t_{e_1} \| t_{e_2} \| \ldots \| t_{e_m} \rangle$$

where for each $c \in \{r, g, b\}$

$$t_c = C[color = c]\langle D[color = c_1] \| D[color = c_2] \rangle$$

with $c_1, c_2 \in \{r, g, b\}$ and $c \neq c_1$, $c \neq c_2$ and $c_1 \neq c_2$ (node variables are omitted in the sake of clarity). Moreover for each edge $(v_i, v_j) \in E$:

$$t_{(v_i,v_j)} = C[color = z_i]\langle D[color = z_j]\rangle$$

If G is 3-colorable the following complete tree is in $Rep_{d_1}(t)$ (node ids from $\mathcal{I}$ are omitted):

$$T = R\langle T_r \| T_g \| T_b\rangle$$

with

$$T_c = C[color = c]\langle D[color = c_1], D[color = c_2]\rangle$$

for all $c \in \{r, g, b\}$, where c_1, c_2 are the two colors different from c and from each other, in some arbitrary order. Indeed consider the mapping ν_{attr} associating to each variable z_i the color of node v_i in the given $\{r, g, b\}$-coloring. It is straightforward to verify that there is a mapping ν_{node} that sends each node variable of $\nu_{attr}(t)$ into a node id of T by preserving either node labels and values of the *color* attribute and the child relation. That is, $\nu = \langle \nu_{node}, \nu_{attr}\rangle$ is a homomorphism from $\underline{rel}(t)$ to T. It follows from Proposition 4.2 that $T \in Rep_{d_1}(t)$.

Conversely assume that there exists a tree T consistent with d_1 and a valuation ν of t such that $(T, \nu, s) \models t$ for some node s of T. Then T has a root of label R and three child subtrees, each of the form $C[color = e]\langle D[color = e_1], D[color = e_2]\rangle$ for some $e, e_1, e_2 \in \mathcal{D}$. The node s of T where t is satisfied has to be the root id of T (since it is the only node of T with label R). Therefore each sub-pattern t_c of t (for $c \in \{r, g, b\}$) has to be satisfied, under the valuation ν, in some child node of the root of T. Moreover t_r, t_g and t_b have to be satisfied into three distinct subtrees of T, because the *color* attribute has three distinct values in the of the roots of t_r, t_g and t_b. It follows that in each child subtree, T_r, T_g and T_b, the colors $\{e, e_1, e_2\}$ coincide with $\{r, g, b\}$, and therefore are all distinct.

Similarly, for each edge $(v_i, v_j) \in E$, the sub-pattern $t_{(v_i,v_j)}$ of t is satisfied, under ν, in some child node of the root of T. Therefore the pair $(\nu(z_i), \nu(z_j))$ coincides with a pair of colors (e, e'), with $e, e' \in \{r, g, b\}$ and $e \neq e'$. It follows that the mapping associating with each node v_i the color $\nu(z_i)$ is a 3-coloring of G. This concludes the proof. □

5.3 Consistency of incomplete DOM-trees

The key feature that was used to obtain NP-hardness for incomplete trees t was the possibility to "collapse" subtrees; i.e., different subtree descriptions of t could represent the same subtree of a tree in $Rep(t)$. This is impossible to do in the case of DOM-trees, where unique ids associated with node descriptions make such "collapse" impossible. We now turn to incomplete DOM-trees, and show that the presence of unique ids lowers the complexity of consistency, even in the presence of DTDs. However, it makes the proofs significantly harder. We show the following.

Theorem 5.21. CONSISTENCY *can be solved in* PTIME *for incomplete DOM-trees.*

Proof. Before we start with the proof, we define the notion of the *Gaifman* graph of a structure $\mathcal{A}$. This is the undirected graph whose nodes are the elements in the domain of $\mathcal{A}$, and such that there is an edge between nodes a and b in the graph if and only if there is a tuple in the interpretation of some relation in $\mathcal{A}$ that contains both a and b.

Now we start with the proof. The first thing that we will do is to find a set of necessary and sufficient conditions on $\underline{rel}(t)$ – for an incomplete DOM-tree t such that $\underline{rel}(t)$ is a structure of vocabulary $\tau_{\Sigma,A}$ – that ensure that $Rep_{\Sigma,A}(T) \neq \emptyset$. In view of Proposition 4.2, to verify whether $Rep_{\Sigma,A}(t) \neq \emptyset$ is equivalent to verify whether there is a tree T over vocabulary $\tau_{\Sigma,A}$ and a homomorphism $\bar{h} : \underline{rel}(t) \to T$. (Notice that since t is an incomplete DOM tree, $adom_{node}(t) \subseteq \mathcal{I}$, and, thus, $\bar{h}$ has to be the identity on $adom_{node}(t)$.) This is precisely the problem that we try to characterize next.

The proof of this result, although not difficult, is quite long and cumbersome. We proceed in a step-by-step fashion by first considering the reduct of $\underline{rel}(t)$ to a restricted vocabulary, and then relaxing these constraints one-by-one. From now on, every time that we say that that there is a homomorphism $\bar{h} : \mathcal{B} \to T$, from some structure $\mathcal{B}$ over a vocabulary $\tau \subseteq \tau_{\Sigma,A}$ into a tree T over vocabulary $\tau_{\Sigma,A}$, we really mean that $\bar{h}$ is a homomorphism from $\mathcal{B}'$ into T, where $\mathcal{B}'$ is the unique expansion of $\mathcal{B}$ to the vocabulary $\tau_{\Sigma,A}$ that satisfies that the interpretation in $\mathcal{B}'$ of each relation symbol in $\tau_{\Sigma,A} \setminus \tau$ is empty.

We start by considering the restriction $\underline{rel}(t)_0$ of $\underline{rel}(t)$ to the vocabulary that includes only the symbols $E, NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$. The question we want to solve is, does there exist a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_0 \to T$? Notice that if this is not the case, then we can immediately conclude that $Rep_{\Sigma,A}(t) = \emptyset$.

However, in order to do this, we start by considering structures over the even more restricted vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$. Our goal is to characterize when a structure $\mathcal{B}$ over this vocabulary can be "completed" into a tree into which all the elements in the domain of $\mathcal{B}$ are siblings. Those structures – that will be called *sisterhoods* – are defined next. Let $\mathcal{B}$ be a (possibly empty) finite structure over vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that satisfies the following:

1. The structure does not contain node variables, i.e. the domain of $\mathcal{B}$ is contained in $\mathcal{I}$;

2. no element belongs to more than one label. Formally, for each element i in the domain of $\mathcal{B}$, there is at most one label $\ell \in \Sigma$ such that i belongs to the interpretation of P_ℓ in $\mathcal{B}$ (it could also be the case that some elements in the domain of $\mathcal{B}$ do not belong to the interpretation of any unary relation symbol P_ℓ in $\mathcal{B}$, for $\ell \in \Sigma$);

3. the structure that is obtained from $\mathcal{B}$ by only considering the relation NS (but without removing elements that do not appear in NS) is a disjoint union of n (nonempty) successor relations $\mathcal{C}_1, \ldots, \mathcal{C}_n$, $n \geq 0$. Notice that some of these successor relations may consist of a single element only, in case that such an element does not appear in the relation NS;

4. there is at most one first child, and this has to be the first element of its own connected component with respect to NS. Formally, the interpretation of FC in $\mathcal{B}$ contains at most one element. Further, if i belongs to the interpretation of FC in $\mathcal{B}$ and $\mathsf{i} \in \mathcal{C}_j$, for $j \in [1, n]$, then i is the first element of $\mathcal{C}_j$ with respect to NS;

5. there is at most one last child, and this has to be the last element of its own connected component with respect to NS. Formally, the interpretation of LC in $\mathcal{B}$ contains at most one element. Further, if i belongs to the interpretation of LC in $\mathcal{B}$ and $\mathsf{i} \in \mathcal{C}_j$, for $j \in [1, n]$, then i is the last element of $\mathcal{C}_j$ with respect to NS; and

6. if the restricion of $\mathcal{B}$ to NS contains more than one connected component, then the first and last child belong to different components. Formally, if $n > 1$, and i_1 and i_2 are elements that belong to the interpretation of FC and LC in $\mathcal{B}$, respectively, then for every $j \in [1, n]$ it must be the case that $\mathsf{i}_1 \notin \mathcal{C}_j$ or $\mathsf{i}_2 \notin \mathcal{C}_j$.

In this case, we say that $\mathcal{B}$ is a *sisterhood*. A sisterhood $\mathcal{B}$ is *connected*, if the the *Gaifman* graph of $\mathcal{B}$ is connected (i.e. the restriction of $\mathcal{B}$ to NS consists of exactly one successor relation). The following trivial claim captures the intuitive idea that sisterhoods are those structures over vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that can "completed" into a tree into which all the elements of the domain of the structure are siblings.

Claim 5.22. *Let $\mathcal{B}$ be a structure over vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ such that the domain of $\mathcal{B}$ is contained in $\mathcal{I}$. Then there exists a tree T and a homomorphism $\bar{h} : \mathcal{B} \to T$ such that all the elements of $\mathcal{B}$ are siblings in T if and only if $\mathcal{B}$ is a sisterhood.*

Now we pass to analyze structures over the extended vocabulary $E, NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$. Let $\mathcal{N}$ be a structure over a vocabulary that contains the symbols E and NS, and let n be in $\mathcal{N}$. Then n is a *generator* in $\mathcal{N}$, if no element n', such that (n, n') belongs to the transitive and reflexive closure of the interpretation of the relation $NS \cup NS^{-1}$ in $\mathcal{N}$, has a parent in $\mathcal{N}$ with respect to E. Intuitively, an element n of $\mathcal{N}$ is a generator if it does not have a parent according to E, nor do any of the elements in $\mathcal{N}$ that are forced to be its siblings (according to NS) have a parent. With this notion in mind we provide next a recursive definition of a class of structures – called hierarchies of sisterhoods – over vocabulary $E, NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$. We prove afterwards that this is exactly the class of structures over vocabulary $E, NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that can be "completed" into a tree.

A *hierarchy of sisterhoods* is a hierarchy of sisterhoods of some level $k \geq 0$, where:

- the unique hierarchy of sisterhoods of level 0 is the empty sisterhood; and
- each hierarchy of sisterhoods $\mathcal{H}$ of level $k+1$, $k \geq 0$, is formed from the disjoint union of
  - a nonempty and connected sisterhood $\mathcal{B}$ with m elements $\{\mathsf{i}_1, \ldots, \mathsf{i}_m\}$, $m > 0$, (intuitively, these correspond to the generators of the hierarchy of sisterhoods), and
  - the disjoint union of hierarchies of sisterhoods

    $$\mathcal{H}_1^1, \ldots, \mathcal{H}_1^{p_1}, \mathcal{H}_2^1, \ldots, \mathcal{H}_2^{p_2}, \cdots, \mathcal{H}_m^1, \ldots, \mathcal{H}_m^{p_m}$$

    such that (1) $p_j \geq 0$, for each $j \in [1, m]$, (2) $\mathcal{H}_j^t$ is of level $\leq k$, for each $j \in [1, m]$ and $t \in [1, p_j]$, (3) if $k > 0$ then for some $j \in [1, m]$ and $t \in [1, p_j]$, $p_j > 0$ and $\mathcal{H}_j^t$ is of level exactly k, and (4) for each $j \in [1, m]$ it is the case that the structure over vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that is realized by the disjoint union of the generators of the $\mathcal{H}_j^t$'s ($t \in [1, p_j]$) is a sisterhood,

  by adding, for each $j \in [1, m]$ and $t \in [1, p_j]$ such that $p_j > 0$ and $\mathcal{H}_j^t$ is nonempty, at least one pair of the form $(\mathsf{i}_j, \mathsf{i}')$ to the interpretation of E, where i' is a generator in $\mathcal{H}_j^t$. (Notice that the unique generators in $\mathcal{H}$ are the ids in $\mathcal{B}$).

It is easy to see that each hierarchy of sisterhoods of level $k > 0$ is nonempty, and that the *Gaifman* graph of each hierarchy of sisterhoods is connected.

Intuitively, the level of $\mathcal{H}$ defines a lower bound on the depth of a smallest tree that can "complete" the hierarchy. For each $j \in [1, m]$ the generators of the $\mathcal{H}_j^t$'s ($t \in [1, p_j]$) correspond to the children of i_j in every tree T that "completes" $\mathcal{H}$. That is why we impose that the structure over vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that is realized by those elements must be a sisterhood (condition (4) above). The way in which we force that those elements correspond exactly to the children of i_j in every tree that "completes" T is by adding, for each $t \in [1, p_j]$ such that $\mathcal{H}_j^t$ is nonempty, at least one pair of the form $(\mathsf{i}_j, \mathsf{i}')$ to the interpretation of E, where i' is a generator in $\mathcal{H}_j^t$.

The following claim captures our intuition regarding hierarchies of sisterhoods and its role in capturing the class of structures over vocabulary $E, NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that can be "completed" into a tree.

Claim 5.23. *Let t be an incomplete DOM-tree and $\underline{rel}(t)_0$ the restriction of $\underline{rel}(t)$ to the vocabulary $E, NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$. Then there is a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_0 \to T$ if and only if $\underline{rel}(t)_0$ is a nonempty disjoint union of hierarchies of sisterhoods.*

Proof. It is easy to prove, by induction on k, that if $\underline{rel}(t)_0$ is a disjoint union of hierarchies of sisterhoods of level at most k then there is a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_0 \to T$. Assume on the other hand that $\underline{rel}(t)_0$ can be "completed" into a tree. Then $\underline{rel}(t)_0$ consists of the disjoint union of different connected components $H_1, \ldots, H_n$. Each one of these components must have at least one generator i (otherwise $\underline{rel}(t)_0$ contains a cycle and it could not be "completed" into a tree). Consider the set S of all the siblings of i with respect to NS. Then the structure over vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ realized by $S' := S \cup \{\mathsf{i}\}$ over $\underline{rel}(t)_0$ must be a connected sisterhood. For each $\mathsf{i}' \in S'$, let $C_{\mathsf{i}'}$ be the set of children of i' with respect to E. Then the structure over vocabulary $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ realized by all the elements in $C_{\mathsf{i}'}$ and its siblings with respect to NS over $\underline{rel}(t)_0$ must be a (not necessarily connected) sisterhood. By continuing in this fashion it is not hard to prove that each H_i $(1 \leq i \leq n)$ is a hierarchy of sisterhoods, and, thus, $\underline{rel}(t)_0$ is a disjoint union of hierarchies of sisterhoods. □

Now, we pass to consider the restriction $\underline{rel}(t)_1$ of $\underline{rel}(t)$ to the vocabulary that includes only the symbols

$$E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC.$$

The question we want to solve again is, does there exist a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_1 \to T$? Notice that if this is not the case, we can immediately conclude that $Rep_{\Sigma,A}(t) = \emptyset$.

As in the previous case, we start by analyzing the even more restrictive vocabulary $NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$. Let $\mathcal{B}$ be a (possibly empty) structure over this vocabulary that satisfies the following:

1. The restriction of $\mathcal{B}$ to the symbols $NS, (P_\ell)_{\ell\in\Sigma}, FC, LC$ is a sisterhood. (We assume that the restriction of $\mathcal{B}$ to NS is formed by the disjoint union of the n (nonempty) successor relations $\mathcal{C}_1, \ldots, \mathcal{C}_n$, $n \geq 0$);

2. the interpretation of NS^* respects the transitive and reflexive closure of NS over each connected component $\mathcal{C}_j$. Formally, for each $j \in [1, n]$, if $\mathsf{i}_1, \mathsf{i}_2 \in \mathcal{C}_j$ and $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to the interpretation of NS^* in $\mathcal{B}$, then $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to the transitive and reflexive closure of the interpretation of NS in $\mathcal{C}_j$, and, thus, in $\mathcal{B}$;

3. the connected components $\mathcal{C}_j$ can be arranged in a sisterhood in such a way that NS^* respects the transitive and reflexive closure of NS over different components. Formally, the simple and directed graph $G_\mathcal{B}$, defined as follows, is a DAG: The set of vertices of $G_\mathcal{B}$ is $\{v_1, \ldots, v_n\}$, and the pair (v_j, v_k) is an edge of $G_\mathcal{B}$, for $j, k \in [1, n]$ with $j \neq k$, if and only if there exist ids $\mathsf{i}_1 \in \mathcal{C}_j$ and $\mathsf{i}_2 \in \mathcal{C}_k$ such that the pair $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to the interpretation of NS^* in $\mathcal{B}$;

4. the connected components $\mathcal{C}_j$ can be arranged in a sisterhood in such a way that, if there is a first child i, then i belongs to the connected component that appears first from left-to-right in the sisterhood. Formally, if i belongs to the interpretation of FC in $\mathcal{B}$, and i belongs to $\mathcal{C}_j$, for $j \in [1, n]$, then v_j has no incoming edges in $G_\mathcal{B}$; and

5. the connected components $\mathcal{C}_j$ can be arranged in a sisterhood in such a way that, if there is a last child i, then i belongs to the connected component that appears last from left-to-right in the sisterhood. Formally, if i belongs to the interpretation of LC in $\mathcal{B}$, and i belongs to $\mathcal{C}_j$, for $j \in [1, n]$, then v_j has no outgoing edges in $G_\mathcal{B}$.

If this is the case, we say that $\mathcal{B}$ is an *extended* sisterhood. Further, we say that $\mathcal{B}$ is *connected* if the *Gaifman* graph of $\mathcal{B}$ is connected. The following trivial claim captures the intuitive idea that extended sisterhoods are those structures over vocabulary $NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that can "completed" into a tree into which all the elements of the domain of the structure are siblings.

Claim 5.24. *Let $\mathcal{B}$ be a structure over vocabulary $NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ such that the domain of $\mathcal{B}$ is contained in $\mathcal{I}$. Then there exists a tree T and a homomorphism $\bar{h} : \mathcal{B} \to T$ such that all the elements of $\mathcal{B}$ are siblings in T if and only if $\mathcal{B}$ is an extended sisterhood.*

Now we continue by analyzing structures over the extended vocabulary $E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$. For every structure $\mathcal{N}$ over a vocabulary that contains the symbols E, NS and NS^*, and element n in $\mathcal{N}$, we say that n is an *extended* generator in $\mathcal{N}$, if no element n', such that (n, n') belongs to the transitive and reflexive closure of the interpretation of the relation $NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1}$ in $\mathcal{N}$, has a parent in $\mathcal{N}$ with respect to E. Intuitively, and as in the previous case, an element is an extended generator if it does not have a parent according to E, nor do any of the elements in $\mathcal{N}$ that are forced to be its siblings (according to $NS \cup NS^*$) have a parent. With this notion in mind we provide next a recursive definition of a class of structures – called extended hierarchies of sisterhoods – over vocabulary $E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$. We prove afterwards that this is exactly the class of structures over vocabulary $E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that can be "completed" into a tree.

An *extended* hierarchy of sisterhoods is a hierarchy of sisterhoods of some level $k \geq 0$, where:

- the unique extended hierarchy of sisterhoods of level 0 is the empty sisterhood; and
- each extended hierarchy of sisterhoods $\mathcal{H}$ of level $k+1$, $k \geq 0$, is formed from the disjoint union of

  – a nonempty and connected extended sisterhood $\mathcal{B}$ with m elements $\{\mathsf{i}_1, \ldots, \mathsf{i}_m\}$, $m > 0$, (intuitively, these correspond to the generators of the extended hierarchy of sisterhoods), and

  – the disjoint union of extended hierarchies of sisterhoods

  $$\mathcal{H}_1^1, \ldots, \mathcal{H}_1^{p_1}, \mathcal{H}_2^1, \ldots, \mathcal{H}_2^{p_2}, \cdots, \mathcal{H}_m^1, \ldots, \mathcal{H}_m^{p_m}$$

  such that (1) $p_j \geq 0$, for each $j \in [1, m]$, (2) $\mathcal{H}_j^t$ is of level $\leq k$, for each $j \in [1, m]$ and $t \in [1, p_j]$, (3) if $k > 0$ then for some $j \in [1, m]$ and $t \in [1, p_j]$, $p_j > 0$ and $\mathcal{H}_j^t$ is of level exactly k, and (4) for each $j \in [1, m]$ it is the case that the structure over vocabulary $NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that is realized by the disjoint union of the generators of the $\mathcal{H}_j^t$'s ($t \in [1, p_j]$) is an extended sisterhood,

  by adding, for each $j \in [1, m]$ and $t \in [1, p_j]$ such that $p_j > 0$ and $\mathcal{H}_j^t$ is nonempty, at least one pair of the form $(\mathsf{i}_j, \mathsf{i}')$ to the interpretation of E, where i' is a generator in $\mathcal{H}_j^t$. (Notice that the unique generators in $\mathcal{H}$ are the ids in $\mathcal{B}$).

It is easy to see that each hierarchy of sisterhoods of level $k > 0$ is nonempty, and that the *Gaifman* graph of each hierarchy of sisterhoods is connected.

It is worth noticing that the definition below can be obtained directly from that of hierarchy of sisterhoods by replacing sisterhoods with extended sisterhoods. As in the previous case, the level of an extended hierarchy of sisterhoods $\mathcal{H}$ defines a lower bound on the depth of a smallest tree that can "complete" $\mathcal{H}$. For each $j \in [1, m]$ the generators of the $\mathcal{H}_j^t$'s ($t \in [1, p_j]$) correspond to the children of i_j in every tree T that "completes" $\mathcal{H}$. That is why we impose that the structure over vocabulary $NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that is realized by those elements must be an extended sisterhood (condition (4) above). The way in which we force that those elements correspond exactly to the children of i_j in every tree that "completes" T is by adding, for each $t \in [1, p_j]$ such that $\mathcal{H}_j^t$ is nonempty, at least one pair of the form $(\mathsf{i}_j, \mathsf{i}')$ to the interpretation of E, where i' is a generator in $\mathcal{H}_j^t$.

The following claim captures our intuition regarding extended hierarchies of sisterhoods and its role in capturing the class of structures over vocabulary $E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that can be "completed" into a tree. The proof of this result goes along the same lines than the proof of Claim 5.23:

Claim 5.25. *Let t be an incomplete DOM-tree and $\underline{rel}(t)_1$ the restriction of $\underline{rel}(t)$ to the vocabulary $E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$. Then there is a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_1 \to T$ if and only if $\underline{rel}(t)_1$ is a nonempty disjoint union of extended hierarchies of sisterhoods.*

Next we consider the restriction $\underline{rel}(t)_2$ of $\underline{rel}(t)$ to the vocabulary that includes only the symbols

$$E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC.$$

The question we want to solve again is, does there exist a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_2 \to T$? Notice that if this is not the case, we can again immediately conclude that $Rep_{\Sigma,A}(t) = \emptyset$.

Let $\mathcal{B}$ be a structure over vocabulary

$$E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$$

that satisfies the following:

- The restriction of $\mathcal{B}$ to $E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ is the disjoint union of p (nonempty) extended hierarchies of sisterhoods $\mathcal{H}_1, \ldots, \mathcal{H}_p$, $p \geq 0$;
- the interpretation of E^* respects the transitive and reflexive closure of E over each extended hierarchy of sisterhoods $\mathcal{H}_j$. Formally, for each $j \in [1, p]$ and pair of elements $\mathsf{i}_1, \mathsf{i}_2 \in \mathcal{H}_j$, if $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to the interpretation of E^* in $\mathcal{B}$, then either $\mathsf{i}_1 = \mathsf{i}_2$ or $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to the relation defined by the union of (i) the interpretation of E in $\mathcal{H}_j$, and (ii) the composition of the interpretation of E in $\mathcal{H}_j$ with the transitive and reflexive closure of the interpretation of $(E \cup NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$ in $\mathcal{H}_j$ (intuitively, i_2 is a descendant of i_1 in any tree that "completes" $\mathcal{B}$). Notice that not every pair that satisfies (i) also satisfies (ii); e.g. it may be the case that $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to E but i_2 does not appear in the relation $(E \cup NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$, i.e. i_2 has neither children nor siblings in $\mathcal{B}$;
- the extended hierarchies of sisterhoods $\mathcal{H}_j$ can be arranged in such a way that E^* respects the transitive and reflexive closure of E over the different extended hierarchies of sisterhoods. Formally, the simple and directed graph $G^{\mathcal{B}}$, defined as follows, is a DAG: The set of vertices of $G^{\mathcal{B}}$ is $\{u_1, \ldots, u_p\}$, and the pair (u_j, u_k) is an edge of $G^{\mathcal{B}}$, for $j, k \in [1, p]$ with $j \neq k$, if and only if there exist ids $\mathsf{i}_1 \in \mathcal{H}_j$ and $\mathsf{i}_2 \in \mathcal{H}_k$ such that the pair $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to the interpretation of E^* in $\mathcal{B}$; and
- for every $j, k \in [1, p]$, the pair (u_j, u_k) is not *conflictive*, where conflictive pairs are defined as follows:
  - for each $j, k \in [1, p]$ with $j \neq k$, if it is the case that, for some $m, m' \in [1, p]$ such that $j \neq m$ and $j \neq m'$, there are ids $\mathsf{i}_1, \mathsf{i}_2 \in \mathcal{H}_j$ and $\mathsf{i}_3 \in \mathcal{H}_m$ and $\mathsf{i}_4 \in \mathcal{H}_{m'}$ such that (1) $(\mathsf{i}_1, \mathsf{i}_3)$ belongs to the interpretation of E^* in $\mathcal{B}$ (that is, each element of $\mathcal{H}_m$ must be a descendant of i_1 in every tree that "completes" $\mathcal{B}$), (2) $(\mathsf{i}_2, \mathsf{i}_4)$ belongs to the interpretation of E^* in $\mathcal{B}$ (that is, each element of $\mathcal{H}_{m'}$ must be a descendant of i_2 in every tree that "completes" $\mathcal{B}$), (3) $\mathsf{i}_1 \neq \mathsf{i}_2$, and neither $(\mathsf{i}_1, \mathsf{i}_2)$ nor $(\mathsf{i}_2, \mathsf{i}_1)$ belongs to the relation defined by the union of (i) the interpretation of E in $\mathcal{H}_m$, and (ii) the composition of the interpretation of E in $\mathcal{H}_m$ with the transitive and reflexive closure of the interpretation of $(E \cup NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$

in $\mathcal{H}_j$ (that is, neither i_1 is a "descendant" of i_2 nor i_2 is a descendant of i_1 in $\mathcal{B}$), and (4) u_j is reachable from both u_m and $u_{m'}$ in $G^{\mathcal{B}}$, then the pair (u_j, u_k) is *conflictive*. Intuitively, this implies that elements of $\mathcal{H}_k$ must be, at the same time, descendants of i_1 and i_2. But the latter is impossible since in every tree that "completes" $\mathcal{B}$ the intersection of the sets of descendants of i_1 and i_2 must be empty.

In this case, we say that $\mathcal{B}$ is a *consistent* union of extended hierarchy of sisterhoods.

Using the same kind of techniques than in the proofs of the previous claims, we can show that the class of consistent union of extended hierarchies of sisterhoods is precisely the class of structures over vocabulary $E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that can be "completed" into a tree. Indeed,

Claim 5.26. *Let t be an incomplete DOM-tree and $\underline{re\ell}(t)_2$ the restriction of $\underline{re\ell}(t)$ to the vocabulary $E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$. Then there is a tree T and a homomorphism $\bar{h} : \underline{re\ell}(t)_2 \to T$ if and only if $\underline{re\ell}(t)_2$ is a nonempty and consistent union of extended hierarchies of sisterhoods.*

Proof. It is clear from the previous discussion that any structure over the vocabulary $E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ that is not a consistent union of extended hierarchies of sisterhoods cannot be completed into a tree. On the other hand, assume that $\mathcal{B}$ is a consistent union of extended hierarchy of sisterhoods $\mathcal{H}_1, \ldots, \mathcal{H}_p$. The idea is try to arrange the different $\mathcal{H}_i$'s in a tree T, but in a way that it respects E^*. We can assume w.l.o.g. that $G^{\mathcal{B}}$ is connected. Since $G^{\mathcal{B}}$ is a DAG there must be a node u_i in it without incoming edges. We choose $\mathcal{H}_i$ as the component that will appear first, when looking top-down, in T. Then for each $\mathcal{H}_j$ such that (u_i, u_j) is an edge of $G^{\mathcal{B}}$, we place $\mathcal{H}_j$ in T, but in a way that it appears below every element $\mathsf{i}_c \in \mathcal{H}_i$ such that for some $\mathsf{i}_r \in \mathcal{H}_j$ it is the case that $(\mathsf{i}_c, \mathsf{i}_r) \in E^*$. Notice that all the the i_c's must be comparable in terms of the descendant relation in T (otherwise, (u_i, u_j) would be a conflictive pair), and, thus, it is possible to place $\mathcal{H}_j$ in T in this way. The process then continues along the same lines, taking advantage of the facts that $G^{\mathcal{B}}$ is a DAG (and, therefore, that neighbors with respect to outgoing edges of elements in $G^{\mathcal{B}}$ can always receive a topological order) and that there are no conflictive pairs in $G^{\mathcal{B}}$ (and, thus, that the process can be continued without falling into inconsistencies). □

We now analyze the case of the restriction $\underline{re\ell}(t)_3$ of $\underline{re\ell}(t)$ to the vocabulary that includes the symbols

$$E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC, Root, Leaf.$$

The question we want to solve again is, does there exist a tree T and a homomorphism $\bar{h} : \underline{re\ell}(t)_3 \to T$? Notice that if this is not the case, we can again immediately conclude that $Rep_{\Sigma,A}(t) = \emptyset$.

Let $\mathcal{B}$ be a structure over vocabulary

$$E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC, Root, Leaf$$

that satisfies the following:

- The restriction of $\mathcal{B}$ to $E, E^*, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ is a the consistent union of p nonempty extended hierarchies of sisterhoods $\mathcal{H}_1, \ldots, \mathcal{H}_p$, $p \geq 0$;
- there is at most one root i and, if there is a root, then it has neither a parent nor a sibling in its connected component. Further, the $\mathcal{H}_i$'s can be arranged in a tree T in such a way that, if there is a root i, then i belongs to the component that appears first, when looking top-down, in T. Formally, the interpretation of *Root* in $\mathcal{B}$ contains at most one element. Further, if i belongs to the interpretation of *Root* in $\mathcal{B}$ and also belongs to $\mathcal{H}_j$, $j \in [1, p]$, then it must be the case that (1) there is no element i' such that $(\mathsf{i}, \mathsf{i}')$ belongs to the interpretation of the relation $(E^{-1} \cup NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$ in $\mathcal{H}_j$ (or equivalently, i is the unique extended generator

in $\mathcal{H}_j$), (2) u_j has no incoming edges in $G^{\mathcal{B}}$, and (3) i does not belong to the interpretation of FC and LC in $\mathcal{H}_j$; and

- leaves have no children. Formally, if i is an element that belongs to the interpretation of *Leaf* in $\mathcal{B}$, then i has no children in $\mathcal{B}$ with respect to E; and
- If (u_j, u_k) is an edge of $G^{\mathcal{B}}$ then the extended generators in $\mathcal{H}_k$ can be placed as proper descendants of some node in $\mathcal{H}_j$. Formally, for every $j, k \in [1, p]$ with $j \neq k$, if for some $\mathsf{i}_1 \in \mathcal{H}_j$ and $\mathsf{i}_2 \in \mathcal{H}_k$ it is the case that $(\mathsf{i}_1, \mathsf{i}_2)$ belongs to the interpretation of E^* in $\mathcal{B}$, then there exists a node i in $\mathcal{H}_j$ such that,

  1. either $\mathsf{i}_1 = \mathsf{i}$, or $(\mathsf{i}_1, \mathsf{i})$ belongs to the interpretation of the relation defined by the union of (i) the interpretation of E in $\mathcal{H}_j$, and (ii) the composition of the interpretation of E in $\mathcal{H}_j$ with the transitive and reflexive closure of the interpretation of $(E \cup NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$ in $\mathcal{H}_j$ (intuitively, i is a descendant of i_1 in every tree that "completes" $\mathcal{B}$),
  2. i does not belong to the interpretation of *Leaf* in $\mathcal{B}$, and
  3. if W is the set of elements i' such that $(\mathsf{i}, \mathsf{i}')$ belongs to the relation defined by the union of (i) the interpretation of E in $\mathcal{H}_j$, and (ii) the composition of the interpretation of E in $\mathcal{H}_j$ with the transitive and reflexive closure of the interpretation of $(NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$ in $\mathcal{H}_j$ (intuitively, W is the set of all the elements in $\mathcal{B}$ that are children of i in any tree that "completes" $\mathcal{B}$), then at least one of the following holds: (1) The *Gaifman* graph of the restriction to NS of the substructure of $\mathcal{H}_j$ induced by the elements in W is not connected; (2) W contains no element in the interpretation of FC in $\mathcal{B}$; (3) W contains no element in the interpretation of LC in $\mathcal{B}$. Intuitively, this represents the fact that there is "room" below i to place the extended generators of $\mathcal{H}_k$.

In this case, we say that $\mathcal{B}$ is *consistent with respect to* $\mathcal{I}$.

Using a proof along the lines of that of Claim 5.26, we can prove that the structures that are consistent with respect to $\mathcal{I}$ are exactly those that can be completed into a tree.

Claim 5.27. *Let t be an incomplete DOM-tree and $\underline{rel}(t)_3$ the restriction of $\underline{rel}(t)$ to the vocabulary $E, E^*, NS, NS^*, (P_\ell)_{\ell \in \Sigma}, FC, LC, Root, Leaf$. Then there is a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_3 \to T$ if and only if $\underline{rel}(t)_3$ is consistent with respect to $\mathcal{I}$.*

Since no DTD is present, the consistency problem for an incomplete DOM-tree t gets reduced to the consistency problem for its restriction without data values. Further, and for the same reason, the consistency problem for t can be reduced to the consistency problem over trees whose sets of labels and attribute names coincide with those that are already present in t. Summing up, given an incomplete DOM-tree t such that $\underline{rel}(t)$ is a structure over $\tau_{\Sigma,A}$, it is the case that $Rep(t) \neq \emptyset$ if and only if $Rep_{\Sigma,A}(t) \neq \emptyset$ iff there is a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_3 \to T$, where $\underline{rel}(t)_3$ is the restriction of $\underline{rel}(t)$ to the vocabulary $E, E^*, NS, NS^*, (P_\ell)_{\ell \in \Sigma}, FC, LC, Root, Leaf$. From Claim 5.27 the latter is equivalent to checking whether $\underline{rel}(t)_3$ is consistent with $\mathcal{I}$. But it is not hard to see that all the properties that define whether a structure over this vocabulary is consistent with $\mathcal{I}$ can be checked in polynomial time (in the size of the structure). Since $\underline{rel}(t)$ can be constructed in polynomial time from t, it follows that Consistency can be solved in PTIME for incomplete DOM-trees. This concludes the proof. □

We can even get tractability for consistency with DTDs if we restrict to $\downarrow^*$-*free* incomplete DOM-trees that do not use the descendant relation (i.e. $\langle\langle \cdot \rangle\rangle$ cannot be used in incomplete tree descriptions). More precisely, we say that an incomplete DOM-tree t is $\downarrow^*$-*free* if all the incomplete tree descriptions used in the definition of t are of the form $\beta\langle f \rangle$.

Theorem 5.28. *For each fixed DTD d,* CONSISTENCY*(d) is solvable in* PTIME *for $\downarrow^*$-free incomplete DOM-trees.*

The proof of this result is in the appendix.

However, the combined complexity (when the DTD is not fixed) is intractable:

Proposition 5.29. *The problem of checking, for a DTD d and an incomplete DOM-tree t, whether $Rep_d(t)$ is nonempty, is* NP*-complete.*

In fact, to get NP-hardness, it suffices to look at $(\downarrow, \|)$-incomplete DOM-trees without attributes and DTDs in which every regular expression defines a finite language.

Proof. It follows from [37] that the following problem is NP-complete: Given an NFA $\mathcal{A}$ over finite alphabet Σ and a subset Σ' of Σ, does $\mathcal{A}$ accept a string w in which every symbol of Σ' is mentioned? The problem remains NP-hard even if restricted to deterministic NFAs that only accept a finite number of strings [11]. This implies that the problem of checking, for a DTD d in which every regular expression is given by a deterministic NFA that accepts a finite language and an $(\downarrow, \|)$-incomplete DOM-tree t, whether $Rep_d(t)$ is nonempty, is also NP-hard. Indeed, given an NFA $\mathcal{A}$ over Σ of the form above and $\Sigma' \subseteq \Sigma$ such that $\Sigma' = \{a_1, \ldots, a_n\}$, one can construct in polynomial time (1) a DTD $d_{\mathcal{A}} = (r, \rho, \alpha)$ such that $\rho(r)$ is $\mathcal{A}$, $\alpha(r) = \emptyset$, and for each $a \in \Sigma$, $\rho(a)$ is the NFA that only accepts the empty string ε and $\alpha(a) = \emptyset$, and (2) a $(\downarrow, \|)$-incomplete DOM-tree $t_{\Sigma'}$ of the following form (we omit node ids, since we know that in DOM trees they are all different): $r\langle a_1 \| a_2 \| \cdots \| a_n \rangle$. It is easy to see that $\mathcal{A}$ accepts a string that mentions every symbol of Σ' if and only if $Rep_{d_{\mathcal{A}}}(t_{\Sigma'})$ is nonempty. □

6 The membership problem

We now consider the next basic computational problem related to incomplete information:

PROBLEM:	MEMBERSHIP
INPUT:	an incomplete tree t, a complete tree T
QUESTION:	is $T \in Rep(t)$?

To test whether $T \in Rep(t)$ one just guesses a homomorphism $h : \underline{rel}(t) \to T$; hence MEMBERSHIP is in NP.

Recall what is known in the relational case. The problem of checking whether R' is in $Rep(R)$ is NP-complete if R is a naïve table, and in PTIME if R is a Codd table, i.e. each variable occurs exactly once in it. We shall prove an analog of this result. We say that t is an incomplete *Codd* tree if every variable from $\mathcal{V}_{\text{attr}}$ occurs at most once in t.

We show that for incomplete trees, the complexity of MEMBERSHIP mimics the relational case (although the proof for the Codd case is quite different from the relational technique [3], which is based on bipartite graph matching; instead we use a technique inspired by CTL model-checking), but it is polynomial for all DOM- and Codd trees.

Theorem 6.1.

- MEMBERSHIP *for $(\downarrow, \|)$-incomplete trees is* NP*-complete.*
- *For incomplete Codd trees,* MEMBERSHIP *is solvable in* PTIME*.*
- *For incomplete DOM-trees,* MEMBERSHIP *is solvable in* PTIME*.*

Proof. We start by showing NP-hardness for $(\downarrow, \|)$-incomplete trees (it was already observed that MEMBERSHIP is in NP). We use a reduction from the 3-COLORING problem.

Let $G = \langle V, E\rangle$ be a graph, where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots e_m\}$. One can construct a $(\downarrow, \|)$-incomplete tree t and a tree T such that $T \in \mathit{Rep}(t)$ if and only if G is 3-colorable:

$$t = R\langle t_{e_1} \| t_{e_2} \| \ldots \| t_{e_m}\rangle$$

where t_{e_i}s are defined as in the proof of Theorem 5.20 (the case of $(\downarrow, \|)$-incomplete trees). The complete tree is:

$$T = R\langle T_r \| T_g \| T_b\rangle$$

defined exactly as in the proof of Theorem 5.20.

If G is 3-colorable then $T \in \mathit{Rep}(t)$. Indeed one can construct a valuation $\nu = (\nu_{node}, \nu_{attr})$ over variables of t from $\mathcal{V}_{\text{node}}$ and $\mathcal{V}_{\text{attr}}$, respectively. The valuation ν_{attr} assigns to each variable x_i the color of node v_i in the given $\{r, g, b\}$-coloring. The valuation ν_{node} maps the root node variable of t into the root of T and all other nodes of t into nodes of T having the same color attribute value and the same level. If i is the root node of T we have $(T, \nu, \mathrm{i}) \models t$, and thus $T \in \mathit{Rep}(t)$.

Conversely assume that $T \in \mathit{Rep}(t)$, then there exists a valuation ν of variables of t such that $(T, \nu, s) \models t$ for some node s of T. The node s has to be the root of T, because it is the only node of label R in T. Therefore for each edge $(v_i, v_j) \in E$, the subtree $t_{(v_i,v_j)}$ of t is satisfied, under ν, in some child node of the root of T. Thus the pair $(\nu(x_i), \nu(x_j))$ coincides with a pair of colors (e, e'), with $e, e' \in \{r, g, b\}$ and $e \neq e'$. It follows that the mapping associating with each node v_i the color $\nu(x_i)$ is a 3-coloring of G.

Next, we move to Membership for incomplete trees under Codd interpretation.

Let T be a complete tree and t be an incomplete tree under Codd interpretation (i.e., each variable from $\mathcal{V}_{\text{attr}}$ occurs at most once in t). We will consider the equivalent syntax of incomplete trees, given below:

$$\begin{aligned} t &:= \beta\langle f\rangle\langle\langle f\rangle\rangle \\ f &:= \varepsilon \mid f_1^< \parallel f_2^< \parallel \ldots \parallel f_k^< \\ f^< &:= t \mid t\ \theta\ f^< \end{aligned} \tag{3}$$

where $f_1^<, \ldots f_k^<$, with $k \geq 1$ are incomplete forests of type $f^<$, the operator θ is either $\rightarrow$ or $\rightarrow^*$, and the node descriptions β are defined as in the classical syntax. We will say that a formula of this syntax is *ordered* if it is a formula of type t or $f^<$.

A parse tree of the above syntax for t can be constructed in polynomial time in the size of t. So in the rest of the proof we will assume we are given a parse tree of t in the above grammar, and nodes of this parse tree will be referred to as *subformulae* of t. Specifically, parse tree nodes of the form f (resp. t) will be referred to as *forest subformulae* (resp. *tree subformulae*); parse tree nodes of the form t or $f^<$ will be referred to as *ordered subformulae*.

We describe an algorithm to check whether $T \in \mathit{Rep}(t)$ inspired by CTL model checking. The idea is to compute, for each (ordered) subformula φ of t the set of nodes of T where φ is satisfied, denoted by $[\![\varphi]\!]^T$. This is done inductively, starting with leaves of the parse tree of t, and stopping when $[\![t]\!]^T$ has been computed. We will prove that $[\![\varphi]\!]^T$ can be correctly computed when sets $[\![\varphi']\!]^T$ have been computed for all (ordered) subformulae φ' of φ.

Formally we define $[\![\cdot]\!]$ for ordered formulae of the above syntax:

- $[\![t]\!]^T$ is the set of nodes s of T such that $(T, \nu, s) \models t$ for some valuation ν of variables occurring in t (both from $\mathcal{V}_{\text{node}}$ and $\mathcal{V}_{\text{attr}}$);
- $[\![f^<]\!]^T$ is the set of nodes s of T such that there exists a set S of following siblings of s in T satisfying $(T, \nu, \{s\} \cup S) \models f^<$, for some valuation ν of variables of $f^<$.

For technical reasons, for an ordered formula φ, we also define $[\![\varphi]\!]^T_{desc}$ as the set of ancestors of nodes of $[\![\varphi]\!]^T$ in T. Similarly we define $[\![\varphi]\!]^T_{sib}$ as the set of preceding siblings of nodes of $[\![\varphi]\!]^T$ in T. Formally,

- $[\![\varphi]\!]^T_{desc}$ is the set of nodes s of T such that there exists a node $s' \in [\![\varphi]\!]^T$ with $s' \neq s$ and $E^*(s, s')$.
- $[\![\varphi]\!]^T_{sib}$ is the set of nodes s of T such that there exists a node $s' \in [\![\varphi]\!]^T$ with $s' \neq s$ and $NS^*(s, s')$.

We now describe a function SEM intended to compute sets $[\![\varphi]\!]^T$, $[\![\varphi]\!]^T_{desc}$ and $[\![\varphi]\!]^T_{sib}$ for an ordered formula φ and an incomplete tree T.

Function SEM The function SEM takes as arguments:

- a complete tree T;
- an ordered formula φ of the above syntax, whose set of ordered subformulae (excluding φ itself) is $\{\varphi_1, \ldots, \varphi_m\}$;
- triples $(S(\varphi_i), S_{desc}(\varphi_i), S_{sib}(\varphi_i))$ of sets of nodes of T, for $i \in [1, m]$.

SEM returns nodes $S(\varphi)$, $S_{desc}(\varphi)$ and $S_{sib}(\varphi)$ computed as follows, depending on φ.

If $\varphi = \beta\langle f_1\rangle\langle\!\langle f_2\rangle\!\rangle$, with $f_1 = f_1^< \| \ldots \| f_k^<$ and $f_2 = f_{k+1}^< \| \ldots \| f_m^<$, let $F_1 = \{f_1^<, \ldots, f_k^<\}$, and $F_2 = \{f_{k+1}^<, \ldots, f_m^<\}$. For each node s of T, the function SEM adds s to $S(\varphi)$ if and only if all the the following holds:

a) $(T, \nu, s) \models \beta$, for some valuation ν of variables occurring in β (both variables from $\mathcal{V}_{\text{node}}$ and $\mathcal{V}_{\text{attr}}$).

b) for each $\psi \in F_1$ there exists a child s' of s in T such that $s' \in S(\psi)$.

c) for each $\psi \in F_2$

- if ψ is of the form $t_1 \rightarrow^* t_2 \rightarrow^* \ldots \rightarrow^* t_r$, for some $r \geq 1$, then either $s \in S_{desc}(\psi)$, or $s \in S(t_j)$ for each $j \in [1, r]$.
- otherwise $s \in S_{desc}(\psi)$.

These conditions on node s can be easily verified in time $O(|\beta| + (|Sub_t(\varphi)| \cdot |Ch(s)|))$, where $Ch(s)$ is the set of children of s in T, and $Sub_t(\varphi)$ is the set of maximal tree subformulae of φ. In fact if we let $\beta = \ell^\mu(x)[@a_1 = z_1, \ldots, @a_m = z_m]$, in order to check a) on node s, one needs to verify that:

- the label of node s in T matches ℓ (that is, either $\ell = \_$ or $\ell = l$); this is done in constant time;
- the node s is in all marking relations of T corresponding to μ; this may be done, depending on the way T is stored, in constant time;
- there is no attribute a_i in β such that $z_i \in \mathcal{D}$ and the value of $@a_i$ on node s is $v \neq z_i$. This can be checked in linear time in the number of attributes of β therefore in time $O(|\beta|)$.

On the whole this requires time $O(|\beta|)$ and verifies precisely a). In fact if it holds, we can construct a valuation ν assigning $x = s$ and assigning to each variable z_i the value of the corresponding attribute in s (if it exists, otherwise z_i is assigned an arbitrary value from $\mathcal{D}$). This is a valuation thanks to the fact that all variables z_i are distinct. Directly from the definition of the semantics of incomplete trees, it follows $(T, \nu, s) \models \beta$.

Conditions b) and c) can be verified in time $O(|Sub_t(\varphi)| \cdot |Ch(s)|)$. In fact F_1 is the set of maximal ordered subformulae of φ, thus it is bounded by the number of maximal tree subformulae of φ. Similarly, to check c) one may need to scan all maximal tree subformulae of φ. The overall cost of computing $S(\varphi)$ is then $O(|T| \cdot (|\beta| + |Sub_t(\varphi)|)$.

If $\varphi = t\ \theta\ f^<$ the set $S(\varphi)$ is computed as follows. For each node s of T, SEM adds s to $S(\varphi)$ if and only if all the following is true:

a) $s \in S(t)$;

b) if $\theta = \rightarrow$, there exists a node $s' \in S(f^<)$ such that $NS(s, s')$ holds in T;

c) if $\theta = \rightarrow^*$, then either $s \in S_{sib}(f^<)$ or $s \in S(f^<)$.

Under suitable representations of T and the sets $S(\varphi_i)$, all these conditions can be verified on s in constant time. Thus in this case $S(\varphi)$ is computed in time $O(|T|)$.

Finally SEM computes sets $S_{desc}(\varphi)$ and $S_{sib}(\varphi)$ by selecting all ancestors and, respectively, all preceding siblings of nodes of $S(\varphi)$. This can be done in a single postfix depth-first right-to-left traversal of the tree, thus in time $(O|T|)$. On the whole SEM runs in time $O(|T| \cdot (|\beta| + |Sub_t(\varphi)|)$, for a tree subformula, and in time $O(|T|)$ for a subformula $\varphi = t\ \theta\ f^<$.

We now prove correctness of SEM:

Lemma 6.2. *For each ordered formula φ with set of ordered subformulae $\{\varphi_i | i \in [1, m]\}$, and for each complete tree T, the function* SEM *over arguments φ, T and*

- $S(\varphi_i) = [\![\varphi_i]\!]^T$ *, for* $i \in [1, m]$,
- $S_{desc}(\varphi_i) = [\![\varphi_i]\!]^T_{desc}$ *, for* $i \in [1, m]$,
- $S_{sib}(\varphi_i) = [\![\varphi_i]\!]^T_{sib}$ *, for* $i \in [1, m]$,

computes the sets: $S(\varphi) = [\![\varphi]\!]^T$ *,* $S_{desc}(\varphi) = [\![\varphi]\!]^T_{desc}$ *, and* $S_{sib}(\varphi) = [\![\varphi]\!]^T_{sib}$ *.*

Proof. Assume $\varphi = \beta\langle f_1 \rangle\langle\langle f_2 \rangle\rangle$. We have to check that $s \in [\![\varphi]\!]^T$ if and only if conditions a), b) and c) checked by the function SEM are satisfied. If $s \in [\![\varphi]\!]^T$, then $(T, \nu, s) \models t$ for some valuation ν of variables of t, this implies a), b) and c) by definition of $[\![\psi]\!]^T$, for subformulae ψ of φ. If conversely properties a), b) and c) are satisfied in s, then we have:

- $(T, \nu_\beta, s) \models \beta$, for some valuation ν_β of variables occurring in β;
- for each $\psi \in F_1$ there exists a child s' of s in T and a set S of following siblings of s' such that $(T, \nu_\psi, s' \cup S) \models \psi$, for some valuation ν_ψ of variables of ψ;
- for each $\psi \in F_2$
  - if ψ is of the form $t_1 \rightarrow^* t_2 \rightarrow^* \ldots \rightarrow^* t_r$, for some $r \geq 1$, then there are two cases. In the first case there exists a descendant s' of s, with $s' \neq s$, and a set S of following siblings of s' (hence still descendants of s) such that $(T, \nu_\psi, s' \cup S) \models \psi$, for some valuation ν_ψ of variables of ψ. The other case is that $(T, \nu_j, s) \models t_j$ for some valuation ν_j of nodes of t_j, for each $j \in [1, r]$;
  - otherwise there exists a descendant s' of s, with $s' \neq s$, and a set S of following siblings of s' such that $(T, \nu_\psi, s' \cup S) \models \psi$, for some valuation ν_ψ of variables of ψ.

In all cases, thanks to the fact that φ is a Codd incomplete tree, each variable (both from $\mathcal{V}_{\text{node}}$ and $\mathcal{V}_{\text{attr}}$) occurs only once in φ, therefore all valuations ν_β, ν_ψ ad ν_j are over a distinct set of variables. This implies that there exists an overall valuation ν of variables of φ. such that $(T, \nu, s) \models \varphi$. A similar argument proves, in the case that $\varphi = t\ \theta\ f^<$, that $S(\varphi) = [\![\varphi]\!]^T$.

Correctness of sets $S_{desc}(\varphi)$ and $S_{sib}(\varphi)$ follows directly from correctness of $S(\varphi)$. This concludes the proof of the lemma. □

The algorithm for checking $T \in Rep(t)$ computes sets $[\![\varphi]\!]^T$, $[\![\varphi]\!]^T_{desc}$ and $[\![\varphi]\!]^T_{sib}$, for each ordered subformula φ of t in a bottom-up fashion. We denote by d_{max} the depth of the leaves in the parse tree of t. For each ordered formula φ, we will denote by $Sub(\varphi)$ the set of all ordered subformulae of φ. The algorithm is reported below, it uses set variables $S(\varphi)$, $S_{desc}(\varphi)$ and $S_{sib}(\varphi)$ associated with each $\varphi \in Sub(t)$. Moreover for each $\varphi \in Sub(t)$, the set of triples $\{(S(\psi),\ S_{desc}(\psi),\ S_{sib}(\psi)) \mid \psi \in Sub(\varphi)\}$ is denoted by L_φ:

Algorithm MEMB(T, t)
Input: A complete tree T; a *Codd* incomplete tree t
Output: Is T in $Rep(t)$?

---

begin
  for $\varphi \in Sub(t)$ **do**
    $S(\varphi) := \emptyset$;
    $S_{desc}(\varphi) := \emptyset$;
    $S_{sib}(\varphi) := \emptyset$;
  enddo

  for $i = d_{max}$ to 1 **do**
    for each $\varphi \in Sub(t)$ of depth i **do**
      $(S(\varphi), S_{desc}(\varphi), S_{sib}(\varphi)) :=$ SEM(T, φ, L_φ);
    enddo
  enddo

  return $S(t) \neq \emptyset$;
end

Correctness of SEM proves that the set $S(\varphi)$ computed at each step coincides with $[\![\varphi]\!]^T$. Therefore in the end of the computation, $S(t)$ is nonempty if and only if there exists a node s of T such that $(T, \nu, s) \models t$, for some valuation ν of variables of t; that is, if and only if $T \in Rep(t)$.

Each tree subformula in $Sub(t)$ having node formula β and maximal tree subformulae $Sub_t(|\varphi|)$ contributes to the running time of the algorithm MEMB with cost $O(|T| \cdot (|\beta| + |Sub_t(\varphi)|)$; while each subformula in $Sub(t)$ of the form $f^<$ contributes with cost $O(|T|)$. So on the whole MEMB runs in time $O(|T| \cdot |t|)$. This shows that MEMBERSHIP for incomplete trees under Codd interpretation is in PTIME.

Finally, we deal with MEMBERSHIP for incomplete DOM-trees. Let T be a complete tree and t an incomplete DOM-tree. By definition, $\underline{rel}(t)$ and T are both two-sorted relational structures over the vocabularies τ_{Σ_t, A_t} and τ_{Σ_T, A_T}, where Σ_t (A_t) and Σ_T (A_T) are respectively the sets of labels (attributes) occurring in t and T. Next, we denote R^t and R^T the relation R in $\underline{rel}(t)$ and in T, respectively.

By Proposition 4.2, T belongs to $Rep(t)$ if and only if there exists a homomorphism from $\underline{rel}(t)$ to T. Thus, in the case of DOM-trees, since no node variable occurs in $\underline{rel}(t)$,

1. $R^t \subseteq R^T$ for every R in $\{E, NS, E^*, NS^*, (P_\ell)_{\ell\in\Sigma_t}, \mathit{Root}, \mathit{Leaf}, \mathit{FC}, \mathit{LC}\}$;
2. there exists a function m from $\mathcal{D}\cup\mathcal{V}_{\text{attr}}$ to $\mathcal{D}$ such that for every $a \in A_t$ and every (n, x) in $A^t_{@a}$, $m(x) = v$, where v is such that (n, v) in $A^T_{@a}$.

Clearly, 1 can be checked in polynomial time in the size of $\underline{re\ell}(t)$. The same holds for 2, since it can be trivially reduced to checking the satisfiability of a set of equalities. Hence, MEMBERSHIP can be solved in polynomial time. This ends the proof of Theorem 6.1. □

7 Query answering

For relational databases, we know that unions of conjunctive queries can be efficiently evaluated over databases with nulls. One just uses the naïve evaluation, which treats nulls as if they were simply different elements of the domain, and then discards tuples that contain nulls from the output. Naïve evaluation correctly computes certain answers [26] and has the same complexity as the usual conjunctive query evaluation. Once negation is added to queries, or the representation mechanism changes, the complexity quickly rises [3].

We want to find classes of queries and incomplete representations that admit tractable query evaluation for computing certain answers. The first obstacle is that for XML queries that produce trees as outputs, the notion of certain answers is far from clear. So for now, since our goal is to broadly outline the tractability boundary, we look at XML queries that produce tuples of values (this, of course, includes Boolean queries). Once we define a query language, we present a few results that rule out several features as immediately leading to intractability. Then we define a class of *rigid* incomplete trees and show that a natural analog of unions of conjunctive queries admits tractable naïve evaluation over them.

7.1 A simple query language

We shall use queries whose free variables range over the domain of attribute values, and thus their results are usual relations. We start with conjunctive queries over trees. These are essentially standard (see, e.g., [10, 25]). We express them in our syntax for incomplete trees, and add existential quantification over variables from $\mathcal{V}_{\text{attr}}$. That is, conjunctive queries $\mathcal{CQ}$ are of the form $q(\bar{x}) = \exists\bar{y}\ t_q(\bar{x}, \bar{y})$, where t_q is an incomplete tree, and $\bar{x}, \bar{y}$ list variables from $\mathcal{V}_{\text{attr}}$. Their semantics on complete trees T is defined as

$$q(T) \;=\; \left\{ \nu_{attr}(\bar{x}) \;\middle|\; \begin{array}{l} (T, \nu, s) \models t_q \;\text{ for some node } s \\ \text{and valuation } \nu = (\nu_{node}, \nu_{attr}) \end{array} \right\}.$$

Recall that in incomplete trees we omit node variables for notational convenience; the semantics of $q(\bar{x})$ of course assumes existential quantification over all node variables.

As our language $\mathcal{UCQ}$ we take unions of conjunctive queries:

$$q_1(\bar{x}) \;\cup\; \ldots \;\cup\; q_k(\bar{x})$$

For (unions of) conjunctive queries, we use the notation $\mathcal{UCQ}(\mathit{structure})$ or $\mathcal{CQ}(\mathit{structure})$, where *structure* refers to the structural information used in incomplete trees t_q. For example, $\exists y\ r\langle \ell_1[@a = x] \to \ell_2[@b = y]\rangle$ is a $\mathcal{CQ}(\downarrow, \to)$-query that returns values of the $@a$ attribute of ℓ_1-children of r that have an ℓ_2-labeled next sibling with a $@b$ attribute.

For $\mathcal{UCQ}$ queries we can define the notion of certain answers since these queries produce relations:

$$\mathit{certain}(q, t) \;=\; \bigcap\{q(T) \mid T \in \mathit{Rep}(t)\}.$$

The main computational problem we consider here is:

PROBLEM:	QUERYANSWERING(q)
INPUT:	an incomplete tree description t, a tuple $\bar{a}$
QUESTION:	is $\bar{a} \in \mathit{certain}(q,t)$?

We also define $\mathit{certain}_d(q,t)$ as $\bigcap\{q(T) \mid T \models d \text{ and } T \in \mathit{Rep}(t)\}$, and a problem QUERYANSWERING(q,d) (query answering with DTDs) where the question is whether $\bar{a} \in \mathit{certain}_d(q,t)$.

We shall also deal with certain answers for Boolean queries (i.e., queries $\exists\bar{y}\, t_q(\bar{y})$ and their unions), and extend the notion of certain answers to them in the standard way. We can code the result of a Boolean query q as a set, with the empty set standing for *false*, and the set $\{()\}$ containing the empty tuple standing for true. Then the definition above applies; of course in this case $\mathit{certain}(q,t)$ is either $\emptyset$ or $\{()\}$, so we can interpret $\mathit{certain}(q,t)$ as *false* or *true*. If $\mathit{Rep}(t)$ is empty, then $\mathit{certain}(q,t)$ is true, since universal quantification over the empty set evaluates to true.

A fragment of the language, namely $\mathcal{UCQ}(\downarrow,\downarrow^*,\|)$, was considered in the study of query answering in XML data exchange [7]. We first provide an upper bound on the complexity of query answering. We show that a counterexample to $\bar{a} \in \mathit{certain}(q,t)$, i.e., a complete tree T so that $\bar{a} \notin q(T)$ can be chosen to be of polynomial size in t and $\bar{a}$. The technique is similar to the "cutting" technique of Theorem 5.1, and the proof is given in the appendix.

Theorem 7.1. *Both* QUERYANSWERING*(q) and* QUERYANSWERING*(q,d) are in* coNP *for all $q \in \mathcal{UCQ}$ and all d.*

7.2 Intractable cases of query answering

We now show that query answering could be intractable, even for unions of conjunctive queries. This contrasts sharply with the relational case, where all unions of conjunctive queries can be evaluated in PTIME.

We can obtain several intractability results by using hardness results for consistency. Note that if we have a class of incomplete trees over which CONSISTENCY is NP-hard, and a class of queries that includes a query false in all trees, then over these classes of incomplete trees and queries, QUERYANSWERING is coNP-hard. This follows from the fact that $\mathit{certain}(\mathit{false},t) = \mathit{true}$ if and only if $\mathit{Rep}(t) = \emptyset$.

With both DTDs and markings, it is easy to write unsatisfiable queries (e.g., $r\langle a\rangle$, where a cannot appear under the root according to the DTD, or $\_\langle \_^{lc} \to \_^{fc}\rangle$ without DTDs). Hence, we have

Corollary 7.2.
- *There exists a DTD d and a query $q \in \mathcal{CQ}(\downarrow)$ such that* QUERYANSWERING*(q,d) is* coNP*-complete over $(\downarrow,\|)$-incomplete trees.*
- *For the classes of $(\downarrow,\to,\star,\mu)$- and $(\downarrow,\downarrow^*,\star,\mu)$-incomplete trees (where $\star$ is either $\|$ or $\to^*$), there exist queries q that use markings such that* QUERYANSWERING*(q) is* coNP*-complete.*

Thus, having DTDs, or markings in trees and queries, immediately gives us coNP-hardness of query answering. But coNP-hardness can occur without DTDs and without markings in queries (and sometimes even without markings in both trees and queries).

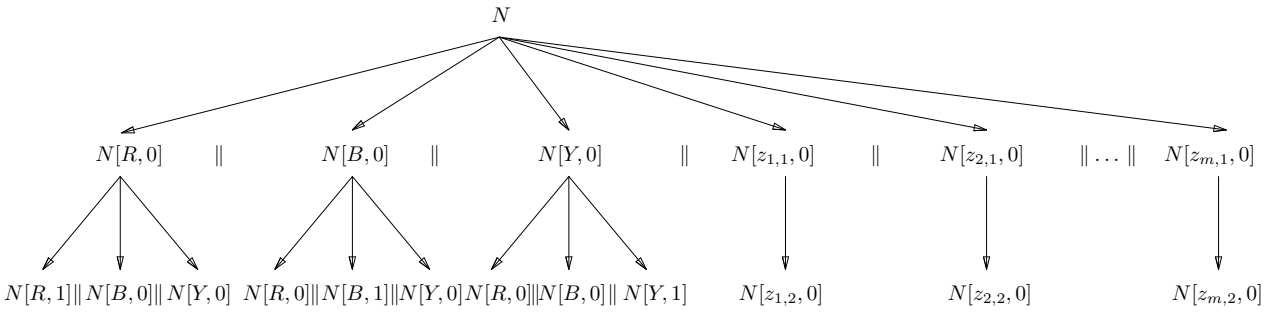
Theorem 7.3. *There is a query $q \in \mathcal{CQ}(\downarrow,\to)$ such that* QUERYANSWERING*(q) is* coNP*-hard over $(\downarrow,\|)$-incomplete trees.*

Moreover, the problem QUERYANSWERING*(q) is* coNP*-hard for $(\downarrow,\|,\downarrow^*,\mu)$-incomplete trees without attributes and $\mathcal{CQ}(\downarrow)$ queries, and for $(\downarrow,\|,\to,\mu)$-incomplete trees without attributes and $\mathcal{CQ}(\downarrow,\to)$ queries.*

Proof. We prove the first statement here (about $\mathcal{CQ}(\downarrow, \rightarrow)$ queries and $(\downarrow, \|)$-incomplete trees) and show the other two reductions in the appendix.

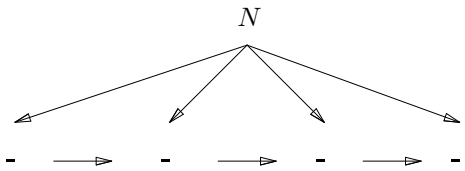
The proof is by reduction from 3-Colorability. Let $G = \langle V, E \rangle$ be a directed graph, with $V = \{v_1, ..., v_n\}$ and $E = \{e_1, ..., e_m\}$. We show how to build a $(\downarrow, \|)$-incomplete tree t from G and a fixed boolean query q in $\mathcal{CQ}(\downarrow, \rightarrow)$ such that *certain*(q, t) evaluates to *false* if and only if G is 3-colorable.

Let t be the following incomplete tree:

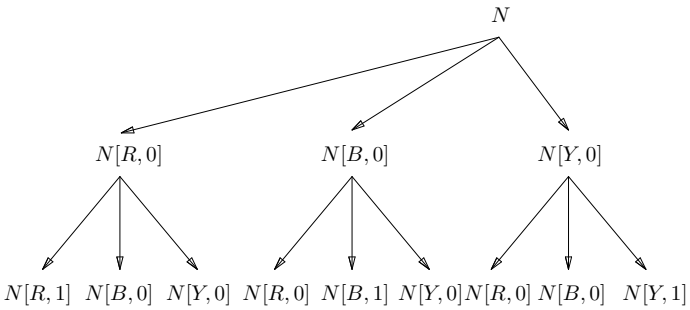


where we annotated every node with $l[C, X]$ to denote that the node is labeled l and has an attribute @color whose value is C and an attribute @distinct whose value is X. Moreover, variables $z_{i,j}$ are defined as follows. We associate a distinct variable to each vertex in V. For each edge $e_i = (v_{i_1}, v_{i_2})$ in E we denote as $z_{i,1}$ and $z_{i,2}$ the variables associated to vertices v_{i_1} and v_{i_2}, respectively. Intuitively the last m children of the root, with their children, encode the edges of the graph.

Now, let q be the boolean query given by the following incomplete tree t_q:



We show that *certain*(q, t) is *false* if and only if G is 3-colorable. In both directions of the proof we refer to the following complete tree T:



Assume first that G is 3-colorable. Then the complete tree T is in $Rep(t)$. In fact if we let $c : V \rightarrow \{R, G, B\}$ be a 3-coloring of G, there exists a homomorphism (h_{node}, h_{null}) from $\underline{rel}(t)$ to T such that $h_{null}(z_{i,j}) = c(v_{i_j})$, where v_{i_j} is the vertex of G whose corresponding variable is $z_{i,j}$.

On the other hand T does not satisfy the query q, since there exist no N-labeled node of T having four distinct children. It follows that *certain*(q, t) is *false*.

Now assume that *certain*(q, t) evaluates to *false*. Then there exists a tree $T' \in Rep(t)$ with no homomorphism from $\underline{rel}(t_q)$. As a consequence, no N-labeled node of T' has more than three children.

Let $\bar{h} = (h_{node}, h_{null})$ be a homomorphism from $\underline{rel}(t)$ to T'; let s_0 be the image according to h_{node} of the root of t. From the fact that $\bar{h}$ is a homomorphism from $\underline{rel}(t)$ to T', and the fact that each N-labeled node of T' has at most three children, it follows that the subtree of T' rooted at s_0 is of the form of the tree T depicted above, up to the grandchildren of s_0.

Then let s_1, s_2 and s_3 be the children of s_0 in T'. In t let $x_1, \ldots, x_m$ be the node ids of the last m children of the root of t, as depicted above, and let y_i be the node variable of the unique child of x_i, for $i \in [1, m]$. Then for each $i \in [1, m]$, the homomorphism h_{node} maps x_i into a node $s \in \{s_1, s_2, s_3\}$ of T'. Moreover h_{node} maps y_i into one of the children of s in T'. Clearly y_i can only be mapped to children of s whose @distinct attribute is 0; these coincide with the children of s whose @color attribute is different from the @color attribute of s.

It follows that for each $i \in [1, m]$, the value of $h_{null}(z_{i,1})$ is in $\{R, G, B\}$, the value of $h_{null}(z_{i,2})$ is in $\{R, G, B\}$, and $h_{null}(z_{i,2}) \neq h_{null}(z_{i,1})$.

Then a 3-coloring for G can be defined by assigning to each vertex v the color $h_{null}(z)$, where z is the variable associated to v.

This completes the proof. □

But so far these results do not say much about the transitive-closure axes in incomplete trees. We now show that with $\downarrow^*$ or $\rightarrow^*$, answering unions of conjunctive queries is coNP-hard. Both reductions are from 3-colorability, with full details in the appendix.

Theorem 7.4.
- *There is a query $q \in \mathcal{UCQ}(\downarrow, \|)$ such that* QueryAnswering*(q) over $(\downarrow, \rightarrow, \downarrow^*)$-incomplete trees is* coNP*-complete.*
- *There is a query $q \in \mathcal{UCQ}(\downarrow, \rightarrow, \rightarrow^*)$ such that* QueryAnswering*(q) over $(\downarrow, \rightarrow, \rightarrow^*)$-incomplete trees is* coNP*-complete.*
- *Both results hold for incomplete DOM-trees as well.*

In the presence of DTDs, we have cases of coNP-hard query answering for very simple queries over incomplete DOM-trees, as the following result shows. The proof is yet another adaptation of a 3-colorability reduction, and we give it in the appendix.

Proposition 7.5. *There exists a DTD d and a query $q \in \mathcal{CQ}(\downarrow, \|)$ such that* QueryAnswering*(q, d) is* coNP*-complete for $(\downarrow, \downarrow^*, \|)$-incomplete DOM-trees.*

7.3 Tractable case: rigid incomplete trees

So far, we have seen that the following features quickly lead to the intractability of query answering for (unions of) conjunctive queries:

1. DTDs; and
2. structural information: transitive-closure axes $\downarrow^*$ and $\rightarrow^*$; union; and markings.

We now exclude these features and obtain a tractable class with respect to query answering. That is, we restrict ourselves to incomplete trees that have neither the transitive closures of axes nor union $\|$ nor markings. We call them *rigid incomplete trees*; they are defined by the grammar:

$$\begin{array}{rcl} t & := & \beta\langle f \rangle \\ f & := & \varepsilon \mid t \rightarrow f \end{array} \tag{4}$$

where node ids are all distinct variables, and markings are not allowed in node descriptions β. This definition mimics (1) except that node descriptions use variables instead of node ids, and may have nulls as values of attributes and wildcard as labels.

Note that each rigid incomplete tree t is consistent. Note also the following property of rigid incomplete trees t: if $h = (h_1, h_2)$ is a homomorphism from t into a complete tree $T \in Rep(t)$, then h_1 is the isomorphism between the tree reduct of t and $h_1(t)$, which is a subtree of T (by tree reducts we mean the structures obtained by deleting relations mentioning attributes, i.e., the pure tree descriptions of t and T).

Our goal is to show that an analog of naïve evaluation will compute certain answers for unions of conjunctive queries over such incomplete trees. We define *naïve-evaluation* as follows. First, each conjunctive query $q(\bar{x}) = \exists \bar{y}\ t_q(\bar{x}, \bar{y})$ is turned into a usual relational conjunctive query by taking $\underline{rel}(t_q)$ and viewing it as a tableau for a query, where $\bar{x}$ are distinguished variables. We shall denote this query by $\underline{rel}(q)_{\bar{x}}$. We then consider the input t, and transform $\underline{rel}(t)$ into $\underline{rel}^*(t)$ by adding reflexive-transitive closures of E and NS.

Then *naïve_eval*(q, t) is the result of evaluating the relational conjunctive query $\underline{rel}(q)_{\bar{x}}$ on the relational database $\underline{rel}^*(t)$ naïvely, and then dropping tuples with nulls. We refer to the result as *naïve_eval*(q, t). This extends to unions of conjunctive queries, simply by taking $\bigcup_i$ *naïve_eval*(q_i, t).

We illustrate this by an example. Suppose we have a query

$$q(x) \ = \ \exists y\ r(n_0)\langle \ell(n_1)[@a = x] \rightarrow^* \_(n_2)[@b = y]\rangle$$

asking for values of the $@a$-attributes of ℓ-children of r-nodes that have a younger sibling with the $@b$-attribute. In the tableau, we shall have tuples (n_0, n_1) and (n_0, n_2) for E, one tuple (n_1, n_2) for NS^*, node n_0 is in P_r and n_1 is in P_ℓ, and pairs $(n_1, x), (n_2, y)$ are in $A_{@a}$ and $A_{@b}$, resp. Since x is the only distinguished variable, this tableau generates a relational conjunctive query $q'(x)$:

$$\exists n_0, n_1, n_2, y\ E(n_0, n_1) \wedge E(n_0, n_2) \wedge NS^*(n_1, n_2) \wedge P_r(n_0) \wedge P_\ell(n_1) \wedge A_{@a}(n_1, x) \wedge A_{@b}(n_2, y).$$

Now suppose we have an incomplete tree

$$t \ = \ r\langle \ell[@a = 1] \rightarrow \ell[@a = u] \rightarrow \ell'[@b = v]\rangle$$

By introducing node variables n_0' for the root and n_1', n_2', n_3' for three children of the root, we create $\underline{rel}(t)$, which has pairs (n_1', n_2') and (n_2', n_3') in NS. By computing $\underline{rel}^*(t)$ we put those pairs, as well as (n_i', n_i') and (n_1', n_3') in NS^*. Evaluating q' naïvely over $\underline{rel}^*(t)$ yields $\{1, u\}$. Eliminating null u, we conclude that *naïve_eval*$(q, t) = \{1\}$. In this case, it is easy to see that $\{1\}$ is the set of certain answers. This correspondence works for all rigid incomplete trees.

Theorem 7.6. *Let t be a rigid incomplete tree, and q a query from $\mathcal{UCQ}$ that does not use markings. Then*

$$certain(q, t) \ = \ naïve\_eval(q, t).$$

In particular, evaluating no-marking queries over rigid incomplete trees has DLOGSPACE *data complexity.*

Proof. In the proof, we shall need to refer explicitly to node variables in queries, so we assume that queries $q(\bar{x})$ are given by incomplete trees $t_q(\bar{n}, \bar{x}, \bar{y})$, where $\bar{n}$ ranges over $\mathcal{V}_{\text{node}}$ and $\bar{x}, \bar{y}$ over $\mathcal{V}_{\text{attr}}$. Hence the query is $q(\bar{x}) = \exists \bar{n}\ \exists \bar{y}\ t_q(\bar{n}, \bar{x}, \bar{y})$ (previously we implicitly assumed existential quantification over all the node variables mentioned in t_q).

The idea of the proof is first to reduce the case of $\mathcal{UCQ}$ queries to $\mathcal{CQ}$ queries, and then, by means of a relational translation, apply the relational results on naïve evaluation. For the first step, we need a lemma.

Lemma 7.7. *If q_1, q_2 are two $\mathcal{UCQ}$ queries and t is a rigid incomplete tree, then $certain(q_1 \cup q_2, t) = certain(q_1, t) \cup certain(q_2, t)$.*

Proof of Lemma 7.7. Since $\mathit{certain}(q_1, t) \cup \mathit{certain}(q_2, t) \subseteq \mathit{certain}(q_1 \cup q_2, t)$ is obvious, we prove the $\supseteq$ inclusion. Suppose $\bar{a} \in \mathit{certain}(q_1 \cup q_2, t)$ but $\bar{a} \notin \mathit{certain}(q_1, t) \cup \mathit{certain}(q_2, t)$. Then we can find two trees $T, T' \in \mathit{Rep}(t)$ such that $\bar{a} \in q_1(T)$, $\bar{a} \notin q_2(T)$ and $\bar{a} \notin q_1(T')$, $\bar{a} \in q_2(T')$. Let $h = (h_1, h_2)$ be a homomorphism from $\underline{rel}(t)$ to T. Let T_0 be the restriction of T to the nodes in the image of h_1. By the observation made earlier, $T_0 \in \mathit{Rep}(t)$ and the tree reducts of t and T_0 are isomorphic. By the monotonicity of q_2, we have $\bar{a} \notin q_2(T_0)$. Hence $\bar{a} \in q_1(T_0)$, for otherwise we would have $\bar{a} \notin q_1(T_0) \cup q_2(T_0)$ and $\bar{a} \notin \mathit{certain}(q_1 \cup q_2, t)$. Thus, we can replace T by T_0. We apply the same reasoning to T' and replace it by a tree T'_0 whose tree reduct is isomorphic to that of t (and thus to that of T_0). Furthermore, we can assume without loss of generality that the node ids in T'_0 and T'_0 are the same (since they are all distinct, and are existentially quantified in queries).

Now fix a valuation ν_{attr} of nulls in t so that it assigns to each null a distinct value in $\mathcal{D}$ that is different from any constant mentioned in t and $\bar{a}$. Let $\nu_{attr}(T_0)$ stand for the tree obtained by applying ν_{attr} to T_0, that is, replacing the second component of the homomorphism $t \to T_0$ with ν_{attr}. Then $\bar{a} \notin q_2(\nu_{attr}(T_0))$; if it were in $q_2(\nu_{attr}(T_0))$, the witnesses for existential quantifiers in q_2 would have witnessed $\bar{a} \in q_2(T_0)$ as well. Likewise, $\bar{a} \notin q_1(\nu_{attr}(T'_0))$. But notice that $\nu_{attr}(T_0) = \nu_{attr}(T'_0)$, and, for this tree T'', we have $T'' \in \mathit{Rep}(t)$. Hence, $\bar{a} \notin q_1(T'') \cup q_2(T'')$ and thus it is not a certain answer to $q_1 \cup q_2$ over t. This contradiction proves Lemma 7.7.

Remark. Note that we only required monotonicity of queries for this lemma.

Lemma 7.7 implies that it suffices to prove the result for a single $\mathcal{CQ}$ query $q(\bar{x}) = \exists \bar{n}\, \exists \bar{y}\, t_q(\bar{n}, \bar{x}, \bar{y})$. Recall that $\underline{rel}(q)_{\bar{x}}$ is the conjunctive query, obtained by viewing $\underline{rel}(q)$ as a tableau, with distinguished variables $\bar{x}$.

First we note that $q(T) = \underline{rel}(q)_{\bar{x}}(T)$ for every such query $q(\bar{x})$. Indeed, if $\bar{a} \in q(T)$, then for some valuation ν and a node s we have $(T, \nu, s) \models t_q(\bar{n}, \bar{a}, \bar{y})$, and thus $T \in \mathit{Rep}(t_q(\bar{n}, \bar{a}, \bar{y}))$. By Proposition 4.2, we conclude that there is a homomorphism $\underline{rel}(t_q(\bar{n}, \bar{a}, \bar{y})) \to T$, which witnesses $\bar{a} \in \underline{rel}(q)_{\bar{x}}(T)$. The other direction (i.e., if $\bar{a} \in \underline{rel}(q)_{\bar{x}}(T)$ then $\bar{a} \in q(T)$) is the same. Hence $q(T) = \underline{rel}(q)_{\bar{x}}(T)$. Therefore,

$$\mathit{certain}(q, t) \;=\; \bigcap\{q(T) \mid T \in \mathit{Rep}(t)\} \;=\; \bigcap\{\underline{rel}(q)_{\bar{x}}(T) \mid T \in \mathit{Rep}(t)\}. \tag{5}$$

Next, we write $T \in \mathit{Rep}_{\text{1-1}}(t)$ if $(T, \nu, r) \models t$, where ν_{node} is a 1-1 onto map from $\bar{n}$ to the node domain of T (and thus r has to be the the root). Since rigid incomplete trees are consistent (by Theorem 5.4), we have $\mathit{Rep}_{\text{1-1}}(t) \subseteq \mathit{Rep}(t)$. Furthermore, by the observation made after the definition of rigidity, for every $T \in \mathit{Rep}(t)$ there is $T_1 \in \mathit{Rep}_{\text{1-1}}(t)$ so that T_1 is contained in T as a relational structure. By monotonicity of conjunctive queries and (5), this implies

$$\bigcap\{\underline{rel}(q)_{\bar{x}}(T) \mid T \in \mathit{Rep}_{\text{1-1}}(t)\} \;=\; \bigcap\{\underline{rel}(q)_{\bar{x}}(T) \mid T \in \mathit{Rep}(t)\}. \tag{6}$$

Let $\overline{m}$ be the set of all node variables used in t (recall that each occurs exactly once), and let $\bar{\imath}$ be a tuple of distinct node ids of the same length as $\bar{n}$. Let $t_{\bar{\imath}}$ be obtained by changing $\overline{m}$ to $\bar{\imath}$. For every two trees $T', T'' \in \mathit{Rep}_{\text{1-1}}(t)$ that only differ in their node ids the output of $\underline{rel}(q)_{\bar{x}}$ is the same (as all node variables are quantified existentially). Thus, in the left-hand side of (6), we can fix node ids in t. Therefore,

$$\bigcap\{\underline{rel}(q)_{\bar{x}}(T) \mid T \in \mathit{Rep}_{\text{1-1}}(t)\} \;=\; \bigcap\{\underline{rel}(q)_{\bar{x}}(T) \mid T \in \mathit{Rep}_{\text{1-1}}(t_{\bar{\imath}})\} \tag{7}$$

where $T \in \mathit{Rep}_{\text{1-1}}(t_{\bar{\imath}})$ means that that $T \in \mathit{Rep}_{\text{1-1}}(t)$ by the valuation that sends node variables to $\bar{\imath}$.

Recall that for relational databases, we use notation $\mathit{Rep}_{\text{CWA}}(R)$ for complete databases obtained by applying valuations to nulls (while $\mathit{Rep}(R)$, under the open world assumption, stands for complete databases that contain results of valuations applied to nulls). Furthermore, $\mathit{certain}_{\text{CWA}}(Q, R)$ stands for $\bigcap\{Q(R') \mid R' \in \mathit{Rep}_{\text{CWA}}(R)\}$.

Now that node variables have been replaced by constants in (7), it is easy to see that $Rep_{1\text{-}1}(t_{\bar{\imath}}) = Rep_{\text{CWA}}(\underline{rel}^*(t_{\bar{\imath}}))$ for incomplete trees that do not use the wildcard (i.e., trees in which the labeling predicates cover the entire domain). Indeed, since all node ids are constants, one can only apply valuation to nulls, and due to the rigidity of t, the only structural information that needs to be added in $T \in Rep(t_{\bar{\imath}})$ is the reflexive-transitive closures and the unary relations. Hence, combining (5)–(7), we derive

$$certain(q,t) \;=\; certain_{\text{CWA}}(\underline{rel}(q)_{\bar{x}}, \underline{rel}^*(t_{\bar{\imath}})). \tag{8}$$

Equation (8) continues to be true for trees that use wildcard. Indeed, in this case one just changes the definition of $Rep_{1\text{-}1}(t)$ slightly so that it does not assign any labeling predicate to wildcard-labeled nodes. Since the domain of labels is infinite, it is easy to see that (6) and (7) remain true (by using trees in *Rep* in which labels for wildcard-labeled nodes are those not used elsewhere in the tree nor in the query). Hence (8) remains true.

Now we show how (8) implies the result. By [26], for evaluation of unions of conjunctive queries, there is no difference between $certain(Q,R)$ and $certain_{\text{CWA}}(Q,R)$ and both are obtained by relational naïve evaluation. Hence, from (8), $certain(q,t)$ is obtained by evaluating naïvely $\underline{rel}(q)_{\bar{x}}$ over $\underline{rel}^*(t_{\bar{\imath}})$. But since for every two tuples of node ids $\bar{\imath}_1$ and $\bar{\imath}_2$ the results of evaluating $\underline{rel}(q)_{\bar{x}}$ naïvely over $\underline{rel}^*(t_{\bar{\imath}_1})$ and $\underline{rel}^*(t_{\bar{\imath}_2})$ are the same, we conclude that $certain(q,t)$ is the result of evaluating $\underline{rel}(q)_{\bar{x}}$ naïvely over $\underline{rel}^*(t)$, that is, $certain(q,t) = naïve\_eval(q,t)$.

Finally, notice that once $\underline{rel}^*(t)$ is computed, evaluating a conjunctive query over it can be done in DLOGSPACE (even AC^0), with respect to data complexity. Computing $\underline{rel}^*(t)$ from $\underline{rel}(t)$ can be done in DLOGSPACE as well, since we need to compute reflexive-transitive closures of successor relations and trees, and both are done in DLOGSPACE. The easiest way to see this is to notice that it suffices to compute deterministic transitive closures, i.e., transitive closures over graphs in which each node has at most one outgoing edge. So for trees, we compute the transitive closure of E^{-1}, which has this property (thus getting the ancestor, rather than the descendant relation), and then reverse the edges to compute E^*. Reversing the edges is done in AC^0, and thus the whole procedure has DLOGSPACE data complexity. This completes the proof. □

Note that Theorem 7.6 applies to *incomplete rigid DOM-trees*, defined just as rigid incomplete trees, except that node ids are now all constants. This is because the rigid structure ensures that every homomorphism $h : \underline{rel}(t) \to T$ is one-to-one.

We have seen in Section 7.2 that the tractability of query answering over the class of rigid trees does not withstand the additions of union, descendant, younger-sibling, or markings. It is also easy to construct examples showing that the naïve evaluation fails with these structural additions. For example, consider $t = r\langle a \| b\rangle$ and $q = r\langle a \to^* b\rangle \cup r\langle b \to^* a\rangle$. We know that $certain(q,t) = true$ but $naïve\_eval(q,t)$ produces *false*. To see why Theorem 7.6 restricts to queries without markings, consider a Boolean query $r\langle \_^{fc}\rangle$ and $t = r\langle a\rangle$. Again naïve evaluation produces *false* but the query is true with certainty.

To see the failure of the naïve evaluation over DOM-trees with additional features, consider an $(\downarrow, \to, \|)$-incomplete DOM-tree $t = r(\mathsf{i}_0)\langle a(\mathsf{i}_1) \| a(\mathsf{i}_2)\rangle$ and a query $q = r\langle \_ \to \_\rangle$. Since $\mathsf{i}_1 \neq \mathsf{i}_2$, we know that r has at least two children, and thus $certain(q,t) = true$, but the naïve evaluation returns false. Similarly, if $t' = r(\mathsf{i}_0)\langle (a(\mathsf{i}_1) \to b(\mathsf{i}_2)) \| (a(\mathsf{i}_3) \to b(\mathsf{i}_4))\rangle$, then for the query $q' = r\langle b \to \_ \to \_\rangle$, the certain answers are true, but the naïve evaluation returns false. Note that this is caused by node ids, and the knowledge that nodes are distinct: if we replace node ids from t and t' with variables, then both $certain(q,t)$ and $certain(q',t')$ would become false.

Class of trees	Consistency	Membership	Query Answering for CQs
Incomplete trees	Dichotomy: PTIME or NP-complete (Theorem 5.4); PTIME without markings (Theorem 5.4);	NP-complete (Theorem 6.1); PTIME for Codd (Theorem 6.1)	in coNP (Theorem 7.1); coNP-complete with $\downarrow^*, \rightarrow^*, \|, \mu$ (Section 7.2) in PTIME for rigid trees (Theorem 7.6)
Incomplete trees + DTDs	NP-complete (Theorem 5.20)	same as above	coNP-complete (Corollary 7.2)
Incomplete DOM-trees	PTIME (Theorem 5.21)	PTIME (Theorem 6.1)	PTIME for rigid trees (Theorem 7.6)
Incomplete DOM-trees + DTDs	PTIME without $\downarrow^*$ (Theorem 5.28)	same as above	coNP-complete (Proposition 7.5)

Figure 5: Summary of the main results

8 Overview of tractability restrictions

Figure 5 presents a summary of the main results of the paper. We now review the choices of the key parameters that lead to tractability of the main computational problems. The key parameters in various models of incomplete XML documents were:

1. the presence of schema information;
2. the presence of markings in node descriptions;
3. structural information (i.e., $\downarrow, \downarrow^*, \rightarrow, \rightarrow^*$ and $\|$); and
4. the presence of node ids.

We have seen that the presence of DTDs, and the presence of markings, makes everything significantly more complicated. Even the simplest cases of consistency and query answering become intractable with DTDs and with markings. So it is natural to suggest that key computational problems for XML with incomplete information be considered without restriction to specific schema information.

The lack of complete structural information is another big obstacle to tractability. Introducing structural uncertainty such as transitive-closure axes and union quickly leads to intractability of both consistency and query answering (Theorems 5.4, 7.3, and 7.4). This happens even for unions of conjunctive queries – the class that is well-behaved with respect to incomplete relational databases.

To achieve tractable query answering over documents with nulls, one needs to restrict not only the class of queries to unions of conjunctive queries but also the class of structural document descriptions so that a portion of a tree is fully described with the child and next-sibling relations. These are rigid incomplete trees: incompleteness only occurs in attribute values and labelings. Then an analog of relational naïve evaluation finds certain answers.

The case when we have explicit node ids is quite different. We can push tractability boundaries further (especially for consistency analysis), but we do so at the expense of algorithms that are significantly more complicated.

9 Future work

There are several possible directions. First, we have only looked at models based on the open world assumption. In the relational case, both open and closed world assumptions (OWA and CWA) are considered, and in many cases the behavior under the CWA is quite different [38]. Many results presented here work for both OWA and CWA but not all. And some existing models (e.g., [4]), fall between CWA and OWA. We have a few preliminary results on the main computational problems under the CWA, but this is a subject of separate future investigation. We also would like to look at analogs of more expressive representations, such as conditional tables [5, 26] or relational representation techniques such as those in [33] to overcome intractability.

Our understanding of models with node ids is not as complete as our understanding of models without ids. And yet this is a fascinating class, because we saw that tractability boundaries can be pushed much further for it.

We would like to address a number of traditional issues related to incomplete information. One example is constraints over documents with incomplete information. It is expected that in the most general form, query answering and consistency analysis will be undecidable (cf. [6, 12]) but one should expect to find reasonable restrictions for decidability and tractability. Another example is using incomplete information in data integration and exchange tasks.

Acknowledgment We are very grateful to the referees for their careful reading of the paper and numerous helpful comments; we also thank one of the referees for suggesting a simpler proof of Proposition 5.29.

This work started when the third and the fourth authors were at the University of Edinburgh. Part of the work was done while the first author was visiting Edinburgh. We gratefully acknowledge support by the FET-Open grant agreement FOX, number FP7-ICT-233599 (Libkin and Sirangelo), EPSRC grants E005039, F028288, and G049165 (Libkin), FONDECYT grant 11080011 and grant P04-067-F from the Millennium Nucleus Centre for Web Research (Barceló), and project TOCAI.IT (Poggi).

References

[1] S. Abiteboul, O. Duschka. Complexity of answering queries using materialized views. In *ACM Symp. on Principles of Database Systems*, 1998 pages 254–263.

[2] J. Albert, D. Giammarresi, D. Wood. Normal form algorithms for extended context-free grammars. *Theoretical Computer Science*, 267(1-2) (2001), 35-47

[3] S. Abiteboul, P. Kanellakis, G. Grahne. On the representation and querying of sets of possible worlds. *Theoretical Computer Science* 78 (1991), 158–187.

[4] S. Abiteboul, L. Segoufin, V. Vianu. Representing and querying XML with incomplete information. *ACM Trans. Database Syst.*, 31 (2006), 208–254.

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

[6] M. Arenas, W. Fan, L. Libkin. On the complexity of verifying consistency of XML specifications. *SIAM J. Comput.* 38 (2008), 841–880.

[7] M. Arenas, L. Libkin. XML data exchange: consistency and query answering. *Journal of the ACM* 55(2): (2008).

[8] P. Barceló, L. Libkin, A. Poggi, C. Sirangelo. XML with incomplete information: models, properties, and query answering. In *ACM Symp. on Principles of Database Systems*, 2009, pages 237–246.

[9] M. Benedikt, W. Fan, F. Geerts. XPath satisfiability in the presence of DTDs. *Journal of the ACM* 55(2): (2008).

[10] H. Björklund, W. Martens, T. Schwentick. Conjunctive query containment over trees. *DBPL'07*, pages 66–80.

[11] H. Björklund, W. Martens, T. Schwentick. Optimizing conjunctive queries over trees using schema information. *Proc. Conf. on Math. Foundations of Comp. Sci.*, 2008, pages 132–143.

[12] A. Calì, D. Lembo, R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. *ACM Symp. on Principles of Database Systems*, 2003, pages 260-271.

[13] D. Calvanese, G. De Giacomo, M. Lenzerini. Semi-structured data with constraints and incomplete information. In *Description Logics*, 1998.

[14] D. Calvanese, G. De Giacomo, M. Lenzerini. Representing and reasoning on XML documents: a description logic approach. *J. Log. Comput.* 9 (1999), 295–318.

[15] B. ten Cate, C. Lutz. The complexity of query containment in expressive fragments of XPath 2.0. *Journal of the ACM* 56(6): (2009).

[16] S. Cohen, B. Kimelfeld, Y. Sagiv. Incorporating constraints in probabilistic XML. In *ACM Symp. on Principles of Database Systems*, 2008, pages 109–118.

[17] C. Date and H. Darwin. *A Guide to the SQL Standard*. Addison-Wesley, 1996.

[18] C. David. Complexity of data tree patterns over XML documents. In *Proc. Conf. on Math. Foundations of Comp. Sci.*, 2008, pages 278–289.

[19] A. Deutsch, V. Tannen. Reformulation of XML queries and constraints. In *Proc. Intl. Conf. on Database Theory*, 2003, pages 225–241.

[20] Document Object Model (DOM). W3C Recommendation, April 2004. http://www.w3.org/TR/DOM-Level-3-Core.

[21] R. Fagin, Ph. Kolaitis, R. Miller, L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science* 336(1): 89–124 (2005).

[22] W. Fan, L. Libkin. On XML integrity constraints in the presence of DTDs. *Journal of the ACM* 49(3): 368-406 (2002).

[23] D. Figueira. Satisfiability of downward XPath with data equality tests. In *ACM Symp. on Principles of Database Systems*, 2009, pages 197–206.

[24] P. Gardner, G. Smith, M. Wheelhouse, U. Zarfaty. Local Hoare reasoning about DOM. In *ACM Symp. on Principles of Database Systems*, 2008, pages 261–270.

[25] G. Gottlob, C. Koch, K. Schulz. Conjunctive queries over trees. *Journal of the ACM* 53 (2006), 238–272.

[26] T. Imielinski, W. Lipski. Incomplete information in relational databases. *Journal of the ACM* 31 (1984), 761–791.

[27] Y. Kanza, W. Nutt, Y. Sagiv. Querying incomplete information in semistructured data. *J. of Comp. and Syst. Sci.* 64 (2002), 655–693.

[28] P. Kolaitis and M. Vardi. A logical approach to constraint satisfaction. In *Finite Model Theory and its Applications*, Springer 2007, pages 339–370.

[29] M. Lenzerini. Data integration: a theoretical perspective. In *ACM Symp. on Principles of Database Systems*, 2002, pages 233–246.

[30] L. Libkin. *Elements of Finite Model Theory*, Springer, 2004.

[31] W. Martens, F. Neven, Th. Schwentick, G. Jan Bex. Expressiveness and complexity of XML Schema. *ACM Trans. Database Syst.* 31(3): 770-813 (2006).

[32] F. Neven, T. Schwentick. On the complexity of XPath containment in the presence of disjunction, DTDs, and variables. *Logical Methods in Computer Science* 2(3): (2006).

[33] D. Olteanu, C. Koch, L. Antova. World-set decompositions: expressiveness and efficient algorithms. *Theoretical Computer Science* 403 (2008), 265–284.

[34] T. Schwentick. A little bit infinite? On adding data to finitely labelled structures. In *STACS'08*.

[35] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL'06*, pages 41–57.

[36] P. Senellart, S. Abiteboul. On the complexity of managing probabilistic XML data. In *ACM Symp. on Principles of Database Systems*, 2007, pages 283–292.

[37] P. Wood. Containment for XPath fragments under DTD Constraints. In *Proc. Intl. Conf. on Database Theory*, 2003, pages 300–314.

[38] M. Vardi. Querying logical databases. *J. of Comp. and Syst. Sci.* 33 (1986), 142–160.

A Additional Proofs

A.1 Proof of Proposition 4.3

a) From the definition of $Rep(t)$ and Proposition 4.2 it follows that:

$Rep^{\Sigma,A}(t) = \{T \mid T \text{ is a tree of vocabulary } \tau_{\Sigma,A}\} \cap \{T \mid \text{there exists a homomorphism } h \text{ from } \underline{rel}(t) \text{ to } T\}$

Hence, by observing that a homomorphism from an incomplete relational structure to a complete relational structure is a valuation, it easily follows that $\underline{rel}(t)$ represents t.
b) Suppose that $\mathbf{D}$ is such that for every D in $Rep(\mathbf{D})$, D is not a tree. It follows that $Rep(\mathbf{D}) \cap \textsc{Trees} = \emptyset$. Hence, the proposition trivially holds, since $\mathbf{D}$ represents any inconsistent incomplete tree.

Suppose now that $\mathbf{D}$ is such that there exists D in $Rep(\mathbf{D})$ such that D is a tree. We next show how to build an incomplete tree $t_{\mathbf{D}}$ starting from $\mathbf{D}$, that represents $t_{\mathbf{D}}$.

Intuitively, $t_{\mathbf{D}}$ can be built starting from the nodes occurring in D and then, for each of them, defining the forests of its children and descendants as the union of "connected components" of nodes among which at least one node is respectively its child or descendant in $\mathbf{D}$.

More precisely, we proceed as follows. First, we define the root r of $t_{\mathbf{D}}$. From the fact that there exists D in $Rep(\mathbf{D})$ such that D is a tree, we know that there exists at most one $n \in \mathcal{I} \cup \mathcal{V}_{\text{node}}$ such that $n \in Root^{\mathbf{D}}$. Thus, if such a node exists, $r = n$. Otherwise, x is inserted in $Root^{\mathbf{D}}$, where x is a fresh variable in $\mathcal{V}_{\text{node}}$. Second, for every $x' \in \mathcal{I} \cup \mathcal{V}_{\text{node}}$ such that $x' \notin Root^{\mathbf{D}}$ and such that for every y, $(y, x') \notin (E^{\mathbf{D}} \cup E^{*\mathbf{D}})$, (r, x') is inserted in $E^{*\mathbf{D}}$. It is easy to see that the rules above can be applied a finite number of times. Moreover, by applying them, the set of trees belonging to $Rep(\mathbf{D})$ does not change.

We are now ready to define $t_{\mathbf{D}}$. This can be done by setting $t_{\mathbf{D}} = tree(r)$, where for every $n \in \mathcal{I} \cup \mathcal{V}_{\text{node}}$, and every $S \subset \mathcal{I} \cup \mathcal{V}_{\text{node}}$, $tree(n)$ and $forest(S)$ are recursively defined as follows:

- $tree(n) = \beta(n)\langle forest(S_1^c) \| forest(S_2^c) \| \dots forest(S_{m^c}^c)\rangle\langle\langle forest(S_1^d) \| forest(S_2^d) \| \dots forest(S_{m^d}^d)\rangle\rangle$ where:
  - $\beta(n) = \ell^{\mu}(n)[@a_1 = z_1, \dots, @a_m = z_m]$, where
    - $(n, z_i) \in A^{\mathbf{D}}_{@a_i}$ for $i \in [1, m]$,
    - $\ell = l$ if $n \in P_l^{\mathbf{D}}$, and $\ell = \_$ otherwise;
    - $\mu = root$ if $n \in Root^{\mathbf{D}}$, $\mu = leaf$ if $n \in Leaf^{\mathbf{D}}$, $\mu = fc$ if $n \in FC^{\mathbf{D}}$, and $\mu = lc$ if $n \in LC^{\mathbf{D}}$;
  - for every $i \in [1, m^c]$, there exists a node $n^c \in S_i^c$ that is a child of n, i.e., such that $(n, n^c) \in E^{\mathbf{D}}$; moreover, S_i^c is the maximal connected component of nodes that contains n^c, i.e., S_i^c is the maximal subset of $\mathcal{V}_{\text{node}} \cup \mathcal{I}$ such that for every n' in S_i^c there exists a path by means of NS and NS^* either from n' to n^c or from n^c to n';
  - for every $i \in [1, m^d]$, there exists a node $n^d \in S_i^d$ that is a descendant of n, i.e., such that $(n, n^d) \in E^{*\mathbf{D}}$; moreover, S_i^d is the maximal connected component of nodes that contains n^d, i.e., S_i^d is the maximal subset of $\mathcal{V}_{\text{node}} \cup \mathcal{I}$ such that for every n' in S_i^d there exists a path by means of NS and NS^* either from n' to n^d or from n^d to n';
- if S is empty, then $forest(S) = \varepsilon$;
- if S is not empty, $forest(S) = n_1\theta_1 n_2 \| n_3\theta_2 n_4 \| \dots \| n_{2k+1}\theta_k n_{2k}$, for some $k \geq 0$, where
  - $S = \{n_1, n_2, \dots, n_{2k}\}$;
  - $\theta_i = \rightarrow$ if $(n_{2i+1}, n_{2i}) \in NS$, and $\theta_i = \rightarrow^*$ if $(n_{2i+1}, n_{2i}) \in NS^*$.

By construction, it is easy to see that $Rep(\underline{rel}(t_{\mathbf{D}})) = Rep(D) \cap \textsc{Trees}$. Hence, $\mathbf{D}$ represents $t_{\mathbf{D}}$.

A.2 Remaining cases from the proof of Theorem 5.4

CONSISTENCY of $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$ and $(\downarrow, \downarrow^*, \rightarrow^*, fc, lc, leaf)$-incomplete trees

The proof that CONSISTENCY of $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$-incomplete trees without attributes is NP-hard follows the lines of the previous reduction, where the next-sibling axis is replaced by the child axis.

Given an instance (S, K) of the shortest common superstring problem, over alphabet Σ, with $S = \{s_1, \ldots, s_n\}$, we define a $(\downarrow, \downarrow^*, \|, fc, lc, leaf)$-incomplete tree t without attributes over alphabet $\Sigma \cup \{R\}$ with $R \notin \Sigma$. The incomplete tree t is constructed from S and K as follows:

$$t = R(x)\langle t_K \rangle \langle\langle t_{s_1} \| \ldots \| t_{s_n} \rangle\rangle$$

where t_K is the incomplete tree of depth K:

$$t_K = \_(x_1)^{fc,lc}\langle \_(x_2)^{fc,lc} \langle \ldots \langle \_(x_K)^{fc,lc,leaf} \rangle \rangle \rangle$$

and for each string $s = a_1 a_2 \cdots a_m \in S$, the incomplete tree t_s is defined as:

$$t_s = a_1 \langle a_2 \langle \ldots \langle a_m \rangle \rangle \rangle$$

(here node variables are omitted).

We claim that $Rep(t) \neq \emptyset$ if and only if there exists a common superstring of S of length not greater than K. Indeed, assume there exists such a superstring w. As in the previous reduction, if $|w| < K$ we pad w with an arbitrary suffix and obtain a word w' of length K. Let $w' = b_1 \cdots b_K$, then the following complete tree is in $Rep(t)$:

$$T = R(\mathsf{i}_0)\langle b_1(\mathsf{i}_1) \langle \ldots \langle b_K(\mathsf{i}_K) \rangle \rangle \rangle$$

Indeed:

- since the subtree of T rooted at i_1 has depth K and is a linear path, there exists a valuation ν_0 with $\nu_0(x_i) = \mathsf{i}_i$ for each $i \in [1, K]$ such that $(T, \nu_0, \mathsf{i}_1) \models t_K$;
- For each $s \in S$, since $b_1 \cdots b_K$ is a superstring of s, there exists some descendant i_s of i_1 in T and a valuation ν_s of node variables of t_s such that $(T, \nu_s, \mathsf{i}_s) \models t_s$

Now we take a valuation ν mapping the root x of t into i_0, coinciding with ν_0 on node variables of t_K, and with ν_s on node variables of t_s, for each $s \in S$. We have $(T, \nu, \mathsf{i}_0) \models t$ and then $T \in Rep(t)$.

Conversely assume that $Rep(t) \neq \emptyset$; then assume $T \in Rep(T)$ is a tree over some alphabet Σ'. There must exist a node i_0 of T and a valuation ν such that $(T, \nu, \mathsf{i}_0) \models t$. Node i_0 must have label R; moreover there must exist nodes $\mathsf{i}_1, \ldots \mathsf{i}_K$ of T, with $\mathsf{i}_0 \rightarrow \mathsf{i}_1 \rightarrow \ldots \mathsf{i}_K$, such that $\nu(x_j) = \mathsf{i}_j$ for $j \in [1, K]$ (the witnesses for nodes of t_K). Also, since nodes x_j of t_K are all labeled both with *fc* and *lc*, nodes $\mathsf{i}_1, \ldots \mathsf{i}_K$ of T have no siblings. Finally, since node x_K of t_K is also labeled as *leaf*, node i_k of T must be a leaf. All this implies that the subtree of T rooted at i_0 is a linear path: $R(\mathsf{i}_0)\langle b_1(\mathsf{i}_1) \langle \ldots \langle b_K(\mathsf{i}_K) \rangle \rangle \rangle$ for some $b_1 \cdots b_K \in \Sigma'^*$.

On the other hand, for each $s \in S$, the tree T must match t_s in some descendant of i_0. Because $R \notin \Sigma$ this descendant must be different from i_0. Therefore for each $s \in S$ there exists $0 < j \leq K$ such that $(T, \nu, \mathsf{i}_j) \models t_s$, and hence s is a substring of $b_1 \cdots b_K$.

As in the previous reduction, we now modify $b_1 \cdots b_K$ by replacing each symbol not in Σ with an arbitrary symbol of Σ. The resulting string has length K and is still a superstring of all strings of S.

This concludes the reduction.

Consistency of $(\downarrow, \downarrow^*, \rightarrow^*, fc, lc, leaf)$-incomplete trees can be proved NP-hard by slightly modifying the previous reduction: we construct incomplete trees $t_i = \_\langle\langle t_{s_i} \rangle\rangle$ for all $i \in [1, n]$ (where node variables are omitted) and

$$t = R(x)\langle t_K \rightarrow^* t_1 \ldots \rightarrow^* t_n \rangle$$

A slight adaptation of the previous proof shows that $Rep(t) \neq \emptyset$ if and only if S has a common superstring of length at most K.

Polynomial time cases.

Incomplete trees without *fc* markings The converse of Lemma 5.15 holds also for incomplete trees with no *fc* markings. The proof of the following lemma is the dual of the the proof of Lemma 5.19, and is not reported:

Lemma A.1. *Given an* str*-incomplete tree t, where* str *does not contain* fc*, and given a valid chase sequence σ for t, if σ is successful, then t is consistent.*

Incomplete trees with neither $\rightarrow$ nor $\downarrow^*$, and incomplete trees with neither $\rightarrow$ nor *leaf*

Lemma A.2. *Given an* str*-incomplete tree t, where $str \cap \{\rightarrow, \downarrow^*\} = \emptyset$, or $str \cap \{\rightarrow, leaf\} = \emptyset$, and given a valid chase sequence σ for t, if σ is successful, then t is consistent.*

Proof. Let $\sigma = D_0, \ldots D_k$, where $D_0 = rel(t)$ and $D_k = chase_\sigma(t)$. For each $i \in [0, n]$, relation *NS* is empty in D_i, thus no *in-sibling* and no *out-sibling* step is applicable in D_i, for all $i \in [1, n]$.

The structure D_0 trivially satisfies properties 1, 2 and 3; then by Claim 5.16, each D_i in the chase sequence, satisfies properties 1, 2 and 3. Moreover in D_k no chase step is applicable; this implies the following properties of $G_{NS}(D_k)$:

- By property 3 and the fact that no *push-fc* and no *push-lc* step is applicable in D_k, the graph $G_{NS}(D_k)$ does not contain directed cycles.
- By the fact that no *merge-fc* and no *push-fc* step is applicable in D_k, each connected component of D_k contains at most one node in *FC*, and this node has no incoming edges in $G_{NS}(D_k)$.
- By the fact that no *merge-lc* and no *push-lc* step is applicable in D_k, each connected component of D_k contains at most one node in *LC*, and this node has no outgoing edges in $G_{NS}(D_k)$.
- By properties 1 and 2, and the fact that no *push-fc* and no *push-lc* step is applicable in D_k, all nodes of $G_{NS}(D_k)$ not in *FC* nor in *LC* have at most one incoming edge and at most one outgoing edge in $G_{NS}(D_k)$.
- By the fact that no *union-fc* and no *union-lc* steps are applicable in D_k, for each node $x \in adom_{node}(D_k)$ there exists at most one connected component of $G_{NS}(D_k)$ having *E-parent* x containing a node in *FC*, and at most one connected component containing a node in *LC* (although they could be the same component).

As a consequence each connected component of $G_{NS}(D_k)$ consists of a set of disjoint simple paths of NS^*-edges $\{p_1, \ldots p_n\}$ (whose nodes are neither in *FC* nor in *LC*) together with a node $x_{fc} \in FC$ or a node $x_{lc} \in LC$ (or both) and possible edges from x_{fc} to the origins of the p_is, as well as from the destination of the p_is to x_{lc}.

Moreover $t(D_k)$ has the following properties:

- By the fact that no *root* step is applicable in D_k, only the root variable of $t(D_k)$ is possibly in the *Root* relation of D_k.

- By the fact that no *leaf* step is applicable in D_k, only leaf variables of $t(D_k)$ can be in the *Leaf* relation.
- By the fact that no *root-child* step is applicable in D_k, no node is both in the *Root* and the *FC* (or *LC*) relation of D_k.

We now construct a complete tree having a homomorphism from D_k. We let h_0 be an arbitrary mapping from $\mathcal{V}_{\text{attr}}$ to $\mathcal{D}$, being the identity on $\mathcal{V}_{\text{node}}$, $\mathcal{I}$ and $\mathcal{D}$. We let $D = h_0(D_k)$ and remark that D has tree shape and has the same above mentioned properties of D_k.

For each subtree t' of $t(D)$ we show how to construct a tree T and a mapping $\nu : adom_{node}(t') \to \mathcal{I}$, sending the root node variable of t' into the root i of T and satisfying:

- $(T, \nu, \mathsf{i}) \models t'^*$ (recall from the proof of Lemma 5.19 that t'^* is t' after the removal of $\{fc, lc\}$ markings from the root).
- for each $x, y \in adom_{node}(t')$, if (x, y) is an *NS*-edge (resp., *NS**-edge) of $G_{NS}(D)$, then $NS(\nu(x), \nu(y))$ (resp., $NS^*(\nu(x), \nu(y))$) holds in T.

We proceed as in the proof of Lemma 5.19, by induction on the structure of t'. Only the induction step needs to be adapted; we describe it in the rest of the proof.

In the case that t is an *str*-incomplete tree, with $str \cap \{\rightarrow, \downarrow^*\} = \emptyset$, relation E^* is empty in D, so we can assume $t' = \beta\langle t_1 \| \dots \| t_n \rangle$. Otherwise if t is an *str*-incomplete tree, with $str \cap \{\rightarrow, leaf\} = \emptyset$, we let $t' = \beta\langle t_1 \| \dots \| t_n \rangle \langle\!\langle t_{n+1} \| \dots \| t_m \rangle\!\rangle$. In both cases we let $\beta = \ell^\mu(x)[@a_1 = v_1, \dots, @a_p = v_p]$.

Assume also that $x_1, \dots x_m$ are the root node variables of $t_1, \dots t_m$ respectively. Assume we have constructed, for each t_i, $i \in [1, m]$, trees T_i with root ids i_i and valuations $\nu_i : adom_{node}(t_i) \to \mathcal{I}$ preserving edges of $G_{NS}(D)$ in T_i and satisfying $(T_i, \nu_i, \mathsf{i}_i) \models t_i^*$.

We now construct the tree T from subtrees $T_1, \dots, T_m$. Let $C_1, \dots C_l$ be all connected components of $G_{NS}(D)$ having *E-parent* x (components $C_1, \dots C_l$ partition $\{x_i | i \in [1, n]\}$). Similarly let $C_{l+1}, \dots C_k$ be all connected components of $G_{NS}(D)$ having *E**-*parent* x (components $C_{l+1}, \dots C_k$ partition $\{x_i | i \in [n+1, m]\}$). We order nodes of $C_1, \dots C_l$ as follows: take all the disjoint paths obtained by removing possible nodes of *FC* or *LC* from each C_i, $i \in [1, l]$; let these paths be $\{p_1, \dots p_q\}$ in an arbitrary order, and let x_{fc} and x_{lc} the (possible) nodes in $C_1 \cup \dots \cup C_l$ belonging to *FC* and *LC*, respectively. If $x_{fc} \neq x_{lc}$ we take the permutation $x_{fc}\ p_1, \dots p_q\ x_{lc}$ of $x_1 \dots x_m$ (one of x_{fc} and x_{lc}, or both of them, may be missing). If this permutation is $x_{i_1} \dots x_{i_m}$, let f_E be the forest $T_{i_1} \dots T_{i_m}$. Otherwise, if $x_{fc} = x_{lc}$, by the fact that no *fc/lc* step is applicable in D_k, we have that x_{fc} is the only node of $C_1 \cup \dots \cup C_l$. Therefore given that $x_{fc} = x_i$ for some $1 \leq i \leq m$, we let $f_E = T_i$.

Similarly we proceed on each single connected component $C \in \{C_{l+1}, \dots C_k\}$: If x_{fc} and x_{lc} are the possible nodes of C in *FC* and *LC* respectively, and $\{p_1, \dots p_q\}$ are the disjoint paths obtained by removing x_{fc} and x_{lc} from C, we take the permutation $\bar{C} = x_{fc}\ p_1 \dots p_q\ x_{lc}$ of nodes of C. Given that $\bar{C} = x_{i_1} \dots x_{i_r}$ for some $i_1, \dots, i_r$ in $[n+1, m]$, we let $f_{\bar{C}}$ the forest $T_{i_1} \dots T_{i_r}$. We construct trees $T_C = B_C\langle f_{\bar{C}} \rangle$ and a tree $T_{E^*} = B'\langle T_{C_{l+1}} \dots T_{C_k} \rangle$ where B_Cs and B' are arbitrary complete node descriptions with new freshly generated ids.

Let $T_0 = B\langle f_E \rangle$ where the node description B is constructed from β as in the base case, using a node id i distinct from all other ids in T_0. In the case that $\{C_{l+1}, \dots C_k\}$ is empty we take $T = T_0$. Otherwise we take $T = T_1$ where T_1 is constructed as follows. Let $l(\mathsf{i}')\langle\varepsilon\rangle$ an arbitrary leaf of T_0; we construct T_1 by composing T_0 with T_{E^*} in the node $l(\mathsf{i}')\langle\varepsilon\rangle$. That is, T_1 is obtained from T_0 by replacing node $l(\mathsf{i}')\langle\varepsilon\rangle$ with $l(\mathsf{i}')\langle T_{E^*} \rangle$.

It is easy to verify that the mapping ν from t' to T sending x into i and coinciding with ν_i on $adom_{node}(t_i)$ preserves edges of $G_{NS}(D)$ in T. Also, using a similar argument as in the proof of Lemma 5.19, one easily proves that $(T_1, \nu, \mathsf{i}_i) \models t_i$ for each $i \in [n+1, m]$, and $(T_0, \nu, \mathsf{i}) \models t_0^*$ – where t_0 denotes $\beta\langle t_1 \| \dots \| t_n \rangle$.

In the case that t is an *str*-incomplete tree with $str \cap \{\rightarrow, \downarrow^*\} = \emptyset$, we have $T = T_0$, and $t' = t_0$, then $(T, \nu, \mathsf{i}) \models t'^*$.

Otherwise – if t is an *str*-incomplete tree with $str \cap \{\rightarrow, \mathit{leaf}\} = \emptyset$ – we have $T = T_1$. By construction, all relations R different from *Leaf* are such that $R^{T_0} \subseteq R^{T_1}$. Moreover in this fragment, $\underline{rel}(t_0^*)$ has empty *Leaf* relation; therefore the fact that ν (naturally extended to the whole $\mathcal{V}_{\text{node}}, \mathcal{I}, \mathcal{D}$ and $\mathcal{V}_{\text{attr}}$) is a homomorphism from $\underline{rel}(t_0^*)$ to T_0 implies that it is also a homomorphism from $rel(t_0^*)$ to T_1. Thus $(T_1, \nu, \mathsf{i}) \models t_0^*$.

Combining this with the fact that $(T_1, \nu, \mathsf{i}_i) \models t_i$ for each $i \in [n+1, m]$, and the fact that i_i is a descendant of i for each $i \in [n+1, m]$, we have $(T_1, \nu, \mathsf{i}) \models t'^*$, and then $(T, \nu, \mathsf{i}) \models t'^*$ also in this case.

This completes the induction, and proves in particular that there exists a tree T and a mapping $\nu : adom_{node}(t(D)) \rightarrow \mathcal{I}$, sending the root node variable of $t(D)$ into the root i of T and preserving edges of $G_{NS}(D)$ such that $(T, \nu, \mathsf{i}) \models t(D)^*$. In the case that the root of $t(D)$ is marked with *fc* (and therefore not with *root*) we modify T by adding an extra root having i as the only child. As in Lemma 5.19, one proves that in any case $(T, \nu, \mathsf{i}) \models t(D)$; moreover ν preserves edges of $G_{NS}(D)$ in T.

This proves (in both considered fragments) that T and ν satisfy conditions of Lemma 5.18 with D, thus there exists a homomorphism h from D to T. Then $h \circ h_0$ is a homomorphism from D_k (that is, $chase_\sigma(t)$) to T.

The proof of the lemma is concluded by Corollary 5.14. □

Incomplete trees with neither $\|$ nor $\rightarrow^*$ Remark that when $\|$ is not allowed, no union of forests is allowed under the same node of the incomplete tree. In particular this also rules out incomplete trees of the form $\beta\langle f \rangle \langle\!\langle f' \rangle\!\rangle$, even when f and f' only use $\rightarrow$ and $\rightarrow^*$.

Lemma A.3. *Given an* str*-incomplete tree t, where $str \cap \{\|, \rightarrow^*\} = \emptyset$, and given a valid chase sequence σ for t, if σ is successful, then t is consistent.*

Proof. Let $\sigma = D_0, \ldots D_k$, where $D_0 = rel(t)$ and $D_k = chase_\sigma(t)$. Then D_0 satisfies property 4, and by Claim 5.16, each D_i in the chase sequence satisfies property 4.

As a consequence D_k is indeed the relational representation of an *str*-incomplete tree (in the considered fragment); let t' denote this incomplete tree. Moreover in D_k no chase step is applicable. In particular:

- By the fact that no *push-fc* and no *push-lc* step is applicable in D_k, we have the following: in each sub-forest $t_1 \rightarrow t_2 \rightarrow \ldots t_k$ of t', only t_1 possibly contains *fc* markings and only t_k possibly contains *lc* markings.
- By the fact that no *root* step is applicable in D_k, only the root of t' possibly contains *root* markings.
- By the fact that no *leaf* step is applicable in D_k, only leaves of t' possibly contain *leaf* markings.
- By the fact that no *root-child* step is applicable in D_k, no node of t' has a *root* marking together with a *fc* (or *lc*) marking.

Let $\tilde{t}'$ denote the incomplete tree obtained from t' by removing all markings. We now let $T' = \delta(\tilde{t}')$, where δ is the function defined above to treat consistency of incomplete trees without markings. In the case that t' contains root markings we let $T = T'$, otherwise we let T be obtained from T' by an adding an new root node having the root of T' as the only child.

We know that $T' \in Rep(\tilde{t}')$, then in both cases also $T \in Rep(\tilde{t}')$. It is easy to verify that this is preserved when markings are added to $\tilde{t}'$, that is, $T \in Rep(t')$ (using the properties of markings listed above).

Therefore by Proposition 4.2, there exists a homomorphism from D_k to T. The proof of the lemma follows from Corollary 5.14. □

A.3 Remaining cases from the proof of Theorem 5.20

First, we deal with $(\downarrow, \|, \rightarrow)$ -incomplete trees. We define a DTD d_2 and reduce the "shortest common superstring" problem to CONSISTENCY(d_2) for $(\downarrow, \|, \rightarrow)$-incomplete trees.

Given an instance (S, K) of the shortest common superstring problem over alphabet Σ, we let $S = \{s_1 \dots s_n\}$. We define a $(\downarrow, \|, \rightarrow)$-incomplete tree t without attributes over alphabet $\Sigma \cup \{F, L, R\}$ with $\{F, L, R\} \cap \Sigma = \emptyset$:

$$t = R(x)\langle f_K \| f_{s_1} \| \dots \| f_{s_n} \rangle$$

where f_K is the incomplete forest:

$$f_K = F(x_0) \rightarrow \_(x_1) \rightarrow \_(x_2) \rightarrow \dots \_(x_K) \rightarrow L(x_{K+1})$$

having exactly K wildcard nodes. For each string $s = a_1 a_2 \cdots a_m \in S$, the incomplete forest f_s is defined as:

$$f_s = a_1 \rightarrow a_2 \dots \rightarrow a_m$$

(where node variables are omitted). Now let d_2 be the DTD:

$$\begin{array}{lll} R & \rightarrow F\Sigma^* L & \\ F & \rightarrow \varepsilon & \\ L & \rightarrow \varepsilon & \\ a & \rightarrow \varepsilon & \forall a \in \Sigma \end{array}$$

We claim that $Rep_{d_2}(t) \neq \emptyset$ if and only if there exists a common superstring of S of length not greater than K. Indeed, assume there exists such a superstring w; we possibly pad w to length K and get $w' = b_1 \cdots b_K$. $|ww_1| = K$. Let $w' = ww_1 = b_1 \cdots b_K$. We now show that the complete tree:

$$T = R(\mathsf{i})\langle F(\mathsf{i}_0) b_1(\mathsf{i}_1) \dots b_K(\mathsf{i}_K) L(\mathsf{i}_{K+1}) \rangle$$

is in $Rep_{d_2}(t)$. In fact:

- T is valid w.r.t d_2;
- the valuation ν_0 mapping x_j to i_j, for each $j \in [0, K+1]$, is such that $(T, \nu_0, \mathsf{i}_0, \dots, \mathsf{i}_{K+1}) \models f_K$;
- since each $s \in S$ is a substring of $b_1 \cdots b_K$, there exist children $\mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}$ of i in T and a valuation ν_s of node variables of f_s such that $(T, \nu_s, \mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}) \models f_s$.

We now take a valuation ν defined so that $\nu(x) = \mathsf{i}$ and ν coincides with ν_0 on node variables of f_K, and with ν_s on node variables of f_s, for each $s \in S$. This gives $(T, \nu, \mathsf{i}) \models t$.

Conversely assume that $Rep_{d_2}(t) \neq \emptyset$, then there exists a tree $T = R(\mathsf{i})\langle F(\mathsf{i}_0) b_1(\mathsf{i}_1) \dots b_l(\mathsf{i}_l) L(\mathsf{i}_{l+1}) \rangle \in Rep_{d_2}(t)$ for some $b_1 \cdots b_l \in \Sigma^*$. Because i, i_0 and i_{l+1} are the only nodes of T of labels R, F and L, respectively, there must exist a valuation ν of node variables of t such that $(T, \nu, \mathsf{i}) \models t$ and $(T, \nu, \mathsf{i}_0, \dots, \mathsf{i}_{l+1}) \models f_K$. This implies:

- $l = K$;
- for each $s \in S$, there exist nodes $\mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}$ in T such that $(T, \nu, \mathsf{i}_{j+1}, \dots, \mathsf{i}_{j+|s|}) \models f_s$. Consequently $b_1 \cdots b_K$ is a superstring of s.

This shows that $b_1 \cdots b_K$ is a superstring of all strings of S, and completes the reduction.

Finally, we deal with $(\downarrow, \|, \downarrow^*)$-incomplete trees. We reduce the "shortest common superstring" problem to CONSISTENCY(d_3) for some fixed DTD d_3 and for $(\downarrow, \|, \downarrow^*)$-incomplete trees. The same argument as in the previous reduction can be used by replacing the child relation with the next-sibling one.

Given an instance (S, K) of the shortest common superstring problem, over alphabet Σ, with $S = \{s_1, \ldots, s_n\}$, we define a $(\downarrow, \|, \downarrow^*)$-incomplete tree t without attributes over alphabet $\Sigma \cup \{L, R\}$ with $\{L, R\} \cap \Sigma = \emptyset$:

$$t = R(x)\langle t_K \rangle \langle\langle t_{s_1} \| \ldots \| t_{s_n} \rangle\rangle$$

where t_K is the incomplete tree of depth $K + 1$:

$$t_K = \_(x_1)\langle \ldots \langle \_(x_K)\langle L(x_{K+1})\rangle\rangle\rangle$$

and for each string $s = a_1 a_2 \cdots a_m \in S$, the incomplete tree t_s is defined as:

$$t_s = a_1\langle a_2\langle \ldots \langle a_m\rangle\rangle\rangle$$

(with node variables omitted).

Now let d_3 be the DTD:

$$\begin{array}{lll} R & \to \Sigma & \\ a & \to \Sigma | L & \forall a \in \Sigma \\ L & \to \varepsilon & \end{array}$$

We claim that $Rep_{d_3}(t) \neq \emptyset$ if and only if there exists a common superstring of S of length not greater than K. Indeed, assume there exists such a superstring $w \in \Sigma^*$. We pad w to length K and obtain a word $w' = b_1 \cdots b_K$. Then the complete tree:

$$T = R(\mathsf{i}_0)\langle b_1(\mathsf{i}_1)\langle \ldots \langle b_K(\mathsf{i}_K)\langle L(\mathsf{i}_{K+1})\rangle\rangle\rangle\rangle$$

is in $Rep_{d_3}(t)$. In fact:

- T is valid w.r.t d_3;
- the valuation ν_0 such that $\nu_0(x_j) = \mathsf{i}_j$, for all $j \in [1, K+1]$, is such that $(T, \nu_0, \mathsf{i}_1) \models t_K$;
- For each $s \in S$, since $b_1 \cdots b_K$ is a superstring of s, there exists some node i_j of T, with $0 < j \leq K$ and a valuation ν_s of node variables of t_s such that $(T, \nu_s, \mathsf{i}_j) \models t_s$.

As in the previous reductions, take a valuation ν mapping x into i_0, coinciding with ν_0 on node variables of t_K, and with ν_s on node variables of t_s, for each $s \in S$. We have $(T, \nu, \mathsf{i}_0) \models t$ and then $T \in Rep_{d3}(t)$.

Conversely assume that $Rep_{d_3}(t) \neq \emptyset$, then there exists a tree $T = R(\mathsf{i}_0)\langle b_1(\mathsf{i}_1)\langle \ldots \langle b_l(\mathsf{i}_l)\langle L(\mathsf{i}_{l+1})\rangle\rangle\rangle\rangle \in Rep_{d_3}(t)$ for some $b_1 \cdots b_l \in \Sigma^*$. Since i_0 is the only node of T of label R, there must exist a valuation ν of node variables of t such that $(T, \nu, \mathsf{i}_0) \models t$. This implies:

- $(T, \nu, \mathsf{i}_1) \models t_K$ and therefore $l = K$;
- for each $s \in S$, there exists a descendant i_j of i_0 in T such that $(T, \nu, \mathsf{i}_j) \models t_s$. Since $R, L \notin \Sigma$, the id i cannot coincide with i_0 nor with i_{l+1}. Consequently $b_1 \cdots b_K$ is a superstring of s.

This shows that there exists a superstring $b_1 \cdots b_K$ of all strings of S, and concludes the reduction.

A.4 Proof of Theorem 5.28

Before we start with the proof of the theorem, we show that a different problem, that we call Constrained Disjoint Matching, can be solved in polynomial time. The reason why we do this is twofold. On the one hand, this result will be later used in the proof of the theorem. We believe that by proving it separately we can obtain a proof of the main theorem that is more modular and easier to understand. On the other hand, we think that the problem Constrained Disjoint Matching may be of independent interest, and, thus, it is worth stating it separately.

Assume that Σ is a finite alphabet. Let $S(\Sigma) = \{s_1, \ldots, s_{2^{|\Sigma|}-1}\}$ be the set of all nonempty subsets of Σ. Assume that $S(\Sigma)$ is disjoint from Σ. For a string $\bar{u} = u_0 u_1 \cdots u_n$ over alphabet $\Sigma \cup S(\Sigma)$ we say that the string $\bar{w} = w_0 w_1 \cdots w_n$ from Σ^* is an *instantiation* of $\bar{u}$, if for each $1 \leq i \leq n$, $u_i = w_i$ if $u_i \in \Sigma$ and $w_i \in u_i$ otherwise. The problem Constrained Disjoint Matching is defined as follows, where $\mathcal{A}$ is a fixed NFA over alphabet Σ:

Problem:	Constrained Disjoint Matching over $\mathcal{A}$
Input:	A finite set $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ of strings from $(\Sigma \cup S(\Sigma))^*$ and a constraint $C \subseteq W \times W$
Question:	Is there a permutation κ of $\{1, \ldots, n\}$, strings $\bar{u}_0, \bar{u}_1, \ldots, \bar{u}_n$ from Σ^*, and an instantiation $\bar{w}'_i$ of $\bar{w}_i$ $(1 \leq i \leq n)$ such that (1) the string $\bar{u}_0 \bar{w}'_{\kappa(1)} \bar{u}_1 \bar{w}'_{\kappa(2)} \bar{u}_2 \cdots \bar{u}_{n-1} \bar{w}'_{\kappa(n)} \bar{u}_n$ is accepted by $\mathcal{A}$, and (2) if $(\bar{w}_i, \bar{w}_j) \in C$, $i = \kappa(i')$ and $j = \kappa(j')$ $(1 \leq i, i', j, j' \leq n)$, then $i' < j'$?

The intuition behind this problem is as follows. The input consists of n strings $\bar{w}_1, \ldots, \bar{w}_n$ over the extended alphabet $\Sigma \cup S(\Sigma)$. Each symbol $s \in S(\Sigma)$ represents a restricted form of wildcard: We allow s to be replaced by an element $\ell \in \Sigma$, but only as long as $\ell \in s$ (recall that s is a nonempty subset of Σ). Then Constrained Disjoint Matching over $\mathcal{A}$ is the problem of finding out if the strings $\bar{w}_1, \ldots, \bar{w}_n$ can be put into some order in a string $\bar{w}$, such that (1) the NFA $\mathcal{A}$ accepts a string that is obtained from $\bar{w}$ by performing a consistent replacement of the "wildcards" in $S(\Sigma)$, (2) no occurrences of the $\bar{w}_i$ overlap in $\bar{w}$, and (3) if the pair $(\bar{w}_i, \bar{w}_j)$ belongs to C then $\bar{w}_i$ appears before $\bar{w}_j$ in $\bar{w}$.

Our goal is to prove the following result:

Lemma A.4. *The problem* Constrained Disjoint Matching over $\mathcal{A}$ *can be solved in polynomial time, for each fixed NFA* $\mathcal{A}$.

The proof of this result is rather long. The first thing that we have to do is to provide a semantic characterization of the class of instances that are accepted by the Constrained Disjoint Matching problem. In order to do that we need to introduce a bunch of new terminology, as well as some intermediate results.

Let $\mathcal{A}$ be a fixed NFA over alphabet Σ, and let $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and $C \subseteq W \times W$ be the input to the Constrained Disjoint Matching problem over $\mathcal{A}$. Since $\mathcal{A}$ is fixed, we can assume w.l.o.g. that it is given by a DFA $(Q, \Sigma, \delta, q_0, F)$, where Q is the set of states, Σ is the alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, q_0 is the initial state, and F is the set of final states.

For each $i \in [1, n]$, let us define $\mathcal{F}(\bar{w}_i)$ to be the set of all functions $\theta : Q \to Q$, such that there is an instantiation $\bar{w}'_i$ of $\bar{w}_i$ that satisfies $\delta(q, \bar{w}'_i) = \theta(q)$, for each $q \in Q$. Then we can prove the following:

Lemma A.5. *The set* $\mathcal{F}(\bar{w}_i)$ *can be constructed in polynomial time in the size of* $\bar{w}_i$*, for each* $i \in [1, n]$.

Proof. Assume that $\bar{w}_i$ is the string $u_1 \cdots u_m$ over alphabet $\Sigma \cup S(\Sigma)$. Given $\theta : Q \to Q$ and $\ell \in \Sigma$, we denote by θ_ℓ the function from Q into Q such that $\theta_\ell(q) = \delta(\theta(q), \ell)$, for each $q \in Q$. We inductively

construct sets $Funct_j$ ($j \in [0, m]$) of functions from Q into Q, as follows: $Funct_0$ only contains the identity function, and for each $1 \leq j \leq m$,

$$Funct_j \;=\; \begin{cases} \{\theta_\ell \mid \theta \in Funct_{j-1}\} & \text{if } u_j = \ell, \text{ for } \ell \in \Sigma; \\ \{\theta_\ell \mid \theta \in Funct_{j-1},\, \ell \in s\} & \text{if } u_j = s, \text{ for } s \in S(\Sigma). \end{cases}$$

It can be easily proved by induction, that for every $j \in [0, m]$ the set $Funct_j$ contains all functions $\theta : Q \to Q$ such that, for some instantiation $\bar{w}'$ of the prefix formed by the first j elements of $\bar{w}$, it is the case that $\delta(q, \bar{w}') = \theta(q)$, for each $q \in Q$. It follows that $\mathcal{F}(\bar{w}_i) = Funct_m$. Further, it is not hard to see that each set $Funct_j$ ($j \in [1, m]$) can be constructed in constant time from $Funct_{j-1}$, and, thus, $\mathcal{F}(\bar{w}_i) = Funct_m$ can be constructed in polynomial time in the size of $\bar{w}_i$. □

Let $G_{W,C}$ be the simple and directed graph defined as follows. The set of vertices of $G_{W,C}$ is $\{v_1, \ldots, v_n\}$ and there is an edge from v_i to v_j ($1 \leq i, j \leq n$) if and only if $(w_i, w_j) \in C$. Notice that if the input $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and $C \subseteq W \times W$ is accepted by Constrained Disjoint Matching over $\mathcal{A}$, then it must be the case that $G_{W,C}$ is a DAG.

Let us now define $G'_{W,C}$ as the vertex-colored graph obtained from $G_{W,C}$ by performing the following coloring. Each vertex v of $G_{W,C}$ is colored with a nonempty set of functions from $\{\theta \mid \theta : Q \to Q\}$. In particular, vertex v_j ($1 \leq j \leq n$) is colored with the set $\mathcal{F}(\bar{w}_j)$. It follows from Lemma A.5, and the fact that $G_{W,C}$ can be constructed in polynomial time in the size of W and C, that $G'_{W,C}$ can be constructed in polynomial time in the size of W and C.

Let Θ be the set of all functions $\theta : Q \to Q$, such that there exists a string $\bar{w}$ over Σ that satisfies $\delta(q, \bar{w}) = \theta(q)$, for each $q \in Q$. Assume, without loss of generality, that Θ is disjoint from Σ and Q. From $\mathcal{A}$ we construct a new automaton $\mathcal{A}' = (Q, \Sigma \cup \Theta, \delta', q_0, F)$ as follows: $\delta'(q, \ell) = \delta(q, \ell)$, for each $q \in Q$ and $\ell \in \Sigma$, and $\delta'(q, \theta) = \theta(q)$, for each $q \in Q$ and $\theta \in \Theta$. Clearly, $\mathcal{A}'$ is a DFA. Further, since $\mathcal{A}$ is fixed, $\mathcal{A}'$ can be constructed in constant time.

Let $\bar{u} = u_1 \cdots u_m$ be a string over $\Sigma \cup \Theta$, and assume that $q_0 q_1 \cdots q_m$ is the unique run of $\mathcal{A}'$ on $\bar{u}$ that starts in the initial state q_0. We define the directed graph $J_{\bar{u}}$ (with self-loops), whose vertices are colored with subsets of $\Sigma \cup \Theta$, as follows:

- The vertices of the graph $J_{\bar{u}}$ are the elements $\{0, \ldots, 2m\}$;
- for each $i, j \in [0, 2m]$ with $i < j$, the pair (i, j) is an edge of $J_{\bar{u}}$;
- for each $i \in [0, 2m]$ such that $i = 2k$, for some $k \in [0, m]$, the pair (i, i) is an edge of $J_{\bar{u}}$;
- for each $i \in [0, 2m]$ such that $i = 2k - 1$, for some $k \in [1, m]$, the vertex i is colored $\{u_{k+1}\}$ in $J_{\bar{u}}$ (intuitively, the vertex $i = 2k - 1$ of $J_{\bar{u}}$ keeps the information about the k-th element of $\bar{u}$); and
- for each $i \in [0, 2m]$ such that $i = 2k$, for some $k \in [0, m]$, the vertex i is colored with the set that contains every symbol p in $\Sigma \cup \Theta$ for which there exists a string $\bar{u}_{q_k,p} = u'_1 \cdots u'_q$ over $\Sigma \cup \Theta$ such that $p = u'_j$, for some $1 \leq j \leq q$, and $\delta'(q_k, \bar{u}_{q_k,p}) = q_k$. Intuitively, the set of colors assigned to $i = 2k$ contains the symbol $p \in \Sigma \cup \Theta$ if and only if there exists a loop from state q_k to state q_k in $\mathcal{A}'$ that goes through a transition labeled p.

Notice that each odd vertex of $J_{\bar{u}}$, for a string $\bar{u}$ in $\Sigma \cup \Theta$, is colored with a set that contains exactly one color, and that the even vertices of $J_{\bar{u}}$ (we consider vertex 0 to be even) are colored with a set that contains zero or more colors. Also, notice that the set of edges of $J_{\bar{u}}$ is

$$\bigcup_{\{i \in [0,2m] \mid i \text{ is odd}\}} \{(i, j) \mid i < j,\, j \in [0, 2m]\} \;\cup \bigcup_{\{i \in [0,2m] \mid i \text{ is even}\}} \{(i, j) \mid i \leq j,\, j \in [0, 2m]\}.$$

Let G be an arbitrary directed graph that is also colored with subsets of $\Sigma \cup \Theta$, and assume that each vertex of G is colored with a set that contains at least one color. Then a function f from the set of vertices of G into the set of vertices of $J_{\bar{u}}$ is a *weak* homomorphism from G into $J_{\bar{u}}$, if the following holds: (1) For each pair v, v' of vertices in G, if (v, v') is an edge in G then $(h(v), h(v'))$ is an edge of $J_{\bar{u}}$, and (2) for every vertex v of G, the set of colors assigned to v in G is not disjoint from the set of colors assigned to $h(v)$ in $J_{\bar{u}}$. Further, we say that h is *coherent with* $\bar{u}$, if for each odd $i \in [0, 2m]$ there is at most one vertex v in G such that $h(v) = i$.

From $\mathcal{A}'$, we construct the set *Witnesses*$(\mathcal{A}')$ that contains all strings $\bar{u} = u_1 \cdots u_m$ over alphabet $\Sigma \cup \Theta$, such that the unique run $q_0 q_1 \cdots q_m$ of $\mathcal{A}'$ over $\bar{u}$ that starts in the initial state q_0 satisfies that (1) $q_m \in F$, and (2) $q_i \neq q_j$, for each $i, j \in [0, m]$ with $i \neq j$. In particular, $m \leq |Q| - 1$. Notice that *Witnesses*$(\mathcal{A}')$ can be constructed in constant time (and, in particular, *Witnesses*$(\mathcal{A}')$ contains a constant number of strings). The following is immediate:

Claim A.6. *For every string $\bar{u}$ in Witnesses$(\mathcal{A}')$, the graph $J_{\bar{u}}$ can be constructed in constant time.*

Further, by a standard application of pumping arguments, we can prove Lemma A.7 below. This lemma provides the desired semantic characterization of the class of instances that are accepted by the Constrained Disjoint Matching problem.

Lemma A.7. *The instance $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and $C \subseteq W \times W$ accepts a constrained disjoint matching over $\mathcal{A}$ if and only if $G_{W,C}$ is a DAG and there exists a string $\bar{u}$ in Witnesses$(\mathcal{A}')$ and a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$.*

Proof. Assume first that $G_{W,C}$ is a DAG and there exists a string $\bar{u} = u_1 \cdots u_m$ in *Witnesses*$(\mathcal{A}')$ and a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$. Assume also that $q_0 q_1 \cdots q_m$ is the unique run of $\mathcal{A}'$ over $\bar{u}$ that starts in the state q_0. For each vertex $v_i \in G_{W,C}$ $(1 \leq i \leq n)$, let θ_i be an arbitrary function from Q to Q that belongs to both the subset of $\{\theta \mid \theta : Q \to Q\}$ that is assigned to v_i and the one that is assigned to $f(v_i)$. Notice that at least one such θ_i must exist since f is a weak homomorphism from $G_{W,C}$ into $J_{\bar{u}}$.

For each vertex $j \in [0, 2m]$ that is of the form $2k$, for some $k \in [0, m]$, let $G^j_{W,C}$ be the subgraph of $G_{W,C}$ that is induced by all the vertices v such that $f(v) = j$. Then $G^j_{W,C}$ is a DAG since $G_{W,C}$ is a DAG. Therefore, there is a topological ordering $\lhd_j$ of the vertices in $G^j_{W,C}$; that is, $\lhd_j$ is an ordering of the vertices of $G^j_{W,C}$, such that $v \lhd_j v'$ whenever there is an edge from v to v' in $G^j_{W,C}$. Assume w.l.o.g. that the set of vertices of $G^j_{W,C}$ is $\{v_{j_1}, \ldots, v_{j_t}\} \subseteq \{v_1, \ldots, v_n\}$ and that $v_{j_1} \lhd_j v_{j_2} \lhd_j \cdots \lhd_j v_{j_t}$.

For each $1 \leq i \leq t$, let $\bar{w}'_{j_i}$ be an arbitrary instantiation of $\bar{w}_{j_i}$ such that for each $q \in Q$, $\delta(q, \bar{w}'_{j_i}) = \theta_{j_i}(q)$. Such an instantiation exists since the subset of $\Sigma \cup \Theta$ assigned to v_{j_i} in $G_{W,C}$ contains the function θ_{j_i}. Recall that for each $\theta \in \Theta$, $\bar{u}_{q_k,\theta}$ is a string over $\Sigma \cup \Theta$ such that $\delta'(q_k, \bar{u}_{q_k,\theta}) = q_k$ and the symbol θ appears in $\bar{u}_{q_k,\theta}$. We define $\bar{u}'_{q_k,\theta_{j_i}}$ as the string that is obtained from $\bar{u}_{q_k,\theta_{j_i}}$ by replacing each appearance of the symbol θ_{j_i} with $\bar{w}'_{j_i}$. (Notice that the string $\bar{u}_{q_k,\theta_{j_i}}$ is well-defined since $f(v_{j_i}) = 2k$ is colored with a set that contains θ_{j_i}). We finally define a word $\overline{u(j)}$ over alphabet $\Sigma \cup \Theta$ as $\bar{u}'_{q_k,\theta_{j_1}} \cdots \bar{u}'_{q_k.\theta_{j_t}}$. In case there is no vertex v in $G'_{W,C}$ such that $f(v) = j$, we simply assume that $\overline{u(j)}$ is the empty string.

Next we define a word $\bar{u}'$ as follows: $\bar{u}' := \overline{u(0)} u_1 \overline{u(2)} u_2 \cdots \overline{u(2m-2)} u_m \overline{u(2m)}$. Clearly, $\bar{u}'$ is accepted by $\mathcal{A}'$.

For each vertex v_i of $G_{W,C}$ such that $f(v_i)$ is odd, let $\bar{w}'_i$ be an arbitrary instantiation of $\bar{w}_i$ such that for each $q \in Q$, $\delta'(q, \bar{w}'_i) = \theta_i(q)$. Such an instantiation exists since the subset of $\Sigma \cup \Theta$ assigned to v_i in $G_{W,C}$ contains the function θ_i. Then, for each $k \in [1, m]$ we define a string $\overline{u(2k-1)}$ over $\Sigma \cup \Theta$ such that $\overline{u(2k-1)} = \bar{w}'_i$ if $2k - 1 = f(v_i)$, for some vertex v_i in $G'_{W,C}$, and $\overline{u(2k-1)}$ is the

unique symbol that belongs to u_k otherwise. (Notice that this replacement is well-defined since each odd vertex of $J_{\bar{u}}$ is the f-image of at most one vertex of $G'_{W,C}$).

Let $\bar{u}''$ be the string $\overline{u(0)}\,\overline{u(1)}\,\overline{u(2)u(3)}\cdots\overline{u(2m-2)}\,\overline{u(2m-1)}\,\overline{u(2m)}$. Clearly, $\bar{u}''$ is accepted by $\mathcal{A}'$. Further, for each string $\bar{w}_i$ $(1 \leq i \leq n)$ there is an instantiation $\bar{w}'_i$ of $\bar{w}_i$ that appears in $\bar{u}''$: In particular, $\bar{w}'_i$ appears in $\overline{u(j)}$ assuming that $f(v_i) = j$. Further, the appearances of the $\bar{w}'_i$'s in $\bar{u}''$ do not overlap. Finally, the appearances of the $\bar{w}'_i$'s respect the constraints in C. This is because, for each $(\bar{w}_i, \bar{w}_j) \in C$, either $f(v_i) < f(v_j)$, or $f(v_i) = f(v_j) = 2k$, for some $k \in [0, m]$, but v_i appears before v_j in the topological ordering $\lhd_{f(v_i)}$. It follows that the instance formed by W and C accepts a constrained disjoint matching over $\mathcal{A}$.

Assume, on the other hand, that the instance $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and $C \subseteq W \times W$ accepts a constrained disjoint matching $\bar{w} = \bar{s}_0\bar{w}'_{\kappa(1)}\bar{s}_1\bar{w}'_{\kappa(2)}\bar{s}_2\cdots\bar{s}_{n-1}\bar{w}'_{\kappa(n)}\bar{s}_n$ over $\mathcal{A}$. That is, each $\bar{s}_j$ $(0 \leq j \leq n)$ is a string over Σ, each $\bar{w}'_i$ $(1 \leq i \leq n)$ is an instantiation of $\bar{w}_i$, κ is a permutation of $\{1, \ldots, n\}$ and $\bar{w}$ is accepted by $\mathcal{A}$. Further, for each $(\bar{w}_i, \bar{w}_j) \in C$ it is the case that if $i = \kappa(i')$ and $j = \kappa(j')$ $(1 \leq i, i', j, j' \leq n)$ then $i' < j'$. As we have mentioned above, $G_{W,C}$ must be a DAG. We prove next that there is a string $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$ and a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$.

For each $1 \leq i \leq n$, let $\theta_{\kappa(i)} : Q \to Q$ be such that $\delta(q, \bar{w}'_{\kappa(i)}) = \theta_{\kappa(i)}(q)$, for each $q \in Q$, and define $\bar{w}'$ as the following string over alphabet $\Sigma \cup \Theta$: $\bar{s}_0\theta_{\kappa(1)}\bar{s}_1\theta_{\kappa(2)}\bar{s}_2\cdots\bar{s}_{n-1}\theta_{\kappa(n)}\bar{s}_n$. Clearly, $\bar{w}'$ is accepted by $\mathcal{A}'$. We iteratively "cut" $\bar{w}'$ until we get a subsequence $\bar{u}$ of it that is accepted by $\mathcal{A}'$ and the unique accepting run of $\mathcal{A}'$ on $\bar{u}$ has no repeated states from $\mathcal{A}'$.

The string $\bar{u}$ is obtained by applying the following procedure to $\bar{w}'$:

1. Set $\bar{w}' = \bar{u}$.
2. Assume that $\bar{u} = u_1 \cdots u_r$ and that $q_0 q_1 \cdots q_r$ is the unique run of $\mathcal{A}'$ on $\bar{u}$ that starts in the initial state q_0. While it holds that for some $1 \leq i < j \leq r$ it is the case that $q_i = q_j$ and there is no $i' < i$ such that $q_{i'} = q_{j'}$ for some $i < j' \leq r$, set $\bar{u}$ to be $u_1 \cdots u_i u_{j+1} \cdots u_r$.
3. Return $\bar{u}$.

Let $\bar{u}$ be the string returned by the procedure above on input $\bar{w}'$. Clearly, $\bar{u}$ is accepted by $\mathcal{A}'$. Further, the unique run of $\mathcal{A}'$ on $\bar{u}$ that starts in q_0 has no repeated states from $\mathcal{A}'$. We show next that there is a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$.

Assume that $\bar{u} = u_1 \cdots u_r$ and that $q_0 \cdots q_r$ is the unique run of $\mathcal{A}'$ over $\bar{u}$ starting in state q_0. Recall that the set of vertices of $J_{\bar{u}}$ is $\{0, \ldots, 2r\}$. We construct a function f from the set of vertices of $G'_{W,C}$ into the set of vertices of $J_{\bar{u}}$ as follows. Let v_i $(1 \leq i \leq n)$ be a vertex of $G'_{W,C}$. Then,

- if it is the case that the symbol θ_i that corresponds to $\bar{w}'_i$ in $\bar{w}'$ has not been "cut" in the process of constructing $\bar{u}$ and appears in the j-th position of $\bar{u}$, $1 \leq j \leq r$, then set $f(v_i) = 2j - 1$; and
- if it is the case that the symbol θ_i that corresponds to $\bar{w}'_i$ in $\bar{w}'$ has been "cut" in the process of constructing $\bar{u}$, and the "cut" in which such symbol θ_i was removed eliminated elements that were either between the j-th and the $(j+1)$-th position of $\bar{u}$ $(1 \leq j < r-1)$, before the position $j = 1$, or after the position $j = r$, then set $f(v_i) = 2j$. Notice that in this case the cut was made on a word that lead from state q_j to state q_j in $\mathcal{A}'$ and had an occurrence of the symbol θ_i inside. This means that the subset of $\Sigma \cup \Theta$ that is assigned to $f(v_i)$ in $J_{\bar{u}}$ is not disjoint from the subset of $\Sigma \cup \Theta$ that is assigned to v_i in $G'_{W,C}$.

It is not hard to see that f as constructed above defines a weak homomorphism from $G'_{W,C}$ into $J_{\bar{u}}$ that is coherent with $\bar{u}$. Indeed, that f is coherent with $\bar{u}$ follows immediately from the construction.

The same for the facts that (1) if (v_i, v_j) $(1 \leq i, j \leq n)$ is an edge of $G_{W,C}$ then $(f(v_i), f(v_j))$ is an edge of $J_{\bar{u}}$ (as the function f respects the relative order of the $\bar{w}'_i$'s in $\bar{w}$) and (2) for each v in $G_{W,C}$, the subset of $\Sigma \cup \Theta$ that is assigned to $f(v)$ in $J_{\bar{u}}$ is not disjoint from the subset of $\Sigma \cup \Theta$ that is assigned to v in $G'_{W,C}$. This finishes the proof of the lemma. □

What we have to do now is to find a procedural characterization of the class of instances $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and $C \subseteq W \times W$ that admit a constrained disjoint matching over $\mathcal{A}$. This is what we do next.

From Lemma A.7, checking whether the instance $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and $C \subseteq W \times W$ admits a constrained disjoint matching over $\mathcal{A}$ is equivalent to checking whether $G_{W,C}$ is a DAG and there exists a string $\bar{u}$ in $\mathit{Witnesses}(\mathcal{A}')$ and a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$. We define next a procedure Weak-Hom-Search that verifies the latter on input W and C:

1. The procedure first checks whether $G_{W,C}$ is a DAG. If not, simply rejects the instance formed by W and C, and concludes with the help of Lemma A.7 that W and C do not admit a constrained disjoint matching over $\mathcal{A}$. If $G_{W,C}$ is a DAG the procedure continues to the next step.

2. Then it constructs a *topological* ordering $\lhd$ of $G_{W,C}$, i.e. $\lhd$ is a linear ordering of the vertices of $G_{W,C}$, such that if (v, v') is an edge of $G_{W,C}$ then $v \lhd v'$. (Recall that $\{v_1, \ldots, v_n\}$ is the set of vertices of $G_{W,C}$. We assume, without loss of generality, that $v_1 \lhd v_2 \lhd \cdots \lhd v_n$).

   (**Observation:** This topological ordering always exists, and can be constructed in polynomial time in the size of $G_{W,C}$, and thus, of W and C, since $G_{W,C}$ is a DAG).

3. The procedure then constructs $G'_{W,C}$. (**Observation:** As we mentioned above, this can be done in polynomial time in the size of W and C).

4. For each $\bar{u} \in \mathit{witness}$, Weak-Hom-Search constructs the graph $J_{\bar{u}}$. (**Observation:** It follows from Claim A.6 that this can be done in constant time).

5. For each $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$ the procedure Weak-Hom-Search does the following. Assume $\bar{u} = u_1 \cdots u_m$. It constructs the set $\mathcal{G}_{\bar{u}}$ of all partial mappings g from the odd vertices of $J_{\bar{u}}$ into the vertices of $G_{W,C}$ that satisfy the following properties:

   - g is 1-to-1; and
   - for each $i' \in \{i \in [0, 2m] \mid i \text{ is odd}\}$ such that $g(i')$ is defined, the unique symbol that belongs to the subset of $\Sigma \cup \Theta$ that is assigned to i' in $J_{\bar{u}}$ also belongs to the subset of $\Sigma \cup \Theta$ that is assigned to $g(i')$ in $G'_{W,C}$.

   (**Observation:** Notice that $\mathcal{G}_{\bar{u}}$ is nonempty (as the partial mapping with empty domain always satisfies the conditions mentioned above). Further, it is not hard to see that there are at most polynomially many functions in $\mathcal{G}_{\bar{u}}$, and that each such mapping is of constant size. Thus, $\mathcal{G}_{\bar{u}}$ can be constructed in polynomial time.

   The intuition behind the set of functions in $\mathcal{G}_{\bar{u}}$ is that these are precisely the functions g such that g^{-1} can be extended into a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$).

6. For each $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$ and $g \in \mathcal{G}_{\bar{u}}$, the procedure does the following. With each vertex v_j $(j \in [1, n])$ of $G'_{W,C}$ it associates a vertex $\mathit{good\_for}(g, v_j)$ of $J_{\bar{u}}$ (i.e. an integer in $[0, 2m]$), using the procedure $\mathit{PossHom}(g)$ described below (assuming that $\mathit{PossHom}(g)$ has not failed at any step $j' < j$):

   (a) If v_j is of the form $g(i')$, for some $i' \in \{i \in [0, 2m] \mid i \text{ is odd}\}$, then $\mathit{good\_for}(g, v_j) = i'$;

(b) if v_j does not belong to the image of g, and there is at least one integer $i \in [0, 2m]$ such that (1) i is even (we consider 0 to be even), (2) the pair $(\mathit{good\_for}(g, v_{j'}), i)$ is an edge of $J_{\bar{u}}$, for each $j' < j$ such that $(v_{j'}, v_j)$ is an edge in $G_{W,C}$, and (3) the set of symbols from $\Sigma \cup \Theta$ that is assigned to v_j in $G'_{W,C}$ is not disjoint from the set of symbols from $\Sigma \cup \Theta$ that is assigned to i in $J_{\bar{u}}$, then we set $\mathit{good\_for}(g, v_j)$ to be the least such integer;

(c) otherwise, the procedure $\mathit{PossHom}(g)$ fails at step j, and stops.

(**Observation:** It is not hard to see that the procedure $\mathit{PossHom}(g)$ can be completed in polynomial time in the size of $G'_{W,C}$, and thus, of W and C. Intuitively, $\mathit{PossHom}(g)$ looks for the least even integer in $[0, 2m]$ that can be assigned to v_j in order to preserve a weak homomorphism that extends g^{-1}).

7. If for some $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$ and $g \in \mathcal{G}_{\bar{u}}$, the procedure $\mathit{PossHom}(g)$ above does not fail at any step $j \leq n$, and for each $i, j \in [1, n]$, if (v_i, v_j) is an edge in $G_{W,C}$ then $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_j))$ is an edge of $J_{\bar{u}}$, the procedure Weak-Hom-Search accepts W and C, and concludes (with the help of Lemma A.9 below) that there exists a weak homomorphism $h : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$. In that case it also concludes (with the help of Lemma A.7) that W and C admit a constrained disjoint matching.

   Otherwise, the procedure Weak-Hom-Search rejects the input W and C, and concludes (with the help of Lemma A.9 below) that there is no weak homomorphism $h : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$, for some $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$. In that case it also concludes (with the help of Lemma A.7) that W and C admit a constrained disjoint matching.

   (**Observation** It is easy to see that this step can also be done in polynomial time).

From all the previous remarks it is easy to conclude the following:

Claim A.8. *The procedure* Weak-Hom-Search *takes polynomial time in W and C.*

We now prove soundness and completeness of the procedure Weak-Hom-Search.

Lemma A.9. *The following are equivalent for each $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and $C \subseteq W \times W$ such that $G_{W,C}$ is a DAG:*

1. *There is a string $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$ and a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$; and*

2. *the procedure* Weak-Hom-Search *accepts input W and C (i.e. for some $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$ and $g \in \mathcal{G}_{\bar{u}}$ the procedure $\mathit{PossHom}(g)$ does not fail at any step $j \leq n$, and for every $i, j \in [1, n]$, if the pair (v_i, v_j) is an edge of $G_{W,C}$ then $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_j))$ is an edge of $J_{\bar{u}}$).*

Proof. We first prove that $(2 \Rightarrow 1)$. Assume that for some $g \in \mathcal{G}_{\bar{u}}$, the procedure $\mathit{PossHom}(g)$ does not fail at any step $j \leq n$, and for every $i, j \in [1, n]$, if the pair (v_i, v_j) is an edge of $G_{W,C}$ then $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_j))$ is an edge of $J_{\bar{u}}$. Since $\mathit{PossHom}(g)$ does not fail at any step $j \leq n$, we can define a function f from the vertices of $G'_{W,C}$ into the vertices of $J_{\bar{u}}$, such that $f(v_j) = \mathit{good\_for}(g, v_j)$, for each $j \in [1, n]$. We prove next that $f : G'_{W,C} \to J_{\bar{u}}$ is a weak homomorphism that is coherent with $\bar{u}$.

That f is a weak homomorphism follows from two facts. First, by assumption, for every $i, j \in [1, n]$, if the pair (v_i, v_j) is an edge of $G'_{W,C}$ then $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_j)) = (f(v_i), f(v_j))$ is an edge of $J_{\bar{u}}$. Second, by definition, the set of symbols from $\Sigma \cup \Theta$ that is assigned to v_j in $G'_{W,C}$ is never disjoint from the set of symbols from $\Sigma \cup \Theta$ that is assigned to $\mathit{good\_for}(g, v_j) = f(v_j)$ in $J_{\bar{u}}$, for each

$j \in [1, n]$. That f is coherent with $\bar{u}$ follows from the fact that for each odd $i \in [0, 2m]$ and vertex v_j in $G'_{W,C}$, $f(v_j) = \mathit{good\_for}(g, v_j) = i$ if and only if $g(i)$ is defined and $g(i) = v_j$ (recall that g is 1-to-1).

We now prove that $(1 \Rightarrow 2)$. Assume that there is a string $\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$ and a weak homomorphism $f : G'_{W,C} \to J_{\bar{u}}$ that is coherent with $\bar{u}$. Let V be the set of all vertices v of $G'_{W,C}$ such that $f(v) \in \{i \in [0, 2m] \mid i \text{ is odd}\}$, and let f_V be the restriction of f to the elements in V. Since f is coherent with $\bar{u}$, f_V^{-1} is a 1-to-1 partial mapping from $\{i \in [0, 2m] \mid i \text{ is odd}\}$ into $G'_{W,C}$. Further, since f is a weak homomorphism, it follows that for every odd $i \in [0, 2m]$ for which f_V^{-1} is defined, it is the case that the unique symbol that belongs to the subset of $\Sigma \cup \Theta$ that is assigned to i in $J_{\bar{u}}$ also belongs to the set of symbols from $\Sigma \cup \Theta$ that is assigned to $f_V^{-1}(i)$ in $G'_{W,C}$. It follows that $g = f_V^{-1}$ belongs to $\mathcal{G}_{\bar{u}}$. We prove next that the procedure $\mathit{WeakHom}(g)$ does not fail at any step $j \leq n$, and that for every $i, j \in [1, n]$, if the pair (v_i, v_j) is an edge of $G_{W,C}$ then $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_j))$ is an edge of $J_{\bar{u}}$.

We prove, by induction, the following, which implies the desired result: For every $j \leq n$, (1) the procedure $\mathit{WeakHom}(g)$ does not fail at step j, (2) $\mathit{good\_for}(g, v_j) \leq f(v_j)$, and (3) for every $i, i' \in [1, j]$, if the pair $(v_i, v_{i'})$ is an edge of $G_{W,C}$ then $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_{i'}))$ is an edge of $J_{\bar{u}}$.

- **Basis case ($j = 1$):** We only have to prove that $\mathit{WeakHom}(g)$ does not fail at step 1, and that $\mathit{good\_for}(g, v_1) \leq f(v_1)$.

  If $v_1 \in V$, then $\mathit{good\_for}(g, v_1) = f(v_1)$, and clearly the procedure $\mathit{WeakHom}(g)$ does not fail at this step. Further, trivially $\mathit{good\_for}(g, v_1) \leq f(v_1)$.

  If $v_1 \notin V$, then it must be the case that $f(v_1)$ is an even integer in $[0, 2m]$, and, since f is a weak homomorphism, that the set of symbols from $\Sigma \cup \Theta$ that is assigned to $f(v_1)$ in $J_{\bar{u}}$ is not disjoint from the set of symbols from $\Sigma \cup \Theta$ that is assigned to v_1 in $G'_{W,C}$. It follows that neither in this case the procedure $\mathit{WeakHom}(g)$ fails at step 1. Further, $\mathit{good\_for}(g, v_1)$ is the least integer $i \in [0, 2m]$ that is even, and such that the set of symbols from $\Sigma \cup \Theta$ that is assigned to i in $J_{\bar{u}}$ is not disjoint from the set of symbols from $\Sigma \cup \Theta$ that is assigned to v_1 in $G'_{W,C}$. It follows that $\mathit{good\_for}(g, v_1) \leq f(v_1)$.

- **Inductive case ($j + 1$, for $j < n$):**

  If $v_{j+1} \in V$, then $\mathit{good\_for}(g, v_{j+1}) = f(v_{j+1})$, and clearly the procedure $\mathit{WeakHom}(g)$ does not fail at this step. Further, trivially $\mathit{good\_for}(g, v_{j+1}) \leq f(v_{j+1})$. Finally, assume that for some $i, i' \in [1, j + 1]$, $(v_i, v_{i'})$ is an edge of $G'_{W,C}$. Then, by definition of $\lhd$, it must be the case that $i < i'$. We consider two cases:

  - $i, i' \in [1, j]$. Then, by induction hypothesis, $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_{i'}))$ is an edge of $J_{\bar{u}}$;
  - $i \in [1, j]$ and $j' = j + 1$. Since f is a weak homomorphism, $(f(v_i), f(v_{j+1}))$ is an edge of $J_{\bar{u}}$. Further, by induction hypothesis, $\mathit{good\_for}(g, v_i) \leq f(v_i)$, and, by definition, $\mathit{good\_for}(g, v_{j+1}) = f(v_{j+1})$. It follows that $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_{j+1}))$ is an edge of $J_{\bar{u}}$.

  If $v_{j+1} \notin V$, then we know that $f(v_{j+1})$ is an even integer in $[0, 2m]$, and, since f is a weak homomorphism, that the set of symbols from $\Sigma \cup \Theta$ that is assigned to $f(v_{j+1})$ in $J_{\bar{u}}$ is not disjoint from the set of symbols from $\Sigma \cup \Theta$ that is assigned to v_{j+1} in $G'_{W,C}$. Further, since f is a weak homomorphism, it follows that for every $i \in [1, j]$, if (v_i, v_{j+1}) is an edge of $G'_{W,C}$ then $(f(v_i), f(v_{j+1}))$ is an edge of $J_{\bar{u}}$. Since by induction hypothesis $\mathit{good\_for}(g, v_i) \leq f(v_i)$, for each

$i \in [1, j]$, it follows that $(\mathit{good\_for}(g, v_i), f(v_{j+1}))$ is an edge of $J_{\bar{u}}$ whenever (v_i, v_{j+1}) is an edge of $G'_{W,C}$, for every $i \in [1, j]$. It follows that neither in this case the procedure $\mathit{WeakHom}(g)$ fails at step $j + 1$.

Further, since $\mathit{good\_for}(g, v_{j+1})$ is the least integer $i \in [0, 2m]$ such that (1) i is even, (2) the set of symbols from $\Sigma \cup \Theta$ that is assigned to i in $J_{\bar{u}}$ is not disjoint from the set of symbols from $\Sigma \cup \Theta$ that is assigned to v_{j+1} in $G'_{W,C}$, and (3) there is an edge $(\mathit{good\_for}(g, v_{j'}), i)$ in $J_{\bar{u}}$, for each $j' \in [1, j]$ such that the pair $(v_{j'}, v_{j+1})$ is an edge in $G'_{W,C}$, it follows that $\mathit{good\_for}(g, v_{j+1}) \leq f(v_{j+1})$.

Finally, assume that for some $i, i' \in [1, j + 1]$, $(v_i, v_{i'})$ is an edge of $G_{W,C}$. Then, by definition of $\lhd$, it must be the case that $i < i'$. We consider two cases:

- $i, i' \in [1, j]$. Then, by induction hypothesis, $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_{i'}))$ is an edge of $J_{\bar{u}}$;
- $i \in [1, j]$ and $j' = j + 1$. By definition of $\mathit{good\_for}(g, v_{j+1})$, it is the case $(\mathit{good\_for}(g, v_i), \mathit{good\_for}(g, v_{j+1}))$ is an edge of $J_{\bar{u}}$.

This finishes the proof of the lemma. □

Finally, putting together Lemmas A.7 and A.9 and Claim A.8, we conclude that the problem Constrained Disjoint Matching over $\mathcal{A}$ can be solved in polynomial time, for each fixed NFA $\mathcal{A}$. This finishes the proof of Lemma A.4.

Now we start with the proof of Theorem 5.28. Assume that the DTD $d = (r, \rho, \alpha)$ is defined over Σ and A. Let $\Sigma_d \subseteq \Sigma$ be the set of all those elements $\ell \in \Sigma$ that are "useful" in d, i.e. the elements $\ell \in \Sigma$ such that there exists a tree T that conforms to d and there is an id i that belongs to the interpretation of P_ℓ in T. Without loss of generality, assume that there exists at least a tree T that conforms to d, and, thus, that $\Sigma_d \neq \emptyset$ and $r \in \Sigma_d$. Further, we denote by $d' = (r, \rho', \alpha')$ the DTD over Σ_d and A, such that for every $\ell \in \Sigma_d$, $\rho'(\ell)$ is the restriction of $\rho(\ell)$ to alphabet Σ_d, and $\alpha'(\ell) = \alpha(\ell)$. It is not hard to see that d' can be constructed in constant time from d.

We construct a procedure CheckConsistency that takes as input an $\downarrow^*$-free incomplete DOM-tree t (over vocabulary $\tau_{\Sigma,A}$). Since t is an $\downarrow^*$-free incomplete DOM-tree, it is the case that the *Gaifman* graph of the restriction of $\underline{rel}(t)$ to E, NS, NS^* is connected. The procedure CheckConsistency accepts t if and only if there is a tree T that conforms to d, and a homomorphism $\bar{h} : \underline{rel}(t) \to T$ (i.e. $\mathit{Rep}_d(t) \neq \emptyset$).

However, CheckConsistency does not accept as input an arbitrary incomplete DOM tree, but a *preprocessed* incomplete DOM tree, as defined next. An incomplete DOM tree t is *preprocessed*, if it satisfies each one of the following:

- The restriction of $\underline{rel}(t)$ to $E, NS, NS^*, (P_\ell)_{\ell \in \Sigma}, FC, LC$ is an extended hierarchy of sisterhoods of level n, for some $n > 0$, as defined in the proof of Theorem 5.21;
- there is a unique extended generator $\mathsf{i} \in \mathit{adom}_{node}(t)$ in $\underline{rel}(t)$;
- for each $\ell \in \Sigma \setminus \Sigma_d$, the interpretation of P_ℓ in $\underline{rel}(t)$ is empty;
- the interpretation of *Root* in $\underline{rel}(t)$ contains at most one element. Further, if $\mathsf{i} \in \mathit{adom}(t)$ belongs to the interpretation of *Root* in $\underline{rel}(t)$, then i is the unique extended generator of $\underline{rel}(t)$ and i does not belong to the interpretation of FC and LC in $\underline{rel}(t)$;
- if $\mathsf{i} \in \mathit{adom}_{node}(t)$ belongs to the interpretation of *Root* and P_ℓ, for some $\ell \in \Sigma_d$, then $\ell = r$;

- the interpretation of P_r in $\underline{rel}(t)$ contains at most one element. Further, if $\mathsf{i} \in adom(t)$ belongs to the interpretation of P_r in $\underline{rel}(t)$, then i is the unique extended generator in $\underline{rel}(t)$ and i does not belong to the interpretation of FC and LC in $\underline{rel}(t)$;

- if i is an element that belongs to the interpretation of *Leaf* in $\underline{rel}(t)$, then i has no children in $\underline{rel}(t)$ with respect to E;

- leaves are labeled with labels that are not forced by the DTD to have children. Formally, if i is an element that belongs to the interpretation of *Leaf* and P_ℓ in $\underline{rel}(t)$, for $\ell \in \Sigma_d$, then the empty string belongs to $\rho'(\ell)$; and

- for each $\mathsf{i} \in adom_{node}(t)$ and $\ell \in \Sigma_d$, if i belongs to the interpretation of P_ℓ in $\underline{rel}(t)$, then it must be the case that the set of all those attributes $@a \in A$, such that i is the first component of some tuple in the interpretation of $A_{@a}$ in $\underline{rel}(t)$, is contained in $\alpha'(\ell)$.

Since the Gaifman graph of the restriction of $\underline{rel}(t)$ to the vocabulary E, NS, NS^* is connected, it follows from the proof of Theorem 5.21 and the definition of what it means that a tree conforms to a DTD, that if an $\downarrow^*$-free incomplete DOM-tree t satisfies that $Rep_d(t) \neq \emptyset$, then it must be the case that t is preprocessed. Further, it is easy to see that one can check in polynomial time whether a $\downarrow^*$-free incomplete DOM-tree t is preprocessed. Therefore, we assume from now on, and without loss of generality, that every input t given to procedure CheckConsistency is preprocessed, since this affects neither the complexity nor the completeness of the proposed solution. That is, in order to prove that there exists a polynomial time procedure that takes as input an $\downarrow^*$-free incomplete DOM-tree t, and accepts t if and only if $Rep_d(t) \neq \emptyset$, it is enough to show that the procedure CheckConsistency as defined below, works in polynomial time, and for every preprocessed and $\downarrow^*$-free incomplete DOM-tree t given as input, CheckConsistency accepts t if and only if $Rep_d(t) \neq \emptyset$. This is what we do next.

We need to introduce first some additional terminology. Let t be a preprocessed and $\downarrow^*$-free incomplete DOM-tree, and assume that the restriction of $\underline{rel}(t)$ to $E, NS, NS^*, (P_\ell)_{\ell \in \Sigma_d}, FC, LC$ is an extended hierarchy of sisterhoods of level $n > 0$:

- We say that the *depth* of the element i in $adom_{node}(t)$ is $k \leq n$, if the substructure of $\underline{rel}(t)$ induced by all those elements i', such that $\mathsf{i}' = \mathsf{i}$ or $(\mathsf{i}, \mathsf{i}')$ belongs to the relation defined by the union of (i) the interpretation of E in $\underline{rel}(t)$, and (ii) the composition of the interpretation of E in $\underline{rel}(t)$ with the transitive and reflexive closure of the interpretation of $(E \cup NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$ in $\underline{rel}(t)$, is an extended hierarchy of sisterhoods of level exactly k (that is, the structure induced by the descendants of i, including i, is an extended hierarchy of sisterhoods of level exactly k);

- an element $\mathsf{i} \in adom_{node}(t)$ is said to be *unlabeled* in $\underline{rel}(t)$, if i does not belong to the interpretation of P_ℓ in $\underline{rel}(t)$, for each $\ell \in \Sigma_d$;

- the extended sisterhood *associated* with element i in $adom_{node}(t)$, is the restriction to $NS, NS^*, (P_\ell)_{\ell \in \Sigma_d}, FC, LC$ of the substructure of $\underline{rel}(t)$ induced by all those elements i', such that $(\mathsf{i}, \mathsf{i}')$ belongs to the relation defined by the union of (i) the interpretation of E in $\underline{rel}(t)$, and (ii) the composition of the interpretation of E in $\underline{rel}(t)$ with the transitive and reflexive closure of the interpretation of $(NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$ in $\underline{rel}(t)$ (intuitively, the elements that belong to the extended sisterhood associated with i are those that are forced to be children of i in every tree T that "completes" $\underline{rel}(t)$).

Next we introduce the procedure CheckConsistency, that takes a preprocessed and $\downarrow^*$-free incomplete DOM-tree t (over vocabulary $\tau_{\Sigma,A}$) as input, and accepts this input if and only if $Rep_d(t) \neq$

$\emptyset$. We assume, without loss of generality, that $\underline{rel}(t)$ is a structure over vocabulary $\tau_{\Sigma,A}$. If $\underline{rel}(t)$ is an extended hierarchy of sisterhoods of level $n > 0$, then the procedure CheckConsistency realizes at most n steps. Let $s_1, \ldots, s_{2^{|\Sigma_d|}-1}$ be an enumeration of the nonempty subsets of Σ_d. After each step $j \in [0, n-1]$, the procedure constructs sets $S_1^{j+1}, \ldots, S_{2^{|\Sigma_d|}-1}^{j+1}$, such that:

- For each $j \in [0, n-1]$, the sets $S_1^{j+1} \setminus S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^{j+1} \setminus S_{2^{|\Sigma_d|}-1}^j$ form a partition of the set of unlabeled elements in $\underline{rel}(t)$ of depth exactly $j+1$.

The basic idea of the procedure is that the unlabeled id i of depth $j+1$ belongs to $S_k^{j+1} \setminus S_k^j$, $j \in [1, n]$ and $k \in [1, 2^{[|\Sigma_d|]} - 1]$, if and only if for the structure $\underline{rel}(t)_{\mathsf{i}}$ that is induced in $\underline{rel}(t)$ by

- the set Desc(i) of all those elements $\mathsf{i}' \in adom_{node}(t)$, such that $\mathsf{i}' = \mathsf{i}$ or $(\mathsf{i}, \mathsf{i}')$ belongs to the relation defined by the union of (i) the interpretation of E in $\underline{rel}(t)$, and (ii) the composition of the interpretation of E in $\underline{rel}(t)$ with the transitive and reflexive closure of the interpretation of $(E \cup NS \cup NS^{-1} \cup NS^* \cup (NS^*)^{-1})$ in $\underline{rel}(t)$, and
- all those elements $d \in adom_{attr}(t)$, such that for some $@a \in A$ and $\mathsf{i}' \in$ Desc(i), the tuple (i', d) belongs to the interpretation of $A_{@a}$ in $\underline{rel}(t)$,

it is the case that s_k is precisely the set of all those labels $\ell \in \Sigma_d$, such that there is a tree T and a homomorphism $\bar{h} : \underline{rel}(t)_{\mathsf{i}}^{\text{unrooted}} \to T$, that satisfies that $h(\mathsf{i}) = \mathsf{i}$ belongs to the interpretation of P_ℓ in T, where for each $\mathsf{i}' \in adom_{node}(t)$, $\underline{rel}(t)_{\mathsf{i}'}^{\text{unrooted}}$ is the restriction of $\underline{rel}(t)_{\mathsf{i}}$ to the vocabulary $\tau_{\Sigma,A} \setminus \{Root\}$. Intuitively, $\mathsf{i} \in S_k^{j+1}$ if and ony if, for each $\ell \in s_k$, there is a way to "complete" into a tree T the structure induced in $\underline{rel}(t)$ by the set of descendants of i, including i, in such a way that i is labeled ℓ in T.

The procedure CheckConsistency is as follows. It makes heavy use of another procedure, HorizontalConsistency, that will be defined below. The procedure stops as soon as t is rejected (this can happen during any step $j \leq n$). If step n of the procedure is completed (i.e. CheckConsistency does not reject t), then CheckConsistency accepts t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) \neq \emptyset$:

1. **Step 0:** For every unlabeled $\mathsf{i} \in adom_{node}(t)$ of depth 1 (i.e. i has no children in $\underline{rel}(t)$ with respect to E), the procedure does the following:

   - If i belongs to the interpretation of *Leaf*, it first computes the set L_{i} of all those labels ℓ in Σ_d, such that the empty string is accepted by $\rho(\ell)$. Notice that $L_{\mathsf{i}} \neq \emptyset$ (because $\Sigma_d \neq \emptyset$).
     The procedure then computes the set A_{i} of all those labels $\ell \in \Sigma_d$, such that the set of all those attributes $@a \in A$, such that i is the first component of some tuple in the interpretation of $A_{@a}$ in $\underline{rel}(t)$, is contained in $\alpha'(\ell)$. If $A_{\mathsf{i}} = \emptyset$, then the procedure rejects the input t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) = \emptyset$.
     If $A_{\mathsf{i}} \neq \emptyset$, then the procedure computes the value of $F_{\mathsf{i}} = L_{\mathsf{i}} \cap A_{\mathsf{i}}$. If $F_{\mathsf{i}} = \emptyset$, then the procedure rejects the input t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) = \emptyset$.
     (**Observation:** Clearly, all these operations can be performed in polynomial time in the size of $\underline{rel}(t)$).
   - If i does not belong to the interpretation of *Leaf*, then the procedure only constructs the set A_{i}, and sets $F_{\mathsf{i}} = A_{\mathsf{i}}$.
     (**Observation:** Clearly, all these operations can be performed in polynomial time in the size of $\underline{rel}(t)$).

Afterwards, the procedure constructs the sets $S_1^1, \ldots, S_{2^{|\Sigma_d|}-1}^1$, in such a way that for each $j \in [1, 2^{|\Sigma_d|}-1]$, S_j^1 is precisely the set of all those unlabeled elements $\mathsf{i} \in adom(t)$ such that the depth of i in $\underline{rel}(t)$ is 1 and $F_{\mathsf{i}} = s_j$.

2. Once this is done, the procedure realizes the following for each $j \in [1, n-1]$:

   Step j $\in [1, n-1]$: For each i in $\underline{rel}(t)$ with depth exactly $j+1$, do the following:

   - If i belongs to the interpretation of P_ℓ, for some $\ell \in \Sigma_d$, then run the procedure HorizontalConsistency with input $[H_{\mathsf{i}}; \ell; S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^j]$, where H_{i} is the extended sisterhood associated with i in $\underline{rel}(t)$, and the sets $S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^j$ are constructed in step $j-1$. (**Observation:** The size of the input $[H_{\mathsf{i}}; \ell; S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^j]$ is linear on the size of $\underline{rel}(t)$).

     If HorizontalConsistency rejects input $[H_{\mathsf{i}}; \ell; S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^j]$, then the procedure CheckConsistency rejects t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) = \emptyset$.

     (**Observation:** Notice that in this case we do not have to check that the set of attributes of i conforms to d since the incomplete DOM-tree t is preprocessed.

     Also, it is not hard to see that if the procedure HorizontalConsistency takes polynomial time in the size of its input, then this step of the procedure CheckConsistency also takes polynomial time in the size of $\underline{rel}(t)$).

   - Otherwise (i.e. if i is unlabeled in $\underline{rel}(t)$), the procedure CheckConsistency constructs the set L_{i} of all those labels $\ell \in \Sigma_d$, such that the procedure HorizontalConsistency accepts input $[H_{\mathsf{i}}; \ell; S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^j]$, where H_{i} is the extended sisterhood associated with i in $\underline{rel}(t)$, and the sets $S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^j$ are constructed in step $j-1$. (**Observation:** The size of the input $[H_{\mathsf{i}}; \ell; S_1^j, \ldots, S_{2^{|\Sigma_d|}-1}^j]$ is linear on the size of $\underline{rel}(t)$. Further, this part of the procedure only makes a constant number of calls to the procedure HorizontalConsistency).

     If $L_{\mathsf{i}} = \emptyset$, then the procedure CheckConsistency rejects t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) = \emptyset$.

     Otherwise, CheckConsistency constructs the set A_{i} of all those labels $\ell \in \Sigma_d$, such that the set of all those attributes $@a \in A$, such that i is the first component of some tuple in the interpretation of $A_{@a}$ in $\underline{rel}(t)$, is contained in $\alpha'(\ell)$. If $A_{\mathsf{i}} = \emptyset$, then the procedure rejects the input t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) = \emptyset$.

     If $A_{\mathsf{i}} \neq \emptyset$, then the procedure computes the value of $F_{\mathsf{i}} = L_{\mathsf{i}} \cap \mathcal{A}_{\mathsf{i}}$. If $F_{\mathsf{i}} = \emptyset$, then the procedure rejects the input t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) = \emptyset$.

     (**Observation:** It is not hard to see that if the procedure HorizontalConsistency takes polynomial time in the size of its input, then this step of the procedure CheckConsistency also takes polynomial time in the size of $\underline{rel}(t)$).

   Afterwards, the procedure constructs the sets $S_1^{j+1}, \ldots, S_{2^{|\Sigma_d|}-1}^{j+1}$, in such a way that for each $k \in [1, 2^{|\Sigma_d|}-1]$, S_k^{j+1} is the union of S_k^j and all those unlabeled elements $\mathsf{i} \in adom(t)$ such that the depth of i in $\underline{rel}(t)$ is $j+1$ and $F_{\mathsf{i}} = s_k$.

3. **Step n:** Since t is preprocessed, $\underline{rel}(t)$ has a unique extended generator $\mathsf{i} \in adom_{node}(t)$ (i.e. there is a unique element $\mathsf{i} \in adom_{node}(t)$ that has neither a parent (with respect to E) nor a sibling (with respect to $NS \cup NS^*$)). Then if i is unlabeled and belongs to the interpretation of

Root in $\underline{re\ell}(t)$, and F_{i} (as constructed in step $n-1$) does not contain r, the procedure CHECK-CONSISTENCY rejects t, and declares (with the help of Lemma A.12 below), that $Rep_d(t) = \emptyset$.

The following is immediate from all the comments above:

Claim A.10. *If the procedure* HORIZONTALCONSISTENCY *takes polynomial time in the size of its input, then* CHECKCONSISTENCY *also takes polynomial time in the size of its input.*

Before studying any kind of properties associated with the procedure CHECKCONSISTENCY, we have to define the procedure HORIZONTALCONSISTENCY. This procedure takes as input an extended sisterhood H, a label $\ell \in \Sigma_d$, and sets $S_1, \ldots, S_{2^{|\Sigma_d|}-1}$, such that $S'_1, \ldots, S'_{2^{|\Sigma_d|}-1}$ forms a partition of the unlabeled ids in H, where for each $j \in [1, 2^{|\Sigma_d|} - 1]$, S'_j is the restriction of S_j to the unlabeled ids in H. But before formally presenting the procedure HORIZONTALCONSISTENCY, we explain what is its role.

Let $\bar{w} = \ell_1, \ldots, \ell_n$, $n > 0$, be a string over alphabet Σ_d. We say that the structure $\mathcal{B}$ over vocabulary $NS, NS^*, (P_\ell)_{\ell \in \Sigma_d}, FC, LC$ *represents* $\bar{w}$, if the domain of $\mathcal{B}$ is $\{\mathsf{i}_1, \ldots, \mathsf{i}_n\}$, where each i_j is a different element in $\mathcal{I}$, the interpretation of NS in $\mathcal{B}$ is the relation $\{(\mathsf{i}_j, \mathsf{i}_{j+1}) \mid j \in [1, n-1]\}$, the interpretation of NS^* in $\mathcal{B}$ is precisely the transitive closure of the interpretation of NS in $\mathcal{B}$, the interpretation of P_ℓ in $\mathcal{B}$, for $\ell \in \Sigma_d$, contains all those i_j, $j \in [1, n]$, such that $\ell_j = \ell$, the interpretation of FC in $\mathcal{B}$ is $\{\mathsf{i}_1\}$, and the interpretation of LC in $\mathcal{B}$ is $\{\mathsf{i}_n\}$. By slightly abusing notation, each time that $\mathcal{B}$ represents a string $\bar{w}$ we simply say that $\mathcal{B}$ *is* the string $\bar{w}$.

Let $\mathcal{A}$ be an NFA over alphabet Σ_d. Then the extended sisterhood H can be *completed* with respect to $\mathcal{A}$ and $S_1, \ldots, S_{2^{|\Sigma_d|}-1}$, if there is a string $\bar{w}$ such that (1) H is a substructure of $\bar{w}$, (2) $\bar{w}$ is accepted by $\mathcal{A}$, and (3) for every unlabeled id i in H that belongs to S_j, $j \in [1, 2^{|\Sigma_d|} - 1]$, it is the case that i belongs to the interpretation of $P_{\ell'}$ in $\bar{w}$, for some ℓ' in s_j. Intuitively, this says that the wildcards in H can take some concrete values constrained by the S_i's, in such a way that there is a superstring of the resulting structure that is accepted by $\mathcal{A}$. Notice that the problem of checking whether H can be "completed" with respect to $\mathcal{A}$ is very close to the problem of checking whether the set $W = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$, formed by the connected components of H with respect to NS, and the set C of constraints given by all the pairs $(\mathcal{C}_i, \mathcal{C}_j)$ such that $i \neq j$ and there is an edge labeled NS^* from an element of $\mathcal{C}_i$ to an element of $\mathcal{C}_j$, accepts a constrained disjoint matching over $\mathcal{A}$. We precisely use this similarity below.

The procedure HORIZONTALCONSISTENCY does the following: It accepts input $[H; \ell; S_1, \ldots, S_{2^{|\Sigma_d|}-1}]$ if and only if H can be completed with respect to $\rho(\ell)$ and $S_1, \ldots, S_{2^{|\Sigma_d|}-1}$. We prove next that HORIZONTAL CONSISTENCY takes polynomial time, by making use of Lemma A.4.

Lemma A.11. *The procedure* HORIZONTALCONSISTENCY *takes polynomial time.*

Proof. Intuitively, what we do is to construct a polynomial time reduction of the problem of checking whether H can be completed with respect to $\rho(\ell)$ and $S_1, \ldots, S_{2^{|\Sigma_d|}-1}$ to the problem of checking whether a set of strings and constraints admit a constrained disjoint matching over $\rho(\ell)$. We show this reduction next.

Let $\mathcal{C}_1, \ldots, \mathcal{C}_n$, $n \geq 0$, be the connected components of the restriction of H to NS (but without removing elements that do not appear in NS). Recall that $S(\Sigma) = \{s_1, \ldots, s_{2^{|\Sigma|}-1}\}$ is the set of all nonempty subsets of Σ, that we assume to be disjoint from Σ. With each component $\mathcal{C}_i$ $(1 \leq i \leq n)$ we associate a string $\bar{w}_i$ over $\Sigma \cup S(\Sigma)$ as follows: If $\mathcal{C}_i$ is the successor relation $\mathsf{i}_1, \ldots, \mathsf{i}_m$, then $\bar{w}_i = u_1 \cdots u_m$ and for each $1 \leq j \leq m$, $u_j = \ell$ $(\ell \in \Sigma)$ if i_j belongs to the interpretation of P_ℓ in H, and $u_j = s_k$ $(s_k \in S(\Sigma))$ if i_j is unlabeled and belongs to S_k.

Let $W = \{\bar{w}_1, \ldots, \bar{w}_n\}$ and assume that the set of constraints $C \subseteq W \times W$ is defined as follows: The pair $(\mathcal{C}_i, \mathcal{C}_j)$ $(1 \leq i, j \leq n)$ belongs to C if and only if $i \neq j$ and there is an element i in $\mathcal{C}_i$ and an element i' in $\mathcal{C}_j$ such that $(\mathsf{i}, \mathsf{i}')$ belongs to NS^*. One would be tempted to say then that H can be completed with respect to $\rho(\ell)$ and $S_1, \ldots, S_{2^{|\Sigma_d|}-1}$ if and only if W and C admit a constrained disjoint matching over $\mathcal{A}$. However, there is a slight detail that has to be taken into consideration: Some of the $\mathcal{C}_i$'s may contain elements labeled with FC and LC, and thus, the corresponding instantiation of the string $\bar{w}_i$ is forced to appear either at the beginning or the end of a constrained disjoint matching of W and C over $\mathcal{A}$. However, this extra constraint can be easily added to Constrained Disjoint Matching without losing tractability. Indeed, the only thing that one has to do is to look for a weak homomorphism of $G_{W,C}$ to $J_{\bar{u}}$ ($\bar{u} \in \mathit{Witnesses}(\mathcal{A}')$) that is coherent with $\bar{u}$ and that sends the strings in W that are distinguished as "first" or "last" to the corresponding nodes in $J_{\bar{u}}$. This can be easily done in polynomial time by adapting the procedure Weak-Hom-Search, which finishes the proof of the lemma. □

We now prove soundness and completeness of the procedure CheckConsistency:

Lemma A.12. *Let t be a preprocessed and $\downarrow^*$-free incomplete DOM-tree. Then $\mathit{Rep}_d(t) \neq \emptyset$ if and only if* CheckConsistency *accepts t.*

Proof. Since t is preprocessed, the restriction of $\underline{re\ell}(t)$ to $E, NS, NS^*, (P_\ell)_{\ell \in \Sigma}, FC, LC$ is an extended hierarchy of sisterhoods of level $n > 0$. We first prove (by induction) the following claim: For every $j \in [0, n-1]$, the procedure CheckConsistency does not reject t during step j if and only if for every id $\mathsf{i} \in \mathit{adom}_{node}(t)$ such that the depth of i is $j+1$, there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$. Here, for each $\mathsf{i}' \in \mathit{adom}_{node}(t)$, $\underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}'}$ refers to the restriction of $\underline{re\ell}(t)_{\mathsf{i}'}$ to $\tau_{\Sigma,A} \setminus \{\mathit{Root}\}$. Further, if CheckConsistency does not fail at step j, then for every unlabeled $\mathsf{i} \in \mathit{adom}_{node}(t)$ of depth $j+1$, it is the case that $\mathsf{i} \in S_k^{j+1}$, for $k \in [1, 2^{|\Sigma_d|} - 1]$, if and only if s_k is precisely the set of all those labels $\ell \in \Sigma_d$, such that there is a tree T that conforms to d and homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$, that satisfies that $\bar{h}(\mathsf{i}) = \mathsf{i}$ belongs to the interpretation of P_ℓ in T.

- **Basis case ($j = 0$):** We first prove that if procedure CheckConsistency does not reject t during step 0, then (*) for every id $\mathsf{i} \in \mathit{adom}_{node}(t)$ such that the depth of i is 1, there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$, and (**) for every unlabeled $\mathsf{i} \in \mathit{adom}_{node}(t)$ of depth 1, it is the case that $\mathsf{i} \in S_k^1$, for $k \in [1, 2^{|\Sigma_d|} - 1]$, if and only if s_k is precisely the set of all those labels $\ell \in \Sigma_d$, such that there is a tree T that conforms to d and homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$, that satisfies that $\bar{h}(\mathsf{i}) = \mathsf{i}$ belongs to the interpretation of P_ℓ in T.

  Let $\mathsf{i} \in \mathit{adom}_{node}(t)$ be an arbitrary id of depth 1.

  1. Suppose that i is unlabeled. We first prove (*). We consider first the case when i belongs to the interpretation of *Leaf* in $\underline{re\ell}(t)$. Since CheckConsistency does not reject t during step 0, it must be the case that both A_{i} and $F_{\mathsf{i}} = L_{\mathsf{i}} \cap A_{\mathsf{i}}$ are nonempty. Let ℓ be an arbitrary element in F_{i}. Since ℓ belongs to Σ_d, we can assume (without loss of generality) that there is a tree T that conforms to d, and such that the interpretation of P_ℓ in T contains the id i. Let T' be the tree obtained from T by removing all proper descendants of i. (Notice that i belongs to the interpretation of *Leaf* in T'). Since $\ell \in L_{\mathsf{i}}$, the empty string belongs to $\rho'(\ell)$, and, therefore, T' also conforms to d.

     Assume that $\alpha'(\ell) = \{@a_1, \ldots, @a_n\}$, and that attribute $@a_k$ takes value $d_k \in \mathcal{D}$ in the element i of T, for each $k \in [1, n]$. Let T'' be the tree obtained from T' as follows. For

every $k \in [1,n]$, if the tuple (i, v) belongs to the interpretation of $A_{@a_k}$ in $\underline{rel}(t)$, for some $v \in \mathcal{D}$, then change the value in T' of the $@a_k$-attribute of i from d_k to v.

It is not hard to see that T'' also conforms to d, and since $\ell \in A_{\mathsf{i}}$, that there is a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}} \to T''$.

We consider second the case when i does not belong to the interpretation of *Leaf* in $\underline{rel}(t)$. Since CheckConsistency does not reject t during step 0, it must be the case that $F_{\mathsf{i}} = A_{\mathsf{i}}$ is nonempty. Let ℓ be an arbitrary element in F_{i}. Since ℓ belongs to Σ_d, we can assume (without loss of generality) that there is a tree T that conforms to d, and such that the interpretation of P_ℓ in T contains the id i.

Assume that $\alpha'(\ell) = \{@a_1, \ldots, @a_n\}$, and that attribute $@a_k$ takes value $d_k \in \mathcal{D}$ in the element i of T, for each $k \in [1,n]$. Let T' be the tree obtained from T as follows. For every $k \in [1,n]$, if the tuple (i, v) belongs to the interpretation of $A_{@a_k}$ in $\underline{rel}(t)$, for some $v \in \mathcal{D}$, then change the value in T of the $@a_k$-attribute of i from d_k to v.

It is not hard to see that T' also conforms to d, and since $\ell \in A_{\mathsf{i}}$, that there is a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}} \to T'$.

We now prove (**). Since CheckConsistency does not reject t during step 0, it means that the sets A_{i} and $F_{\mathsf{i}} = A_{\mathsf{i}} \cap L_{\mathsf{i}}$ are nonempty. (We assume, without loss of generality, that in the case when i does not belong to the interpretation of *Leaf* in $\underline{rel}(t)$, $L_{\mathsf{i}} = \Sigma_d$). By definition, i belongs to S_k^{j+1} if and only if $F_{\mathsf{i}} = s_k$, for each $k \in [1, 2^{|\Sigma_d|} - 1]$. Take an arbitrary element $\ell \in s_k$ (recall that $s_k = F_{\mathsf{i}}$). Then by the same argument given above, we know that there exists a tree T that conforms to d, and such that (a) the interpretation of P_ℓ in T contains the id i, and (b) there exists a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$. On the other hand, assume that $\ell \notin s_k$. Then either $\ell \notin L_{\mathsf{i}}$, that is, i belongs to the interpretation of *Leaf* in $\underline{rel}(t)$ and the empty string does not belong to $\rho'(\ell)$, or $\ell \notin A_{\mathsf{i}}$, that is, there is a tuple of the form $(\mathsf{i}, \cdot)$ in the interpretation of $A_{@a}$ in $\underline{rel}(t)$, such that $@a \notin \alpha'(\ell)$. In any of the two cases, it is clear that there is no tree T that conforms to d, and such that (a) the interpretation of P_ℓ in T contains the id i, and (b) there exists a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$. This proves (**).

2. Suppose, on the other hand, that i belongs to the interpretation of P_ℓ in $\underline{rel}(t)$, for some $\ell \in \Sigma_d$. We only have to prove (*). We consider first the case when i belongs to the interpretation of *Leaf* in $\underline{rel}(t)$. Then by assumption on $\underline{rel}(t)$, the empty string belongs to $\rho'(\ell)$. Further, since ℓ belongs to Σ_d, we can assume that there is a tree T that conforms to d, and such that the interpretation of P_ℓ in T contains the id i. Let T' be the tree obtained from T by removing all proper descendants of i'. It is clear that T' also conforms to d.

   Assume that $\alpha'(\ell) = \{@a_1, \ldots, @a_n\}$, and that attribute $@a_k$ takes value $d_k \in \mathcal{D}$ in the element i of T, for each $k \in [1,n]$. Let T'' be the tree obtained from T' as follows. For every $k \in [1,n]$, if the tuple (i, v) belongs to the interpretation of $A_{@a_k}$ in $\underline{rel}(t)$, for some $v \in \mathcal{D}$, then change the value in T' of the $@a_k$-attribute of i from d_k to v.

   It is not hard to see that T'' also conforms to d, and since $\ell \in A_{\mathsf{i}}$, that there is a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}} \to T''$.

   The other case, that is, when i does not belong to the interpretation of *Leaf* in $\underline{rel}(t)$, is similar.

We now prove the following: If for every id $\mathsf{i} \in adom_{node}(t)$ such that the depth of i is 1, there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$, then CheckConsistency does not fail during step 0. It is enough to prove that for every unlabeled i of depth 1, the sets A_{i} and $F_{\mathsf{i}} = L_{\mathsf{i}} \cap A_{\mathsf{i}}$ are nonempty. (We assume, without loss of generality, that if i does not belong to *Leaf*, then $L_{\mathsf{i}} = \Sigma_d$). Take an arbitrary id i of depth 1.

1. Suppose first that i belongs to the interpretation of *Leaf* in $\underline{re\ell}(t)$. Take a tree T that conforms to d, and such that there is a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \rightarrow T$. Then $\bar{h}(\mathsf{i}) = \mathsf{i}$ also belongs to the interpretation of *Leaf* in T. Assume that i belongs to the interpretation of P_ℓ in T, for $\ell \in \Sigma_d$. Thus, the empty string belongs to $\rho(\ell)$, and, therefore, ℓ belongs to L_{i}. It is also not hard to see that $\ell \in A_{\mathsf{i}}$, and, therefore, that both A_{i} and F_{i} are nonempty.
2. Suppose second that i does not belong to the interpretation of *Leaf* in $\underline{re\ell}(t)$. Take a tree T that conforms to d, and such that there is a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \rightarrow T$. Assume that i belongs to the interpretation of P_ℓ in T, for $\ell \in \Sigma_d$. Then it must be the case that $\ell \in A_{\mathsf{i}}$, and, therefore, that $A_{\mathsf{i}} = F_{\mathsf{i}}$ is nonempty.

- **Inductive case ($j+1$, for $j \leq n-2$):** We first prove that if procedure CheckConsistency does not reject t during step $j+1$, then (*) for every id $\mathsf{i} \in adom_{node}(t)$ such that the depth of i is $j+2$, there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \rightarrow T$, and (**) for every unlabeled $\mathsf{i} \in adom_{node}(t)$ of depth $j+2$, it is the case that $\mathsf{i} \in S^{j+2}_k$, for $k \in [1, 2^{|\Sigma_d|} - 1]$, if and only if s_k is precisely the set of all those labels $\ell \in \Sigma_d$, such that there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}} \rightarrow T$, that satisfies that $\bar{h}(\mathsf{i}) = \mathsf{i}$ belongs to the interpretation of P_ℓ in T.

  Let $\mathsf{i} \in adom_{node}(t)$ be an arbitrary id of depth $j+2$, and assume that H_{i} is the extended sisterhood associated with the element i in $\underline{re\ell}(t)$.

  1. Suppose first that i is unlabeled. We first prove (*). Since CheckConsistency does not reject t during step $j+1$, it must be the case that the sets L_{i}, A_{i} and $F_{\mathsf{i}} = L_{\mathsf{i}} \cap A_{\mathsf{i}}$ are nonempty. Let ℓ be an arbitrary element in F_{i}. Since ℓ belongs to L_{i}, it is the case that HorizontalConsistency accepts input $[H_{\mathsf{i}}; \ell; S^{j+1}_1, \ldots, S^{j+1}_{2^{|\Sigma_d|}-1}]$, where sets $S^{j+1}_1, \ldots, S^{j+1}_{2^{|\Sigma_d|}-1}$ are obtained in step j of the procedure CheckConsistency. Thus, H_{i} can be completed with respect to $\rho(\ell)$ and $S^{j+1}_1, \ldots, S^{j+1}_{2^{|\Sigma_d|}-1}$, i.e. that there is a string $\bar{w}$ over alphabet Σ_d, such that (1) H_{i} is a substructure of $\bar{w}$, (2) $\bar{w}$ belongs to $\rho'(\ell)$, and (3) for every unlabeled i' in H_{i} that belongs to S^{j+1}_k, $k \in [1, 2^{|\Sigma_d|} - 1]$, it is the case that for some $\ell' \in s_k$, i' belongs to the interpretation of $P_{\ell'}$ in $\bar{w}$.

     First of all, for each id i' in the domain of $\bar{w}$, choose a tree $T[\mathsf{i}']$ that satisfies the following:

     - If i' belongs to H_{i}, and i' belongs to the interpretation of $P_{\ell'}$ in H_{i}, for $\ell' \in \Sigma_d$, then $T[\mathsf{i}']$ conforms to d and there is a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}'} \rightarrow T[\mathsf{i}']$. Clearly, $T[\mathsf{i}']$ exists by induction hypothesis, since the depth of i' is strictly less than $j+2$. Further, $\bar{h}(\mathsf{i}') = \mathsf{i}'$ belongs to the interpretation of $P_{\ell'}$ in $T[\mathsf{i}']$;
     - if i' belongs to H_{i}, i' is unlabeled in $\underline{re\ell}(t)$, and i' belongs to the interpretation of $P_{\ell'}$ in $\bar{w}$, for $\ell' \in \Sigma_d$, then $T[\mathsf{i}']$ conforms to d and there is a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}'} \rightarrow T[\mathsf{i}']$, that satisfies that $\bar{h}(\mathsf{i}') = \mathsf{i}'$ belongs to the interpretation of $P_{\ell'}$ in $T[\mathsf{i}']$. Clearly, $T[\mathsf{i}']$ exists by induction hypothesis, since the depth of i' is strictly less than $j+2$, and if i' belongs to the interpretation of $P_{\ell'}$ in $\bar{w}$ then i' belongs to S^{j+1}_k, for some $k \in [1, 2^{|\Sigma_d|} - 1]$, such that $\ell' \in s_k$; and
     - if i' does not belong to H_{i}, and i' belongs to the interpretation of $P_{\ell'}$ in $\bar{w}$, for $\ell' \in \Sigma_d$, then $T[\mathsf{i}']$ conforms to d and i' belongs to the interpretation of $P_{\ell'}$ in $T[\mathsf{i}']$. Clearly, $T[\mathsf{i}']$ exists, because ℓ' belongs to Σ_d.

     Since $\ell \in \Sigma_d$, we know that there is a tree $T(\mathsf{i})$ that conforms to d and such that i belongs to the interpretation of P_ℓ in $T(\mathsf{i})$. Let $T(\mathsf{i})_1$ be the tree obtained from $T(\mathsf{i})$ by removing all proper descendants of i in $T(\mathsf{i})$. Further, assume that $\alpha'(\ell) = \{@a_1, \ldots, @a_n\}$, and that attribute $@a_k$ takes value $d_k \in \mathcal{D}$ in the element i of $T(\mathsf{i})$, for each $k \in [1, n]$. Let $T(\mathsf{i})_2$ be

the tree obtained from $T(\mathsf{i})_1$ as follows. For every $k \in [1,n]$, if the tuple (i, v) belongs to the interpretation of $A_{@a_k}$ in $\underline{re\ell}(t)$, for some $v \in \mathcal{D}$, then change the value in $T(\mathsf{i})_1$ of the $@a_k$-attribute of i from d_k to v.

Assume that the domain of $\bar{w}$ is $\{\mathsf{i}'_1, \ldots, \mathsf{i}'_p\}$, and that NS is interpreted in $\bar{w}$ as the set of all pairs of the form $(\mathsf{i}'_k, \mathsf{i}'_{k+1})$, for $k \in [1, p-1]$. Let T be the tree obtained from $T(\mathsf{i})_2$ by appending the ordered forest $T[\mathsf{i}'_1]_\downarrow T[\mathsf{i}'_2]_\downarrow \cdots T[\mathsf{i}'_p]_\downarrow$ as the children of i in $T(\mathsf{i})_2$, where for each $k \in [1,p]$, $T[\mathsf{i}'_k]_\downarrow$ is the subtree of $T[\mathsf{i}'_k]$ rooted at i'_k. It is not hard to see that T also conforms to d, and since $\ell \in A_\mathsf{i}$, that there is a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_\mathsf{i} \to T$.

Now we prove (**). Since CheckConsistency does not reject t during step $j+1$, it means that the sets L_i, A_i, and $F_\mathsf{i} = L_\mathsf{i} \cap A_\mathsf{i}$ are nonempty. By definition, i belongs to S^{j+1}_k if and only if $F_\mathsf{i} = s_k$, for each $k \in [1, 2^{|\Sigma_d|} - 1]$. Take an arbitrary element $\ell \in s_k (= F_\mathsf{i})$. Then by the same argument given above, there exists a tree T that conforms to d, and such that (a) the interpretation of P_ℓ in T contains the id i, and (b) there exists a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_\mathsf{i} \to T$. On the other hand, assume that $\ell \notin s_k = F_\mathsf{i}$. Then either $\ell \notin L_\mathsf{i}$, that is, H_i cannot be completed with respect to $\rho(\ell)$ and $S^{j+1}_1, \ldots, S^{j+1}_{2^{|\Sigma_d|}-1}$, or $\ell \notin A_\mathsf{i}$, that is, there is a tuple of the form $(\mathsf{i}, \cdot)$ in the interpretation of $A_{@a}$ in $\underline{re\ell}(t)$, such that $@a \notin \alpha'(\ell)$. In the latter case, it is clear that there cannot be a tree T that conforms to d, and such that (a) the interpretation of P_ℓ in T contains the id i, and (b) there exists a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_\mathsf{i} \to T$. In the former case, the same holds: Assume, on the contrary, that there exists a tree T that conforms to d, and a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_\mathsf{i} \to T$. Then it must be the case that all elements in H_i are children of i in T. Assume that i belongs to the interpretation of P_ℓ in T, for $\ell \in \Sigma_d$, and let $\bar{w}$ be the string formed by the ordered children of i in T. It follows by induction hypothesis, that for every $k \in [1, 2^{|\Sigma_d|} - 1]$ and i' in H_i, if $\mathsf{i}' \in S^{j+1}_k$ then $\bar{h}(\mathsf{i}') = \mathsf{i}'$ must belong to the interpretation of $P_{\ell'}$ in T, for some $\ell' \in s_k$ (since the depth of i' is strictly less than $j+2$ and the restriction of $\bar{h}$ to $\underline{re\ell}(t)_{\mathsf{i}'}$ is a homomorphism from $\underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}'}$ into T). It follows that (1) H_i is a substructure of $\bar{w}$, (2) $\bar{w}$ belongs to $\rho'(\ell)$, and (3) for every unlabeled id i' in H_i that belongs to S^{j+1}_k, $k \in [1, 2^{|\Sigma_d|} - 1]$, it is the case that i' belongs to the interpretation of $P_{\ell'}$ in $\bar{w}$, for some ℓ' in s_k. This shows that H_i can be completed with respect to $\rho(\ell)$ and $S^{j+1}_1, \ldots, S^{j+1}_{2^{|\Sigma_d|}-1}$, and, thus, that HorizontalConsistency accepts input $[H_\mathsf{i}; \ell; S^n_1, \ldots, S^n_{2^{|\Sigma_d|}-1}]$

2. The other case, that is, when i belongs to the interpretation of P_ℓ in $\underline{re\ell}(t)$, for some $\ell \in \Sigma_d$, can be handled similarly.

We prove next that if for every id $\mathsf{i} \in adom_{node}(t)$ such that the depth of i is $j+2$, there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_\mathsf{i} \to T$, then CheckConsistency does not fail during step $j+1$. The procedure CheckConsistency loops over all unlabeled elements i of depth $j+2$.

1. Suppose first that i belongs to the interpretation of P_ℓ in $\underline{re\ell}(t)$, for some $\ell \in \Sigma_d$. Then we need to show that HorizontalConsistency accepts input $[H_\mathsf{i}; \ell; S^{j+1}_1, \ldots, S^{j+1}_{2^{|\Sigma_d|}-1}]$, where H_i is the extended sisterhood associated with i in $\underline{re\ell}(t)$, and the sets $S^{j+1}_1, \ldots, S^{j+1}_{2^{|\Sigma_d|}-1}$ are obtained during the step j of the procedure CheckConsistency. We know that there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)^{\text{unrooted}}_\mathsf{i} \to T$. But then for every unlabeled id i' in H_i, the restriction of $\bar{h}$ to $\underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}'}$ is a homomorphism from $\underline{re\ell}(t)^{\text{unrooted}}_{\mathsf{i}'}$ into T. It follows by induction hypothesis, since the depth of i' is strictly less than $j+2$, that if i' belongs to S^{j+1}_k, $k \in [1, 2^{|\Sigma_d|} - 1]$, then $\bar{h}(\mathsf{i}') = \mathsf{i}'$ must belong to the interpretation of $P_{\ell'}$ in T, for some $\ell' \in s_k$. It follows that the string $\bar{w}$ formed

by the ordered children of i in T, satisfies that (1) H_{i} is a substructure of $\bar{w}$, (2) $\bar{w}$ belongs to $\rho'(\ell)$, and (3) for every unlabeled id i' in H_{i} that belongs to S_k^{j+1}, $k \in [1, 2^{|\Sigma_d|} - 1]$, it is the case that i' belongs to the interpretation of $P_{\ell'}$ in $\bar{w}$, for some ℓ' in s_k. This shows that H_{i} can be completed with respect to $\rho(\ell)$ and $S_1^{j+1}, \ldots, S_{2^{|\Sigma_d|}-1}^{j+1}$, and, thus, that HorizontalConsistency accepts input $[H_{\mathsf{i}}; \ell; S_1^{j+1}, \ldots, S_{2^{|\Sigma_d|}-1}^{j+1}]$

2. Suppose second that i is unlabeled. Then we need to show that the sets A_{i}, L_{i}, and $F_{\mathsf{i}} = A_{\mathsf{i}} \cap L_{\mathsf{i}}$ are nonempty. We know that there is a tree T that conforms d and a homomorphism $\bar{h} : \underline{re\ell}(t)_{\mathsf{i}}^{\text{unrooted}} \to T$. Assume that $\bar{h}(\mathsf{i}) = \mathsf{i}$ belongs to the interpretation of P_ℓ in T, for some $\ell \in \Sigma_d$. We claim that $\ell \in A_{\mathsf{i}} \cap F_{\mathsf{i}}$, and, thus, that A_{i}, L_{i}, and $F_{\mathsf{i}} = A_{\mathsf{i}} \cap L_{\mathsf{i}}$ are nonempty. We prove this next.

   It is clear that $\ell \in A_{\mathsf{i}}$. We prove that $\ell \in F_{\mathsf{i}}$. It is enough to prove that HorizontalConsistency accepts input $[H_{\mathsf{i}}; \ell; S_1^{j+1}, \ldots, S_{2^{|\Sigma_d|}-1}^{j+1}]$, where H_{i} is the extended sisterhood associated with i in $\underline{re\ell}(t)$, and the sets $S_1^{j+1}, \ldots, S_{2^{|\Sigma_d|}-1}^{j+1}$ are obtained during the step j of the procedure CheckConsistency. But this can be done exactly as in the previous case.

Now we prove Lemma A.12 using the previous claim. Assume that the restriction of $\underline{re\ell}(t)$ to $E, NS, NS^*, (P_\ell)_{\ell\in\Sigma}, FC, LC$ is an extended hierarchy of sisterhoods of level $n > 0$, and that i is the unique extended generator of $\underline{re\ell}(t)$. Assume first that CheckConsistency accepts t; we will prove that $Rep_d(t) \neq \emptyset$. If CheckConsistency accepts t, then the procedure does not reject t during any step $j < n$. There are three different cases to consider when CheckConsistency accepts t:

1. The first case occurs when i belongs to the interpretation of P_ℓ in $\underline{re\ell}(t)$, for some $\ell \in \Sigma_d$.

   Assume first that i does not belong to the interpretation of *Root* in $\underline{re\ell}(t)$. Since CheckConsistency does not fail during step $n-1$, it follows from the claim that there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)_{\mathsf{i}}^{\text{unrooted}} \to T$. But then $\bar{h}$ is also a homomorphism from $\underline{re\ell}(t)_{\mathsf{i}}$ into T. Since in this case $\underline{re\ell}(t)_{\mathsf{i}} = \underline{re\ell}(t)$, it follows that $Rep_d(t) \neq \emptyset$.

   Assume, otherwise, that i belongs to the interpretation of *Root* in $\underline{re\ell}(t)$. Then it must be the case that $\ell = r$. Since CheckConsistency does not fail during step $n-1$, it follows from the claim that there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)_{\mathsf{i}}^{\text{unrooted}} \to T$. It is clear then that $\bar{h}(\mathsf{i}) = \mathsf{i}$ is the root of T, and, therefore, that $\bar{h}$ is also a homomorphism from $\underline{re\ell}(t)_{\mathsf{i}}$ into T. Since in this case $\underline{re\ell}(t)_{\mathsf{i}} = \underline{re\ell}(t)$, it follows that $Rep_d(t) \neq \emptyset$.

2. The second case occurs when i is unlabeled and does not belong to the interpretation of *Root* in $\underline{re\ell}(t)$.

   Since CheckConsistency does not fail during step $n-1$, it follows from the claim that there is a tree T that conforms to d and a homomorphism $\bar{h} : \underline{re\ell}(t)_{\mathsf{i}}^{\text{unrooted}} \to T$. But then $\bar{h}$ is also a homomorphism from $\underline{re\ell}(t)_{\mathsf{i}}$ into T. Since in this case $\underline{re\ell}(t)_{\mathsf{i}} = \underline{re\ell}(t)$, it follows that $Rep_d(t) \neq \emptyset$.

3. The third case occurs when i is unlabeled and belongs to the interpretation of *Root* in $\underline{re\ell}(t)$, and $r \in F_{\mathsf{i}}$ (where F_{i} is constructed in step $n-1$ of the procedure).

   Since CheckConsistency does not fail during step $n-1$, it follows from the claim that there is a tree T that conforms to d, such that there is a homomorphism $\bar{h} : \underline{re\ell}(t)_{\mathsf{i}}^{\text{unrooted}} \to T$ and i belongs to the interpretation of P_r in T. It is clear then that $\bar{h}(\mathsf{i}) = \mathsf{i}$ is the root of T, and, therefore, that $\bar{h}$ is also a homomorphism from $\underline{re\ell}(t)_{\mathsf{i}}$ into T. Since in this case $\underline{re\ell}(t)_{\mathsf{i}} = \underline{re\ell}(t)$, it follows that $Rep_d(t) \neq \emptyset$.

In each one of the three cases we conclude that $Rep_d(t) \neq \emptyset$.

Assume, on the other hand, that CheckConsistency does not accept t; we will prove that $Rep_d(t) = \emptyset$. There are two different cases to consider when CheckConsistency rejects t:

1. The first case is when CheckConsistency rejects t during step $j < n$.

   Then from the previous claim, there is an id i' of depth $j+1$, such that for every tree T that conforms to d it is not the case that there is a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}'} \to T$. It follows that there is no tree T that conforms to d, and such that there is a homomorphism $\bar{h} : \underline{rel}(t) \to T$. We conclude that $Rep_d(t) = \emptyset$.

2. The second case is when CheckConsistency does not reject t during any step $j < n$, i is unlabeled and belongs to the interpretation of *Root* in $\underline{rel}(t)$, and F_{i} (as constructed in step n of the procedure CheckConsistency) does not contain r.

   Then from the previous claim, for every tree T that conforms to d and such that there exists a homomorphism $\bar{h} : \underline{rel}(t)^{\text{unrooted}}_{\mathsf{i}} \to T$, it must be the case that $\bar{h}(\mathsf{i}) = \mathsf{i}$ does not belong to the interpretation of P_r in T. It follows that there is no tree T that conforms to d, and such that there is a homomorphism $\bar{h} : \underline{rel}(t)_{\mathsf{i}} \to T$ (because i belongs to the interpretation of *Root* in $\underline{rel}(t)$). We conclude that $Rep_d(t) = \emptyset$.

This finishes the proof of Lemma A.12. □

Finally, from Claim A.10 and Lemmas A.11 and A.12, it follows that Consistency(d) is in PTIME, for $\downarrow^*$-free incomplete DOM-trees. This finishes the proof of Theorem 5.28.

A.5 Proof of Theorem 7.1

Fix a DTD $d = (r, \rho, \alpha)$ and a query $q(\bar{x})$. Assume that $q(\bar{x})$ is a union of queries of the form $\exists \bar{y}\, t(\bar{x}, \bar{y})$, where $t(\bar{x}, \bar{y})$ is an incomplete tree (i.e. $t(\bar{x}, \bar{y})$ has no node ids). To prove that the complexity of computing certain answers is in coNP, it suffices to show that there exists a polynomial $p(x)$, that depends only on d and q, such that the following holds: If for an incomplete tree t and a tuple $\bar{s}$ of elements from $\mathcal{D}$ it is is the case that $\bar{s} \notin certain_d(q, t)$, then there exists a tree T in $Rep_d(t)$, such that $\bar{s} \notin q(T)$ and the size of T is bounded by $p(|t|)$. Then the problem of checking whether $\bar{s} \notin certain_d(q, t)$ is in NP, and, thus, QueryAnswering(q, d) is in coNP.

Let t be an incomplete tree and assume that $\bar{s} \notin certain_d(q, t)$. Then there exists a tree $T_0 \in Rep_d(t)$ such that $\bar{s} \notin q(T_0)$. What we do first is to construct, from T_0, another tree in $Rep_d(t)$, such that $\bar{s}$ does not belong to the evaluation of q over such tree and the length of each path of the tree is polynomial.

Since $T_0 \in Rep_d(t)$, there exists a homomorphism $\bar{h} : \underline{rel}(t) \to T_0$. Define the *skeleton* of T_0, denoted by $sk(T_0)$, recursively as follows: (1) If a node s is the root of T_0 or belongs to the image of $\bar{h}_0$, then s belongs to $sk(T_0)$; and (2) if the nodes s_1 and s_2 of T_0 belong to $sk(T_0)$, then so it does its least common ancestor. It is easy to see that the size of $sk(T_0)$ is at most quadratic in the size of t.

First of all we construct, from T_0, a tree T_0' as follows: Every node in T_0, except those in $sk(T_0)$, is given new attribute values, which are fresh and distinct values from $\mathcal{D}$. Clearly, $T_0' \in Rep_d(t)$. Further, $\bar{s} \notin q(T_0')$. Assume otherwise. Then for some disjunct $\exists \bar{y}\, t(\bar{x}, \bar{y})$ of $q(\bar{x})$, the tree T_0' satisfies $\exists \bar{y}\, t(\bar{s}, \bar{y})$, and thus, T_0' satisfies $t(\bar{s}, \bar{c}')$ for some tuple $\bar{c}'$. But then, if the tuple $\bar{c}$ is obtained from $\bar{c}'$ by changing newly created attributes in T_0' to those they replaced, we would have that T_0 satisfies $t(\bar{s}, \bar{c})$, contradicting $\bar{s} \notin q(T_0)$. We define $sk(T_0') = sk(T_0)$. In the following, we prune the paths of T_0' using *vertical shortcuts* as defined next.

Vertical shortcuts: Clearly, there is a union of conjunctive queries $\varphi(\bar{x})$ over vocabulary $\tau_{\Sigma,A}$ that is equivalent to $q(\bar{x})$. Assume that the quantifier depth of φ is $k \geq 0$. Notice that k depends only on φ. Further, let $K \geq 0$ be the number of different *rank-k types* (c.f., [30]) of trees over vocabulary $\tau_{\Sigma,A}$ with one distinguished element. Again, K only depends on k, and thus, on φ. Let m be the size of Σ. We define M to be $K \cdot m + 1$.

Consider an arbitrary vertical path $s_1 \dots s_{M+4}$ in T'_0, such that none of nodes $s_1, \dots, s_{M+3}$ belongs to $sk(T'_0)$ and s_{M+4} has a descendant in $sk(T'_0)$. Because the length of this path is bigger than $M+3$, there exist two indexes $1 < j_1 < j_2 < M+4$, such that s_{j_1} and s_{j_2} have the same label in T'_0 and the rank-k types of $(T'_0(s_{j_1}|_{s_{M+4}}), s_{M+4})$ and $(T'_0(s_{j_2}|_{s_{M+4}}), s_{M+4})$ coincide, where $T'_0(s_{j_1}|s_{M+4})$ (resp. $T'_0(s_{j_2}|s_{M+4})$) is the subtree of T'_0 induced by all elements that are descendants of s_{j_1}, including s_{j_1} (resp. descendants of s_{j_2}, including s_{j_2}), that are not proper descendants of s_{M+4}. Let $T'_0(s_{j_1} \uparrow s_{j_2})$ be the tree obtained from T'_0 by replacing the tree rooted at s_{j_1} with the tree rooted at s_{j_2}. We say that $T'_0(s_{j_1} \uparrow s_{j_2})$ is a *vertical shortcut* of T'_0.

It is not hard to see that the vertical shortcut $T'_0(s_{j_1} \uparrow s_{j_2})$ still conforms to d. It is also possible to prove that every element in $sk(T'_0)$ belongs to $T'_0(s_{j_1} \uparrow s_{j_2})$. Indeed, assume for the sake of contradiction, that there exists an element s in the image of $sk(T'_0)$ that does not belong to $T'_0(s_{j_1} \uparrow s_{j_2})$. Then s belongs to the subtree rooted at s_k, for some $k \in [j_1, j_2 - 1]$. But then s_k is the least common ancestor of s and any descendant s' of s_{j_2} that belongs to $sk(T'_0)$. It follows that s_k belongs to $sk(T'_0)$, which is a contradiction. In addition, it is not hard to see that $\bar{h}_0 : \underline{re\ell}(t) \to T'_0(s_{j_1} \uparrow s_{j_2})$ is a homomorphism. Thus, $T'_0(s_{j_1} \uparrow s_{j_2}) \in Rep_d(t)$. Finally, it is also possible to prove - using a standard Ehrenfeucht-Fraïssè game argument (c.f., [30]) - that $(T'_0(s_{j_1} \uparrow s_{j_2}), \bar{s})$ and $(T'_0, \bar{s})$ are indistinguishable by FO formulas of quantifier depth $\leq k$, and thus, $\bar{s} \notin q(T'_0(s_{j_1} \uparrow s_{j_2}))$.

Applying the process of vertical shortcutting inductively, we obtain a tree T_1 that conforms to d, the mapping $\bar{h}_0 : \underline{re\ell}(t) \to T_1$ is a homomorphism, and $\bar{s} \notin q(T_1)$. We define $sk(T_1) = sk(T'_0) = sk(T_0)$. Notice that it may still be the case that some vertical paths in T_1 are not of polynomial length. This may happen, for instance, if there is a subtree rooted at a node s in T_0 that does not contain a node in $sk(T_0)$, but that has a vertical path that is not of polynomial length. In order to prune the long vertical paths of T_1, we construct from T_1 a new tree T_2 as follows: The tree T_2 is obtained from T_1 by replacing every subtree rooted at a node s that does not contain an element in $sk(T_1)$ with a fixed-size subtree in such a way that for the resulting subtree, say T'_1, it is still the case that T'_1 conforms to d and $\bar{s} \notin q(T'_1)$. (This can be done by applying, to the subtree rooted at node s, the same kind of shortcutting techniques that we present here). Clearly, every element in $sk(T_1)$ belongs to T_2, and $\bar{h}_0 : \underline{re\ell}(t) \to T_2$ is a homomorphism. We define $sk(T_2) = sk(T_1)$. Further, T_2 conforms to d. Finally, using the same kind of techniques than in the proof of Theorem 5.1, it is possible to prove that there exists a polynomial $p_1(x)$, that depends only on d and $q(\bar{x})$, such that the length of each path in T_1 is at most $p_1(t)$.

From T_2 we now construct a new tree, such that this tree belongs to $Rep_{\Sigma,A}(t)$, it conforms to d, and the number of children of each one of its nodes is polynomially bounded.

Horizontal shortcuts: Let $\sigma_1, \dots, \sigma_t$ be an enumeration of all rank-k types of trees over vocabulary $\tau_{\Sigma,A}$, and let K' be the number of different rank-k types of strings over the alphabet $\{\sigma_1, \dots, \sigma_t\}$. Notice that K' depends only on k, and thus, on $q(\bar{x})$. Further, let p be the maximum number of states of an NFA of the form $\rho(\ell)$, for $\ell \in \Sigma$. We define M' to be $K' \cdot p + 1$.

Let $s_1 \dots s_{M'+4}$ be a horizontal path in T_2, such that no subtree rooted at a node of the form s_j, for $j \in [1, M'+3]$, has an element in $sk(T_2)$. Further, assume that the parent s of the elements in this path is labeled ℓ. Choose an arbitrary accepting run π of the NFA $\rho(\ell)$ over the children of s. Since the length of the path is strictly bigger than $M'+3$, there exist two indexes $1 < j_1 < j_2 < M'+4$, such that $\pi(s_{j_1}) = \pi(s_{j_2})$ and the rank-k types of the strings $\sigma_{s_{j_1}} \sigma_{s_{j_1}+1} \cdots \sigma_{s_{M'+4}}$ and $\sigma_{s_{j_2}} \sigma_{s_{j_2}+1} \cdots \sigma_{s_{M'+4}}$

coincide, where for an arbitrary node s in T_2, σ_s is the rank-k type of the subtree rooted at s. Let $T_2(s_{j_1} \leftarrow s_{j_2})$ be the tree obtained from T_2 by removing the subtrees rooted at $s_{j_1}, \ldots, s_{j_2-1}$.

It is not hard to see that $T_2(s_{j_1} \leftarrow s_{j_2})$ conforms to d, that every element of $sk(T_2)$ belongs to $T_2(s_{j_1} \leftarrow s_{j_2})$, and $\bar{h}_0 : \underline{rel}(t) \rightarrow T_2(s_{i_1} \leftarrow s_{i_2})$ is a homomorphism. Further, it is also possible to prove – again using a standard Ehrenfeucht-Fraïssè game argument – that $(T_2(s_{j_1} \leftarrow s_{j_2}), \bar{s})$ and $(T_2, \bar{s})$ are indistinguishable by FO formulas of quantifier depth $\leq k$, and thus, $\bar{s} \notin q(T_2(s_{j_1} \leftarrow s_{j_2}))$.

By inductively applying the horizontal shortcutting technique, we obtain a tree T_3 that conforms to d, every element of $sk(T_2) = sk(T_0)$ belongs to T_3 and $\bar{h}_0 : \underline{rel}(t) \rightarrow T_3$ is a homomorphism. Further, in T_3 the following holds: (1) Every path is of length at most $p_1(|t|)$, (2) every node that has a (not necessarily proper) descendant in $sk(T_0)$, has at most $(|sk(T_0)|+1) \cdot (M'+4)$ children, and (3) every subtree rooted at a node that does not have a descendant in $sk(T_0)$ has size bounded by a fixed number N. It follows that the size of T_3 is bounded by

$$O(|sk(T_0)| \cdot p_1(|t|) \cdot (|sk(T_0)|+1) \cdot (M'+4) \cdot N),$$

and, hence, T is polynomial in the size of t. This concludes the proof of the theorem.

A.6 Remaining cases from the proof of Theorem 7.3

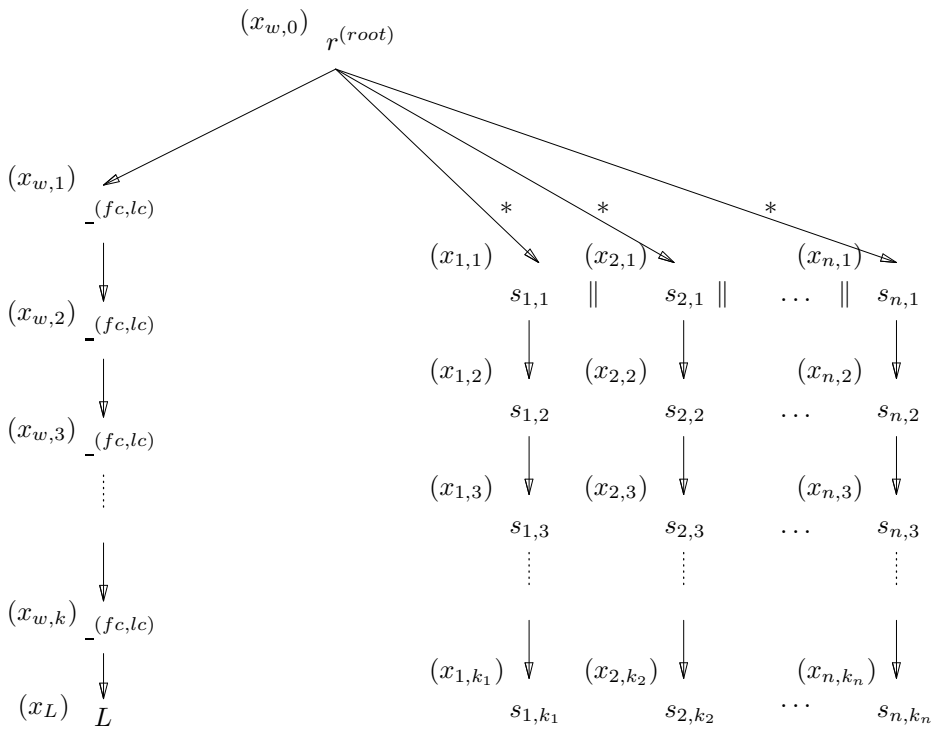
We prove that there exists a query q in $\mathcal{CQ}(\downarrow)$ such that QueryAnswering(q) is coNP-hard for $(\downarrow, \|, \downarrow^*, \mu)$-incomplete trees, even without attributes. Thus, by Theorem 7.1, QueryAnswering(q) is coNP-complete.

The proof is by reduction from the following:

Problem:	Shortest Common Superstring
Input:	finite alphabet Σ, finite set S of strings from Σ^*, and a positive integer K
Question:	is there a string $w \in \Sigma^*$ with $\|w\| \leq K$ such that each string $s \in S$ is a substring of w, i.e. $w = w_0 s w_1$ where $w_0, w_1 \in \Sigma^*$?

Consider an instance of the problem above with $S = \{s_1, s_2, \ldots, s_n\}$ and $s_i = s_{i,1}s_{i,2}\ldots s_{i,k_i}$, for $i \in \{1, n\}$ and $k_i \geq 1$. We next show how to build a $(\downarrow, \|, \downarrow^*, \mu)$-incomplete tree t, such that given a query q in $\mathcal{CQ}(\downarrow)$ and $a \in \mathcal{D}$, $a \in \mathit{certain}(q, t)$ if and only if there exists no string $w_1 \ldots w_k \in \Sigma^k$ that is a superstring of each s_i in S.

Let t be the following incomplete tree:



where $L \notin \Sigma$, $x_L \in \mathcal{V}_{\text{node}}$, for every $i \in \{1, n\}$, $j \in \{1, k_i\}$, $x_{i,j} \in \mathcal{V}_{\text{node}}$ and for every $h \in \{0, k\}$, $x_{w,h} \in \mathcal{V}_{\text{node}}$.

Moreover, let q be the query $q(a) = t_q()$, where $a \in \mathcal{D}$ and t_q is the following:

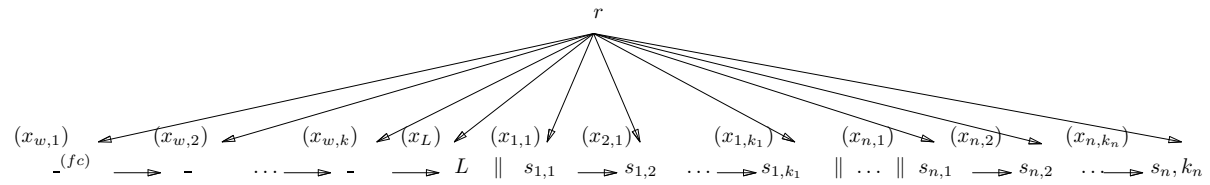
We next show that $a \in \mathit{certain}(q, t)$ if and only if there exists no string $w \in \Sigma^*$ with $|w| \leq K$ that is a superstring of each s_i in S.

$\Rightarrow$ Assume that $a \in \mathit{certain}(q, t)$. Then, for every $T \in \mathit{Rep}(t)$, $a \in q(T)$. Thus, for every T there exists a node that is a child of a node labeled L. It follows that there exists not a single evaluation $\nu = (\nu_{node}, \nu_{null})$, such that for every $i \in \{1, n\}$, $\nu_{node}(x_{i,1}) = \nu_{node}(x_{w,h_i})$ for some $h_i \in \{1, k\}$. Hence there exists no string $w \in \Sigma^*$ with $|w| \leq K$ that is a superstring of each s_i in S.

$\Leftarrow$ Assume that $a \notin \mathit{certain}(q, t)$. It follows that there exists $T \in \mathit{Rep}(t)$ such that $a \notin q(T)$. Let ν be the evaluation such that $(T, \nu, s) \models t$, where $\nu = (\nu_{node}, \nu_{null})$, and $s = \nu_{node}(x_{w,0})$ is the root of T. Since $a \notin q(T)$, $\nu(x_L)$ is a leaf of T. Moreover, given the markings occurring in t, $\nu(x_{w,h})$ has a unique child, for every $h \in \{0, k\}$. Thus, ν maps every node $x_{i,j}$ into the same node as some $x_{w,h}$, i.e. for every i, j, there exists $h \in \{1, k\}$ such that $\nu_{node}(x_{i,j}) = \nu_{node}(x_{w,h})$. Hence, $w_1, w_2, \ldots, ...w_k$ is a superstring of each s_i in S, where w_h is the label of $\nu_{node}(x_{w,h})$, for every $h \in \{1, k\}$.

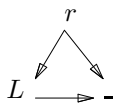
Finally, we prove that there exists a query q in $\mathcal{CQ}(\downarrow, \rightarrow)$ such that QueryAnswering(q) is coNP-hard for $(\downarrow, \rightarrow, \|, \mu)$-incomplete trees, even without attributes.

The proof is again by reduction from the Shortest Common Superstring problem. Let t be the following incomplete tree:



where $L, x_L, x_{i,j}, x_{w,h}$ are as in the previous proof.

Moreover let q be the query $q(a) = t_q()$, where $a \in \mathcal{D}$ and t_q is the following:



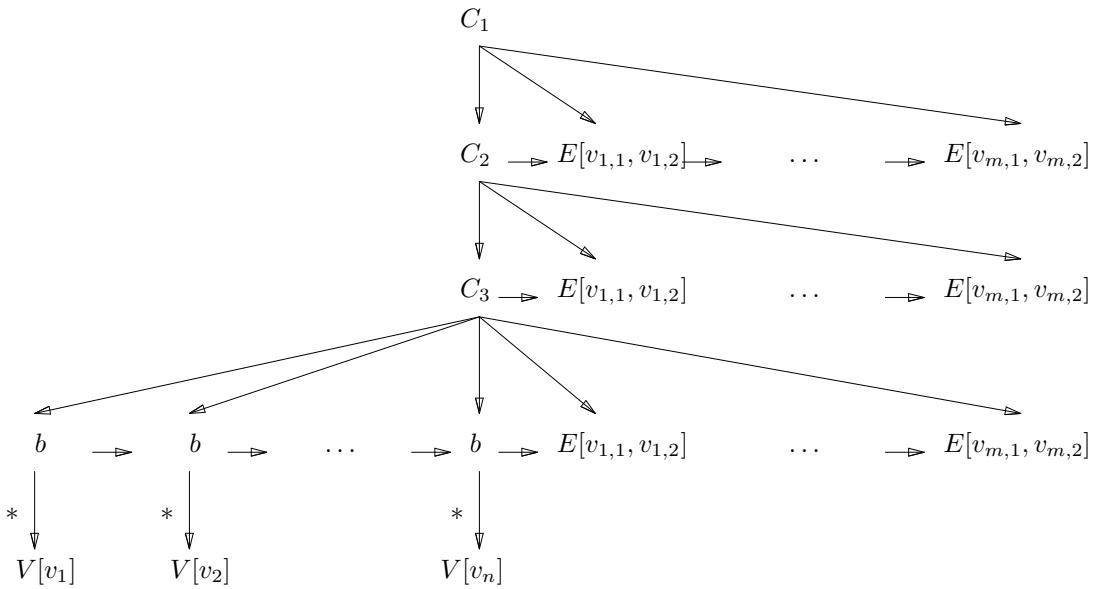
By proceeding similarly to the previous proof, it can be easily shown that $a \in \mathit{certain}(q, t)$ if and only if there exists no string $w \in \Sigma^*$ with $|w| \leq K$ that is a superstring of each s_i in S.

A.7 Proof of Theorem 7.4

We next show that there exists a query $q \in \mathcal{UCQ}(\downarrow, \|)$ so that QueryAnswering(q) over $(\downarrow, \rightarrow, \downarrow^*)$-incomplete trees is coNP-hard. From Proposition 7.1, QueryAnswering(q) is coNP-complete.

The proof is by reduction from 3-Colorability. Given a graph $G = \langle V, E \rangle$, with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$, for each $i \in [1, m]$, we let e_i be $(v_{i,1}, v_{i,2})$ with $v_{i,1}, v_{i,2} \in V$. We next show how to build a $(\downarrow, \rightarrow, \downarrow^*)$-incomplete tree t and a fixed boolean query $q \in \mathcal{UCQ}(\downarrow, \|)$ such that $\mathit{certain}(q, t)$ is *false* if and only if G is 3-colorable.

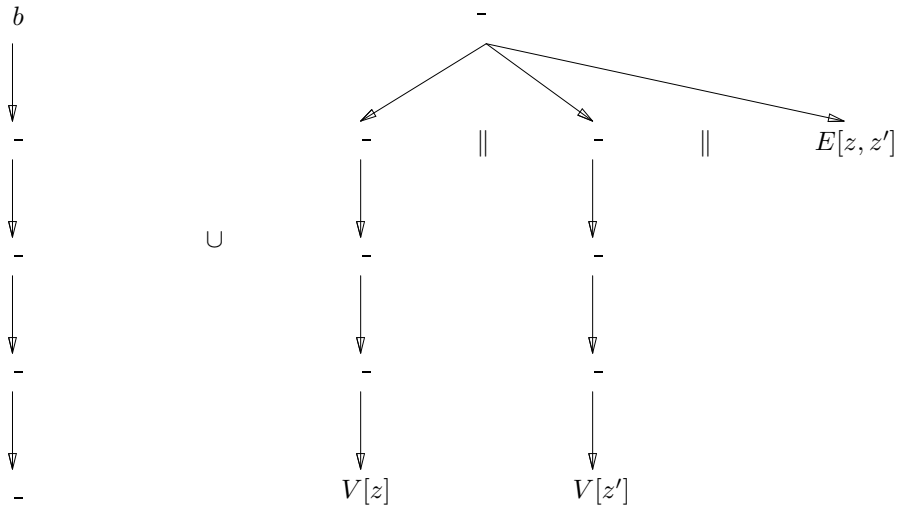
Let t be the following incomplete tree:



where the notation $E[v_{i,1}, v_{i,2}]$ indicates that the node is labeled E and has two attributes whose values are respectively $v_{i,1}$ and $v_{i,2}$. Similarly, the notation $V[v_j]$ indicates that the node is labeled V and has an attribute whose value is v_j. Notice that we use the vertices of the graph as attribute values in t, and all attribute values of t are constants (that is, we assume $\mathcal{D} \supseteq V$). Intuitively, nodes labeled E encode the edges of the graph, while nodes labeled V encode the vertices of the graph.

Given a node s of a complete tree, in what follows we call l-ancestor (l-descendant) of s, a node s' such that there exists a path of l child-edges from s' to s (from s to s') in the tree.

Now, let q be a union of two boolean conjunctive queries $q = q_1 \cup q_2$ where $q_1 = t_{q_1}$ and $q_2 = \exists z, z'\, t_{q_2}(z, z')$, and $t_{q_1} \cup\ t_{q_2}(z, z')$ is the following:



Here the notation used for node descriptions is the same as in t above.

It is straightforward to verify that, given a complete tree T:

- $q_1(T)$ is *true* if and only if there exists a b-labeled node of T which has a 4-descendant;
- $q_2(T)$ is *true* if and only if there is a pair of nodes of T with node descriptions $V[v]$ and $V[v']$, having a common 4-ancestor s_0, such that s_0 has a child with node description $E[v, v']$.

We next show that there exists a tree $T \in \mathit{Rep}(t)$ such that $q(T)$ is *false* if and only if G is 3-colorable.

Assume first that G is 3-colorable, and let $c : V \to [1, 3]$ be a 3-coloring of G. Then we construct a tree T encoding c as follows. In t, depicted above, we replace each descendant edge with a path of child edges. In particular, if $c(v_i) = j$ (with $j \in [1, 3]$), then the descendant edge of t terminating in the node with description $V[v_i]$ is replaced with a path of j child edges. Nodes in this path, between node descriptions b and $V[v_i]$, are assigned a new label, different from the ones occurring in t. This defines a complete tree T which, by construction, is in $\mathit{Rep}(t)$. Intuitively the depth of the V-labeled nodes of T encodes the color associated to the corresponding vertex of G. In other words, the 4-ancestor of the node of T with description $V[v_i]$ gives the color associated to v_i: either C_1 or C_2 or C_3.

Furthermore $q(T)$ is false, in fact:

- $q_1(T)$ is false, since no b labelled node of T has a 4-descendant.
- Assume by contradiction that $q_2(T)$ is true, then there exist V-labeled nodes of T, with node descriptions $V[v_i]$ and $V[v_j]$, having a common 4-ancestor s_0, such that s_0 has a child with node description $E[v_i, v_j]$.

  Now observe that the 4-ancestor of each V-labeled node of T is a C_l-labeled node of T ($l \in [1, 3]$). Each C_l-labeled node of T has a child with node description $E[v, v']$ if and only if (v, v') is an edge of G. Therefore (v_i, v_j) is an edge of G.

  On the other hand, by construction of T, if two V-labeled nodes of T have a common 4-ancestor, their corresponding vertices of G are assigned the same color by c. Hence $c(v_i) = c(v_j)$; this contradicts the fact that c is a 3-coloring.

This proves one direction. For the other direction assume that there exists a tree $T \in \mathit{Rep}(t)$ with $q(T) = \mathit{false}$. We next prove that G has a 3-coloring.

Since $T \in \mathit{Rep}(t)$, there exists a homomorphism h from $\underline{\mathit{rel}}(t)$ to T. Let s_1, s_2 and s_3 be the images by h of the nodes of t with labels C_1, C_2 and C_3. Let also $s_{v_1}, \ldots, s_{v_n}$ be the images by h of the nodes of t with descriptions $V[v_1], \ldots, V[v_n]$. Moreover each node s_{v_i} of T has a k_i-ancestor with label b (for some k_i) which is a child of s_3. Because $q(T)$ is false, and in particular $q_1(T)$ is *false*, we have that $1 \leq k_i \leq 3$, for each $i \in [1, n]$.

We now define a function $c : V \to [1, 3]$ assigning to each vertex v_i of G the color $c(v_i) = k_i$. Intuitively we let the depth of nodes s_{v_i} of T encode the color assigned to the vertex v_i of G.

Now assume, by contradiction, that c is not a 3-coloring of G. Then there exists and edge (v_i, v_j) of G such that $c(v_i) = c(v_j)$. By construction of c, this implies that nodes s_{v_i} and s_{v_j} of T have the same depth, that is, $k_i = k_j$. As a consequence nodes s_{v_i} and s_{v_j} have the same 4-ancestor $s \in \{s_1, s_2, s_3\}$.

Moreover, since (v_i, v_j) is an edge of G, the node s has a child with description $E[v_i, v_j]$. It follows that there exists a valuation ν such that $(T, \nu, s) \models t_{q_2}$ having $\nu(z) = v_i$ and $\nu(z') = v_j$. This implies $q_2(T) = \mathit{true}$, which is a contradiction, thus concluding the reduction from 3-Colorability.

Notice that this reduction does not use the fact that node ids in t are from $\mathcal{V}_{\text{node}}$, and no homomorphism can map two nodes of t into the same node of a tree (either by rigidity or because of distinct labels). Therefore the reduction holds verbatim for DOM incomplete trees.

We now show that there exists a query q from $\mathcal{UCQ}(\downarrow, \rightarrow, \rightarrow^*)$ such that QueryAnswering(q) over $(\downarrow, \rightarrow, \rightarrow^*)$-incomplete trees (as well as over $(\downarrow, \rightarrow, \rightarrow^*)$-incomplete DOM trees) is coNP-hard.

Assume again that G is the graph $\langle V, E\rangle$, with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. Let t be

$$t = r\langle t_1 \to t_2 \to \ldots \to t_n \to \beta_{e_1} \to \ldots \to \beta_{e_m}\rangle$$

where intuitively the trees β_{e_j} encode edges of the graph, and the trees t_i encode assignments of colors to vertices v_i. That is,

$$\beta_{(v_i,v_j)} = E[@n_1 = v_i, @n_2 = v_j]$$

and

$$t_i = A\langle C[@c = 0] \to C[@c = 1] \to C[@c = 2] \to^* N[@n = v_i]\rangle$$

where $0, 1, 2$ and v_i, for $i \in [1, n]$, are assumed to be in $\mathcal{D}$, that is, they are constants in t.

Intuitively t_i encodes the color assigned to v_i as the third preceding sibling of node $N[@n = v_i]$. Node ids of t are all distinct and are omitted, they can be either all from $\mathcal{V}_{\text{node}}$ (and in this case t is a $(\downarrow, \rightarrow, \rightarrow^*)$-incomplete tree) or all from $\mathcal{I}$ (and in this case t is a $(\downarrow, \rightarrow, \rightarrow^*)$-incomplete DOM tree).

We now define a boolean query $q = q_1 \cup q_2$ (independent of G) where q_1 and q_2 are from $\mathcal{CQ}(\downarrow, \rightarrow, \rightarrow^*)$, and show that G is 3-colorable if and only if the certain answer $\mathit{certain}(q_1 \cup q_2, t)$ is false. The query q_1 is represented by the following incomplete tree without attributes (and intuitively asks where there exists a node labeled A with at least seven children):

$$t_{q_1} = A\langle \_ \to \_ \to \_ \to \_ \to \_ \to \_ \to \_\rangle$$

and $q_2 = \exists z_1, z_2, z\ t_{q_2}(z_1, z_2, z)$ where:

$$t_{q_2} = r\langle t_{z_1} \to^* t_{z_2} \to^* E[@n_1 = z_1, @n_2 = z_2]\rangle$$

and, for $w = z_1, z_2$:

$$t_w = A\langle C[@c = z] \to \_ \to \_ \to N[@n = w]\rangle$$

Assume G is 3-colorable and let $c : V \to [0, 2]$ be a 3-coloring of G. We now construct a tree $T \in \mathit{Rep}(t)$ from c, and show that $q(T)$ is false. This will imply $\mathit{certain}(q, t)$ false. Intuitively T

coincides with t in the rigid part, and uses the distance of nodes $N[@n = v_i]$ from their C siblings to encode colors assigned to v_i. It is formally defined as follows:

$$T = r\langle T_1\ T_2 \dots T_n\ T_{e_1} \dots T_{e_m}\rangle$$

where trees

$$T_{(v_i,v_j)} = E[@n_1 = v_i, @n_2 = v_j]$$

and

$$T_i = A\langle C[@c = 0]\ C[@c = 1]\ C[@c = 2]\ D \dots D\ N[@n = v_i]\rangle.$$

The number of D nodes separating the node $N[@n = v_i]$ from its C siblings is defined as the color $c(v_i)$. Therefore notice that the third preceding sibling of the node $N[@n = v_i]$ in T is a node $C[@c = c(v_i)]$.

Node ids are omitted in T, but it should be remarked that in the case t is an incomplete tree, they are new fresh distinct node ids from $\mathcal{I}$. Otherwise (if t is an incomplete DOM tree), all node ids of T, except the D nodes, are defined as coinciding with their corresponding node in t.

By construction, $T \in \mathit{Rep}(t)$. We now show that $q(T) = \mathit{false}$. Indeed all A-labeled nodes of T have at most six children, so $q_1(T) = \mathit{false}$. Now assume, by contradiction that $q_2(T)$ is true. Then t_{q_2} must be satisfied in the root of T – being the root the only node of T with label r. This implies that, for some values v_i, v_j and e: 1) the root of T has a child with node description $E[@n_1 = v_i, @n_2 = v_j]$, and 2) there exist two nodes of T with description $N[@n = v_i]$ and $N[@n = v_j]$, respectively, having both a third preceding sibling with node description $C[@c = e]$.

Now notice that by construction of T: 1) if a node of T with description $N[@n = v]$ has a third preceding sibling with node description $C[@c = e]$, then $c(v) = e$. Therefore $c(v_i) = c(v_j) = e$. 2) the root of T has a child with node description $E[@n_1 = v, @n_2 = v']$ if and only if (v, v') is an edge of G. Thus (v_i, v_j) is an edge of G. This contradicts the fact that c is a coloring of G. Hence $q_2(T) = \mathit{false}$, thus $q(T) = \mathit{false}$ and $\mathit{certain}(q, t) = \mathit{false}$.

Now assume that $\mathit{certain}(q, t) = \mathit{false}$. We next prove that there exists a 3-coloring of G. We know that there exists a tree $T \in \mathit{Rep}(t)$ such that $q_1(T)$ and $q_2(T)$ are false. Let h be a homomorphism from t to T, let also u be the root id of t and u_i, for $i \in [1, n]$, the root ids of subtrees t_i of t, and u_{e_j} the id of β_{e_j}, for all $j \in [1, m]$. (Remark that these node ids maybe either from $\mathcal{V}_{\text{node}}$ or $\mathcal{I}$). Since $q_1(T)$ is false, all A-labeled nodes of T have at most six children. This is true in particular for nodes $h(u_i)$ of T. Moreover, by the fact that $T \in \mathit{Rep}(t)$, the sequence of children of $h(u_i)$ must contain a subsequence $C[@c = 0]\ C[@c = 1]\ C[@c = 2]\ s_i\ N[@n = v_i]$, where s_i must be a possibly empty sequence of at most two nodes. Let i_i be the id of the last node ($N[@n = v_i]$) of this subsequence, then notice that the third preceding sibling of i_i in T is a node $C[@c = |s_i|]$. We now show that the mapping assigning color $|s_i|$ to each node v_i of G is a 3-coloring. Indeed assume by contradiction that this is not a 3-coloring, then there exists an edge (v_i, v_j) of G with $|s_i| = |s_j|$. This implies that the third preceding siblings of nodes i_i and i_j of T have both label C and an attribute $@c = |s_i|$. Moreover, because h is a homomorphism, the parents $h(u_i)$ and $h(u_j)$ of i_i and i_j have both parent $h(u)$ of label r. In turn $h(u)$ must have children $E[@n_1 = v_i, @n_2 = v_j]$ and $E[@n_1 = v_j, @n_2 = v_i]$ (because we can assume G undirected) which follow $h(u_n)$.

As a consequence t_{q_2} is satisfied in node $h(u)$ of T, this contradicts the hypothesis that $q_2(T)$ is false, and concludes the reduction and the proof of the theorem.

A.8 Proof of Proposition 7.5

We next prove that there exists a DTD d and a query q in $\mathcal{CQ}(\downarrow, \|)$ such that QueryAnswering(q, d) is coNP-hard for $(\downarrow, \downarrow^*, \|)$-incomplete DOM-trees. Thus, by Proposition 7.1, QueryAnswering(q, d) is coNP-complete for $(\downarrow, \downarrow^*, \|)$-incomplete DOM-trees.

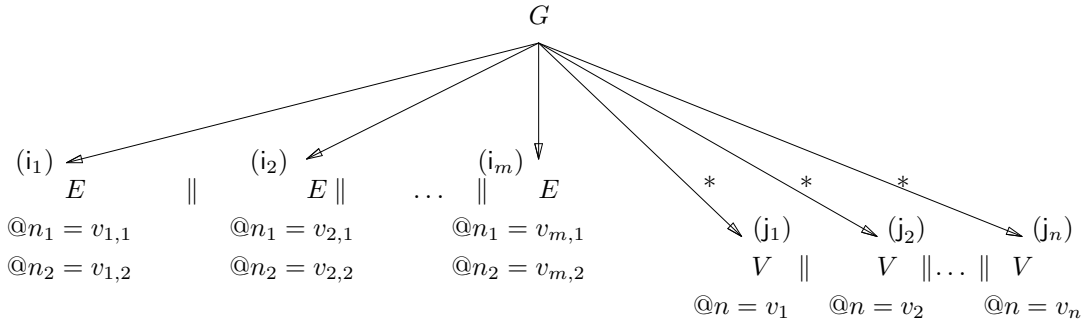
Let d be the following DTD:

$$\begin{array}{ll} G & \rightarrow E^* C_1 C_2 C_3 \\ C_i & \rightarrow V^*, i = 1, 2, 3 \\ V & \rightarrow \varepsilon \\ E & \rightarrow \varepsilon \end{array}$$

where E has two attributes n_1 and n_2 and V has an attribute n. So, a node labeled E encodes an edge between two vertices which are specified in the attributes n_1 and n_2. Moreover, a node labeled V encodes a vertex, which is specified in the attribute n, and whose parent encodes the color assigned to the vertex, that can be either C_1,C_2,C_3.

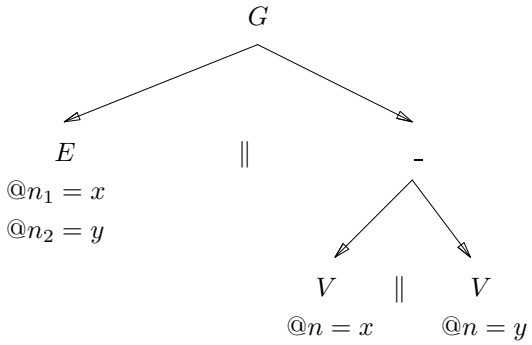
The proof is by reduction from 3-Colorability. Let $G = \langle V, E \rangle$ be a directed graph, with $V = \{v_1, ..., v_n\}$ and $E = \{e_1, ..., e_m\}$. We show how to build a $(\downarrow, \downarrow^*, \|)$-incomplete DOM-tree t and a query q in $\mathcal{CQ}(\downarrow, \|)$ (whose size does not depend on G) such that $\mathit{certain}_d(q, t)$ is *false* if and only if G is 3-colorable.

Let t be the following incomplete DOM-tree:



where $\mathsf{i}_1, \ldots, \mathsf{i}_m, \mathsf{j}_1, \ldots, \mathsf{j}_n$ are distinct elements of $\mathcal{I}$, and $(v_{i,1}, v_{i,2}) = e_i$ for every $i \in [1, m]$. Intuitively, the set of nodes labeled E and V encode respectively the set of edges and vertices of the graph.

Now, let q be the query $q = \exists x, y\ t_q(x, y)$, where t_q is the following incomplete tree:



We next show that $\mathit{certain}_d(q, t)$ is *false* if and only if G is 3-colorable.

Assume first that $\mathit{certain}_d(q, t)$ is *false*. Let T be a tree in $\mathit{Rep}_d(t)$ such that $q(T)$ is *false*. Since T satisfies the DTD d, for each $i \in [1, n]$, the node j_i of T (with label V) has a parent labeled C_k, for some $k \in [1, 3]$. Define a coloring function c associating to each vertex v_i of G the label C_k of the parent of j_i in T. The mapping c is a 3-coloring of G. In fact, assume by contradiction that c is not a 3-coloring. Then there exists two vertices, v_p and v_l such that j_p and j_l have a parent with label C_k in T, and $(v_p, v_l) \in E$. Because T satisfies d, the vertices j_p and j_l must have the same parent labeled C_k, which must in turn be a child of the root of T (labeled G). On the other hand, the root of T must have a child with label E and $(@n_1, @n_2) = (v_p, v_l)$. Then T satisfies the query q, which is a contradiction.

Assume now that G is 3-colorable and let $c : V \to \{C_1, C_2, C_3\}$ be a 3-coloring of G. Construct a tree T having a root labeled G with: 1) one E-labeled child with node id i_j for each edge e_j of G, $j \in [1, m]$, such that the value of attributes $(@n_1, @n_2)$ equals e_j; 2) three children with labels C_1, C_2 and C_3 respectively. Each of these nodes with label C_k has a set of V-labeled children assigned as follows: the node with label C_k has a V-labeled child with node id j_i and attribute value v_i for each vertex v_i of G having $c(v_i) = C_k$. No other node is in T. Clearly $T \in \mathit{Rep}_d(T)$. Moreover, since c is a 3-coloring, $q(T)$ is false. This shows that $\mathit{certain}_d(q, t)$ is *false* and concludes the proof of the proposition.