Rapport de Recherche

An axiom system for XML and an algorithm for filtering XFD

Joshua Amavi, Mirian Halfeld Ferrari
LIFO, Université d'Orléans

Rapport n^o **RR-2012-03**

An axiom system for XML and an algorithm for filtering XFD

A step to interoperability

Joshua Amavi Mirian Halfeld Ferrari

Université d'Orléans - LIFO - Orléans - France
{joshua.amavi, mirian}@univ-orleans.fr

Abstract. XML has become the *de facto* format for data exchange. A web application expected to deal with XML documents conceived on the basis of divers sets of (local) constraints would be expected to test documents with respect to all non contradictory constraints imposed by these original (local) sources. The goal of this paper is to introduce an optimized algorithm for computing the maximal set of XML functional dependencies (XFD) over multiple systems. The basis of our method is a sound and complete axiom system which is provided for relative XFD allowing two kinds of equality. This paper covers important theoretical and applied aspects of XFD. Not only do we present proofs and algorithms but we also show the result of some experiments that emphasize the interest of our approach.

Keywords: XML, Functional dependencies, XFD, constraints, interoperability

1 Introduction

Let $S_1, \dots, S_n$ be local systems, as shown in Figure 1, which deal with sets of XML documents $X_1, \dots, X_n$, respectively, and that inter-operate with a global, integrated system S. System S may receive information from any local system. Each set X_i conforms to schema constraints $\mathcal{D}_i$ and to integrity constraints $\mathcal{F}_i$ and follows an ontology O_i. Our goal is to associate system S to type and integrity constraints which represent a conservative evolution of local constraints. More precisely, given different triples $(\mathcal{D}_1, \mathcal{F}_1, O_1), \dots, (\mathcal{D}_n, \mathcal{F}_n, O_n)$, we are interested in generating $(\mathcal{D}, cover\mathcal{F}, \mathcal{A})$, where:
(i) $\mathcal{D}$ is an extended type that accepts any local document;
(ii) $cover\mathcal{F}$ is a set of XFD equivalent to $\mathcal{F}$ the biggest set of functional dependencies (XFD), built from $\mathcal{F}_1, \dots, \mathcal{F}_n$, that can be satisfied by all documents in $X_1, \dots, X_n$ and
(iii) $\mathcal{A}$ is an ontology alignment that guides the construction of $\mathcal{D}$ and $\mathcal{F}$ in terms of semantics mapping.

This paper focus only on the generation of $cover\mathcal{F}$ which contains the XFD for which no violation is possible when considering document sets $X_1, \dots, X_n$.

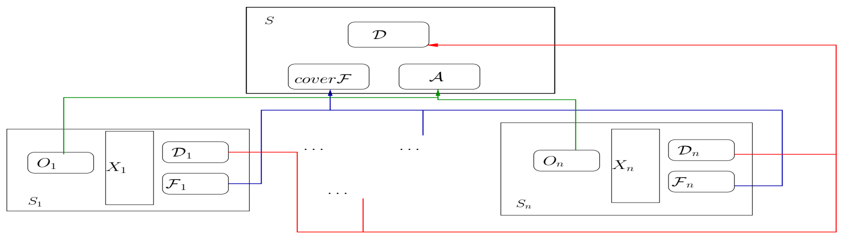

Fig. 1. System S interoperability. Schema and integrity constraints in system S are built from constraints in local system. Type $\mathcal{D}$ accepts documents from all types $\mathcal{D}_1 \ldots \mathcal{D}_n$. Integrity constraints in *cover*$\mathcal{F}$ are obtained by filtering non-contradictory XFD obtained from $\mathcal{F}_1, \ldots, \mathcal{F}_n$. The alignment $\mathcal{A}$, built from ontologies $O_1, \ldots, O_n$, allows finding equivalent concepts. Each local system deals with a set of XML document X_i.

It is important to notice that our algorithm is based on an axiom system and, thus, obtains *cover*$\mathcal{F}$ from $\mathcal{F}_1, \ldots, \mathcal{F}_n$, disregarding data.

The contribution of this paper is twofold. On one hand we introduce an axiom system together with the proofs of its soundness and completeness. On the other hand, we present an efficient way for computing, on the basis of our axiom system, the set *cover*$\mathcal{F}$. We prove that the obtained set *cover*$\mathcal{F}$ has good properties and some experiments show the efficiency of our approach.

The rest of this paper is organized as follows. Section 2 illustrates our goal with an example. Section 3 presents some background while Section 4 introduces our XFD. Section 5 focuses on our axiom system. An algorithm for computing the closure of a set of paths, *w.r.t.* a set of XFD, is presented in Section 6. Section 7 introduces our method for computing $\mathcal{F}$ and *cover*$\mathcal{F}$ while in Section 8 we discuss on some experiments. Finally, Section 9 comments on some related work and Section 10 concludes the paper.

2 Motivating Example

Let the XML trees in Figure 2 be documents from two different local sources. Each document is valid *w.r.t.* the functional dependencies presented in Table 1, *i.e.*, documents in X_1 are valid *w.r.t.* $\mathcal{F}_1$, those in X_2 are valid *w.r.t.* $\mathcal{F}_2$.

In the XML domain, a functional dependency (XFD) is defined by paths over a tree. Each path selects a node on a tree. Values or positions of the selected nodes are gathered to build tuples that will be used to verify whether a given XML document satisfies an XFD. For example, consider the XFD *f: (univ/, (undergraduate/courses/course/codeC,* $\rightarrow$ *undergraduate/courses/course/titleC))* on the first document of Figure 2. It specifies that the context is *univ*, *i.e.*, that the constraint should be verified on data below a node labelled *univ.* In this context, f entails the construction of tuples composed by values obtained by following the paths:

univ/undergraduate/courses/course/codeC, univ/undergraduate/courses/course/titleC. As in the relational model, a document is valid *w.r.t.* f if any two tuples agreeing on values obtained from *univ/undergraduate/courses/course/codeC* also agree on values obtained from *univ/undergraduate/courses/course/titleC*. Therefore, in a university the code of a course determines its name. Similarly, the XFD *f1: univ/(undergraduate/courses/course/codeC* $\rightarrow$ *undergraduate/courses/course/prerequisiteC))* entails tuples where the path *univ/undergraduate/courses/course/prerequisiteC)* leads us to obtain sub-trees having roots labelled *prerequisiteC* (*i.e.*, sub-trees containing information about prerequisites). A document is valid *w.r.t.* $f1$ if any two tuples agreeing on values obtained from *univ/undergraduate/courses/-course/codeC* also agree on values obtained from *univ/undergraduate/courses/course/prerequisiteC*, *i.e.*, obtained sub-trees are isomorphic.

Now consider the first three XFD in Table 1, concerning source 1. They indicate that in a university, the code of a course determines its name, its domain and its prerequisites. In other words, a course is identified by its code.

From the alignment of local ontologies we assume that Table 2 is available, making the correspondence among paths on the different local sets of documents. Thus, it is possible to conclude that, for instance, XFD *f: (univ/, (undergraduate/courses/course/codeC,* $\rightarrow$ *undergraduate/courses/course/titleC))* and *(univ/, (courses/course/codeC,* $\rightarrow$ *courses/course/nameC))* are equivalent, *i.e.*, they represent the same constraint since they involve the same concepts: in a university, the code of a course determines its name.

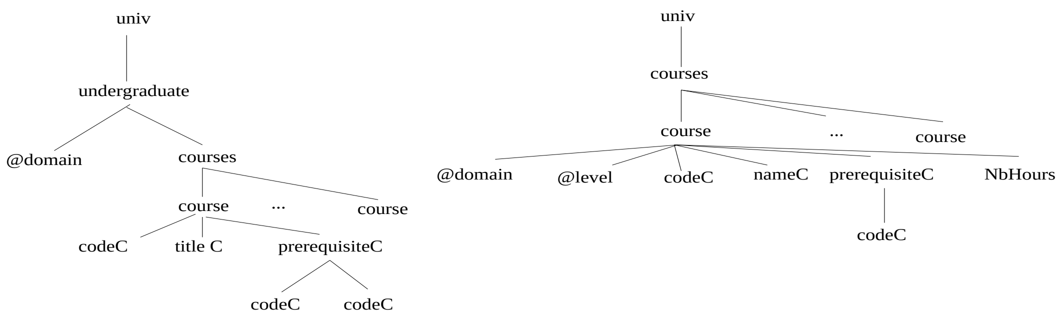

Fig. 2. Two XML documents from different local sources.

Assuming that we have only these two local sources, we want to obtain, from $\mathcal{F}_1$ and $\mathcal{F}_2$, the biggest set of XFD $\mathcal{F}$ that does not contradict any document in X_1 and X_2. To reach this goal, we should consider *all* XFD derivable from $\mathcal{F}_1$ and $\mathcal{F}_2$, which may result in very big sets of XFD. Indeed, the set $\mathcal{F}$ is, usually, a very big one - too big to work with. A better solution consists in computing *cover*$\mathcal{F}$, *a cover of* $\mathcal{F}$ (*i.e.*, a (usually) smaller set of XFD that is equivalent to $\mathcal{F}$), without computing *all* XFD derivable from $\mathcal{F}_1$ and $\mathcal{F}_2$. In this paper, we propose an algorithm that generates this set of XFD.

$\mathcal{F}$	XFD
1	$(univ/, (undergraduate/courses/course/codeC, \rightarrow undergraduate/courses/course/titleC))$
1	$(univ/, (undergraduate/courses/course/codeC, \rightarrow undergraduate/courses/course/prerequisiteC))$
1	$(univ/, (undergraduate/courses/course/codeC, \rightarrow undergraduate/@domain))$
2	$(univ/, (courses/course/codeC, \rightarrow courses/course/nameC))$
2	$(univ/, (courses/course/codeC, \rightarrow courses/course/@domain))$
2	$(univ/, (courses/course/codeC, \rightarrow courses/course/@level))$
2	$(univ/, (\{courses/course/nameC, courses/course/@level\} \rightarrow courses/course/NbHours))$

Table 1. XFD in $\mathcal{F}_1$ and $\mathcal{F}_2$.

Paths from $\mathcal{D}_1$	Paths from $\mathcal{D}_2$
univ/undergraduate/courses/course/codeC	*univ/courses/course/codeC*
univ/undergraduate/courses/course/titleC	*univ/courses/course/nameC*
univ/undergraduate/courses/course/prerequisitesC	*univ/courses/course/prerequisitesC*
univ/undergraduate/courses/course/prerequisitesC/codeC	*univ/courses/course/prerequisitesC/codeC*
univ/undergraduate/@domain	*univ/courses/course/@domain*

Table 2. Extract of the translation table

In our example, the resulting *cover*$\mathcal{F}$ would contain XFD of Table 3. Let us analyse this solution. In Table 3, the first and the fourth XFD are equivalent. They are kept in *cover*$\mathcal{F}$ since all documents in X_1 and X_2 are valid *w.r.t.* it. The same reasoning is applied for the second and third XFD in Table 3. The two last XFD involve concepts that occur only in X_2 and, thus, cannot be violated by documents in X_1. Notice that the XFD $(univ/, (undergraduate/courses/course/codeC, \rightarrow undergraduate/courses/course/prerequisiteC))$ in $\mathcal{F}_1$, which states that courses with the same code have the same set of prerequisites, is not in $\mathcal{F}$. The reason is that, according to the ontology alignment, this XFD is equivalent to *(univ/, (courses/course/codeC, → courses/course/prerequisitesC))* in $\mathcal{F}_2$. However, as $\mathcal{F}_2$ does not contain this XFD, documents in X_2 may violate it (since the involved concepts exist in X_2).

$(univ/, (undergraduate/courses/course/codeC, \rightarrow undergraduate/courses/course/titleC))$
$(univ/, (undergraduate/courses/course/codeC, \rightarrow undergraduate/@domain))$
$(univ/, (courses/course/codeC, \rightarrow courses/course/@domain))$
$(univ/, (courses/course/codeC, \rightarrow courses/course/nameC))$
$(univ/, (courses/course/codeC, \rightarrow courses/course/@level))$
$(univ/, (\{courses/course/nameC, courses/course/@level\} \rightarrow courses/course/NbHours))$

Table 3. XFD in the resulting set $\mathcal{F}$.

3 Preliminaries

Our work uses XFD such as those in [2, 4]. This section recalls and introduces some important concepts necessary in the definition of our XFD semantics. As usual, XML document are seen as unranked labelled trees. We assume a finite alphabet Σ and a set of positions U which denotes the set $\mathbb{N}^*$ of all finite strings of positive integers with the empty string ϵ (*i.e.*, Dewey numbering).

Definition 1 (Prefix relation for positions). *The prefix relation $\preceq_{pos}$ in U is: $u \preceq_{pos} v$ iff $uw = v$ for some $w \in U$. We say that D ($D \subseteq U$) is closed under prefixes if $u \preceq_{pos} v$, $v \in D$ implies $u \in D$.* $\square$

Definition 2 (XML Document). *Let $\Sigma = \Sigma_{ele} \cup \Sigma_{att} \cup \{data\}$ be an alphabet where Σ_{ele} is the set of element names and Σ_{att} is the set of attribute names. An XML document is represented by a tuple $\mathcal{T} = (t, type, value)$. The tree t is the function $t : dom(t) \to \Sigma \cup \{\lambda\}$ where $dom(t)$ is the set closed under prefixes of positions $u.j$, such that $(\forall j \geq 0)$ $(u.j \in dom(t)) \Rightarrow (\forall i \quad 0 \leq i < j)$ $(u.i \in dom(t))$; where i and $j \in \mathbb{N}$ and $u \in U$. The symbol ϵ denoting the empty sequence is the root position and the empty tree is $\{(\epsilon, \lambda)\}$, with $\lambda \notin \Sigma$. Given a tree position p, function $type(t, p)$ returns a value in {data, element, attribute}.*

Similarly, $value(t,p) = \begin{cases} p & \text{if } type(t,p) = element \\ val \in \boldsymbol{V} & otherwise \end{cases}$

where $\boldsymbol{V}$ is an infinite recursively enumerable domain. $\square$

As many other authors, we distinguish two kinds of equality in an XML tree, namely, *value and node equality.* Two nodes are *value equal* when they are roots of isomorphic sub-trees. Two nodes are *node equal* when they are at the same position. To combine both equality notions we use the symbol E, that can be represented by V for value equality, or N for node equality. Notice that our value equality definition does not take into account the document order. For instance, in Figure 3, nodes in positions 1.1.1 and 0.1.1 are value equal, but nodes 1.1 and 0.1 are not (elements *title* in their subtrees are associated to different data values).

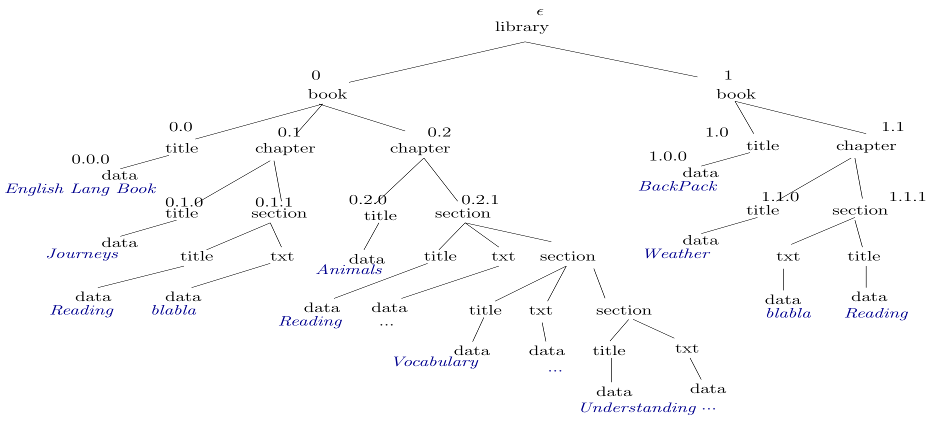

Fig. 3. XML document: a library of English books. Each node has a position and a label. For instance, $t(\epsilon) = library$ and $t(0.1.1) = section$.

Linear paths are used to address parts of an XML document. In our work, paths are defined by using the path language PL.

Definition 3 (Path Language and Paths). *Let PL be a path language where a path is defined by*

$$\rho ::= [] \mid \ l \mid \rho/\rho \mid \rho//l$$

where $[]$ is the empty path, l is a label in Σ, "/" is the concatenation operation, "//" represents a finite sequence (possibly empty) of labels. Notice that $l/[] = []/l = l$ and $[]//l = //l$.
We distinguish between paths using the wild-card // and simple paths (those with no wild-card) and we denote by $\mathbb{P}$ the set of all possible rooted simple paths that may occur in an XML tree t respecting a given schema $\mathcal{D}$. □

In this work we consider that the set $\mathbb{P}$ is generated from a given schema $\mathcal{D}$. Notice that $\mathbb{P}$ is a *finite* set of simple (top-down) paths and, in this way, the schema from which it is obtained should ensure a limited *depth* of label repetitions. In other words, the language $L(D)$, obtained from a finite state automaton D (which is built from a given type $\mathcal{D}$), should be finite. Such kind of schema can be expressed, for instance, by a non-recursive DTD. In this way, we are more general than [2, 4], where $\mathbb{P}$ was the set containing only all possible paths in *one* given tree.

It is important to notice that one path with wild-card can be associated to a *set* of simple paths in $\mathbb{P}$. This set of simple paths is the language $L(A_P)$, where A_P is a finite-state automaton obtained on the basis of the two definitions bellow.

Definition 4 (Finite state automaton B_P built from P). *Given a path P in PL over Σ, we define the FSA B_P as a 5-tuple $(Q, \Sigma, I, \delta, F)$ where Q is a finite set of states, s_0 is the initial state, δ is the transition function $\delta : Q \times (\Sigma \cup \{any\}) \to Q$ and F is the final state. The construction of B_P is done by parsing P, starting with $Q = \{s_0\}$. Let $s = s_0$ be the current state in B_P. If P is the empty path then $F = \{s_0\}$, else while the end of P is not reached, let $\mathcal{C} = \Sigma \cup \{[\ ]\}$ and let a be the next $\mathcal{C}$ symbol in P:*
(1) If $a \in \Sigma$ comes alone or a comes on the right of a '/' symbol (i.e., $/a$), then we add a new state s_i ($i \in \mathbb{N}$) to Q and define $\delta(s, a) = s_i$. Let $s = s_i$ be the current state in B_P.
(2) If $a \in \Sigma$ comes on the right of a '//' symbol (i.e., $//a$), we add a new state s_i to Q and define $\delta(s, a) = s_i$. We also define $\delta(s, any) = s$, where any is any symbol in Σ. Let $s = s_i$ be the current state in B_P.
(3) If $a =' [\ ]'$ comes on the right of a '/' symbol, we do nothing and continue parsing P.
At the end of P, the current state is added to F. □

The automaton B_P of Definition 4 has always the same format since paths in PL are similar to restricted regular expressions. The deterministic FSA equivalent to B_P is easily obtained (just with more transitions).

Definition 5 (Finite state automaton A_P associated to P in the context of $\mathbb{P}$). *Given a path P in PL and a set $\mathbb{P}$ of simple paths, the FSA A_P is the one defined by path P such that $L(A_P) = L(B_P) \cap \mathbb{P}$, where B_P is the FSA of Definition 4.* □

Example 1. We suppose a DTD, concerning a library of English books as the one in Figure 3, which constrains sections to have at most two lower levels of subsections. The FSA D built from this DTD is the one in Figure 4(a) and the language $L(D) = \mathbb{P}$. Notice that D recognizes the language of prefixes of the paths defined by the given DTD. Let $P = $ *library//section/title*. The FSA built from P and its deterministic counterpart are in Figure 4(b). Clearly $L(B_P) \cap \mathbb{P}$ gives the set of simple paths associated to P in the context of $\mathbb{P}$. In other words, this set contains only the simple paths in $L(B_P)$ that trees respecting $\mathbb{P}$ may have, *i.e.*, { *library/book/chapter/section/title, library/book/chapter/section/section/title, library/book/chapter/section/section/section/title*}. Clearly, $A_P = D \cap B_P$. □

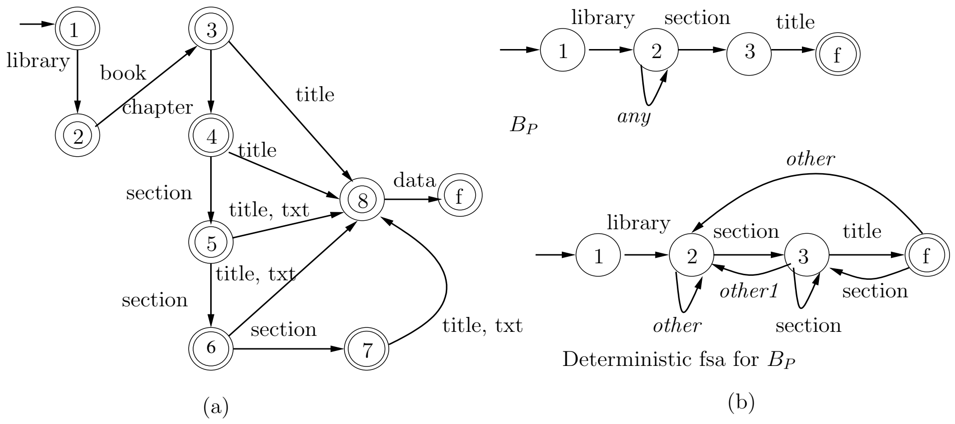

Fig. 4. (a) Automaton D built from a DTD ($L(D)) = \mathbb{P}$). (b) Automaton B_P (Def. 4) for $P = $ *library//section/title* and its deterministic counterpart (where *other* stands for $\Sigma \setminus \{section\}$ and *other1* stands for $\Sigma \setminus \{section, title\}$).

A path P is **valid** if: (i) it conforms to the syntax of PL, (ii) $L(A_P) \neq \emptyset$, (iii) for all label $l \in P$, if $l = data$ or $l \in \Sigma_{att}$, then l is the last symbol in P.

In this work, given a path P in PL we define the following functions:

– $Last(P) = l_n$ where l_n is the last label on path P.
– $Parent(P) = \{l_1/\dots/l_{n-1} \mid l_1/\dots/l_{n-1}/l_n \in L(A_P) \text{ for } n > 1\}$, the set of simple paths starting at a node labelled by l_1 and ending at the parent of l_n (where $Last(P) = l_n$).

Definition 6 (Prefix of a path). *Let P be a valid path and let A_P be its associated finite state automaton (Definition 5). Let $PREFIX(A_P)$ be the finite* *state automaton that accepts the language containing all prefixes of* $L(A_P)$[1]*. A* prefix *of* P *is a simple path* P' *such that* $P' \in L(PREFIX(A_P))$ *(and thus we note* $P' \preceq P$*). This notion is extended for non simple paths* Q *and* P *as follows: we write* $Q \preceq_{PL} P$*, if* $L(A_Q) \subseteq L(PREFIX(A_P))$*.* □

Definition 7 (The longest common prefix (path intersection)). *Let* P *and* Q *be valid paths and let* A_P *and* A_Q *be their associated finite state automaton (Definition 5). The longest common prefix of* P *and* Q *is the path* $P \cap Q$ *that describes the set of simple paths* $\{P' \cap Q' \mid P' \in L(A_P) \wedge Q' \in L(A_Q)\}$*. The longest common prefix of two simple paths* P' *and* Q' *(denoted* $P' \cap Q'$*) is the simple path* R *where* $R \preceq P'$ *and* $R \preceq Q'$ *and there is no path* R' *such that* $R \prec R'$*,* $R' \preceq P'$ *and* $R' \preceq Q'$*.* □

From Definition 7, notice that simple paths in the longest common prefix of paths P and Q are obtained by the intersection of each *two simple paths* described by P and Q. We refer to Appendix A for the computation of the longest common prefix.

Example 2. Consider the XML document of Figure 3 and the set $\mathbb{P} = L(D)$ of Figure 4. The simple path */library/book/chapter/section*, is a prefix for path */library/book/chapter/section/section/title*. Given P' =*/library/book/chapter/-section/section/title* and Q' = */library/book/chapter/section/section/txt*, their longest common prefix is */library/book/chapter/section/section*.
Given P = */library//section/title/data* and Q = */library//section/txt/data*, the longest common prefix $P \cap Q$ can be written as */library//section* since $P \cap Q = \{$ */library/book/chapter/section/*, */library/book/chapter/section/section/*, */library/book/chapter/section/section/section* $\}$ □

Definition 8 (Instance of a path P over t). *Let* P *be a path in* PL*,* A_P *the finite-state automaton associated to* P *in the context of* $\mathbb{P}$*, and* $L(A_P)$ *the language accepted by* A_P*. Let* $I = p_1/\ldots/p_n$ *be a sequence of positions such that each* p_i *is a direct descendant of* p_{i-1} *in* t*. Then* I *is an instance of* P *over a given tree* t *if and only if the sequence* $t(p_1)/\ldots/t(p_n) \in L(A_P)$*. We denote by* $Instances(P, t)$ *the set of all instances of* P *over* t*.* □

Some particular notations on path instances are now introduced:

– Given a path instance $I = p_1/\ldots/p_n$, we define $Last(I) = p_n$, the last position on I.
– Given $I = p_1/\ldots/p_n$ (for $n > 1$), we denote by $Parent(I)$ the sequence $p_1/\ldots/p_{n-1}$.
– Let $I = p_1/\ldots/p_n$ and $J = u_1/\ldots/u_m$ be path instances. We say that I is a prefix of instance J (also denoted by $I \preceq J$) if $n \leq m$ and $p_1 = u_1, \ldots, p_n = u_n$. We say that $I = J$ when $I \preceq J$ and $J \preceq I$.

[1] The automaton $PREFIX(A_P)$ is obtained from A_P (which has F as the set of final states and q_0 as the initial state) by replacing F by $F' = F \cup \{q \mid q$ is a state in A_P, reachable from q_0 and from which we can reach a state in $F\}$.

– Let $I = p_1/\ldots/p_n$ and $J = u_1/\ldots/u_m$ be path instances. The longest common prefix of these two instances, denoted by $I \cap J$, is the path instance $K = v_1/\ldots/v_k$ where $K \preceq J$ and $K \preceq I$ and there is no path instance K' such that $K \prec K'$, $K' \preceq J$ and $K' \preceq I$.

Example 3. In the document of Figure 3, $I_1 = \epsilon/0/0.1/0.1.0/0.1.0.0$, $I_2 = \epsilon/0/0.2/0.2.0/0.2.0.0$ and $I_3 = \epsilon/1/1.1/1.1.0/1.1.0.0$ are instances of path $P =$ */library/book/chapter/title/data*. We have $Last(I_1) = 0.1.0.0$ and $Parent(I_1) = \epsilon/0/0.1/0.1.0$. The longest common prefix of I_1 and I_2 is $\epsilon/0$ and of I_1 and I_3 is ϵ. □

Notice that the longest common prefix allows the identification of the least common anscestor and that Definition 7 gives the path for it.

We now remark that, in this paper, we will only deal with complete trees.

Definition 9 (Complete trees). *Let $\mathbb{P}$ be a set of simple paths associated to an XML document $\mathcal{T}$. We say that $\mathcal{T}$ is complete w.r.t. $\mathbb{P}$ if whenever there exist path P and P' in the associated $\mathbb{P}$ such that $P' \prec P$ and there exist an instance I' for P' such that node $v' \in Last(I')$, then there exists an instance I for P such that $v \in Last(I)$ and v' is an ancestor of v.* □

Example 4. Let $\mathbb{P}$ be composed by paths $R/A/C$, $R/A/D$, R/B and their prefixes. Figure 5 shows complete and non complete trees *w.r.t.* $\mathbb{P}$.

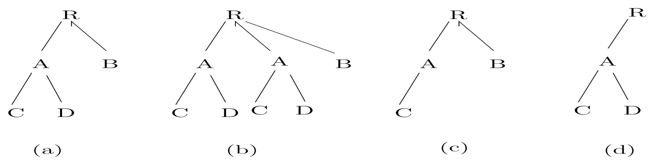

Fig. 5. Examples of complete trees ((a) and (b)) and non complete trees ((c) and (d)) *w.r.t.* $\mathbb{P}$.

Given two paths P and Q, the following definition allows us to verify whether two given path instances match on the longest common prefix of P and Q.

Definition 10 (Testing instances of the longest common prefix of paths). *Let P and Q be two valid paths over a tree t. The boolean function isInst_lcp(P, I, Q, J) returns true when all the following conditions hold: (i) $I \in Instances(P, t)$; (ii) $J \in Instances(Q, t)$ and (iii) $I \cap J$ is an instance of a path in $P \cap Q$; otherwise, it returns false.* □

Now we introduce the notion of branching paths also called a pattern in the literature [3, 17]. A branching path is a non-empty set of simple paths having a common prefix. The projection of a tree over a branching path determines the tree positions corresponding to the given path. Thus, as defined below, this projection is a set of prefix closed simple path instances that respect some important conditions.

Definition 11 (Branching path). *A branching path is a finite set of prefix-closed (simple) paths on a tree t.* □

Definition 12 (Projection of a tree $\mathcal{T}$ over a branching path M). *Let M be a branching path over a tree $\mathcal{T}$. Let $Long_M$ be the set of paths in M that are not prefix of other paths in M. Let $SetPathInst$ be the set of (simple) path instances that verifies:*

1. *For all paths $P \in Long_M$ there is one and only one instance $inst \in Instances(P, t)$ in the set $SetPathInst$.*
2. *For all $inst \in SetPathInst$ there is a path $P \in Long_M$ such that $inst \in Instances(P, t)$.*
3. *For all instances $inst$ et $inst'$ in $SetPathInst$, if $inst \in Instances(P, t)$ and $inst' \in Instances(Q, t)$, then $isInst\_lcp(P, inst, Q, inst')$ is true.*

A projection of $\mathcal{T}$ over M, denoted by $\Pi_M(\mathcal{T})$, is a tuple $(t^i, type^i, value^i)$ where $type^i(t^i, p) = type(t, p)$, $value^i(t^i, p) = value(t, p)$ and t^i is a function $\Delta \to \Sigma$ in which:

- *$\Delta = \bigcup_{inst\ \in\ SetPathInst}\{p \mid p$ is a position in $inst\}$*
- *$t^i(p) = t(p)$, $\forall p \in \Delta$* □

Given the projection of two branching paths, $\Pi_{M_1}(\mathcal{T})$ and $\Pi_{M_2}(\mathcal{T})$, the union $\Pi_{M_1}(\mathcal{T}) \cup \Pi_{M_2}(\mathcal{T})$ is naturally obtained by considering all the path instances used to obtain each projection.

Example 5. Consider the XML document of Figure 3. Let M be a branching path defined from the set {*library/book/title, /library/book/chapter/title, /library/book/chapter/section/title* }, *i.e.*, M contains these paths and all their prefixes.
An example of a projection of $\mathcal{T}$ over M is the one where $t(\epsilon) = library$, $t(0) = book$, $t(0.0) = title$, $t(0.1) = chapter$, $t(0.1.0) = title$, $t(0.1.1) = section$, and $t(0.1.1.0) = title$. However, if we take $t(0) = book$, $t(0.0) = title$, $t(0.1) = chapter$, $t(0.1.0) = title$, $t(0.2.1) = section$, and $t(0.2.1.0) = title$ we do not have a projection of $\mathcal{T}$ over M. Indeed, in Definition 12, if we consider $P =$*/library/book/chapter/-title* and its instance $inst =$ *ϵ/0/0.1/0.1.0* together with $Q =$ */library/book/-chapter/section/title* and its instance $inst' =$ *ϵ/0/0.2/0.2.1/0.2.1.0*, we obtain $isInst\_lcp(P, inst, Q, inst') = false$. Notice that the longest common paths $P \cap Q$ is */library/book/chapter/*. □

From Definition 12, we remark that the projection of $\mathcal{T}$ over a branching path M contains exactly one instance of every path in M. In the following, when needed, *we denote by $\Pi_M(\mathcal{T})[P]$ the unique instance of the simple path P in $\Pi_M(\mathcal{T})$*. Indeed, when we write $\Pi_M(\mathcal{T})[P]$ we restrict the projection of T over M to the instance (in the projection) of one simple path P.

Lemma 1. *Let $\Pi_M(\mathcal{T})$ be a projection of a tree $\mathcal{T}$ over a branching path M. For each two simple paths P and Q in M if $I = \Pi_M(\mathcal{T})[P]$ and $J = \Pi_M(\mathcal{T})[Q]$ then we have $isInst\_lcp(P, I, Q, J) = true$.* $\square$

Proof: Let P and Q be two simple paths in a branching path M. Let $\Pi_M(\mathcal{T})$ be the projection of M over a tree $\mathcal{T}$ and let I and J be the instances of P and Q in $\Pi_M(\mathcal{T})$. We have the following cases:

1. Case 1: $P \prec Q$. In this case $P \cap Q = P$. Since I and J are the unique path instances of P and Q in $\Pi_M(\mathcal{T})$, we know that $I \cap J = I$. Thus we have that $isInst\_lcp(P, I, Q, J) = true$.
2. Case 2: $P \not\prec Q$ and $Q \not\prec P$. We can distinguish three different situations. (i) P and Q are in $Long_M$. By Definition 12 we know that $isInst\_lcp(P, I, Q, J) = true$. (ii) $P \in Long_M$ and $Q \notin Long_M$. Let $Q' \in Long_M$ such that $Q \prec Q'$. Let $J' = \Pi_M(\mathcal{T})[Q']$. By Definition 12, we know that $isInst\_lcp(P, I, Q', J') = true$ and thus $I \cap J'$ is the instance of $P \cap Q'$ in the projection $\Pi_M(\mathcal{T})$. We also know that $I \cap J' \prec J'$. From item 1 we know that $isInst\_lcp(Q, J, Q', J') = true$ and thus $J \prec J'$. As $J \not\prec I$ (since $Q \not\prec P$) we have that $I \cap J' \prec J$. Since $J \prec J'$ we conclude that $I \cap J = I \cap J'$ and thus $isInst\_lcp(P, I, Q, J) = true$. (iii) $P \notin Long_M$ and $Q \notin Long_M$ is similar to the situation (ii) $\square$

4 Functional Dependencies in XML

Usually, a functional dependency in XML (XFD) is denoted by $X \to Y$ (where X and Y are sets of paths) and it imposes that for each pair of tuples[2] t_1 and t_2 if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$. We assume that an XFD has a single path on the right-hand side and possibly more than one path on the left-hand side - generalizing the proposals in [1, 17, 13, 18]. The dependency can be imposed in a specific part of the document, and, for this reason, we specify a *context path.*

Definition 13 (XML Functional Dependency). *Given an XML tree t, an XML functional dependency (XFD) is an expression of the form*

$$f = (C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\}\ \to Q\ [E]))$$

where C is a path that starts from the root of t (context path) ending at the context node; $\{P_1, \ldots, P_k\}$ is a non-empty set of paths in t and Q is a single path in t, both P_i and Q start at the context node. The set $\{P_1, \ldots, P_k\}$ is the left-hand side (LHS) or determinant of an XFD, and Q is the right-hand side (RHS) or the dependent path. The symbols $E_1, \ldots, E_k, E$ represent the equality type associated to each dependency path. When symbols E or $E_1, \ldots, E_k$ are omitted, value equality is the default choice. $\square$

Notice that in an XFD the set of paths $\{C/P_1, \ldots, C/P_k, C/Q\}$ defines branching paths and that, as in [18], our XFD definition allows the combination of two kinds of equality.

[2] Tuples formed by the values or nodes found at the end of the path instances of X and Y in a document T.

Definition 14 (XFD Satisfaction). *Let $\mathcal{T}$ be an XML document and $f = (C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\} \rightarrow Q\ [E]))$ an XFD. Let M be a branching path defined from f. We say that $\mathcal{T}$ satisfies f (noted by $\mathcal{T} \models f$) if and only if for all $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ that are projections of $\mathcal{T}$ on M and that coincide at least on their prefix C, we have:*

If $\tau^1[C/P_1, \ldots, C/P_k] =_{E_i, i\in[1\ldots k]} \tau^2[C/P_1, \ldots, C/P_k]$ then $\tau^1[C/Q] =_E \tau^2[C/Q]$ where τ^1 (resp. τ^2) is the tuple obtained from $\Pi^1_M(\mathcal{T})$ (resp. $\Pi^2_M(\mathcal{T})$). □

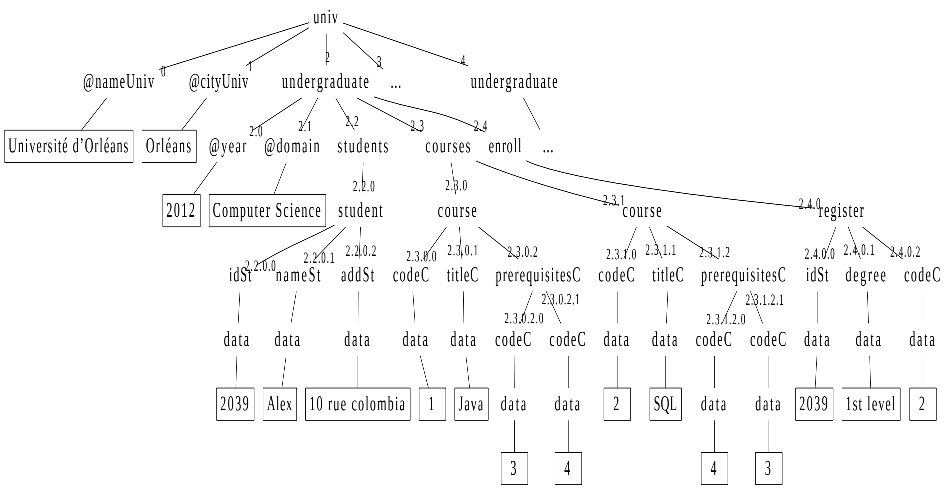

Fig. 6. XML document concerning the first degree (undergraduate) at a university

Example 6. Consider the following XFD on document of Figure 6.

XFD1: $univ//courses/, (\{course/codeC\} \rightarrow course/titleC)$
Considering the set of courses of an undergraduate domain, courses having the same code have the same title.

XFD2: $univ/, (\{undergraduate//course/codeC\} \rightarrow undergraduate//course/titleC)$
Considering the set of all courses in an university, courses having the same code have the same title.

XFD3: $univ/, (\{undergraduate/@year, undergraduate/@domain, undergraduate//register/idSt\} \rightarrow undergraduate//register/degree)$.
At the university, for a given year and a given domain, a student can be enrolled only for courses that correspond to the same degree.

XFD4: $univ//students/, (\{student/idSt\} \rightarrow student[N])$
Considering the set of students of an undergraduate domain, no two students have the same number and each student appears once. □

An XML document $\mathcal{T}$ satisfies a set of XFD $\mathcal{F}$, denoted by $\mathcal{T} \models \mathcal{F}$, if $\mathcal{T} \models f$ for all f in $\mathcal{F}$. Usually it is important to reason whether a given XFD f is also satisfied on $\mathcal{T}$ when $\mathcal{F}$ is satisfied. The following definition introduces this notion.

Definition 15 (XFD Implication). *Given a set $\mathcal{F}$ of XFDs we say that $\mathcal{F}$ implies f, denoted by $\mathcal{F} \models f$, if for every XML tree $\mathcal{T}$ such that $\mathcal{T} \models \mathcal{F}$ then $\mathcal{T} \models f$.* $\square$

Based on the notion of implication we can introduce the definition of closure for a set of XFDs.

Definition 16 (Closure of a set of XFD). *The closure of a set of XFD $\mathcal{F}$, denoted by $\mathcal{F}^+$, is the set containing all the XFDs which are logically implied by $\mathcal{F}$, i.e., $\mathcal{F}^+ = \{f \mid \mathcal{F} \models f\}$.* $\square$

Notation: In the rest of this paper, given an XFD $(C, (X \to A))$ where $X = \{P_1, \ldots, P_n\}$ is a set of paths and A is a path, we use C/X as a shorthand for the set $\{C/P_1, \ldots, C/P_n\}$.

5 Axiom System

To find which XFDs f are also satisfied when a given set of XFDs $\mathcal{F}$ is satisfied we need inference rules that tell how one or more dependencies imply other XFDs. In this section we present our axiom system and prove that it is sound (we cannot deduce from F any false XFD) and complete (from a given set $\mathcal{F}$, the rules allow us to deduce all the true dependencies). Our axiom system is close to the one proposed in [16], but has two important differences: our XFD are defined *w.r.t.* a context (and not always *w.r.t.* the root) and we use two kinds of equality.

Definition 17. - Inference Rules for XFDs: Given a tree $\mathcal{T}$ and XFD defined over paths in $\mathbb{P}$, our axioms are:

A1: **Reflexivity**
$(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_i\,[E_i])), \forall i \in [1 \ldots n]$.

A2: **Augmentation**
If $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$ then
$(C, (\{P_0\,[E_0], P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{P_0\,[E_0], Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$.

A3: **Transitivity**
If $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$ and $(C, (\{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\} \to S\,[E_s]))$ then $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to S\,[E_s]))$.

A4: **Branch Prefixing**
If $(C, (\{P'_1\,[E'_1], \ldots, P'_n\,[E'_n]\} \to P_{n+1}\,[E_{n+1}]))$ and there exist paths $C/P_1, \ldots, C/P_n$ (not necessarily distinct) such that:
(i) $P'_i \cap P_{n+1} \preceq_{PL} P_i$ and
(ii) $P_i \preceq_{PL} P'_i$ or $P_i \preceq_{PL} P_{n+1}$
then $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+1}\,[E_{n+1}]))$.

A5: **Ascendency**
If $Last(P) \in \Sigma_{ele}$ and Q is a prefix for P then $(C, (P\,[N] \to Q\,[N]))$.

A6: **Attribute Uniqueness**
If $Last(P) \in \Sigma_{att}$ then $(C, (Parent(P)\,[E] \to P\,[E]))$.

A7: **Root Uniqueness**
$(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to []\,[E_{n+1}]))$.

A8: **Context Path Extension**
If $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+1}\,[E_{n+1}]))$ and there is a path Q such that $P_1 = Q/P'_1, \ldots, P_{n+1} = Q/P'_{n+1}$
then $(C/Q, (\{P'_1\,[E_1], \ldots, P'_n\,[E_n]\} \to P'_{n+1}\,[E_{n+1}]))$.

Example 7. A given university has one or more undergraduate specialties (first degree) and, for each of them, we store its domain and year together with information concerning students, courses and enrollment. Figure 6 shows part of this XML document over which we illustrate the intuitive meaning of our axioms.

A1 : $univ/undergraduate/, (\{idSt, nameSt, addSt\} \to addSt)$. As usual, the reflexivity axiom concerns trivial functional dependencies.

A2 : If $univ//courses/, (\{course/codeC\} \to course/titleC)$
then $univ//courses/, (\{course/codeC, course/prerequisitesC\} \to \{course/titleC, course/prerequisitesC\})$.
Clearly, if courses having the same code correspond to only one title then courses with the same code and the same set of prerequisites also correspond to one title and set of prerequisites.

A3 : If $univ/undergraduate/, (\{courses//codeC\} \to course[N])$
and $univ/undergraduate/(\{courses/course[N]\} \to titleC[N])$
then $univ/undergraduate/(\{courses//codeC\} \to titleC[N])$.
In the context of an undergraduate specialty, we consider that a course is uniquely defined by *codeC* (*i.e.*, *CodeC* is the key), and thus there is no two courses with the same *codeC*. Moreover, a course has only one title node. In this context, we can derive by axiom A3, that given *codeC*, one can determine the unique *titleC* associated to it.

A4 : If $univ/, (\{undergraduate/domain, courses//codeC\} \to undergraduate/enroll//degree)$
then we can say, for instance, that:
$univ/, (\{undergraduate/domain, undergraduate/courses/course\} \to undergraduate/enroll//degree)$
or $univ/, (\{undergraduate/domain, undergraduate/courses\} \to undergraduate/enroll//degree)$
or $univ/, (undergraduate \to undergraduate/enroll//degree)$
We consider the XFD stating that all courses having *codeC* in the same domain correspond to the same degree. From this XFD, we can deduce, among others XFD, an XFD stating that an undergraduate specialty is associated to only one degree. In other words, an undergraduate specialty prepares to only one degree (*e.g.*, Bachelor's)

A5 : Given a path $P = undergraduate//register/idSt$, by A5 we can derive, for instance, $univ/, (\{undergraduate//register/idSt\} \to undergraduate//register)$.

A6 Given $P = undergraduate/@year$, by A6, we derive that $univ/, (\{undergraduate[N]\} \to @year[N]$.

A8 If *univ/udergraduate,* ({*/students/student/idSt*} $\to$ */students/student/nameSt*) then *univ/udergraduate/students,* ({*/student/idSt*} $\to$ */student/nameSt*). If, in the context of an undergraduate domain, the *idSt* identifies the name of a student; this is also true in the context of *students.*

Notice that $A5$ does not hold when dealing with value equality. The tree on Figure 6 violates the XFD $univ/, (\{undergraduate//course/prerequisitesC\,[V]\} \to undergraduate//course\,[V])$. Indeed $Last(2.3.0.2) =_V Last(2.3.1.2)$ but $Last(2.3.0) \neq_V Last(2.3.1)$. □

The set of axioms in Definition 17 establishes an inference system with which one can derive other XFDs.

Definition 18. - XFD Derivation: Given a set $\mathcal{F}$ of XFDs, we say that an XFD f is derivable from the functional dependencies in $\mathcal{F}$ by the set of inference rules in Definition 17, denoted by $\mathcal{F} \vdash f$, if and only if there is a sequence of XFDs $f_1, f_2, \ldots, f_n$ such that (i) $f = f_n$ and (ii) for all $i = 1, \ldots, n$ the XFD f_i is in $\mathcal{F}$ or it is obtainable from $f_1, f_2, \ldots f_{i-1}$ by means of applying an axiom A1-A8 (from Definition 17). □

5.1 Soundness of the Axiom System

In this section we prove that our axiom system is sound, *i.e.*, our axioms always lead to true conclusions when we deal with complete XML trees. We start by proving some lemmas. The first one deals with properties concerning the longest common prefix of paths. The following example illustrates the situation it concerns.

Example 8. We consider the XML document in Figure 6 and the following paths:
$P_K = univ/undergraduate/@domain$.
$P_J = univ/undergraduate/students/student/idSt$.
$P_I = univ/undergraduate/students/student/nameSt$.

In this situation we have $P_I \cap P_J = univ/undergraduate/students/student$ and $P_J \cap P_K = univ/undergraduate/$. Clearly, $P_J \cap P_K \preceq P_I \cap P_J$. Then, consider path instances where $isInst\_lcp(P_I, I, P_J, J) = true$ and $isInst\_lcp(P_I, I, P_K, K) = true$. For instance, let instance $K = \epsilon/2/2.1$, instance $J = \epsilon/2/2.2/2.2.0/2.2.0.0$ and $I = \epsilon/2/2.2/2.2.0/2.2.0.1$. Notice that in this case we also have $isInst\_lcp(P_J, J, P_K, K) = true$. □

The above example suggests that a kind of transitivity property could be established for the function *isInst_lcp*. The following lemma proves that this is actually possible.

Lemma 2. Let $\mathcal{T}$ be an XML document and $\mathbb{P}$ its associated set of simple paths. Let P_I, P_J, P_K be distinct paths in $\mathbb{P}$. If we have:

1. $P_J \cap P_K \preceq P_I \cap P_J$ and
2. *isInst_lcp*$(P_I, I, P_J, J) = true$ and *isInst_lcp*$(P_I, I, P_K, K) = true$

then *isInst_lcp*$(P_J, J, P_K, K) = true$. □

Proof: In the following, let v_i, v_j and v_k be the nodes corresponding to *Last*(I), *Last*(J) and *Last*(K), respectively. From the conditions stated in the lemma, we can distinguish the three situations illustrated in Figure 7. When $P_J \cap P_K \prec P_I \cap P_J$ we have the situation shown in Figure 7(a). When $P_J \cap P_K = P_I \cap P_J$ we can have the situations in Figure 7(b) or (c).

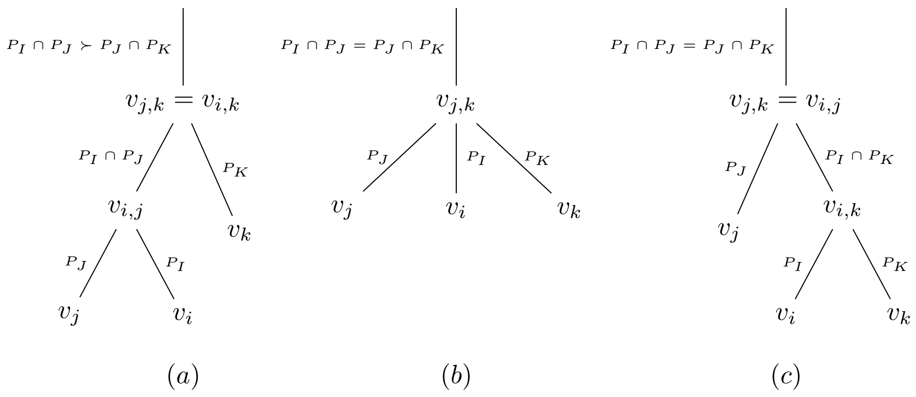

Fig. 7. Three situations obtained from the conditions of Lemma 2.

From condition 2 and Definition 12 we know that $I \cap J$ is the instance of the longest common prefix of P_I and P_J ($P_I \cap P_J$). The same reasoning is applied for $I \cap K$. We denote by $v_{i,j}$ the node *Last*$(I \cap J)$ and by $v_{i,k}$ the node *Last*$(I \cap K)$. Clearly, $v_{i,j}$ is an ancestor for v_i and v_j while $v_{i,k}$ is an ancestor for both v_i and v_k.
Case 1 - Figure 7(a): In this case, we have $P_I \cap P_K = P_J \cap P_K$. From this fact together with condition 1, we can say that node $v_{i,k}$ is an ancestor of $v_{i,j}$. Therefore $v_{i,k}$ is also an ancestor of v_j. Since $P_I \cap P_K = P_J \cap P_K$ we have that *Last*$(J \cap K) =$ *Last*$(I \cap K) = v_{i,k}$. Thus, *isInst_lcp*$(P_J, J, P_K, K) = true$.
Case 2 - Figure 7(b): In this case, we also have $P_I \cap P_K = P_J \cap P_K$ and the proof is similar to the previous one.
Case 3 - Figure 7(c): In this case, we have $P_J \cap P_K \prec P_I \cap P_K$. From condition 1 and the fact that $P_J \cap P_K \prec P_I \cap P_K$, we can say that node $v_{i,j}$ is an ancestor of $v_{i,k}$. Therefore $v_{i,j}$ is also an ancestor of v_k. Since $P_I \cap P_J = P_J \cap P_K$ we have that *Last*$(J \cap K) =$ *Last*$(I \cap J) = v_{i,j}$. Thus, *isInst_lcp*$(P_J, J, P_K, K) = true$. □

The following lemma deals with the extension of a branching path (and its projection) by the addition of a path (and its instance).

Lemma 3. Let M be a branching path over a complete tree $\mathcal{T}$ and let P' be a valid path. Let M' be the branching path obtained by the union $M \cup \{P'\}$. If there is a projection $\Pi_M(\mathcal{T})$ then there exists an instance of P' on $\mathcal{T}$ (denote by $\Pi_{\{P'\}}(\mathcal{T})[P']$) such that $\Pi_{M'}(\mathcal{T}) = \Pi_M(\mathcal{T}) \cup \Pi_{\{P'\}}(\mathcal{T})$. □

Proof: Following Definition 12, we consider $Long_M = \{P_1, \ldots, P_n\}$ and the set of instances $SetPathInst_M = \{\Pi_M(\mathcal{T})[P_1], \ldots, \Pi_M(\mathcal{T})[P_n]\}$. We rename P' as P_{n+1}. Let us consider the set of paths $\{P_1 \cap P_{n+1}, \ldots, P_n \cap P_{n+1}\}$. These paths can be totally ordered *w.r.t.* the prefix relation ,$\preceq$, since each path is a prefix of P_{n+1}. Then, we relabel the subscripts of $P_1, \ldots, P_n$ such that:

$$i < j \Rightarrow (P_i \cap P_{n+1}) \preceq (P_j \cap P_{n+1}) \tag{1}$$

In the lights of (1), let us consider the path P_n. As $\mathcal{T}$ is complete, there exists an instance $\Pi_{\{P_{n+1}\}}(\mathcal{T})[P_{n+1}]$ of the path P_{n+1} such that:

$$isInst\_lcp(P_n, \Pi_M(\mathcal{T})[P_n], P_{n+1}, \Pi_{\{P_{n+1}\}}(\mathcal{T})[P_{n+1}]) = true \tag{2}$$

Let $\Pi_{M'}(\mathcal{T}) = \Pi_M(\mathcal{T}) \cup \Pi_{\{P_{n+1}\}}(\mathcal{T})$. The proof consists in showing that the set of path instances $SetPathInst_{M'} = SetPathInst_M \cup \Pi_{\{P_{n+1}\}}(\mathcal{T})[P_{n+1}]$ also verifies the three conditions in Definition 12. The verification of the first two conditions is obvious. We now consider the verification of the third condition, namely:

$$\forall\, i, j \in [1, \ldots, n+1], isInst\_lcp(P_i, \Pi_{M'}(\mathcal{T})[P_i], P_j, \Pi_{M'}(\mathcal{T})[P_j]) = true \tag{3}$$

By hypothesis, the construction of $SetPathInst_M$ assumes that $\forall\, i, j \in [1, \ldots, n]$, $isInst\_lcp(P_i, \Pi_M(\mathcal{T})[P_i], P_j, \Pi_M(\mathcal{T})[P_j]) = true$. Since $\Pi_{M'}(\mathcal{T})[P_i] = \Pi_M(\mathcal{T})[P_i]$ (for $i \in [1, \ldots, n]$), and by considering (2), the condition (3) is reduced to:

$$\forall\, i \in [1, \ldots, n-1], isInst\_lcp(P_i, \Pi_M(\mathcal{T})[P_i], P_{n+1}, \Pi_{M'}(\mathcal{T})[P_{n+1}]) = true \tag{4}$$

Finally, by using the Lemma 2 with $P_I = P_n$, $P_J = P_{n+1}$, $P_K = P_i$, and the fact that $(P_i \cap P_{n+1}) \preceq (P_n \cap P_{n+1})$ (from (1)), we obtain (4) which ends the proof of the lemma. □

To illustrate Lemma 3, consider the following example.

Example 9. We consider the XML document of Figure 6. Let $M = \{$*univ/undergraduate/courses/course/codeC* , *univ/undergraduate/courses/course/titleC*$\}$. We consider the projection of M which contains the path instances $\epsilon/2/2.3/2.3.0/2.3.0.0$ and $\epsilon/2/2.3/2.3.0/2.3.0.1$. Let $P' =$ *univ/undergradute/@domain* and $M' = M \cup P$. Instance $(\epsilon/2/2.1)$ of P' is the one for which we have $\Pi_{M'}(\mathcal{T}) = \Pi_M(\mathcal{T}) \cup \Pi_{\{P'\}}(\mathcal{T})$. □

The next example illustrates a special situation where an XFD not satisfied by a given document has at the left-hand side a path which is a prefix of the path on the right-hand side.

Example 10. We consider the example in Figure 6, the XFD

$f = (univ/undergradute/, (\{courses//titleC, courses//prerqtC\}$
$\rightarrow courses//prerqtC/CodeC))$

and the branching path M definded by f. The document of Figure 6 does not satisfy f. Notice that $P_2 = univ/undergradute/courses/course/prerqtC$ in the left-hand side of f is a prefix for $P = univ/undergradute/courses/course/prerqtC/CodeC$ in right-hand side of f. Also remark that we can find two projections of the XML tree over M such that $Last(\Pi^1_M(\mathcal{T})[C/P_2]) =_N Last(\Pi^2_M(\mathcal{T})[C/P_2])$: the two projections ending on node 2.3.0.2 □

The following lemma proves that in situations as the one illustrated by Example 10 we can always find two projections of the XML tree over the branching path M such that $Last(\Pi^1_M(\mathcal{T})[C/P_j]) =_N Last(\Pi^2_M(\mathcal{T})[C/P_j])$, where P_j is the path on the left-hand side which is a prefix of the one on the right-hand side.

Lemma 4. Let $\mathcal{T}$ be an XML document, $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow P_{n+1}\,[E_{n+1}]))$ an XFD and let M be the branching path $\{C/P_1, \ldots, C/P_{n+1}\}$. If $\mathcal{T} \not\models f$ and there exists a $j \in [1 \ldots n]$ such that $P_j \preceq P_{n+1}$ then we can find two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for M in $\mathcal{T}$ such that $Last(\Pi^1_M(\mathcal{T})[C/P_j]) =_N Last(\Pi^2_M(\mathcal{T})[C/P_j])$. □

Proof: The proof is by contradiction. Suppose that for any two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path M in $\mathcal{T}$ we have $Last(\Pi^1_M(\mathcal{T})[C/P_j]) \neq_N Last(\Pi^2_M(\mathcal{T})[C/P_j])$. Since $\mathcal{T} \not\models f$, then from Definition 14, we can deduce that there exists two projections (let $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ be these two instances) for the branching path M in $\mathcal{T}$ such that $\tau^1[C/P_1, \ldots, C/P_n] =_{E_i, i\in[1\ldots n]} \tau^2[C/P_1, \ldots, C/P_n]$ and $\tau^1[C/P_{n+1}] \neq_{E_{n+1}} \tau^2[C/P_{n+1}]$.

– If $E_j = N$ then $Last(\Pi^1_M(\mathcal{T})[C/P_j]) =_N Last(\Pi^2_M(\mathcal{T})[C/P_j])$ which contradicts the assumption $Last(\Pi^1_M(\mathcal{T})[C/P_j]) \neq_N Last(\Pi^2_M(\mathcal{T})[C/P_j])$.
– Otherwise, if $E_j = V$ then $Last(\Pi^1_M(\mathcal{T})[C/P_j]) =_V Last(\Pi^2_M(\mathcal{T})[C/P_j])$. First suppose that the instances $\Pi^1_M(\mathcal{T})[C/P_j]$, $\Pi^2_M(\mathcal{T})[C/P_j]$ are prefix for two instances of path P_{n+1}. Let $\Pi^1_M(\mathcal{T})[C/P_j]$ be the prefix for I and J and let $\Pi^2_M(\mathcal{T})[C/P_j]$ be the prefix for I' and J'.
  1. If $Last(I) \neq_V Last(J)$ then consider that:
     • $\Pi^1_M(\mathcal{T})[C/P_{n+1}] = I$;
     • there is a projection $\Pi^3_M(\mathcal{T})$ that coincides with $\Pi^1_M(\mathcal{T})$ except for the instance of path P_{n+1}, since we make $\Pi^3_M(\mathcal{T})[C/P_{n+1}] = J$.
     In this case, $Last(\Pi^1_M(\mathcal{T})[C/P_j]) =_N Last(\Pi^3_M(\mathcal{T})[C/P_j])$ and, thus, we have a contradiction *w.r.t.*the intial assumption which says that for any projection of M on $\mathcal{T}$, the last nodes of P_j's instance are not node equal. The same argument is used when $Last(I') \neq_V Last(J')$.
  2. Else, we are in the situation where $Last(I) =_V Last(J)$ and $Last(I') =_V Last(J')$. Since $Last(\Pi^1_M(\mathcal{T})[C/P_{n+1}]) \neq_V Last(\Pi^2_M(\mathcal{T})[C/P_{n+1}])$, this implies that also $Last(\Pi^1_M(\mathcal{T})[C/P_j]) \neq_V Last(\Pi^2_M(\mathcal{T})[C/P_j])$. We have a contradiction with our premise which says that $Last(\Pi^1_M(\mathcal{T})[C/P_j]) =_V Last(\Pi^2_M(\mathcal{T})[C/P_j])$.

Now suppose that each instance $\Pi^1_M(\mathcal{T})[C/P_j]$ and $\Pi^2_M(\mathcal{T})[C/P_j]$ is a prefix for only one P_{n+1}'s instance. Since $Last(\Pi^1_M(\mathcal{T})[C/P_{n+1}]) \neq_V Last(\Pi^2_M(\mathcal{T})[C/P_{n+1}])$, this implies that also $Last(\Pi^1_M(\mathcal{T})[C/P_j]) \neq_V Last(\Pi^2_M(\mathcal{T})[C/P_j])$. We have again a contradiction with our premise that $Last(\Pi^1_M(\mathcal{T})[C/P_j]) =_V Last(\Pi^2_M(\mathcal{T})[C/P_j])$. $\square$

We can now prove the soundness of our axiom system on complete XML trees.

Theorem 1. *Axioms A1-A8 are sound for XFD on complete XML trees.* $\square$

Proof: We consider a complete XML document $\mathcal{T} = (t, type, value)$.

A1: Let $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_i\,[E_i]))$. The proof is by contradiction. Assume that $\mathcal{T} \not\models f$. From Definition 14, we can deduce that there exists two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path $M = \{C/P_1, \ldots, C/P_n\}$ in $\mathcal{T}$ such that $\tau^1[C/P_1, \ldots, C/P_n] =_{E_i, i\in[1\ldots n]} \tau^2[C/P_1, \ldots, C/P_n]$ and for a $j \in [1\ldots n]$, $\tau^1[C/P_j] \neq_{E_j} \tau^2[C/P_j]$. To satisfy the left-hand side equality we know that $\tau^1[C/P_j] =_{E_j} \tau^2[C/P_j]$. But this is a contradiction with the fact that $\tau^1[C/P_j] \neq_{E_j} \tau^2[C/P_j]$.

A2: Let $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$ and $f' = (C, (\{P_0\,[E_0], P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{P_0\,[E_0], Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$. The proof is by contrapositive. We show that if $\mathcal{T} \not\models f'$ then $\mathcal{T} \not\models f$. Let us define $Q_0 = P_0$ and $E'_0 = E_0$. From Definition 14, we can deduce that there exist Q_j with $j \in [0, \ldots, m]$ such that $\tau^1[C/Q_j] \neq_{E'_i} \tau^2[C/Q_j]$. Suppose first that $Q_j = Q_0$. Then by using the same arguments as in $A1$, we obtain a contradiction since from Definition 14, $\tau^1[C/Q_0] =_{E'_0} \tau^2[C/Q_0]$ but $\tau^1[C/Q_0] \neq_{E'_0} \tau^2[C/Q_0]$. Thus suppose that $j > 0$. As $\mathcal{T} \not\models f'$, from Definition 14 we have that $\forall\, i \in [1, \ldots, n]$ $\tau^1[C/P_i] =_{E_i} \tau^2[C/P_i]$ and that there exist a $j > 0$, such that $\tau^1[C/Q_j] \neq_{E'_j} \tau^2[C/Q_j]$. Therefore $\mathcal{T} \not\models f$ as claimed.

A3: Let $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$, $f' = (C, (\{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\} \to S\,[E_s]))$ and $f'' = (C, (P_1\,[E_1], \ldots, P_n\,[E_n]\} \to S\,[E_s]))$. The argument is by contrapositive: we show that if $\mathcal{T} \not\models f''$ then either $\mathcal{T} \not\models f'$ or $\mathcal{T} \not\models f$. Let us define $P_{n+1} = S$ and $E_{n+1} = E_s$. Since $\mathcal{T} \not\models f''$ then by using Definition 14, there exist two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path $M = \{C/P_1, \ldots, C/P_{n+1}\}$ in $\mathcal{T}$ such that $\tau^1[C/P_1, \ldots, C/P_n] =_{E_i, i\in[1\ldots n]} \tau^2[C/P_1, \ldots, C/P_n]$ and $\tau^1[C/P_{n+1}] \neq_{E_{n+1}} \tau^2[C/P_{n+1}]$. We extend the branching path $M = \{C/P_1, \ldots, C/P_{n+1}\}$ to obtain $M' = \{C/P_1, \ldots, C/P_{n+1}, C/Q_1, \ldots, C/Q_m\}$. From several applications of Lemma 3 we know that we can build instances $\Pi^1_{M'}(\mathcal{T})$ from $\Pi^1_M(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ from $\Pi^2_M(\mathcal{T})$. In instances $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ the two following situations are possible (where u^1 and u^2 are tuples from $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ respectively):

1. $u^1[C/Q_1, \ldots, C/Q_m] =_{E'_i, i\in[1\ldots m]} u^2[C/Q_1, \ldots, C/Q_m]$ and in this case we have $\mathcal{T} \not\models f'$ because $u^1[C/P_{n+1}] \neq_{E_{n+1}} u^2[C/P_{n+1}]$,

2. or there is $j \in [1 \ldots m]$ such that $u^1[C/Q_j] \neq_{E'_j} u^2[C/Q_j]$. In this case we have $\mathcal{T} \not\models f$.

We can conclude that $A3$ is correct.

A4: Let $f = (C, (\{P'_1\,[E'_1], \ldots, P'_n\,[E'_n]\} \rightarrow P_{n+1}\,[E_{n+1}]))$ and $f' = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow P_{n+1}\,[E_{n+1}]))$. The proof is by contradiction. Suppose that $\mathcal{T} \models f$ but $\mathcal{T} \not\models f'$. From Axiom $A1$, we can assume that for all $i \in [1 \ldots n]$, $P_i \neq P_{n+1}$. From Definition 14, we can deduce that there exists two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path $M = \{C/P_1, \ldots, C/P_{n+1}\}$ in $\mathcal{T}$ such that $\tau^1[C/P_1, \ldots, C/P_n] =_{E_i, i \in [1 \ldots n]} \tau^2[C/P_1, \ldots, C/P_n]$ and $\tau^1[C/P_{n+1}] \neq_{E_{n+1}} \tau^2[C/P_{n+1}]$. We now show that there exist two projections $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$, construct from $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$, for the branching path $M' = \{C/P'_1, \ldots, C/P'_n, C/P_{n+1}\}$ in $\mathcal{T}$ such that:

$$u^1[C/P'_1, \ldots, C/P'_n] =_{E'_i, i \in [1 \ldots n]} u^2[C/P'_1, \ldots, C/P'_n] \text{ and} \tag{5}$$

$$u^1[C/P_{n+1}] \neq_{E_{n+1}} u^2[C/P_{n+1}]. \tag{6}$$

However from our hypothesis we know that for all two projections $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ such that (5) is satisfied then we have $u^1[C/P_{n+1}] =_{E_{n+1}} u^2[C/P_{n+1}]$. If $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ really exist, we have a contradiction with (6) and the axiom $A4$ will be satisfied.

We detail the proof by showing that it is possible to obtain two projections for M' satisfying (5) and (6). We start by considering that $\Pi^1_{M'}(\mathcal{T})[C/P_i] = \Pi^1_M(\mathcal{T})[C/P_i]$ and $\Pi^2_{M'}(\mathcal{T})[C/P_i]) = \Pi^2_M(\mathcal{T})[C/P_i]\ \forall\, i \in [1 \ldots n+1]$.

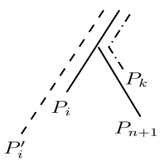

Fig. 8. Graphical representation of paths to be projected on a tree. Here, path $P_k \prec P_{n+1}$ where P_k is one of the paths in the left-handside of XFD f' and $P_i \preceq P'_i$.

1. If $\exists\, k \in [1 \ldots n]$ such that $P_k \preceq P_{n+1}$ (Figure 8) then, from Lemma 4, we can consider that :

$$Last(\Pi^1_{M'}(\mathcal{T})[C/P_k]) =_N Last(\Pi^2_{M'}(\mathcal{T})[C/P_k]). \tag{7}$$

By considering $\Pi^1_{M'}(\mathcal{T})$ we know that $\forall\, i \in [1 \ldots n]$,

$$\begin{aligned} &\mathit{isInst\_lcp}(C/P_i, \Pi^1_{M'}(\mathcal{T})[C/P_i], C/P_{n+1}, \Pi^1_{M'}(\mathcal{T})[C/P_{n+1}]) = \mathit{true} \\ \text{and } &\mathit{isInst\_lcp}(C/P_i, \Pi^1_{M'}(\mathcal{T})[C/P_i], C/P_k, \Pi^1_{M'}(\mathcal{T})[C/P_k]) = \mathit{true}. \end{aligned} \tag{8}$$

We have $\textit{isInst\_lcp}(C/P_i, \Pi^1_{M'}(\mathcal{T})[C/P_i], C/P_k, \Pi^2_{M'}(\mathcal{T})[C/P_k]) = \mathit{true}$ from (7) and (8). Then, by using Lemma 2 with $I = \Pi^2_{M'}(\mathcal{T})[C/P_k]$, $J = \Pi^1_{M'}(\mathcal{T})[C/P_i]$, $K = \Pi^2_{M'}(\mathcal{T})[C/P_{n+1}]$, and $(P_i \cap P_{n+1}) \preceq (P_k \cap P_{n+1})$ we obtain $\forall\, i \in [1 \ldots n]$,

$$\textit{isInst\_lcp}(C/P_i, \Pi^1_{M'}(\mathcal{T})[C/P_i], C/P_{n+1}, \Pi^2_{M'}(\mathcal{T})[C/P_{n+1}]) = \mathit{true}. \quad (9)$$

Since t is complete there exist instances J_i such that $\forall\, i \in [1 \ldots n]$, $\Pi^1_{M'}(\mathcal{T})[C/P_i] \preceq J_i$ and $J_i \in \mathit{Instances}(C/P'_i, t)$. Let $\forall\, i \in [1 \ldots n]$, $\Pi^1_{M'}(\mathcal{T})[C/P'_i] = \Pi^2_{M'}(\mathcal{T})[C/P'_i] = J_i$. Then by using Lemma 2, (8), (9) we obtain $\forall\, i, j \in [1 \ldots n+1]$ (recall that we consider that $P'_{n+1}{=}P_{n+1}$):

$$\textit{isInst\_lcp}(C/P'_i, \Pi^1_{M'}(\mathcal{T})[C/P'_i], C/P'_j, \Pi^1_{M'}(\mathcal{T})[C/P'_j]) = \mathit{true} \quad (10)$$

$$\text{and } \textit{isInst\_lcp}(C/P'_i, \Pi^2_{M'}(\mathcal{T})[C/P'_i], C/P'_j, \Pi^2_{M'}(\mathcal{T})[C/P'_j]) = \mathit{true}. \quad (11)$$

Thus, in this case, it is possible to have projections $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ satisfying 5 and 6.

2. Otherwise if $\forall\, i \in [1 \ldots n]$, $P_i \not\preceq P_{n+1}$ and $P_i \preceq P'_i$ we can have the following situations:
   (a) If we consider node equality, we have $\mathit{Last}(\Pi^1_{M'}(\mathcal{T})[C/P_i]) =_N \mathit{Last}(\Pi^2_{M'}(\mathcal{T})[C/P_i])$ (Figure 9(a)). Since t is complete there exist an instance J_i such that $\Pi^1_{M'}(\mathcal{T})[C/P_i] \preceq J_i$ and $J_i \in \mathit{Instances}(C/P'_i, t)$. Let $\Pi^1_{M'}(\mathcal{T})[C/P'_i] = \Pi^2_{M'}(\mathcal{T})[C/P'_i] = J_i$.
   (b) If we consider value equality, we have $\mathit{Last}(\Pi^1_{M'}(\mathcal{T})[C/P_i]) =_V \mathit{Last}(\Pi^2_{M'}(\mathcal{T})[C/P_i])$. Since t is complete there exist instances J^1_i, J^2_i such that $\Pi^1_{M'}(\mathcal{T})[C/P_i] \preceq J^1_i$, $\Pi^2_{M'}(\mathcal{T})[C/P_i] \preceq J^2_i$ and $J^1_i, J^2_i \in \mathit{Instances}(C/P'_i, t)$.
      - If $\mathit{Last}(J^1_i) =_V \mathit{Last}(J^2_i)$ then let $\Pi^1_{M'}(\mathcal{T})[C/P'_i] = J^1_i$ and $\Pi^2_{M'}(\mathcal{T})[C/P'_i] = J^2_i$ (Figure 9(b)).
      - Otherwise if $\mathit{Last}(J^1_i) \neq_V \mathit{Last}(J^2_i)$ then, since $P_i \preceq P'_i$ and $\mathit{Last}(\Pi^1_{M'}(\mathcal{T})[C/P_i]) =_V \mathit{Last}(\Pi^2_{M'}(\mathcal{T})[C/P_i])$, there exists two instances J^3_i, J^4_i such that $\Pi^1_{M'}(\mathcal{T})[C/P_i] \preceq J^3_i$, $\Pi^2_{M'}(\mathcal{T})[C/P_i] \preceq J^4_i$ and $J^3_i, J^4_i \in \mathit{Instances}(C/P'_i, t)$, $\mathit{Last}(J^1_i) =_V \mathit{Last}(J^4_i)$ and $\mathit{Last}(J^2_i) =_V \mathit{Last}(J^3_i)$. In this case, let $\Pi^1_{M'}(\mathcal{T})[C/P'_i] = J^1_i$ and $\Pi^2_{M'}(\mathcal{T})[C/P'_i] = J^4_i$ (Figure 9(c)).

   Now, we know that:

$$\textit{isInst\_lcp}(C/P_i, \Pi^1_{M'}(\mathcal{T})[C/P_i], C/P'_i, \Pi^1_{M'}(\mathcal{T})[C/P'_i]) = \mathit{true} \text{ and}$$

$$\textit{isInst\_lcp}(C/P_i, \Pi^1_{M'}(\mathcal{T})[C/P_i], C/P_{n+1}, \Pi^1_{M'}(\mathcal{T})[C/P_{n+1}]) = \mathit{true}.$$

   By using Lemma 2 with $I = \Pi^1_{M'}(\mathcal{T})[C/P_i]$, $J = \Pi^1_{M'}(\mathcal{T})[C/P'_i]$, $K = \Pi^1_{M'}(\mathcal{T})[C/P_{n+1}]$, and $(P'_i \cap P_{n+1}) \preceq (P_i \cap P_{n+1})$, we obtain (10). Proceeding in a similar way with projection $\Pi^2_{M'}(\mathcal{T})$ we obtain (11).

Since $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ exist and conditions (5), (6) are satisfied, we can conclude that $A4$ is sound.

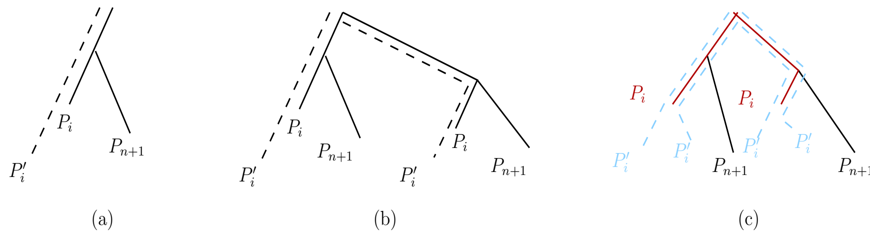

Fig. 9. Graphical representation of paths and possible projections. Case (a): Node equality for last nodes in P_i. Both projections $\Pi^1_{M'}(\mathcal{T})$ and $\Pi^2_{M'}(\mathcal{T})$ have the same instance for path P'_i. Case (b) and (c): Value equality for last nodes in P_i. In case (b) there is only one instance of P'_i ($P_i \preceq P'_i$). In case (c) there are two instances of P'_i ($P_i \preceq P'_i$).

A5: Let $f = (C, (P\,[N] \to Q\,[N]))$. The proof is by contradiction. Suppose that $\mathcal{T} \not\models f$. From Definition 14 there exist two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path $M = \{C/P, C/Q\}$ in $\mathcal{T}$ such that $\tau^1[C/P] =_N \tau^2[C/P]$ and $\tau^1[C/Q] \neq_N \tau^2[C/Q]$. However, since $Last(P) \in \Sigma_{ele}$ and $\tau^1[C/P] =_N \tau^2[C/P]$ we have, due to the node equality, that the instance of the path C/P is the same in $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$. Then, since C/Q is a prefix of C/P, the path C/Q has the same instance in $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$. Therefore $\tau^1[C/Q] =_N \tau^2[C/Q]$ and we have a contradiction with our previous assumption that $\tau^1[C/Q] \neq_N \tau^2[C/Q]$.

A6: Let $f = (C, (Parent(P)[E] \to P[E]))$. The proof is again by contradiction. Suppose that $\mathcal{T} \not\models f$. From Definition 14 there exist two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path $M = \{C/P, C/Parent(P)\}$ in $\mathcal{T}$ such that $\tau^1[C/Parent(P)] =_E \tau^2[C/Parent(P)]$ and $\tau^1[C/P] \neq_E \tau^2[C/P]$. However, since $Parent(P) \prec P$ and $\mathcal{T} \not\models f$, then by applying Lemma 4 we obtain that the instance of the path $C/Parent(P)$ is the same in $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$. From the definition of an XML tree, we know that the instance $\Pi^1_M(\mathcal{T})[Parent(P)] = \Pi^2_M(\mathcal{T})[Parent(P)]$ has only one attribute child for the label $Last(P)$ and so the instance of the path C/P must be the same in $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$. Thus $\tau^1[C/P] =_E \tau^2[C/P]$, which is a contradiction with the initial assumption that $\tau^1[C/P] \neq_E \tau^2[C/P]$.

A7: Since the context node is unique, this axiom is automatically satisfied.

A8: Let $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+1}\,[E_{n+1}]))$ and $f' = (C/Q, (\{P'_1\,[E_1], \ldots, P'_n\,[E_n]\} \to P'_{n+1}\,[E_{n+1}]))$. The proof is by contrapositive. We show that if $\mathcal{T} \not\models f'$ then $\mathcal{T} \not\models f$. Assume that $\mathcal{T} \not\models f'$. From Definition 14, we can deduce that there exists two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path $M = \{C/Q/P'_1, \ldots, C/Q/P'_{n+1}\}$ in $\mathcal{T}$ such that $\tau^1[C/Q/P'_1, \ldots, C/Q/P'_n] =_{E_i, i \in [1 \ldots n]} \tau^2[C/Q/P'_1, \ldots, C/Q/P'_n]$ and $\tau^1[C/Q/P'_{n+1}] \neq_{E_{n+1}} \tau^2[C/Q/P'_{n+1}]$. Since $P_1 = Q/P'_1, \ldots, P_{n+1} = Q/P'_{n+1}$ then $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ are also projections for the branching path $M = \{C/P_1, \ldots, C/P_{n+1}\}$. By considering the projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$, we have $\tau^1[C/P_{n+1}] \neq_{E_{n+1}} \tau^2[C/P_{n+1}]$. Thus we obtain $\mathcal{T} \not\models f$.

□

5.2 Additional Inferece Rules for the Axiom System

From the inference rules in the axiom system introduced in Definition 17, we can derive other rules which will be useful and will greatly simplify the proof of the completeness.

Definition 19. - Additional Inference Rules: Given a tree $\mathcal{T}$ and XFD defined over paths in $\mathbb{P}$, our additional axioms are:

A9: **Union**
If $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow Q\,[E_{n+1}]))$ and $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow R\,[E_{n+2}]))$ then $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow \{Q\,[E_{n+1}], R\,[E_{n+2}]\}))$.

A10: **Decomposition**
If $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$ and the set of paths $\{R_1, \ldots, R_k\} \subseteq \{Q_1, \ldots, Q_m\}$ then $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow \{R_1\,[E''_1], \ldots, R_k\,[E''_k]\}))$.

A11: **Pseudotransitivity**
If $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$ and $(C, (\{Q_1\,[E'_1], \ldots, Q_m\,[E'_m], R_1\,[E''_1], \ldots, R_k\,[E''_k]\} \rightarrow S\,[E_s]))$ then $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n], R_1\,[E''_1], \ldots, R_k\,[E''_k]\} \rightarrow S\,[E_s]))$.

A12: **Subtree Uniqueness**
If $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow P_{n+1}\,[E_{n+1}]))$ and for all $i \in [1 \ldots n]$ the longest common prefix of C/P_i and C/P_{n+1} is the context path C (*i.e.*, $P_i \cap P_{n+1} = \{[]\}$) then $(C, (P_0\,[E_0] \rightarrow P_{n+1}\,[E_{n+1}]))$ for any path C/P_0.

The intuition of axioms A9, A10 and A11 is the usual one. Let us consider an example to illustrate A12. Assume we have $univ/undergraduate, (\{students//idSt\} \rightarrow domain)$ then $univ/undergraduate(\{courses//codeC\} \rightarrow domain)$, *i.e.*, in the context of one undergraduate speciality, a student is associated to only one domain. Then, by $A12$, we can deduce, for instance, that courses having the same $codeC$ belong to the same domain.

Next we prove the soundness of these additional inference rules.

Theorem 2. *Axioms A9-A12 are sound for XFDs on complete XML trees.* □

Proof: We consider a complete tree $\mathcal{T}$.

A9: Let $f_1 = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow Q\,[E_{n+1}]))$. We can augment f_1 with $\{C/P_1, \ldots, C/P_n\}$ by using Axiom $A2$ to derive $f'_1 = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow \{P_1\,[E_1], \ldots, P_n\,[E_n], Q\,[E_{n+1}]\}))$.
Let $f_2 = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow R\,[E_{n+2}]))$. We can augment f_2 with C/Q by using Axiom $A2$ to derive $f'_2 = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n], Q\,[E_{n+1}]\} \rightarrow \{Q\,[E_{n+1}], R\,[E_{n+2}]\}))$.
Now from f'_1, f'_2 and by using Axiom $A3$ we derive $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow \{Q\,[E_{n+1}], R\,[E_{n+2}]\}))$.

A10: Let $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$. Since $\{R_1, \ldots, R_k\} \subseteq \{Q_1, \ldots, Q_m\}$, then by using Axiom $A1$ and $A9$ we derive $f' = (C, (\{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\} \to \{R_1\,[E''_1], \ldots, R_k\,[E''_k]\}))$.
Now from f, f' and by using Axiom $A3$ we derive $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{R_1\,[E''_1], \ldots, R_k\,[E''_k]\}))$.

A11: Let $f_1 = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m]\}))$. We can augment f_1 with $\{C/R_1, \ldots, C/R_k\}$ by using Axiom $A2$ to derive $f'_1 = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n], R_1\,[E''_1], \ldots, R_k\,[E''_k]\} \to \{Q_1\,[E'_1], \ldots, Q_m\,[E'_m], R_1\,[E''_1], \ldots, R_k\,[E''_k]\}))$.
Let $f_2 = (C, (\{Q_1\,[E'_1], \ldots, Q_m\,[E'_m], R_1\,[E''_1], \ldots, R_k\,[E''_k]\} \to S\,[E_s]))$. From f'_1, f_2 and by using Axiom $A3$ we derive $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n], R_1\,[E''_1], \ldots, R_k\,[E''_k]\} \to S\,[E_s]))$.

A12: Let $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+1}\,[E_{n+1}]))$. With Axiom $A7$ we have $\forall\ C/P_0$, $f' = (C, (P_0\ [E_0] \to []\ [E]))$. $\forall i \in [1 \ldots n]$ let $P'_i = []$, we have $P_i \cap P_{n+1} = [] \preceq_{PL} P'_i \preceq_{PL} P_i$ and by using Axiom $A4$ on the XFD f we obtain $f'' = (C, ([]\ [E] \to P_{n+1}\ [E_{n+1}]))$. Finally by using Axiom $A3$ on f' and f'', we obtain $(C, (P_0\ [E_0] \to P_{n+1}\ [E_{n+1}]))$.

□

5.3 Completeness of the Axiom System

Before tackling the completeness issue, it is important to define the closure of a set of paths *w.r.t.* a set of XFDs.

Definition 20. - Closure of a set of Paths: Let X be a set of paths and let C be a path defining a context. The closure of (C, X) with respect to $\mathcal{F}$, denoted by $(C, X)^+_{\mathcal{F}}$, is the set of path $\{P_1, \ldots, P_n\}$ such that $(C, (X \to \{P_1, \ldots, P_n\}))$ can be deduced from $\mathcal{F}$ by the axiom system in Definition 17. In other words, $(C, X)^+_{\mathcal{F}} = \{C/P \mid \mathcal{F} \vdash (C, (X \to P))\}$. When there is no ambiguity about the set $\mathcal{F}$ being used, we just note $(C, X)^+$. □

As in the relational model, the central fact about the closure of a set of paths is that it enables us to tell on a glance whether an XFD follows from a set $\mathcal{F}$ by the axiom system. The next lemma tells us how.

Lemma 5. Let $X = \{C/P_1, \ldots, C/P_n\}$ and $Y = \{C/P_{n+1}, \ldots, C/P_{n+m}\}$ be two sets of paths. We have $\mathcal{F} \vdash (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{P_{n+1}\,[E_{n+1}], \ldots, P_{n+m}\,[E_{n+m}]\}))$ iff $Y \subseteq X^+$. □

Proof:

1. Let $Y \subseteq X^+$. By the definition of X^+ we know that $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+1}\,[E_{n+1}])), \ldots, (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+m}\,[E_{n+m}]))$. By applying the union rule $A9$ we have $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{P_{n+1}\,[E_{n+1}], \ldots, P_{n+m}\,[E_{n+m}\}))$.

2. Let $\mathcal{F} \vdash (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to \{P_{n+1}\,[E_{n+1}], \ldots, P_{n+m}\,[E_{n+m}\}))$. By the decomposition rule $A10$ we know that $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+1}\,[E_{n+1}])), \ldots, (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P_{n+m}\,[E_{n+m}]))$. Thus each path $C/P_{n+1} \ldots C/P_{n+m}$ is in X^+ and we have $Y \subseteq X^+$. □

To prove the completeness of our axiom system, we would like to define a special tree having two instances (except for the root node) for every path $P \in \mathbb{P}$. However, the following examples show that depending on the conditions imposed on paths, it is not possible to have *two* instances for *every* path $P \in \mathbb{P}$.

Example 11. We want to build a complete tree having exactly two instances for each path in $\mathbb{P}$. Let us consider value equality and two paths P and Q such that $P \prec Q$. We denote by I_{P_1} and I_{P_2} the two instances of P on a tree t. We denote by I_{Q_1} and I_{Q_2} the two instances of Q on a tree t. Suppose that $Last(I_{P_1}) =_V Last(I_{P_2})$ and $Last(I_{Q_1}) \neq_V Last(I_{Q_2})$. Based on this situation, the functional dependency $P \to Q$ is not satisfied by this tree. Then, we can apply Lemma 4, to conclude that there is an instance of P which is a prefix of both (distinct) instances of Q. As we want just two instances for each path, to have two instances of Q we should have $Last(I_{P_1}) =_N Last(I_{P_2})$. In other words, in this situation, we cannot have a tree with two instances for P. Indeed, Figure 10 illustrates that, a tree having two instances of P and respecting the constraints $Last(I_{P_1}) =_V Last(I_{P_2})$ and $Last(I_{Q_1}) \neq_V Last(I_{Q_2})$ must have four instances of Q.
Now let us consider that a node equality condition is imposed on the instances of a path P. In this situation we have $Last(I_{P_1}) =_N Last(I_{P_2})$. Clearly, in this case, P has only one instance. □

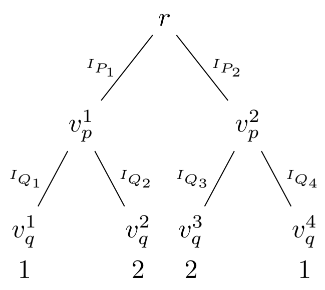

Fig. 10. An XML tree with two instances value equal for the path P and four instances for the path Q with I_{P_1} prefix of I_{Q_1}, I_{Q_2} and I_{P_2} prefix of I_{Q_3}, I_{Q_4}.

Based on Example 11, we introduce the definition of our special tree, having *at most* two instances for each path in $\mathbb{P}$.

Definition 21. - Two-instance Tree: Let $\mathcal{F}$ be a set of XFDs. Let $\mathcal{T} = (t, type, value)$ be an XML document where the tree t, built according to the construction properties below, is called *two-instance tree*. Let $\mathbb{P}$ be the set of paths associated to $\mathcal{T}$ and let $X \subseteq \mathbb{P}$. We denote by $|Instances(P,t)|$ the number of instances of a path P in t.

CONSTRUCTION PROPERTIES:

1. For each $P \in \mathbb{P}$, $|Instances(P,t)|$ is at most 2 (I_1 or I_2) and:
   (a) when $|Instances(P,t)| = 2$ we have $P \in X^+$ iff $Last(I_1) =_V Last(I_2)$;
   (b) when $|Instances(P,t)| = 2$ we have $P \in \mathbb{P} \backslash X^+$ iff $Last(I_1) \neq_V Last(I_2)$
   (c) if $|Instances(P,t)| = 1$ then $Last(P) \in \Sigma_{ele}$ or $Last(P) \in \Sigma_{att}$
2. For each $P \in \mathbb{P}$, $|Instances(P,t)| = 2$ except when:
   (a) $Last(P)$ is the root of t, or
   (b) by considering value equality, $|Instances(P,t)| = 2$ provokes the violation of condition 1 for another path $Q \in \mathbb{P}$ with $P \prec Q$, or
   (c) $X[E] \to P[N]$ or
   (d) $Last(P) \in \Sigma_{att}$ and $Parent(P)$ verifies condition 2b or condition 2c. □

Lemma 6. Let $\mathcal{F}$ be a set of XFD. Let $\mathcal{T} = (t, type, value)$ be an XML document where t is a two-instance tree. Let $\mathbb{P}$ be the set of paths associated to $\mathcal{T}$ and let $X \subseteq \mathbb{P}$. The following properties hold for t:

1. If $P \in \mathbb{P}$ and $|Instances(P,t)| = 1$ then $P \in X^+$.
2. If $P, Q \in \mathbb{P}$, $P \preceq Q$ and there is an instance $I_P \in Instances(P,t)$, and instances I_{Q_1} and $I_{Q_2} \in Instances(Q,t)$ such that $I_P \preceq I_{Q_1}$ and $I_P \preceq I_{Q_2}$ then $|Instances(P,t)| = 1$.
3. If $P \in \mathbb{P}$ then $P \in X^+$ iff $\mathcal{T} \models X \to P$. □

Proof:

1. From Definition 21, we know that if $|Instances(P,t)| = 1$ then one of the conditions 2a-2d holds.
   Firstly, we consider case 2a, and we suppose that $Last(P)$ is the root of t. From axiom $A7$ we have $\forall P_i \in X^+$, $P_i \to P$ and so $P \in X^+$. Secondly, consider case 2c, and suppose that $X[E] \to P[N]$. In this case $P \in X^+$ is straightforward.

   Thirdly, consider case 2b. We suppose that $Last(P)$ is different from the root and not in Σ_{att}. Let I_{P_1} be the instance of P on t. The proof is by induction on the number of paths $Q_1, \ldots, Q_n$ ($\forall i \in [1 \ldots n], Q_i \in \mathbb{P}$ and $P \prec Q_i$) imposing P to have only one instance on t (as stated by condition 2b of Definition 21).

   We have the following situations for $n = 1$ (only Q_1 imposes P to have only one instance on t):

(a) If $|Instances(Q_1, t)| = 1$ then according to Definition 21(2b), there exist a path Q' that imposes the path Q_1 to have only one instance. Since $P \prec Q_1$ the path Q' imposes also P to have only one instance on t. This case is not possible because Q_1 is supposed to be the only path that imposes P to have only one instance on t.

(b) If $|Instances(Q_1, t)| = 2$ and $Q_1 \in X^+$ then let $I_{Q_{1,1}}$ and $I_{Q_{1,2}}$ be these two instances of Q_1. Let the subtree concerning the instance of P have the format illustrated on Figure 11(a). As $Q_1 \in X^+$, we have $Last(I_{Q_{1,1}}) =_V Last(I_{Q_{1,2}})$. In order to ensure that t is a two-instance tree, we have to verify whether it is not possible to have a two-instance tree with two instances for P that respects the conditions imposed on Q_1. Let us consider the tree t' identical to t except for the subtree concerning the instances of P in Figure 11(b). The tree t' has two instances of P and two instances of Q_1 ($I'_{Q_{1,1}}$ and $I'_{Q_{2,1}}$) such that $Last(I'_{Q_{1,1}}) =_V Last(I'_{Q_{2,1}}) =_V Last(I_{Q_{1,1}})$. Notice that tree t' is a two-instance tree, no matter whether $P \in X^+$ or $P \notin X^+$. On the one hand, $P \in X^+$ when for the two instances, I'_{P_1} and I'_{P_2}, of P on t' we have $Last(I'_{P_1}) =_V Last(I'_{P_2})$. On the other hand, $P \notin X^+$ when $Last(I'_{P_1}) \neq_V Last(I'_{P_2})$. Thus t' is a two instance tree. The existence of a two-instance tree t' having two instances of P proves that, when $Q_1 \in X^+$ and $|Instances(Q_1, t)| = 2$, tree t does not respect condition 2b of Definition 21. Thus, t is not a two-instance tree. In other words, it is not possible to consider $Q_1 \in X^+$ with Q_1 having 2 instances and being the path that imposes P to have only one instance.

(c) If $|Instances(Q_1, t)| = 2$ and $Q_1 \notin X^+$ then let $I_{Q_{1,1}}$ and $I_{Q_{1,2}}$ be these two instances of Q_1. From Definition 21, we have $Last(I_{Q_{1,1}}) \neq_V Last(I_{Q_{1,2}})$. Let the subtree concerning the instance of P have the format illustrated on Figure 12(a). In order to ensure that t is a two-instance tree, we have to verify whether it is not possible to have a two-instance tree with two instances for P that respects the conditions imposed on Q_1. We have:
   - Consider that $P \notin X^+$ and the tree t' identical to t except for the subtree concerning the instances of P in Figure 12(b). With the same arguments as in Case 1b, the existence of a two-instance tree t' contradicts the fact that t is a two-instance tree.
   - Finally consider that $P \in X^+$. Since $Q_1 \notin X^+$, by using Lemma 4, we can deduce that a tree with two instances of P is the tree t', identical to t except for the subtree concerning the instances of P which is illustrated in Figure 12(c). Clearly t' is not a two-instance tree because $|Instances(Q_1, t)| = 4$. Therefore the fact that t is a two-instance tree, is justified and consequently we have $P \in X^+$.

In conclusion, the only situation justifying the fact that P has one instance in t is the one where there exist a path $Q_1 \in \mathbb{P}$ such that $P \prec Q_1$, $|Instances(Q_1, t)| = 2$ and $Q_1 \notin X^+$. Thus, when $|Instances(P, t)| = 1$, we have $P \in X^+$.

Now suppose that $\forall\, m < n$, the paths $Q_1, \ldots, Q_m$ impose P to have only one instance on t, $P \in X^+$ and $\forall\, i \in [1 \ldots m]$, $P \prec Q_m$, $|Instances(Q_m, t)| = 2$ and $Q_m \notin X^+$. We now prove that $P \in X^+$ when the paths $Q_1, \ldots, Q_n$ impose P to have only one instance on t. We have:

- If there exist a path R such that $Parent(R) = P$ and $|Instances(R, t)| = 1$ then there exists a path Q' which imposes R to have one instance. By induction hypothesis, $R \prec Q'$, $|Instances(Q', t)| = 2$, $Q' \notin X^+$ and $R \in X^+$. By applying the same reasonning of situation 1c above, we conclude that $P \in X^+$.
- Otherwise, we are in the situation where for all path R_i we have: $Parent(R_i) = P$ and $|Instances(R_i, t)| = 2$. We can distinguish two cases:
  - For all R_i we have $R_i \in X^+$. This case is similar to the situation 1b. Therefore, we can build a tree with two instances of P which is a two-instance tree respecting the constraints over paths R_i. Thus the tree t (with only one instance of P) is not a two-instance tree.
  - There is a path R_j such that $R_j \notin X^+$. This case is similar to the the situation 1c. Therefore, we cannot build a tree with two instances of P which is a two-instance tree respecting the constraints over paths R_j. Thus, we conclude that $P \in X^+$.

Finally, consider case 2d of Definition 21. We consider that $Last(P) \in \Sigma_{att}$. We first prove by contradiction that $|Instances(Parent(P), t)| = 1$. Suppose that $|Instances(Parent(P), t)| \neq 1$. Thus, $|Instances(Parent(P), t)| = 2$, because t is a two-instance tree. Now, since t is a complete tree, each of the two instances of $Parent(P)$ must have a node attribute $Last(P)$ and so $|Instances(P, t)| = 2$. We obtain a contradiction with the hypothesis $|Instances(P, t)| = 1$. In conclusion, the tree t has only one instance for the path $Parent(P)$. Notice that, since $Last(Parent(P)) \in \Sigma_{ele}$, we have already proved that $Parent(P) \in X^+$. By using the axiom Attribute Uniqueness ($A6$) we have $Parent(P) \to P$ and then we conclude that $P \in X^+$.

2. The proof is done by contradiction and we suppose that $|Instances(P, t)| \neq 1$. Then, by considering the above document $\mathcal{T}$, $|Instances(P, t)|$ must be equal to 2. By hypothesis, we know that there is an instance I_P such that $I_P \preceq I_{Q_1}$ and $I_P \preceq I_{Q_2}$. Now, let I'_P be the second instance of P. Since t is complete, $\exists\, I_{Q_3} \in Instances(Q, t)$ such that $I_{P'} \preceq I_{Q_3}$. Thus $|Instances(Q, t)| \geq 3$. This is a contradiction with the fact that t has exactly one or two instances for each path in $\mathbb{P}$. This properties is illustrated in Figure 13.
3. ($\to$): From Definition 21 and Lemma 6(1), we have that if $P \in X^+$ then $|Instances(P, t)| = 2$ and there exist I_1, I_2 instances of the path P such that $I_1 =_V I_2$ or $|Instances(P, t)| = 1$. Since $X \subseteq X^+$, we have also that for all $P_i \in X$, $|Instances(P_i, t)| = 2$ and there exist I_1^i, I_2^i instances of the path P_i such that $I_1^i =_V I_2^i$ or $|Instances(P_i, t)| = 1$. Hence by using Definition 14

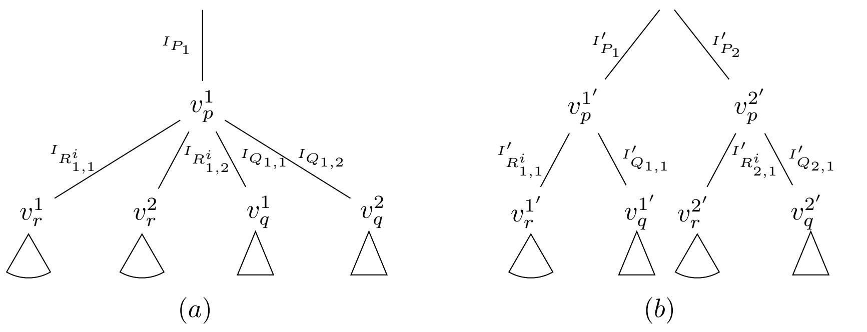

Fig. 11. Construction of a new two-instance tree t' with $Q \in X^+$.

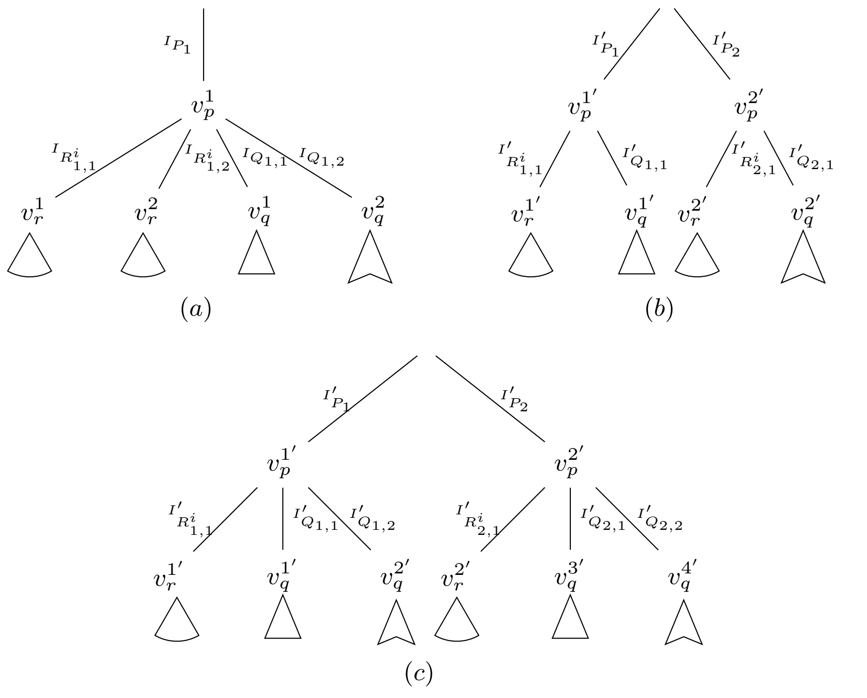

Fig. 12. Construction of a new two-instance tree t' with $Q \notin X^+$.

we conclude that $\mathcal{T} \models X \to P$.
($\leftarrow$): If $|Instances(P,t)| = 1$ then by using Lemma 6(1) we have $P_i \in X^+$. Else if $|Instances(P,t)| = 2$, we know from the hypothesis that the two

instances I_1, I_2 of the paths P in $\mathcal{T}$ are such that $I_1 =_V I_2$. By using Definition 21 we have $P_i \in X^+$. $\square$

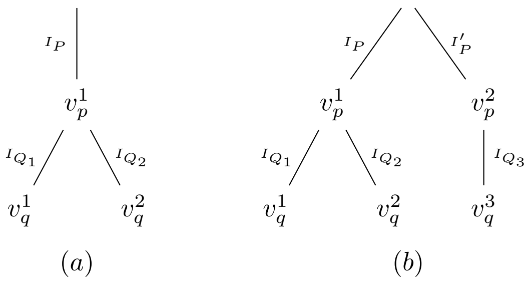

Fig. 13. Adding another instance of path P to the tree in (a) implies $|Instances(Q, t)| \geq 3$ for the tree in (b).

We now prove that the axiom system introduced in Definition 17 is complete. In other words, given a set of XFD $\mathcal{F}$, by using our inference rules, we can derive all XFD f such that $\mathcal{F} \models f$.

Theorem 3. *If $\mathcal{F} \models f$ then $\mathcal{F} \vdash f$.* $\square$

Proof: The proof is by contrapositive: we show that if $\mathcal{F} \not\vdash f$ then $\mathcal{F} \not\models f$.
Let $f = (C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow \{P_{n+1}\,[E_{n+1}] \ldots P_{n+m}[E_{n+m}]\}))$. Then, we consider that $X = \{C/P_1, \ldots, C/P_n\}$, $Y = \{C/P_{n+1}, \ldots, C/P_{n+m}\}$ and that both X and Y are in a given $\mathbb{P}$.
If $\mathcal{F} \not\models f$ then there must be an XML document that satisfies $\mathcal{F}$ but does not satisfy f. The proof consists in showing the existence of such a document.

Let us suppose an XML document $\mathcal{T} = (t, type, value)$ where t is a two-instance tree defined on the set of paths $X = \{C/P_1, \ldots, C/P_n\}$.

Fact 1: $\mathcal{T} \models \mathcal{F}$.

The proof is by contradiction. We suppose that $\mathcal{T} \not\models g$, where g is an XFD $(C,(\{Q_1\,[E_1], \ldots, Q_k\,[E_k]\} \rightarrow Q_{k+1}\,[E_{k+1}]))$ in $\mathcal{F}$. From Definition 14, as $\mathcal{T} \not\models g$, we can deduce that there exists two projections $\Pi^1_M(\mathcal{T})$ and $\Pi^2_M(\mathcal{T})$ for the branching path $M = \{C/Q_1, \ldots, C/Q_{k+1}\}$ in $\mathcal{T}$ such that:

$$\tau^1[C/Q_1, \ldots, C/Q_k] =_{E_i, i\in[1\ldots k]} \tau^2[C/Q_1, \ldots, C/Q_k] \text{ and} \tag{12}$$

$$\tau^1[C/Q_{k+1}] \neq_{E_{k+1}} \tau^2[C/Q_{k+1}]. \tag{13}$$

From (13) we have that $\Pi^1_M(\mathcal{T})[C/Q_{k+1}] \neq \Pi^2_M(\mathcal{T})[C/Q_{k+1}]$ and $|Instances(Q_{k+1}, t)| = 2$. From Definition 21(1), we obtain:

$$Q_{k+1} \notin X^+. \tag{14}$$

From Definition 12, we know that the instances of two paths belonging to the same branching path match on their longest common prefix path. Formally, for all combination of paths Q_i and Q_j such that $1 \leq i \leq k+1$ and $1 \leq j \leq k+1$, we have:

$\underline{\text{Considering } \Pi^1_M(\mathcal{T})}$

$$isInst\_lcp(C/Q_i, \Pi^1_M(\mathcal{T})[C/Q_i], C/Q_j, \Pi^1_M(\mathcal{T})[C/Q_j]) = true \text{ and} \tag{15}$$

$\underline{\text{Considering } \Pi^2_M(\mathcal{T})}$

$$isInst\_lcp(C/Q_i, \Pi^2_M(\mathcal{T})[C/Q_i], C/Q_j, \Pi^2_M(\mathcal{T})[C/Q_j]) = true \tag{16}$$

and we can also determine the following special nodes for $1 \leq i \leq k$:

$$v^1_{i,k+1} = Last(\Pi^1_M(\mathcal{T})[C/Q_i] \cap \Pi^1_M(\mathcal{T})[C/Q_{k+1}]) \text{ and} \tag{17}$$

$$v^2_{i,k+1} = Last(\Pi^2_M(\mathcal{T})[C/Q_i] \cap \Pi^2_M(\mathcal{T})[C/Q_{k+1}]) \tag{18}$$

From (15) and (16), together with Definition 10 we know that positions $v^1_{i,k+1}$ and $v^2_{i,k+1}$ exist in t. We have to consider two cases:

(a) $v^1_{i,k+1} = v^2_{i,k+1}$
(b) $v^1_{i,k+1} \neq v^2_{i,k+1}$

and choose for each $i \in [1 \dots k]$, a path $R_i \in \mathbb{P}$ respecting the following property:

$$R_i \in X^+ \text{ and } Q_i \cap Q_{k+1} \preceq R_i \preceq Q_i \tag{19}$$

– When we are in case (a), we consider $R_i = Q_i \cap Q_{k+1}$ - which clearly respects property (19). This case is illustrated in Figure 14(a).
From (13) we know that $\Pi^1_M(\mathcal{T})[C/Q_{k+1}] \neq_{E_{k+1}} \Pi^2_M(\mathcal{T})[C/Q_{k+1}]$. Since $v^1_{i,k+1} = v^2_{i,k+1}$, intersections in (17) and (18) are equal and correspond to an instance of R_i (call it I_{R_i}) *i.e.*,

$$I_{R_i} = \Pi^1_M(\mathcal{T})[C/Q_i] \cap \Pi^1_M(\mathcal{T})[C/Q_{k+1}] = \Pi^2_M(\mathcal{T})[C/Q_i] \cap \Pi^2_M(\mathcal{T})[C/Q_{k+1}]. \tag{20}$$

We can now see that:
$I_{R_i} = \Pi^1_M(\mathcal{T})[C/Q_i] \cap \Pi^1_M(\mathcal{T})[C/Q_{k+1}] \preceq \Pi^1_M(\mathcal{T})[C/Q_{k+1}]$ and that
$I_{R_i} = \Pi^2_M(\mathcal{T})[C/Q_i] \cap \Pi^2_M(\mathcal{T})[C/Q_{k+1}] \preceq \Pi^2_M(\mathcal{T})[C/Q_{k+1}]$
and that, by applying Lemma 6(2), we have $|Instances(R_i, \mathcal{T})| = 1$.
Then, from Lemma 6(1), we obtain that $R_i \in X^+$.

– When we are in case (b), we consider $R_i = Q_i$ which also respects property (19). This case is illustrated in Figure 14(b).
  As $v^1_{i,k+1} \neq v^2_{i,k+1}$, instances $\Pi^1_M(\mathcal{T})[C/Q_i]$ and $\Pi^2_M(\mathcal{T})[C/Q_i]$ are always distinct and so are the instances associated to the path R_i. More precisely, $|Instances(R_i, t)| = 2$. From this fact and the situation described in (12), we obtain that $\Pi^1_M(\mathcal{T})[C/Q_i] =_V \Pi^2_M(\mathcal{T})[C/Q_i]$. By considering Definition 21(1) we conclude that $R_i \in X^+$.

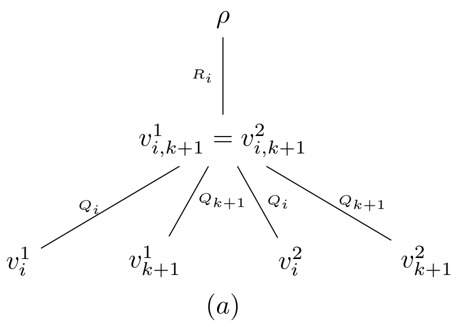

Fig. 14. Illustration of the two cases (a)$v^1_{i,k+1} = v^2_{i,k+1}$ and (b)$v^1_{i,k+1} \neq v^2_{i,k+1}$.

Now, from property (19), the XFD $g = (C,(\{Q_1\,[E_1], \ldots, Q_k\,[E_k]\} \rightarrow Q_{k+1}\,[E_{k+1}]))$ in $\mathcal{F}$, and the axiom Branch Prefixing (Definition 17, axiom $A4$) we deduce the XFD $g' = (C,(\{R_1\,[E_1], \ldots, R_k\,[E_k]\} \rightarrow Q_{k+1}\,[E_{k+1}]))$. Next, we assume that if $\{R_1, \ldots, R_k\} \subseteq X^+$ then $Q_{k+1} \in X^+$. Indeed, by Definition 20 and the axiom Union (Definition 19, axiom $A9$), we know that $X \rightarrow \{R_1\,[E_1], \ldots, R_k\,[E_k]\}$. From this rule and g', we derive $X \rightarrow Q_{k+1}$ by using the axiom Transitivity ($A3$). Thus, from Definition 20, we obtain $Q_{k+1} \in X^+$ which contradicts (14): $Q_{k+1} \notin X^+$. Thus, we conclude that $\mathcal{T} \models g$ for any $g \in \mathcal{F}$. In other words, $\mathcal{T} \models \mathcal{F}$.
Fact 2: $\mathcal{T} \not\models f$

Recall, from the beginning of our proof, that f is the XFD $X \rightarrow Y$. As $X \subseteq X^+$, for all $P_i \in X$ we have $Last(\Pi^1_M(\mathcal{T})[P_i]) =_{E_i} Last(\Pi^2_M(\mathcal{T})[P_i])$ for two given projections of M on $\mathcal{T}$. From our hypothesis, $\mathcal{F} \not\vdash f$ and so $Y \not\subseteq X^+$. Thus there is at least one path $P \in Y$ having instances I_1 and I_2 such that $Last(I_1) \neq_E Last(I_2)$. We deduce that $\mathcal{T} \not\models f$.

In conclusion we have built a tree $\mathcal{T}$ such that $\mathcal{T} \models \mathcal{F}$ and $\mathcal{T} \not\models f$ which establishes the proof of Theorem 3. □

6 An Algorithm for Computing the Closure of a Set of Paths

In this section we present an algorithm for computing $(C, X)^+$ and we prove that this computation is sound and complete.

Algorithm 1 Closure Algorithm of a set of Paths

```
Input: A finite set of paths $\mathbb{P}$, a set of XFD $\mathcal{F}$, a context path $C$, and a set of paths
    $X$ such that $C/X \subseteq \mathbb{P}$
Output: The set of paths $(C, X)^+$
 1: $T = \{P \mid P \preceq Q$ and $Q \in X\}$
 2: $V = \{P \mid Q \in T, Parent(P) = Q$ and $Last(P) \in \Sigma_{att}\}$
 3: $X^{(0)} = T \cup V$
 4: while $X^{(i)} \neq X^{(i-1)}$ do
 5:     $Y = \{P \mid$ there exist an XFD $(C, (\{P_1 [E_1], \ldots, P_n [E_n]\} \to P [E_{n+1}]))$ in $\mathcal{F}$
        such that one of the conditions below is satisfied:

    $(a)$    $\{P_1, \ldots, P_n\} \subseteq X^{(i)}$ or
    $(b)$    $\forall k \in [1, \ldots, n]$, $P_k \cap P = \{[]\}$ or
    $(c)$    $\exists P'_1, \ldots, P'_n \in X^{(i)}$ and $\forall k \in [1, \ldots, n]$, $P_k \cap P \preceq P'_k$ and $(P'_k \preceq P_k$ or
             $P'_k \preceq P)$

        $\}$
 6:     $T = \{P \mid P \preceq Q$ and $Q \in Y\}$
 7:     $V = \{P \mid Q \in T, Parent(P) = Q$ and $Last(P) \in \Sigma_{att}\}$
 8:     $X^{(i+1)} = X^{(i)} \cup T \cup V$
 9: end while
10: return $C/X^{(i)}$
```

In Algorithm 1, set T contains the prefixes of each path in X (line 1). Set V contains all the paths ending on attributes and having a path in T as its parent (line 2). We compute the closure of (C, X) in the while loop. In line 5 we compute the set Y. For each XFD f, with the form $(C, (\{P_1 [E_1], \ldots, P_n [E_n]\} \to P [E_{n+1}]))$, in $\mathcal{F}$, we insert P in Y when one of the following conditions hold:

(a) all paths of the left-hand side of f are in $X^{(i)}$, or

(b) axiom A12 can be applied or

(c) if there are paths $P'_1, \ldots, P'_n$ in $X^{(i)}$ that verify conditions of Axiom A4, then A4 is applied.

In lines 6 and 7 we compute the new sets T and V which are added to $X^{(i)}$. The loop ends when no new path can be added to $X^{(i)}$.

Theorem 4. *The set which is returned by Algorithm 1 is* $(C, X)^+$. □

Proof: First of all, the algorithm terminates: since $\mathbb{P}$ is finite, we must eventually reach j such that $X^{(j)} = X^{(j+1)}$. Thus we have a guarantee that the loop stops and the set $X^{(j)}$ is returned by the algorithm. We now prove that X^+ is $X^{(j)}$

for this value of j.
Soundness ($X^{(j)} \subseteq X^{+}$): We show by induction on i that if a path P is placed in $X^{(i)}$ during Algorithm 1, then P is in X^{+} (or $X \to P$ from Definition 20).

- Basis: $i = 0$. From Line 3 of Algorithm 1, the path P is in X, or T or V (notice that $X \subseteq T$ since each path in X is its proper prefix). If P is in X then, by Reflexivity ($A1$), we have $X \to P$. Otherwise if P is in T then there exist a path Q in X such that $P \preceq Q$. We know that $X \to Q$. By Ascendency ($A5$) we have $Q \to P$ and by Transitivity ($A3$) we obtain $X \to P$. Or else, if P is in V then $Last(P) \in \Sigma_{att}$ and there exist a path Q in T such that $Q = Parent(P)$. We know that $X \to Q$. By Attribute Uniqueness ($A6$) we have $Q \to P$ and by Transitivity ($A3$) we obtain $X \to P$.
- Induction: Let $i > 0$ and assume that $X^{(i-1)}$ contains only paths in X^{+}. We prove that if P is placed in $X^{(i)}$ then P is in X^{+}.
  1. From line 8 of Algorithm 1, we have $X^{(i)} = X^{(i-1)} \cup Y \cup T \cup V$. If P is in $X^{(i-1)}$ then by induction hypothesis P is in X^{+}.
  2. Otherwise, if the path P is in Y then there exist an XFD $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \to P\,[E_{n+1}]))$ in $\mathcal{F}$ such that one of the conditions a-b of Line 5 is satisfied. Let us analyse these three conditions:
     (a) Firstly we suppose that $Y = \{P_1, \ldots, P_n\} \subseteq X^{(i-1)}$. Since $Y \subseteq X^{(i-1)}$, we know $Y \subseteq X^{+}$ by inductive hypothesis. Thus, $X \to Y$ by Lemma 5. By Transitivity ($A3$), $X \to Y$ and $Y \to P$ imply $X \to P$ and so P is in X^{+}.
     (b) Secondly we suppose that $\forall\, k \in [1, \ldots, n]$, $P_k \cap P = \{[]\}$. From Subtree Uniqueness ($A11$) we deduce that for any path Q, $Q \to P$. Let Q be a path in $X^{(i-1)}$. By Transitivity ($A3$), $X \to Q$ and $Q \to P$ imply $X \to P$. Thus, X is in $(C, X)^{+}$.
     (c) Finally, we suppose that $\exists\, P'_1, \ldots, P'_n \in X^{(i-1)}$ and $\forall\, k \in [1, \ldots, n]$, $P_k \cap P \preceq P'_k$ and ($P'_k \preceq P_k$ or $P'_k \preceq P$). If the set $Y = \{P'_1, \ldots, P'_n\}$ respect these conditions then by Branch Prefixing ($A4$), we have $Y \to P$. Since $Y \subseteq X^{(i-1)}$, by inductive hypothesis and Lemma 5, we have $X \to Y$. By Transitivity ($A3$), $X \to Y$ and $Y \to P$ imply $X \to P$.
  3. Or else, if P is in $T \cup V$ then with the same arguments as in the basis step, we assume that P is in X^{+}.

Completeness ($X^{+} \subseteq X^{(j)}$): Now we prove that if P is in X^{+} then P is in $X^{(j)}$. The proof is by contradiction. Suppose that P is in X^{+} but P is not in $X^{(j)}$. Recall that Algorithm 1 returns an answer only if $X^{(j)} = X^{(j+1)}$.

We suppose a two-instance tree t defined on $X^{(j)}$. This two-instance tree t has two instances such that their last nodes are equal for paths in $X^{(j)}$, and their last nodes are not equal for other paths. We claim that t satisfies $\mathcal{F}$. If not, let $U \to V$ be an XFD in $\mathcal{F}$ that is violated by t. Then $U \subseteq X^{(j)}$ and V cannot be a subset of $X^{(j)}$. If the violation occurs then we can use the same argument in the proof of Theorem 3(Fact 1). Thus, as $V \not\subseteq X^{(j)}$ and $U \to V$ is not violated, the set $X^{(j+1)}$ should contain V. Therefore, $X^{(j+1)}$ cannot be the same as $X^{(j)}$

as supposed.

Consequently, the two-instance tree t must also satisfy $X \to P$. The reason is that P is assumed to be in X^+, and therefore, $X \to P$ follows from $\mathcal{F}$ by the axiom system. Since the axiom system is sound, any tree satisfying $\mathcal{F}$ also satisfies $X \to P$. But the only way $X \to P$ could hold in t is if P is in $X^{(j)}$. Hence we have a contradiction because P is supposed not to be in $X^{(j)}$. We conclude that P is in $X^{(j)}$ which is the set returned by Algorithm 1. □

7 Computing Functional Dependencies for Interoperability

Given XFD sets $\mathcal{F}_1, \ldots, \mathcal{F}_n$, our goal is to propose an algorithm that computes a set *cover*$\mathcal{F}$ of XFD equivalent to the biggest set $\mathcal{F}$ containing XFD that are not in contradiction with any set $\mathcal{F}_1, \ldots, \mathcal{F}_n$. In other words, all documents in $X_1, \ldots, X_n$ valid *w.r.t.* $\mathcal{F}_1, \ldots, \mathcal{F}_n$ should stay valid *w.r.t.* $\mathcal{F}$.

To obtain $\mathcal{F}$, two strategies can be considered. On one hand, we may consider the computation of $\mathcal{F}$ itself, which is more natural to conceive. This computation takes into account *all* the XFD in $\mathcal{F}_1^+$ and $\mathcal{F}_2^+$ (which are very expensive to compute), and generates a *too* big set $\mathcal{F}$. On the other hand, we can consider the computation of a set *cover*$\mathcal{F}$, built from $\mathcal{F}_1$ and $\mathcal{F}_2$, equivalent to $\mathcal{F}$. This computation should take into account some important situations and ensure that all XFD f, derived by both $\mathcal{F}_1$ and $\mathcal{F}_2$, will also be derived by $\mathcal{F}$. This second strategy requires a finer algorithm but produces a manipulable set of XFD. This report presents both strategies.

Notice that, for the sake of simplicity, we suppose only two local sources, but our algorithms can be easily extended for n local sources. Translation functions Φ_1 and Φ_2 are available. These functions work on the translation table (obtained from the ontology alignment $\mathcal{A}$): given a path P from, for instance $\mathbb{P}_2$, $\Phi_1(P)$ gives its equivalent path in $\mathbb{P}_1$, if it exists; otherwise it returns the identity. The function Φ_2 works on a symmetric way. Indeed, we note i and $\bar{i}$ to indicate symmetric sources (*e.g.*, when $i = 1$, $\bar{i} = 2$).

As input, both algorithms receive the local sets of XFD together with the set of possible paths given by each local schema.

7.1 Algorithm for Computing $\mathcal{F}$

Algorithm 2 generates $\mathcal{F}$ as expected. Notice that to know whether an XFD follows from two given sets $\mathcal{F}_1$ and $\mathcal{F}_2$, one have to consider the closure of these two sets. Algorithm 2 considers each local set $\mathcal{F}_i^+$. Then, each XFD $f = (C, (X \to B))$ in $\mathcal{F}_i^+$ is checked and added to $\mathcal{F}$ when one of the following properties hold:
(i) There is no path in the source $\bar{i}$ equivalent to the right-hand side of f (line 4). Thus, documents in $\bar{i}$ do not violate f.
(ii) There is no set of paths in the source $\bar{i}$ equivalent to the set on the left-hand

Algorithm 2 Set of XFD ensuring the interoperability of S *w.r.t.* S_1 and S_2

```
Input:
    – A set of XFDs $\mathcal{F}_1$ for schema $\mathcal{D}_1$
    – A set of XFDs $\mathcal{F}_2$ for schema $\mathcal{D}_2$
    – The set of paths $\mathbb{P}_1, \mathbb{P}_2$ specified by $\mathcal{D}_1$ and $\mathcal{D}_2$
    – Translation functions $\Phi_1$ and $\Phi_2$
Output: The set of XFD $\mathcal{F}$ for the integrated system
 1: $\mathcal{F} = \emptyset$
 2: for $i = 1$ to 2 do
 3:     for each $(C, (X \to B)) \in \mathcal{F}_i^+$ do
 4:         if $\Phi_{\bar{i}}(C/B) \notin \mathbb{P}_{\bar{i}}$ then
 5:             $\mathcal{F} = \mathcal{F} \cup \{(C, (X \to B))\}$
 6:         else if $\Phi_{\bar{i}}(C/X) \not\subseteq \mathbb{P}_{\bar{i}}$ then
 7:             $\mathcal{F} = \mathcal{F} \cup \{(C, (X \to B))\}$
 8:         else if $\Phi_{\bar{i}}(C/B) \in \Phi_{\bar{i}}(C, X)^+_{\mathcal{F}_{\bar{i}}}$ then
 9:             $\mathcal{F} = \mathcal{F} \cup \{(C, (X \to B))\}$
10:         end if
11:     end for
12: end for
13: return $\mathcal{F}$
```

side of f (line 6). Since no set of paths in the source $\bar{i}$ correspond to X, no document in $\bar{i}$ violates f.
(iii) In the source $\bar{i}$, there is a path equivalent to C/B that belongs to the closure of a set of paths equivalent to C/X (line 8). Therefore, XFD f exists in both sources and can be added to $\mathcal{F}$.

We prove that $\mathcal{F}$ is the subset of $\mathcal{F}_i^+ \cup \mathcal{F}_{\bar{i}}^+$ that contains all the XFD in the intersection $\mathcal{F}_i^+ \cap \mathcal{F}_{\bar{i}}^+$ together with all those XFD whose left-hand side or right-hand side concerns concepts existing only in $\mathcal{D}_i$. To present this theorem, we first define the following sets of XFD.

Given two sets of XFD, $\mathcal{F}_i$ and $\mathcal{F}_{\bar{i}}$, we define set $\mathcal{K}_i$ of XFD as:

$$\mathcal{K}_i = \{X \to A \mid X \to A \in \mathcal{F}_i^+ \text{ and } [((X \subseteq \mathbb{P}_i) \text{ and } (X \not\subseteq \mathbb{P}_{\bar{i}})) \text{ or } (A \in (\mathbb{P}_i \setminus \mathbb{P}_{\bar{i}})]\}$$

Intuitively, $\mathcal{K}_i$ contains all the XFD f which can be obtained from $\mathcal{F}_i$ but that cannot be violated by documents in $X_{\bar{i}}$ due to one of the two reasons:
(a) the right-hand side of f is a path B which belongs to $\mathbb{P}_i$ but not to $\mathbb{P}_{\bar{i}}$ or
(b) the left-hand side of f is a set of paths X which is included in $\mathbb{P}_i$ but not in $\mathbb{P}_{\bar{i}}$.

Theorem 5. *The set $\mathcal{F}$, returned by Algorithm 2, is the union $\mathcal{F} = (\mathcal{F}_1^+ \cap \mathcal{F}_2^+)$ $\cup\ \mathcal{K}_1 \cup \mathcal{K}_2$.* □

Proof: Straightforward from Algorithm 2. □

Theorem 6. *The set $\mathcal{F}$, returned by Algorithm 2, is the biggest subset of $\mathcal{F}_1^+ \cup \mathcal{F}_2^+$ such that we can guarantee, without considering data, that $X_1 \models \mathcal{F}$ and $X_2 \models \mathcal{F}$.* $\square$

Proof: Proving that $X_1 \models \mathcal{F}$ and $X_2 \models \mathcal{F}$ is straightforward. Next we prove by contradiction that $\mathcal{F}$ is the biggest subset of $\mathcal{F}_1^+ \cup \mathcal{F}_2^+$ ensuring this property. Suppose there is a set G of XFD that is a subset of $\mathcal{F}_1^+ \cup \mathcal{F}_2^+$ such that $\mathcal{F} \subset G$ and that we always have $X_i \models G$ for $i = 1, 2$. Let $(C, (X \to A))$ be an XFD in G that is not in $\mathcal{F}$. As $(C, (X \to A)) \notin \mathcal{F}$, we know that:

1. C/A is in $\mathbb{P}_1$ and in $\mathbb{P}_2$; otherwise the XFD is added to $\mathcal{F}$ in step 4 of Algorithm 2.
2. C/X is included in $\mathbb{P}_1$ and $\mathbb{P}_2$; otherwise the XFD is added to $\mathcal{F}$ in step 6 of Algorithm 2.
3. $C/A \in X^+_{\mathcal{F}_1}$ and $C/A \notin X^+_{\mathcal{F}_2}$, or vice-versa; otherwise the XFD is added to $\mathcal{F}$ in step 8 of Algorithm 2.

From 3, we know that $(C, (X \to A)) \in \mathcal{F}_1^+$ but $(C, (X \to A)) \notin \mathcal{F}_2^+$. In this case, from items 1-2 above, documents in X_2 may violate $(C, (X \to A))$. This is a contradiction with the assumption that it is always true that $X_i \models G$ for $i = 1, 2$. $\square$

Corollary 1. $\mathcal{F}^+ \cap (\mathcal{F}_1^+ \cup \mathcal{F}_2^+) = \mathcal{F}$. $\square$

Proof: Let $(C, (X \to A))$ be an XFD in $\mathcal{F}^+$ which is not in $\mathcal{F}$. Thus $(C, (X \to A))$ is not in $\mathcal{F}_1^+ \cup \mathcal{F}_2^+$, otherwise, from Theorem 6, it is not always true that $X_i \models (C, (X \to A))$, which is impossible (since $(C, (X \to A)) \in \mathcal{F}^+$ and that we have the assurance that $X_i \models \mathcal{F}$ for $i = 1, 2$). $\square$

7.2 Towards an optimized version

Algorithm 2 depends on two main procedures: Algorithm 1 which computes the closure of a set of paths $((C, X)^+)$ and an algorithm for computing the closure of a set of XFD $(\mathcal{F}^+)$. As expected, the computation of $\mathcal{F}^+$ is the bottleneck of our method. Our computation consists in generating all the possible subsets X of the set of paths $\mathbb{P}$ (*i.e.*, $2^{|\mathbb{P}|} - 1$ subsets) and in computing X^+ for each of them. For each path $P \in X^+$, the XFD $X \to P$ is added to $\mathcal{F}^+$. For computing $\mathcal{F}^+$, we have to call $2^{|\mathbb{P}|}$ times Algorithm 1.

Algorithm 2 uses Algorithm 1 to compute each $\mathcal{F}_i^+$. Each $\mathcal{F}_i^+$ has about $2^{|\mathbb{P}|}$ XFD and, for each of them, we have to test conditions from lines 4-8. The complexity of Algorithm 2, set aside the computation of $\mathcal{F}_i^+$, is $O(2^{|\mathbb{P}|}.g)$ where g is the complexity of Algorithm 1.

It is important to notice that we cannot just change $\mathcal{F}_i^+$ by $\mathcal{F}_i$ in line 3 of Algorithm 2. To understand this problem, let us consider sets $\mathcal{F}_1$ and $\mathcal{F}_2$ from which we can derive an XFD $f = (C, (X \to B))$ by different derivation sequences. Suppose that in $\mathcal{F}_1$ we have $f_1, \ldots, f_k, \ldots, f$ while in $\mathcal{F}_2$ we have

$f'_1, \ldots, f'_k, \ldots, f$. Moreover, we assume that, due to conditions stated in lines 4, 6 and 8, the dependencies f_k and f'_k are not included in $\mathcal{F}$ and, thus, the derivation of f is not possible from the new set $\mathcal{F}$ built by Algorithm 2. This would be a mistake, since f is derived by both $\mathcal{F}_1$ and $\mathcal{F}_2$. This is why Algorithm 2 takes into account all XFD in the closure of $\mathcal{F}_1$ and $\mathcal{F}_2$. Now, in order to optimize Algorithm 2 we build a new version which use $\mathcal{F}_i$ instead of $\mathcal{F}_i^+$ with some modifications. The complexity of this optimized version does not have the factor $2^{|\mathbb{P}|}$ present on the complexity of Algorithm 2.

7.3 Algorithm for Computing *cover$\mathcal{F}$*

Algorithm 3 Computation of *cover$\mathcal{F}$* (set of XFD ensuring the interoperability of S *w.r.t.* S_1 and S_2)

Input:
- A set of XFDs $\mathcal{F}_1$ for schema $\mathcal{D}_1$
- A set of XFDs $\mathcal{F}_2$ for schema $\mathcal{D}_2$
- The set of paths $\mathbb{P}_1, \mathbb{P}_2$ specified by $\mathcal{D}_1$ and $\mathcal{D}_2$
- Translation functions Φ_1 and Φ_2

Output: The set of XFD *cover$\mathcal{F}$* for the integrated system

```
1: $cover\mathcal{F} = \emptyset$
2: for $i = 1$ to 2 do
3:     $G = \mathcal{F}_i$
4:     for each $(C, (X \rightarrow B)) \in G$ do
5:         if $\Phi_{\bar{i}}(C/B) \notin \mathbb{P}_{\bar{i}}$ then
6:             $cover\mathcal{F} = cover\mathcal{F} \cup \{(C, (X \rightarrow B))\}$
7:         else if $\Phi_{\bar{i}}(C/X) \not\subseteq \mathbb{P}_{\bar{i}}$ then
8:             $cover\mathcal{F} = cover\mathcal{F} \cup \{(C, (X \rightarrow B))\}$
9:         else if $\Phi_{\bar{i}}(C/B) \in \Phi_{\bar{i}}(C, X)^+_{\mathcal{F}_{\bar{i}}}$ then
10:            $cover\mathcal{F} = cover\mathcal{F} \cup \{(C, (X \rightarrow B))\}$
11:        else
12:            $H = closure1Step(C, B, \mathcal{F}_i) \setminus \{C/B\}$
13:            $G = G \cup \{(C, (X \rightarrow D)) \mid C/D \in H\}$
14:            $K = inverseClosure1Step(C, X, \mathcal{F}_i) \setminus \{C/X\}$
15:            $G = G \cup \{(C, (Y \rightarrow B)) \mid C/Y \in K\}$
               % Recall that $C/Y$ is a shorthand for $\{C/A_1, \ldots, C/A_n\}$ and that $K$ is
               a set of path sets.
16:            $G = G \cup \{(C, (Z \rightarrow B)) \mid (C, (Z \rightarrow B))$ is obtained by using Axiom $A4$
               on $(C, (X \rightarrow B))\}$ % Notice that $Z$ is a set of prefixes of
               paths in $X$ or $B$
17:        end if
18:    end for
19: end for
20: return $cover\mathcal{F}$
```

In this section we present Algorithm 3 that generates the set $cover\mathcal{F}$ of XFD equivalent to the biggest set $\mathcal{F}$. As in Algorithm 2 each XFD $f = (C, (X \to B))$ in $\mathcal{F}_i$ is checked and added to $cover\mathcal{F}$ according to the properties (i)-(iii) established on page 35 Section 7.1.

From line 12 to 16, Algorithm 3 takes into account the fact that, working with $\mathcal{F}_i$, some XFD in $\mathcal{F}_i^+$ may be neglected. To understand this problem, let us consider sets $\mathcal{F}_1$ and $\mathcal{F}_2$ from which we can derive an XFD $f = (C, (X \to B))$ by different derivation sequences. Suppose that in $\mathcal{F}_1$ we have $f_1, \ldots, f_k, \ldots, f$ while in $\mathcal{F}_2$ we have $f'_1, \ldots, f'_k, \ldots, f$. Moreover, we assume that, due to conditions stated in lines 5, 7 and 9, the dependencies f_k and f'_k are not included in $\mathcal{F}$ and, thus, the derivation of f is not possible from the new set $cover\mathcal{F}$ built by Algorithm 3. This would be a mistake, since f is derived by both $\mathcal{F}_1$ and $\mathcal{F}_2$. One solution (as mentioned in the beginning of this section) would be to start with (in line 4) the closure of $\mathcal{F}_1$ and $\mathcal{F}_2$. However, this solution implies the generation of a too big and, thus, not manipulable set of XFD. Algorithm 3 does better: when the test in line 9 fails, it computes all XFD $f_j = (C, (Y \to A))$ such that:
(i) $C/X \in (C, Y)^+$ and $A = B$ or
(ii) $C/Y = C/X$ and $C/A \in (C, B)^+$ or
(iii) $C/A = C/B$ and f_j is obtained by using Axiom $A4$ on f.
Tests from lines 12-16 are then performed on these computed XFD. In this way, we do not compute the entire closure of a set $\mathcal{F}_i$ but, when necessary, we calculate part of it. This computation is done by using $closure1Step$ and $inverseClosure1Step$. Function $closure1Step$ computes one step of the closure of a set of paths. Its implementation consists in deleting from Algorithm 1 the while loop: in this way all the instructions are executed only once. Function $inverseClosure1Step$ consider XFD inversely and computes an "inverse closure" one step backward.

The following example illustrates the computation performed in lines 12-16 of Algorithm 3.

Example 12. Let $\mathcal{F}_1 = \{(C, (A \to B)), (C, (B \to M)), (C, (M \to D)), (C, (D \to E)), (C, (O \to Z))\}$ and let $\mathcal{F}_2 = \{(C, (A \to B)), (C, (B \to M)), (C, (B \to O)), (C, (O \to E)), (C, (D \to N))\}$. Without lines 12-16 in Algorithm 3, the XFD $(C, (A \to E))$, derivable from both $\mathcal{F}_1$ and $\mathcal{F}_2$, would not be derived from $cover\mathcal{F}$.
Let us consider part of the execution of Algorithm 3. Table 4 shows the XFD we obtain when considering each XFD in $\mathcal{F}_1$ (line 3 of Algorithm 3). The first column of this table shows the XFD in G being verified. The second column indicates XFD that are added to G due to lines 12-16. Finally the last column shows XFD that are inserted in $cover\mathcal{F}$.
Table 4 is obtained by following the execution of Algorithm 3. For instance, let us consider the third line in Table 4: the case when the XFD $(C, (M \to D))$ in $\mathcal{F}_1$ is taken in line 4 of Algorithm 3. This XFD does not verify any condition among conditions in lines 5, 7 and 9. When line 12 is executed, the set $H = \{C/E\}$ is computed, since $closure1Step(C,\ D,\ \mathcal{F}_1)$ gives $\{C/D, C/E\}$. Thus,

G **(XFD being considered)**	**Add to** G	*cover*$\mathcal{F}$ **contains**
$(C, (A \to B))$		$(C, (A \to B))$ (cond. line 9)
$(C, (B \to M))$		$(C, (B \to M))$ (cond. line 9)
$(C, (M \to D))$	$(C, (M \to E))$ $(C, (B \to D))$ $(C, ([\,] \to D))$	
$(C, (D \to E))$	$(C, (M \to E))$ $(C, ([\,] \to E))$	
$(C, (O \to Z))$		$(C, (O \to Z))$ (cond. line 5)
$(C, (M \to E))$	$(C, (B \to E))$ $(C, ([\,] \to E))$	
$(C, (B \to D))$	$(C, (B \to E))$ $(C, (A \to D))$ $(C, ([\,] \to D))$	
$(C, (B \to E))$		$(C, (B \to E))$ (cond. line 9)
$(C, (A \to D))$	$(C, (A \to E))$ $(C, ([\,] \to D))$	
$(C, (A \to E))$		$(C, (A \to E))$ (cond. line 9)

Table 4. Computation of (part) of *cover*$\mathcal{F}$: XFD obtained when considering $\mathcal{F}_1$

the XFD $(C, (M \to E))$ is added to G (line 13). When line 14 is executed, the set $K = \{\{C/B\}\}$ is computed, since *inverseClosure1Step*$(C, M, \mathcal{F}_1)$ gives $\{\{C/B\}, \{C/M\}\}$. Thus, the XFD $(C, (B \to D))$ is added to G (line 15). When line 16 is executed, the XFD $(C, ([\,] \to D))$ is added to G. Notice that these three XFD are analysed later (lines 6 and 7 of Table 4). They are not included in *cover*$\mathcal{F}$, but generate other XFD as, for instance, $(C, (A \to E))$, which is finally added to *cover*$\mathcal{F}$. □

Function *inverseClosure1Step* considers XFD inversely and works one step backward, starting from a giving set of paths. It is implemented by Algorithm 4. The computed result is a set S containing sets of paths. Lines 2-9 represent the initialisation of S. To this end, Algorithm 4 stores in set R all sets of paths that imply the given path P due to axioms A1, A5 and A6.

- If *Last*(P) is an element (line 3), by applying A5 we have $C, (Q \to P)$ for all path Q having the path P as a prefix. Remember that $P \preceq P$.
- If *Last*(P) is an attribute (line 5), by applying A6 we have $C, (Parent(P) \to P)$. The set S is initialized with two sets: one containing P itself (A1) and another containing $Parent(P)$ (A6).
- If *Last*(P) is *data* (line 8) the set S is initialized with $\{P\}$ due to A1.

In line 10, Y_1 contains all the set of paths Z appearing on the left-hand side of an XFD that has P as its right-hand side, *i.e.*, $C, (Z \to P)$. In line 11, Y_2 contains all the set of paths respecting the conditions for applying the Axiom $A4$ on an XFD that has P as its right-hand side. Finally, in line 12, the result is computed from Y_1, Y_2 and R by performing a *distributing union* defined as follows: Let S_1

Algorithm 4 One Step of Inverse Closure

```
Input: A finite set of paths $\mathbb{P}$, a set of XFD $\mathcal{F}$, a context path $C$, and a set of paths
    $X$ such that $C/X \subseteq \mathbb{P}$
Output: The set of paths $inverseClosure1Step(C, X, \mathcal{F})$
1: $S = \{\emptyset\}$
2: for each $P \in X$ do
3:     if $Last(P) \in \Sigma_{ele}$ then
4:         $R = \{\{Q\} \mid P \preceq Q\}$
5:     else if $Last(P) \in \Sigma_{att}$ then
6:         $R = \{\{P\}\} \cup \{\{Parent(P)\}\}$
7:     else
8:         $R = \{\{P\}\}$
9:     end if
10:    $Y_1 = \{\{P_1, \ldots, P_n\}$ | there exist an XFD $(C, (\{P_1\,[E_1], \ldots, P_n\,[E_n]\} \rightarrow$
       $P'\,[E_{n+1}]))$ in $\mathcal{F}$ such that $P' = P$ $\}$
11:    $Y_2 = \{\{P_1, \ldots, P_n\}$ | there exist an XFD $(C, (\{P'_1\,[E_1], \ldots, P'_n\,[E_n]\} \rightarrow$
       $P'\,[E_{n+1}]))$ in $\mathcal{F}$ such that $P' = P$ and $\forall k \in [1, \ldots, n]$, $P'_k \cap P \preceq P_k$ and
       $(P_k \preceq P'_k$ or $P_k \preceq P)\}$
12:    $S = S \uplus (R \cup Y_1 \cup Y_2)$ % *where $\uplus$ stands for the distributing union of two sets*
13: end for
    % *$C/S$ a contracted form to express $C/P$ for each path $P$ appearing in a set $Z \in S$*
14: return $C/S$
```

and S_2 be sets of path sets. The set $S_1 \uplus S_2$ contains all sets resulting from the union of each $W_1 \in S_1$ to each $W_2 \in S_2$. The following example illustrates the computation of Algorithm 4.

Example 13. Let us consider three XFD in $\mathcal{F}$: $(C, (D \rightarrow O))$, $(C, (\{A, B\} \rightarrow O))$ and $(C, (\{M, N\} \rightarrow Q))$. Moreover, we assume that $Q \prec J$ and that $O \prec I$. Let $X = \{O, Q\}$ where O and Q are paths reaching element nodes. Following Algorithm 4, we obtain, step by step, the results shown below:

	1st Step of the **for** loop	2nd Step of the **for** loop
path $P \in X$	O	Q
set R (line 3)	$\{\{O\}, \{I\}\}$	$\{\{Q\}, \{J\}\}$
set Y_1 (line 10)	$\{\{A, B\}, \{D\}\}$	$\{\{M, N\}\}$
set Y_2 (line 11)	$\{\{[\,]\}\}$	$\{\{[\,]\}\}$
set S (line 12)	$\{\{O\}, \{I\}, \{A, B\}, \{D\}, \{[\,]\}\}$	$\{\{O, Q\}, \{O, J\}, \{O, M, N\}, \{O, [\,]\}, \{I, Q\}, \{I, J\}, \{I, M, N\}, \{I, [\,]\}, \{A, B, Q\}, \{A, B, J\}, \{A, B, M, N\}, \{A, B, [\,]\}, \{D, Q\}, \{D, J\}, \{D, M, N\}, \{D, [\,]\}, \{[\,], Q\}, \{[\,], J\}, \{[\,], M, N\}, \{[\,]\}\}$

The computed result indicates that XFD such as $(C, (\{I, Q\} \rightarrow X))$ or $(C, (\{A, B, M, N\} \rightarrow X))$ are derivable from our initial set $\mathcal{F}$. Thus, Algorithm 4 returns a set $S = \{\{C/O, C/Q\}, \{C/O, C/J\} \ldots \{C/[\,]\}\}$. Notice also that, in this case, in line 14 of Algorithm 3 the set K will contain all sets in S except $\{C/O, C/Q\}$ which corresponds to our X. $\square$

7.4 Properties of *cover$\mathcal{F}$*

In this section we prove that Algorithm 3 works correctly, and fulfil our goals.

Lemma 7. Let $\mathcal{F}$ be a set of XFD such that $(C, (X \to Y)) \in \mathcal{F}$. Let $(C, (Z_1 \to Z_2))$ be an XFD different from $(C, (X \to Y))$.

If $\mathcal{F} \vdash (C, (Z_1 \to Z_2))$ then $G \vdash (C, (Z_1 \to Z_2))$ where G is the set of XFD defined as follows:

$$\begin{aligned} G = \mathcal{F} \cup (&\{(C, (X \to V)) \mid V \in \mathit{closure1Step}(C, Y, \mathcal{F})\} \\ &\cup \{(C, (W \to Y)) \mid W \in \mathit{inverseClosure1Step}(C, X, \mathcal{F})\} \\ &\cup \{(C, (\{P'_1, \ldots, P'_n\} \to Y)) \mid X = \{P_1, \ldots, P_n\} \text{ and } \forall\, k \in [1, \ldots, n], \\ &\quad P_k \cap Y \preceq P'_k \text{ and } (P'_k \preceq P_k \text{ or } P'_k \preceq Y)\}) \ \setminus \{(C, (X \to Y))\}. \end{aligned}$$ $\square$

Proof:

- Suppose that $(C, (Z_1 \to Z_2)) \in \mathcal{F}$. Since $(C, (Z_1 \to Z_2))$ is different from $(C, (X \to Y))$, and G contains each XFD in $\mathcal{F}$ except $(C, (X \to Y))$ then $G \vdash (C, (Z_1 \to Z_2))$.
- Otherwise $(C, (Z_1 \to Z_2)) \notin \mathcal{F}$. As $F \vdash (C, (Z_1 \to Z_2))$, there exist a sequence α of XFD containing XFD in $\mathcal{F}$ such that α derives $(C, (Z_1 \to Z_2))$. When α does not contain $(C, (X \to Y))$, it is obvious that $G \vdash (C, (Z_1 \to Z_2))$.
  We consider now that α contains $(C, (X \to Y))$.
  1. Let α_1 be the derivation sequence $f_1 \ldots f_n$ such that :
     (a) each f_i (for $1 \leq i \leq n$) is in α and
     (b) each f_i (for $1 \leq i \leq n$) is of the form $X_i \to X$, *i.e.*, $X_1, \ldots, X_n$ are the paths in $\mathit{inverseClosure1Step}(C, X, \mathcal{F})$.
     We say that α_1 is the sub-sequence of α which derives the set of paths X in one step.
     Since $\mathcal{F} \vdash (C, (X_i \to X))$ and $\mathcal{F} \vdash (C, (X \to Y))$ then by transitivity $\mathcal{F} \vdash (C, (X_i \to Y))$. We construct the sequence α' by replacing, in α, the sub-sequence
     $$\alpha_1, (C, (X \to Y))$$
     by
     $$(C, (X_1 \to Y)), \ldots, (C, (X_n \to Y)).$$
     By considering α' we have proved that $G \vdash (C, (Z_1 \to Z_2))$ since each XFD $(C, (X_i \to Y)) \in G$.
  2. Similarly to step 1, let α_2 be the sub-sequence of α which derives the paths $Y_1, \ldots, Y_n$ in one step from the path Y. The paths $Y_1, \ldots, Y_n$ are in $\mathit{closure1Step}(C, Y, \mathcal{F})$. Since $\mathcal{F} \vdash (C, (X \to Y))$ and $\mathcal{F} \vdash (C, (Y \to Y_i))$ $\forall i \in [1 \ldots n]$ then by transitivity $\mathcal{F} \vdash (C, (X \to Y_i))$. We construct the sequence α'' by replacing the sub-sequence $(C, (X \to Y)), \alpha_2$ by

$(C,(X \to Y_1)),\ldots,(C,(X \to Y_n))$ in α. By considering α'' we have proved that $G \vdash (C,(Z_1 \to Z_2))$ since each XFD $(C,(X \to Y_i)) \in G$.

3. Similarly to step 1, let α_3 be the sub-sequence of α which derives the path Y in one step from each set of paths X_i ($i \in [1,\ldots,n]$) and by using Axiom $A4$ on the XFD $(C,(X \to Y))$. The sets of paths $X_1,\ldots,X_n$ are in $inverseClosure1Step(C,Y,\mathcal{F})$. We construct the sequence α''' by deleting the XFD $(C,(X \to Y))$ in α. Since each XFD $(C,(X_i \to Y)) \in G$ then by considering α''' we have proved that $G \vdash (C,(Z_1 \to Z_2))$.

□

Now, given two sets of XFD, $\mathcal{F}_i$ and $\mathcal{F}_{\bar{i}}$, we recall the definition of the set $\mathcal{K}_i$ of XFD as:

$$\mathcal{K}_i = \{X \to A \mid X \to A \in \mathcal{F}_i^+ \text{ and } [((X \subseteq \mathbb{P}_i) \text{ and } (X \not\subseteq \mathbb{P}_{\bar{i}})) \text{ or } (A \in (\mathbb{P}_i \backslash \mathbb{P}_{\bar{i}})]\} \tag{21}$$

Intuitively, $\mathcal{K}_i$ contains all the XFD f which can be obtained from $\mathcal{F}_i$ but that cannot be violated by documents in $X_{\bar{i}}$ due to one of the two reasons:
(a) the right-hand side of f is a path B which belongs to $\mathbb{P}_i$ but not to $\mathbb{P}_{\bar{i}}$ or
(b) the left-hand side of f is a set of paths X which is included in $\mathbb{P}_i$ but not in $\mathbb{P}_{\bar{i}}$. In Section 7.1 we show an algorithm, starting with $\mathcal{F}_1^+$ and $\mathcal{F}_2^+$, instead of $\mathcal{F}_1$ and $\mathcal{F}_2$, that computes the set

$$\mathcal{F} = (\mathcal{F}_1^+ \cap \mathcal{F}_2^+) \cup \mathcal{K}_1 \cup \mathcal{K}_2 \tag{22}$$

and we prove some properties of $\mathcal{F}$.

Theorem 7. *The set cover$\mathcal{F}$, returned by Algorithm 3, is equivalent to (or is a cover of) the set of XFD $\mathcal{F} = (\mathcal{F}_1^+ \cap \mathcal{F}_2^+) \cup \mathcal{K}_1 \cup \mathcal{K}_2$ (cover$\mathcal{F} \equiv \mathcal{F}$).* □

Proof: We prove that: (A) Algorithm 3 terminates, and that (B) *cover*$\mathcal{F} \equiv \mathcal{F}$.

(A) For each XFD $h = (C,(Y \to A))$ which is considered in line 4 of Algorithm 3, we have that h is added to *cover*$\mathcal{F}$ due to tests in lines 5-10 or h is analysed and implies zero or several XFD added to G (lines 11-15).
Recall that the set $(C,Y)^+$ *w.r.t.* $\mathcal{F}_i$ (denoted by $(C,Y)^+_{\mathcal{F}_i}$) is finite and computed by adding, at each iteration k of Algorithm 1, a set of paths Z_k to $(C,Y)^+_{\mathcal{F}_i}$. The process goes on until no more inclusions are possible in $(C,Y)^+_{\mathcal{F}_i}$. The set of paths Z_k ($1 \leq k \leq n$) is obtained from successive calls of function *closure1Step.* More precisely, we have that $Z_n \subseteq closure1Step(C,Z_{n-1},\mathcal{F}_i)$, ..., $Z_2 \in closure1Step(C,Y,\mathcal{F}_i)$.
Functions $closure1Step(C,A,\mathcal{F}_i)$ and $inverseClosure1Step(C,Y,\mathcal{F}_i)$ just follow XFD forward or backward, "visiting" paths in $(C,Y)^+_{\mathcal{F}_i}$. Thus, as $(C,Y)^+_{\mathcal{F}_i}$ is finite the number of calls to $closure1Step(C,A,\mathcal{F}_i)$ and $inverseClosure1Step(C,Y,\mathcal{F}_i)$ are also finite.
Thus, Algorithm 3 terminates because the only possible situations are the following ones:

- A new XFD, added to G due to XFD h, is also added to $cover\mathcal{F}$.
- A new XFD, found in lines 12-16 from the XFD h, corresponds to an XFD already treated and, thus, is not added to G (recall that G is a set).
- XFD $h = (C, (Y \to A))$ is such that $inverseClosure1Step(C, Y, \mathcal{F}_i) = \emptyset$ and $closure1Step(C, A, \mathcal{F}_i) = \emptyset$, and then no XFD is added to G.

(B) For proving that $cover\mathcal{F} \equiv \mathcal{F}$, we will prove that:
$(B1)$ $\forall f \in cover\mathcal{F}$, $\mathcal{F} \vdash f$ and
$(B2)$ $\forall f \in \mathcal{F}$ we have that $cover\mathcal{F} \vdash f$.

$(B1)$ In fact, we can prove that $cover\mathcal{F} \subseteq \mathcal{F}$ (which is stronger than just proving that $\mathcal{F} \vdash f$ for any XFD $f \in cover\mathcal{F}$).
From Algorithm 3, two kinds of XFD are added to $cover\mathcal{F}$:

1. Those that are in $\mathcal{F}_1$ or $\mathcal{F}_2$ and succeed the tests in lines 5, 7 and 9. Clearly, according to (21) and (22), these XFD are also in $\mathcal{F}$.
2. Those derived from XFD in $\mathcal{F}_1$ or $\mathcal{F}_2$ (lines 12 to 16) and that also succeed the tests in lines 5, 7 and 9.
   To prove that these XFD are also in $\mathcal{F}$, let us consider an XFD $g = (C, (X \to B))$ in G for which lines 12 to 16 of Algorithm 3 are executed. From line 13 we obtain an XFD $g_1 = (C, (X \to D))$ such that $C/D \in (C, B)^+_{\mathcal{F}_i}$, which is in $\mathcal{F}_i^+$. From line 15 we obtain an XFD $g_2 = (C, (Y \to B))$ such that $C/X \in (C, Y)^+_{\mathcal{F}_i}$, which is in $\mathcal{F}_i^+$. From line 16 we obtain an XFD $g_3 = (C, (Y \to B))$ such that $C/B \in (C, Y)^+_{\mathcal{F}_i}$, which is in $\mathcal{F}_i^+$. Since these new XFD are in $\mathcal{F}_1^+$ or $\mathcal{F}_2^+$ and satisfy conditions stated in lines 5, 7 and 9 then they are also in $\mathcal{F}$ (see (21) and (22)).

We have just proved that $cover\mathcal{F} \subseteq \mathcal{F}$.

(B2) The final step is to prove that for each XFD $f \in \mathcal{F} = (\mathcal{F}_1^+ \cap \mathcal{F}_2^+) \cup \mathcal{K}_1 \cup \mathcal{K}_2$ then f is also in $cover\mathcal{F}^+$ (*i.e.*, $cover\mathcal{F} \vdash f$).

1. Let $(C, (Y \to A))$ be an XFD in $\mathcal{F}_1^+ \cap \mathcal{F}_2^+$. Thus, we know that $(C/Y \cup \{C/A\}) \subseteq \mathbb{P}_1$, $(C/Y \cup \{C/A\}) \subseteq \mathbb{P}_2$, $C/A \in (C, Y)^+_{\mathcal{F}_1}$ and $C/A \in (C, Y)^+_{\mathcal{F}_2}$.
   (a) If $(C, (Y \to A)) \in \mathcal{F}_1$ then, as $C/A \in (C, Y)^+_{\mathcal{F}_2}$, from line 9 of Algorithm 3, we have $(C, (Y \to A)) \in cover\mathcal{F}$. Clearly, $(C, (Y \to A)) \in cover\mathcal{F}^+$.
   (b) Otherwise, if $(C, (Y \to A)) \in \mathcal{F}_2$, with the same arguments as those in 1a, we conclude that $(C, (Y \to A)) \in cover\mathcal{F}$ and so $(C, (Y \to A)) \in cover\mathcal{F}^+$.
   (c) Otherwise, we have that $f = (C, (Y \to A)) \notin \mathcal{F}_1$ and $(C, (Y \to A)) \notin \mathcal{F}_2$. As $C/A \in (C, Y)^+_{\mathcal{F}_1}$, there is a derivation sequence $\alpha = f_1, \ldots, f_n$ which derives f.
   Firstly, assume that $cover\mathcal{F}$ contains all the XFD of $\mathcal{F}_1$ taking part in the derivation sequence α. In this case, it is straightforward that $cover\mathcal{F} \vdash f$.

Now assume the contrary, *i.e.*, there is at least one XFD of $\mathcal{F}_1$ that takes part in the derivation sequence α but does not belong to $cover\mathcal{F}$. Denote it by f_k. We know that f_k does not satisfy conditions on lines 5, 7, 9 and, thus, is considered in lines 12-16. From lines 12-16, we know that f_k is deleted from G and that some other XFD h is inserted in G. From Lemma 7, we know that $G \vdash f$.

All new functional dependencies h are going to be considered in line 4. If they satisfy conditions in lines 5, 7, 9 they are added to $cover\mathcal{F}$. Otherwise, they are analysed in lines 12-16 and the process goes on until f is added to G and, thus, to $cover\mathcal{F}$.

To see why f is eventually added to G, recall the following facts. We know that $f = (C, (Y \to A))$ and that $f \in \mathcal{F}_1^+$. We also know that new XFD h which is not inserted in $cover\mathcal{F}$, provokes the insertion in G of XDF having the form $(C, (Z \to W))$ where $C/Z \subseteq (C, Y)^+$ and $C/W \subseteq (C, Y)^+$ *w.r.t.* $\mathcal{F}_1$. This is true because when computing $(C, Y)^+$ *w.r.t.* $\mathcal{F}_1$ we find paths C/Z and C/W which are included in $(C, Y)^+$ during a step of Algorithm 1. More precisely, there are sets of paths $C/Z_1 \ldots C/Z_n \in (C, Y)^+$ such that $C/W \subseteq closure1Step(C, Z_n, \mathcal{F}_1)$, ..., $Z_2 \subseteq closure1Step(C, Z_1, \mathcal{F}_1), Z_1 \subseteq closure1Step(C, Z, \mathcal{F}_1)$. As f is one of the XFD $(C, (Z \to W))$ described above, it will eventually be added to G, and selected in line 9 to be in $cover\mathcal{F}$ (recall that $C/A \in (C, Y)^+_{\mathcal{F}_1}$ and $C/A \in (C, Y)^+_{\mathcal{F}_2}$).

2. Let $f = (C, (Y \to A)) \in \mathcal{K}_1$. In this situation $f \in \mathcal{F}_1^+$, $(C/Y \cup \{C/A\}) \subseteq \mathbb{P}_1$, and $(C/Y \cup \{C/A\}) \not\subseteq \mathbb{P}_2$. We prove that $f \in cover\mathcal{F}^+$. As $C/A \in (C, Y)^+_{\mathcal{F}_1}$, there is a sequence $\alpha = f_1, \ldots, f_n$ corresponding to a derivation of f. With the same arguments as those in 1c, we know that $cover\mathcal{F}$ contains all XFD which are needed to the derivation of f or f is added to G. Then f is considered in line 4 and f is added to $cover\mathcal{F}$ in line 6 or 8 because $(C/Y \cup \{C/A\}) \subseteq \mathbb{P}_1$, and $(C/Y \cup \{C/A\}) \not\subseteq \mathbb{P}_2$.
3. Let $f = (C, (Y \to A)) \in \mathcal{K}_2$. The proof that $f \in cover\mathcal{F}^+$, is similar to case 2.

□

7.5 Complexity of Algorithm 3

Algorithm 3 depends on Algorithm 1 which computes the closure of a set of paths $((C, X)^+)$, and on Algorithm 4 which computes just one step of the inverse closure of a set of paths. In Algorithm 1, we notice that:

- The loop at line 4 is executed at most $|\mathbb{P}|$ times when we consider that at each iteration just one path in $\mathbb{P}$ is added to $X^{(i)}$.
- Checking that $X^{(i)} \neq X^{(i+1)}$ has a complexity, in the worst case, of $O(|\mathbb{P}|^2)$ when $X^{(i)} = X^{(i+1)} = \mathbb{P}$.
- For computing the set Y in line 5, we consider all XFD in $\mathcal{F}$ and check if one of the conditions (a)-(c) is satisfied for each XFD. The complexity of this

part is $O(|f|.|\mathbb{P}|.|\mathcal{F}|)$ where $|\mathcal{F}|$ is the cardinality of $\mathcal{F}$ and $|f|$ is the size of the longest XFD in $\mathcal{F}$.
– For computing the prefixes and attributes in lines 6 and 7, the runtime is $O(|\mathbb{P}|)$.

Thus, the time complexity of Algorithm 1 is $O(|\mathbb{P}|.(|f|.|\mathbb{P}|.|\mathcal{F}| + |\mathbb{P}|^2 + |\mathbb{P}|))$ or, factorising, $O(|\mathbb{P}|^2.(|f|.|\mathcal{F}|+|\mathbb{P}|+1))$. Therefore the running time of Algorithm 1, in the worst (unlikely) case, is $O(|\mathbb{P}|^2.(|f|.|\mathcal{F}| + |\mathbb{P}|))$.

The running time of Algorithm 4 is based on the following remarks. At each iteration, we compute S on the basis the two sets Y_1 and Y_2 which contain sets of path.

– The computation of Y_1 is done in time $O \mid \mathcal{F} \mid$.
– To compute Y_2 we need to consider all XFD in $\mathcal{F}$ and, for each $f \in \mathcal{F}$ having P on its right-hand side, we compute the prefixes of all paths on its left-hand side. Then the computation of Y_2 is done in time $O \mid \mathcal{F} \mid .(n.m + m^n)$, where m is the number of labels on the longest path in f and n is the number of path on the left-hand side of f. In fact, we know that:
  (i) Prefixes of a path Q are obtained in time $O(m)$. We have to compute prefixes for n paths.
  (ii) The new sets of path in Y_2 are obtained by combining each prefix of a path P_1 (on the left-hand side of f) with a prefix of paths $P_2 \dots P_n$. This computation is done in time $O(m^n)$.
– The number of sets of path in R is m; in Y_1 is $\mid \mathcal{F} \mid$ and in Y_2 is $m^n \times \mid \mathcal{F} \mid$. At each iteration we compute the distributing union of two sets S_1 and S_2 which complexity is $O(|S_1|.|S_2|)$. As we have $\mid X \mid$ iterations the computation of S is done in time $O((m + \mid \mathcal{F} \mid + m^n \times \mid \mathcal{F} \mid)^{|X|})$.

In the worst case, Algorithm 3 will treat about $|\mathcal{F}_i|.|\mathbb{P}_i|$ XFD for each set $\mathcal{F}_i$. The worst case occurs when for each XFD $f = (C, (X \to P))$ in $\mathcal{F}_i$, $(C, X)^+$ contains $|\mathbb{P}_i|$ paths and just one path is added to $(C, X)^+$ in each step of the loop of Algorithm 1 and no XFD is added to *cover*$\mathcal{F}$. Hence, in this case, lines 12-15 of Algorithm 3 will be executed $|\mathbb{P}_i|$ times for each XFD in $\mathcal{F}_i$. The complexity of Algorithm 3 is $O(|\mathcal{F}_i|.|\mathbb{P}_i|.(g + h))$ where g is the complexity of Algorithm 1 and h is the complexity of Algorithm 4.

8 Experimental Results

In order to examine the performances of Algorithm 3, we run several experiments on synthetic data. Recall that Algorithm 3 has as input two local systems $S_1 = (\mathcal{D}_1, \mathcal{F}_1, O_1)$ and $S_2 = (\mathcal{D}_2, \mathcal{F}_2, O_2)$, and computes the set *cover*$\mathcal{F}$ which contains only the XFD for which no violation is possible when considering document sets for S_1 and S_2. We take into account two parameters in the experiments: (i) the number of paths obtained from $\mathcal{D}_1$ and $\mathcal{D}_2$, and (ii) the number of XFD in $\mid \mathcal{F}_1 \mid + \mid \mathcal{F}_2 \mid$.

Tree $\mathcal{T}$ (Figure 16) guides the way we perform our experiments. $\mathcal{T}$ is built by repeating the pattern tree in Figure 15 several times. To perceive the difference between the sub-trees of $\mathcal{T}$, we relabel the nodes of the pattern tree by adding the index k ($k \geq 1$). Thus, we say *sub-tree k* to refer to the kth tree pattern in $\mathcal{T}$.

Our experiments consist in generating *cover*$\mathcal{F}$ from sets $\mathcal{F}_1$ and $\mathcal{F}_2$ which increase at each test by assuming the existence of bigger sets of paths $\mathbb{P}_1$ and $\mathbb{P}_2$ and, therefore, larger trees $\mathcal{T}$. In the text, we usually refer to tree $\mathcal{T}$ to indicate the type of documents (the schema) we are dealing with. In this context, let us define $\mathbb{P}_1^j$ as the set of paths containing all the paths in the tree $\mathcal{T}$ except the paths $C/R_{1,k}/G_{1,k}$ (with $k \leq j$), and $\mathbb{P}_2^j$ as the set of paths containing all the paths in the tree $\mathcal{T}$ except the paths $C/R_{1,k}/F_{1,k}$ (with $k \leq j$). We suppose that the set of paths $\mathbb{P}_1^j$ (respectively $\mathbb{P}_2^j$) is generated from $\mathcal{D}_1$ (respectively $\mathcal{D}_2$).

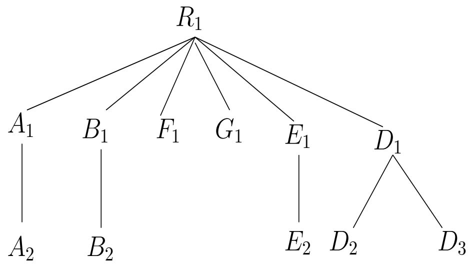

Fig. 15. Pattern tree

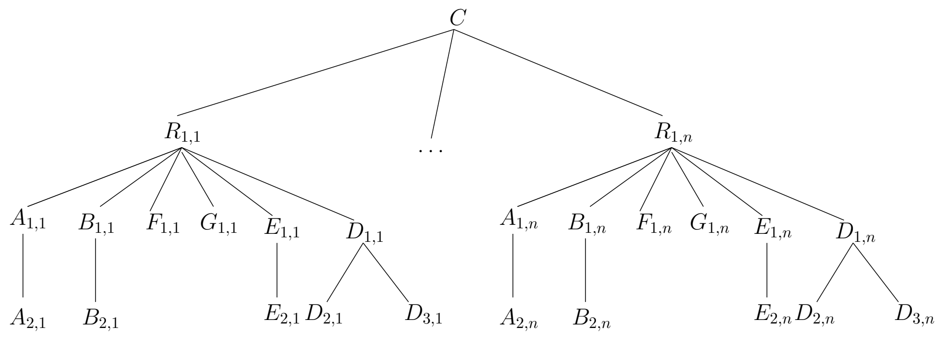

Fig. 16. Tree $\mathcal{T}$ built by repeating, under the same root, n times the pattern tree in Figure 15.

The set of XFD $\mathcal{F}_1^j$ (respectively $\mathcal{F}_2^j$) is defined over paths in $\mathbb{P}_1^j$ (respectively $\mathbb{P}_2^j$). Table 5 shows the XFD in $\mathcal{F}_1^j$ and $\mathcal{F}_2^j$. Sets $\mathcal{F}_1^j$ and $\mathcal{F}_2^j$ contain both XFD (1) and (2). However XFD (3a), (4a) and (5a) are only in $\mathcal{F}_1^j$ and XFD (3b), (4b)

and ($5b$) are only in $\mathcal{F}_2^j$. With XFD ($4a$) and ($5a$), we can derive the XFD (6) $(C/R_{1,k}, (\{A_{1,k}/A_{2,k}, B_{1,k}/B_{2,k}\} \rightarrow E_{1,k}/E_{2,k}))$ and with XFD ($4b$) and ($5b$), we can can also derive the XFD (6). Hence, $\mathcal{F}_1^j$ and $\mathcal{F}_2^j$ derive XFD (6) but by different ways. We can remark that $|\mathcal{F}_1^1| = |\mathcal{F}_2^1| = 5$, $|\mathcal{F}_1^2| = |\mathcal{F}_2^2| = 10$ and $\mathcal{F}_1^1 \subset \mathcal{F}_1^2$, $\mathcal{F}_2^1 \subset \mathcal{F}_2^2$.

$\mathcal{F}_1^j$	$\mathcal{F}_2^j$
(1) $(C/R_{1,k}, (\{A_{1,k}, B_{1,k}\} \rightarrow D_{1,k}))$	(1) $(C/R_{1,k}, (\{A_{1,k}, B_{1,k}\} \rightarrow D_{1,k}))$
(2) $(C/R_{1,k}, (\{D_{1,k}\} \rightarrow E_{1,k}))$	(2) $(C/R_{1,k}, (\{D_{1,k}\} \rightarrow E_{1,k}))$
($3a$) $(C/R_{1,k}, (\{E_{1,k}\} \rightarrow F_{1,k}))$	($3b$) $(C/R_{1,k}, (\{E_{1,k}\} \rightarrow G_{1,k}))$
($4a$) $(C/R_{1,k}, (\{A_{1,k}/A_{2,k}, B_{1,k}/B_{2,k}\} \rightarrow D_{1,k}/D_{2,k}))$	($4b$) $(C/R_{1,k}, (\{A_{1,k}/A_{2,k}, B_{1,k}/B_{2,k}\} \rightarrow D_{1,k}/D_{3,k}))$
($5a$) $(C/R_{1,k}, (\{D_{1,k}/D_{2,k}\} \rightarrow E_{1,k}/E_{2,k}))$	($5b$) $(C/R_{1,k}, (\{D_{1,k}/D_{3,k}\} \rightarrow E_{1,k}/E_{2,k}))$

Table 5. Contents of the XFD set $\mathcal{F}_1^j$ and $\mathcal{F}_2^j$ using in the experiments

The algorithm was implemented in Java and the tests have been done on an Intel Quad Core i3-2310M with 2.10GHz and 8GB of memory. We have used three scenarios for performing our tests.

In the first scenario we examine the influence of the size of $\mathcal{F}_1$ and $\mathcal{F}_2$ on the execution time of Algorithm 3. We have used $\mathcal{F}_1^j$ and $\mathcal{F}_2^j$, such that $1 \leq j \leq 45$. Figure 17 shows reasonable execution time (approximately 2 minutes) for computing *cover*$\mathcal{F}$ from sets of XFD $\mathcal{F}_1$ and $\mathcal{F}_2$ where $|\mathcal{F}_1| + |\mathcal{F}_2| = 450$. Figure 17 also shows that the size of *cover*$\mathcal{F}$ increases slightly.

In the second scenario we examine again the influence of the size of $\mathcal{F}_1$ and $\mathcal{F}_2$ on the execution time of Algorithm 3. Notice that, in the first scenario, the functional dependencies involving index k concerns only one subtree. Now, in this second scenario, we allow an XFD involving index $k = 1$ to derive an XFD involving index $k = 2$, and so on. To do this, we add to $\mathcal{F}_1^j$ (respectively $\mathcal{F}_2^j$) the XFD of the form ($7a$) $(C, (\{R_{1,k}/E_{1,k-1}/E_{2,k-1}\} \rightarrow R_{1,k}/D_{1,k}/D_{2,k}))$, respectively ($7b$) $(C, (\{R_{1,k}/E_{1,k-1}/E_{2,k-1}\} \rightarrow R_{1,k}/D_{1,k}/D_{3,k}))$, with $2 \leq k \leq j$.

As shown in Figure 18, the execution time for computing *cover*$\mathcal{F}$ is more important than the one obtained with the first scenario. For instance, for sets $\mathcal{F}_1$ and $\mathcal{F}_2$ (such that $|\mathcal{F}_1|+|\mathcal{F}_2| = 262$) we need 53 minutes to compute *cover*$\mathcal{F}$. This behaviour is explained by two facts:

- XFD of the form ($7a$) and ($7b$) are not added to *cover*$\mathcal{F}$ due to condition in line 9 of Algorithm 3. Checking this condition is an expensive task because the computation of $(C, R_{1,k}/E_{1,k-1}/E_{2,k-1})^+_{\mathcal{F}_i}$ involves many paths.
- For this example, lines 12-15 of Algorithm 3 generate many XFD increasing dramatically the number of XFD in *cover*$\mathcal{F}$. Indeed, $|cover\mathcal{F}|$ has about 10610 XFD when $|\mathcal{F}_1^j| + |\mathcal{F}_2^j|$ is 262.

In the third scenario, we compare Algorithm 2 (which computes $\mathcal{F}$) with Algorithm 3 (which computes *cover*$\mathcal{F}$). Recall that in Section 7, we have shown

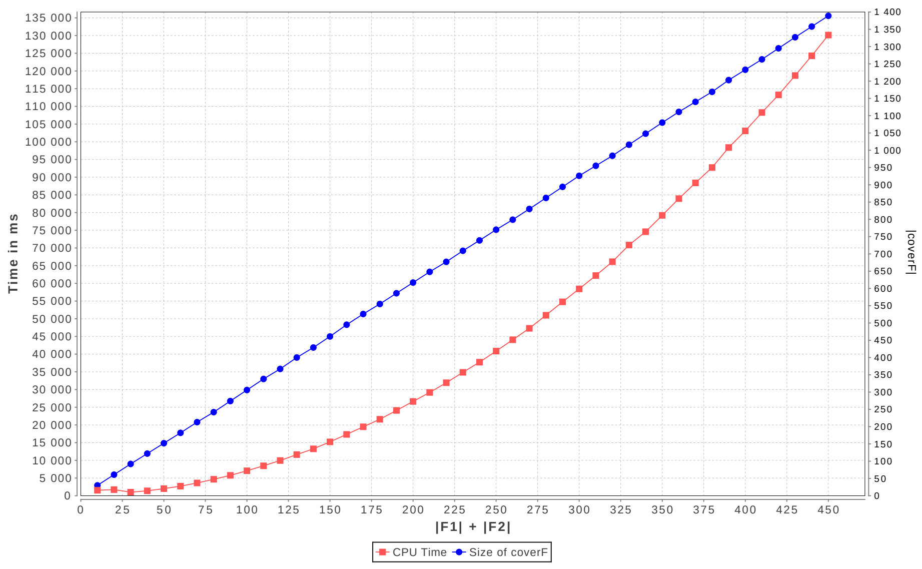

Fig. 17. Scenario 1: CPU time for the computing of $cover\mathcal{F}$ and the evolution of its size

that $cover\mathcal{F}$ is equivalent to $\mathcal{F}$. Now, Table 6 compares these two algorithms. Line 1 in Table 6 shows the results with sets $\mathcal{F}_1^1$ and $\mathcal{F}_2^1$ while line 4 shows the result with $\mathcal{F}_1^2$ and $\mathcal{F}_2^2$, and line 5 shows the result with $\mathcal{F}_1^3$ and $\mathcal{F}_2^3$, the same sets used in scenario 1. When computing the set $\mathcal{F}$ for sets $\mathcal{F}_1^3$ and $\mathcal{F}_2^3$ with Algorithm 2, we obtain an *out-of-memory* error after 5 minutes. For the same sets of XFD, Algorithm 3 takes approximately 2.9 seconds and $|cover\mathcal{F}| = 92$. Since test concerning line 5 does not produce a result for Algorithm 2, we perform tests of line 2 and 3 on a modified tree, *i.e.*, on $\mathcal{T}$ without the leaves. In other words, we delete nodes $A_{2,2}$, $B_{2,2}$, $D_{2,2}$, $D_{3,2}$ and $E_{2,2}$ from a tree $\mathcal{T}$ with $k = 2$ sub-trees. The tree considered in line 3 contains nodes $F_{1,2}$ and $G_{1,2}$ in addition to nodes in the tree considered in line 2. As expected, in all cases, Algorithm 3 is much more efficient than Algorithm 2. Moreover, the size of $\mathcal{F}$ grows dramatically while the size of $cover\mathcal{F}$ increases slightly.

	$\|\mathbb{P}_1\| + \|\mathbb{P}_2\|$	$\|\mathcal{F}_1\|$	$\|\mathcal{F}_2\|$	$\|\mathcal{F}_1^+\|$	$\|\mathcal{F}_2^+\|$	$\|\mathcal{F}\|$	t_1 (ms)	$\|cover\mathcal{F}\|$	t_2 (ms)
1	12	5	5	3 835	3 835	5 755	19 211	30	160
2	17	7	7	3 865	3 865	5 785	24 401	32	195
3	18	8	8	3 928	3 928	5 911	26 476	34	204
4	23	10	10	7 670	7 670	11 510	165 460	61	503
5	34	15	15	?	?	?	> 5min	92	2 949

Table 6. Comparison: t_1 is the time needed to compute $\mathcal{F}$ and t_2 is the time needed to compute $cover\mathcal{F}$.

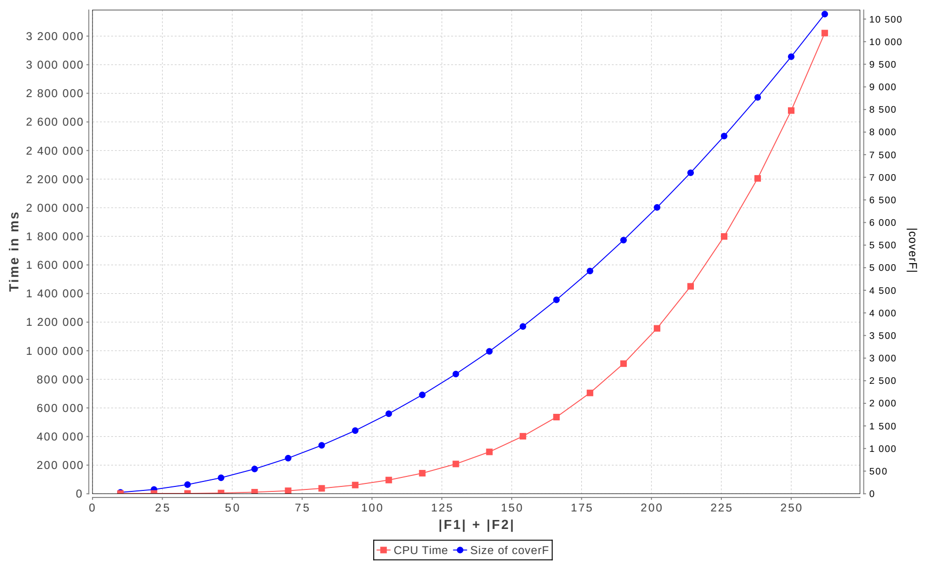

Fig. 18. Scenario 2: CPU time for the computing of *cover*$\mathcal{F}$ and the evolution of its size

Our experiments confirm the time complexity presented in Section 7.5, and reinforce the importance of computing the smaller set *cover*$\mathcal{F}$ instead of the equivalent set $\mathcal{F}$ considered in Section 7.1. The worst case happens when documents have many equivalent paths and derivations that involve a lot of paths. We have used Algorithm 1 to test, successfully, the equivalence between $\mathcal{F}$ and *cover*$\mathcal{F}$ on several examples. These tests contribute to the validation of the correction of our method.

9 Related Work

We suppose a system which is expected to receive XML documents from different sources, each one submitted to some local integrity constraints. The motivation of this paper comes from the idea of giving to this a system the capability of verifying all the original (local) constraints whose satisfaction can be ensured by any local valid documents. To achieve our goal we have seen the necessity of using an axiom system.

We decided to adopt the XFD definition presented in [2] for which a validation algorithm is introduced in [4]. Indeed, [4] offers a general algorithm to verify the satisfaction of different integrity constraints, including XFD. A complementary approach is presented in [8], where the idea of incremental validation is considered via some static verification of functional dependencies *w.r.t.* updates. In [8], XFD are defined as tree queries which augments considerably the complexity of an implementation, and the necessary dynamic part of the verification (*i.e.* considering values existing in the XML document) still has to be done for parts involved in the updates.

In this paper we propose an axiom system, for XFD as those in [2, 4], together with an efficient algorithm for computing the closure of a set of paths. Our work on the axiom system is comparable to the one proposed by [16]. The main differences are: (i) we propose a more powerful path language allowing the use of a wild-card; (ii) our XFD are verified *w.r.t.* a context and not only *w.r.t.* the root, *i.e.*, XFD can be relative; (iii) our XFD can be defined by taking into account two types of equality: value and node equality and (iv) we use simpler concepts (such as branching paths, projection) which, we believe, allow us to prove that our axiom system is sound and complete in a clearer way.

Several other proposals for defining XML functional dependencies (XFD) exist. We refer to [1, 9, 12, 15, 17, 16, 19] as some examples and to [9, 18] for a comparison among some of them. The path language used in [1, 17, 16] allows only the specification of simple paths. In [18, 4], we find a more powerful path language. All these approaches present paths as unary queries, but [9] whose path language is n-ary. Different XFD proposals entail different axiomatisation system, such as those in [9, 11, 16].

We use our axiom system in the development of a practical tool: to filter local XFD in order to obtain a set containing only XFD that cannot be violated by any local XML document. The goal of our global system is to deal with data coming from any local source, but not to perform data fusion. In this context, our work presents an original point of view, since we are not interested in putting together all the local information, but just in manipulating them. Indeed, our approach does not deal with data, only with the available constraints (XFD in our case).

Some work consider schema integration of local database into a global. Usually, this proposal comes together with the idea of data fusion. XML data fusion is considered in papers such as [5, 14]. In [14], the author considers the following problem: given an XFD $f : X \to A$, it can be violated when local data is put together - even when f is satisfied by each local data separately. He proposes a probabilistic approach to decide which data is the most reliable. In [7] data exchange is considered. A target schema is built from a source schema (a mapping is given). The goal of their proposal is to construct an instance over the target schema, based on the source and the mapping, and to answer queries against the target data in a way consistent with the source data.

In several modern applications the schema often changes as the information grows and different people have inherently different ways of modelling the same information. In the XML domain, one should easily imagine a situation where the schema is not agreed upon in advance but is designed incrementally, according to the needs of different users. Our global (integrated) system is also intended to be a consensus one, a starting point for constraint evolution when the demand for dealing with new and old information exists.

Different works have already considered XML type evolution, but these proposals usually do not take into account the evolution of associated integrity constraints whose role is extremely important in the maintenance of consistent information. In [10] authors offer as a perspective to apply to XML their proposal of adapting (or re-formulating) functional dependencies according to schema changes. This is done in [15] where authors consider the problem of constraint evolution in conformance with type evolution. The type evolution proposed in [6] is well adapted to our proposes; it seems possible to combine their type evolution with our XFD filter in order to generate a set of constraints allowing interoperability.

10 Conclusions

In this paper we were motivated by applications on a multiple system environment and we presented a method for establishing the biggest set of XFD that can be satisfied by any document conceived to respect local XFD. One important originality of our work is the fact that we do not deal with data, only with the available constraints (XFD in our case). Our approach is not only interesting for multiple system applications, but also in a conservative constraint evolution perspective. To reach our goals, a new axiom system, built for XFD defined over a context and two kinds of equality, was introduced. Moreover the paper presented the proof of its soundness and completeness and an efficient algorithm to compute path closures.

As some future directions that follow from this work, we mention:

– By using the schema evolution method of [6] together with our computation of $cover\mathcal{F}$, the generation of a new type and a new set of integrity constraints that will allow interoperability without abolishing constraint verification. We are currently working on a platform that puts together these tools.
– The extension of our method to other kinds of integrity constraints such as inclusion constraints.
– An incremental computation of $cover\mathcal{F}$, following the evolution of local constraints or systems.
– The detection of local XFD that are not selected in $cover\mathcal{F}$ but that could be included in it by correcting the associated documents that do not respect them.

References

1. Arenas, M., Libkin, L.: A normal form for XML documents. ACM Transactions on Database Systems (TODS) **29 No.1** (2004)
2. Bouchou, B., Cheriat, A., Halfeld Ferrari, M., Laurent, D., Lima, M.A., Musicante, M.: Efficient constraint validation for updated XML databases. Informatica **31**(3), 285–310 (2007)
3. Bouchou, B., Halfeld Ferrari, M., Lima, M.: Contraintes d'intégrité pour xml. visite guidée par une syntaxe homogène. Technique et Science Informatiques **28**(3), 331–364 (2009)

4. Bouchou, B., Halfeld Ferrari, M., Lima, M.A.V.: A grammarware for the incremental validation of integrity constraints on XML documents under multiple updates. Transactions on Large-Scale Data and Knowledge-Centered Systems, TLDKS Journal **6**(LNCS 7600) (2012)
5. Cecchin, F., de Aguiar Ciferri, C.D., Hara, C.S.: XML data fusion. In: DaWak, pp. 297–308 (2010)
6. Chabin, J., Halfeld Ferrari, M., Musicante, M.A., Réty, P.: Minimal tree language extensions: A keystone of XML type compatibility and evolution. In: ICTAC 2010, 7th International Colloquium of Theoretical Aspects of Computing, *Lecture Notes in Computer Science*, vol. 6255, pp. 60–75. Springer (2010)
7. Chirkova, R., Libkin, L., Reutter, J.L.: Tractable xml data exchange via relations. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, pp. 1629–1638 (2011)
8. Gire, F., Idabal, H.: Regular tree patterns: a uniform formalism for update queries and functional dependencies in XML. In: EDBT/ICDT Workshops (2010)
9. Hartmann, S., Trinh, T.: Axiomatising functional dependencies for XML with frequencies. In: FoIKS, pp. 159–178 (2006)
10. He, Q., Ling, T.W.: Extending and inferring functional dependencies in schema transformation. In: Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, pp. 12–21 (2004)
11. Kot, L., White, W.M.: Characterization of the interaction of XML functional dependencies with DTDs. In: ICDT- 11th International Conference on Database Theory, pp. 119–133 (2007)
12. Lee, M.L., Ling, T.W., Low, W.L.: Designing functional dependencies for XML. In: Extending Database Technology (EDBT), pp. 124–141 (2002). URL `citeseer.ist.psu.edu/article/lee02designing.html`
13. Liu, J., Vincent, M.W., Liu, C.: Functional dependencies, from relational to XML. In: Ershov Memorial Conference, pp. 531–538 (2003)
14. Pankowski, T.: Reconciling inconsistent data in probabilistic XML data integration. In: Sharing Data, Information and Knowledge, 25th British National Conference on Databases - BNCOD, pp. 75–86 (2008)
15. Shahriar, M.S., Liu, J.: Preserving functional dependency in XML data transformation. In: ADBIS '08: Proceedings of the 12th East European conference on Advances in Databases and Information Systems, pp. 262–278. Springer-Verlag (2008)
16. Vincent, M., Liu, J., Mohania, M.: The implication problem for 'closest node' functional dependencies in complete XML documents. Journal of Computer and System Sciences **78**(4), 1045 – 1098 (2012)
17. Vincent, M.W., Liu, J., Liu, C.: Strong functional dependencies and their application to normal forms in XML. ACM Trans. Database Syst. **29**(3), 445–462 (2004)
18. Wang, J., Topor, R.: Removing XML data redundancies using functional and equality-generating dependencies. In: ADC '05: Proceedings of the 16th Australasian database conference, pp. 65–74. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2005)
19. Zhao, X., Xin, J., Zhang, E.: XML functional dependency and schema normalization. In: HIS '09: Proceedings of the 9th International Conference on Hybrid Intelligent Systems, pp. 307–312 (2009)

A Algorithm for computing the longest common prefix

Given A_P the FSA associated to path P and A_R the FSA associated to path R, Algorithm 5 computes the FSA $A_{P \cap R}$ associated to the longest common prefix of P and R. We suppose that automata A_P and A_R are deterministic.

Let $A_P = (Q_P, \Sigma, I_P, \delta_P, F_P)$ and $A_R = (Q_R, \Sigma, I_R, \delta_R, F_R)$. Algorithm 5 is similar to the intersection algorithm of two FSA. In this way it is explained by the following items:

– Each state of the resulting automaton $A_{P \cap R}$ is *pair-state* (p, q) such that $p \in Q_P$ and $q \in Q_R$.
– The initial *pair-states* in $A_{P \cap R}$ comes from the set $I_P \times I_R$.
– A *pair-state* (p', q') is added to the set of *pair-states* Q when there exist $(p, q) \in Q$ and $a \in \Sigma$ such that $\delta_P(p, a) = p'$ and $\delta_R(q, a) = q'$ (in other words (p', q') is reached from (p, q)).
– The only difference between Algorithm 5 and the intersection algorithm is the way of computing the set of final *pair-states*. In the intersection algorithm a *pair-state* (p, q) is final when $p \in F_P$ and $q \in F_R$. In Algorithm 5, a *pair-state* (p, q) is final when $p \in F_P$ or $q \in F_R$, or there exist $a, b \in \Sigma$ such that $\delta_P(p, a)$ and $\delta_R(q, b)$ are defined and $a \neq b$.

Example 14. Consider the XML document of Figure 3 and the set $\mathbb{P} = L(D)$ of Figure 4. Given $P =$ */library//section/title/data* and $R =$ */library//section/txt/data*, the longest common prefix $P \cap R$ can be written as */library//section* since $P \cap R = \{$ */library/book/chapter/section/, /library/book/chapter/section/section/, /library/book/chapter/section/section/section* $\}$. In Figure 19 we have the FSA associated to paths P and R, and the FSA computed by Algorithm 5 for $P \cap R$.

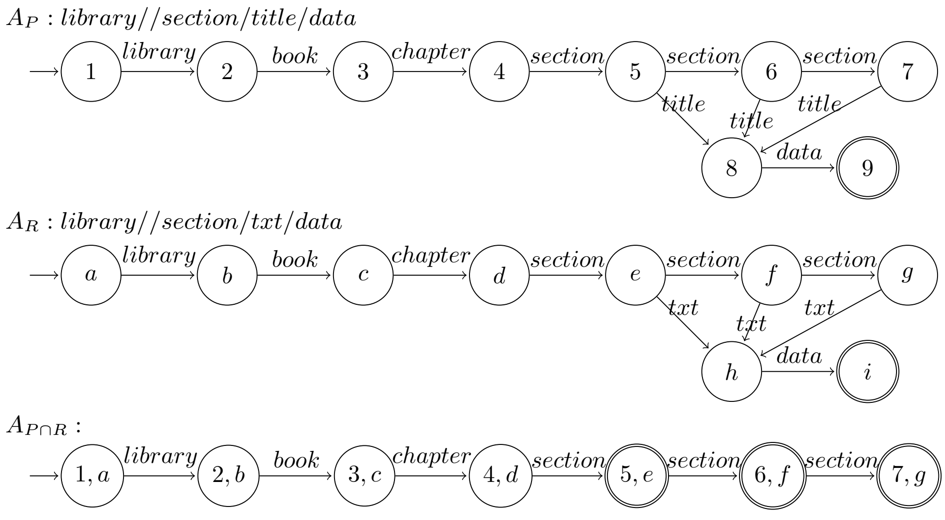

Fig. 19. FSA associated to the paths P, R and $P \cap R$.

Algorithm 5 Longest Common Prefix Algorithm of two Paths

Input:

- $A_P = (Q_P, \Sigma, I_P, \delta_P, F_P)$, the FSA associated to path P
- $A_R = (Q_R, \Sigma, I_R, \delta_R, F_R)$, the FSA associated to path R

Output: $A_{P \cap R}$, the FSA associated to path $P \cap R$ (the longest common prefix of P and R)

```
 1: $Q = \emptyset$                % *initialization of the set of pair-states*
 2: $I = I_P \times I_R$           % *initialization of the set of initial pair-states*
 3: $F = \emptyset$                % *initialization of the set of final pair-states*
 4: $worklist = \emptyset$         % *list for storing the states which are not yet treated*
 5: $worklistprec = \emptyset$ % *list for storing the states which are already treated*
 6: **for each** $(p, q) \in I$ **do**
 7:     $worklist.add((p, q))$
 8: **end for**
 9: **while** $worklist.size() > 0$ **do**
10:     $(p, q) = worklist.removeFirst()$
11:     $worklistprec.add((p, q))$ % $(p, q)$ *is marked as treated*
12:     $Q = Q \cup \{(p, q)\}$
13:     % *computing of new pair-states*
14:     **for each** $a \in \Sigma$ such that $\delta_P(p, a)$ and $\delta_R(q, a)$ are defined **do**
15:         $\delta((p, q), a) = (\delta_P(p, a), \delta_R(q, a))$
16:         **if** $(\delta_P(p, a), \delta_R(q, a)) \notin worklistprec$ **then**
17:             $worklist.add((\delta_P(p, a), \delta_R(q, a)))$
18:         **end if**
19:     **end for**
20:     % *checking if* $(p, q)$ *is final pair-state*
21:     **if** $p \in F_P$ or $q \in F_R$ **then**
22:         $F = F \cup \{(p, q)\}$
23:     **else if** $\exists\, a, b \in \Sigma$ such that $\delta_P(p, a)$ and $\delta_R(q, b)$ are defined and $a \neq b$ **then**
24:         $F = F \cup \{(p, q)\}$
25:     **end if**
26: **end while**
27: **return** $(Q, \Sigma, I, \delta, F)$
```