Online Appendix to A Normal Form for XML Documents

MARCELO ARENAS and LEONID LIBKIN
University of Toronto, Toronto, Ontario, Canada

A. PROOF OF SECTION 7

A DTD D can be inconsistent in the sense that there is no XML tree T such that $T \models D$. For example, a recursive DTD containing a rule $P(a) = a$ is not consistent; there is no a finite XML tree satisfying this rule. In this section we only consider consistent DTDs, since the implication problem for inconsistent DTDs is trivial and it can be checked in linear time whether a DTD is consistent [Fan and Libkin 2001].

A.1 Proof of Theorem 7.1

To prove this theorem we start by introducing some terminology. Given a simple DTD $D = (E, A, P, R, r)$ and $p, p' \in paths(D)$ such that p is a proper prefix of p', we say that p' *can be nullified from* p if p' is of the form $p.w_1.\ldots.w_n$, where $w_i \in E \cup A \cup \{\mathrm{S}\}$ ($i \in [1, n]$) and either (1) $P(last(p))$ contains $w_1?$ or w_1^*; or (2) there is $i \in [1, n-1]$ such that $P(w_i)$ contains $w_{i+1}?$ or w_{i+1}^*. Intuitively, p' can be nullified from p if there exists and XML tree T conforming to D and a tree tuple t in T such that $t.p \neq \perp$ and $t.p' = \perp$. For example, if $P(r) = a$, $P(a) = b^*$ and $P(b) = c$, then $r.a.b.c$ can be nullified from r and $r.a$, but it cannot be nullified from $r.a.b$. Given $S \subseteq paths(D)$, we say that p' *can be nullified from* S if p' can be nullified from p, where p is the longest common prefix of p' and a path from S.

The following is proved by the same argument as Lemma A.6 shown in electronic Appendix A.3.

The authors were supported in part by grants from the Natural Sciences and Engineering Research Council of Canada and from Bell University Laboratories.
Authors' addresses: M. Arenas, Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 3G4; email: marenas@cs.toronto.edu; L. Libkin, Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Ontario, Canada M5S 3H5; email: libkin@cs.toronto.edu.

LEMMA A.1. *Given a simple DTD D, a set Σ of functional dependencies over D and $S \cup \{p\} \subseteq paths(D)$, $(D, \Sigma) \not\models S \rightarrow p$ if and only if there is an XML tree T and a path q prefix of p such that $T \models (D, \Sigma)$, $tuples_D(T) = \{t_1, t_2\}$, $t_1.S = t_2.S$, $t_1.S \neq \perp$, $t_1.p \neq t_2.p$, $t_1.p \neq \perp$, $t_2.p \neq \perp$, $t_1.q \neq t_2.q$ and*

—*For each $s \in paths(D)$, if s can be nullified from $S \cup \{p\}$, then $t_1.s = t_2.s = \perp$.*
—*For each $s \in paths(D)$, if q is not a prefix of s and s cannot be nullified from $S \cup \{p\}$, then $t_1.s = t_2.s$ and $t_1.s \neq \perp$.*

To prove that the implication problem for simple DTDs can be solved in polynomial time, we use the technique of [Sagiv et al. 1981] and code constraints with propositional formulas. That is, for each simple DTD D and set of functional dependencies $\Sigma \cup \{S \rightarrow p\}$ over D, we will define a propositional formula φ such that $(D, \Sigma) \not\models S \rightarrow p$ if and only if φ is satisfiable. This formula will be of the form $\varphi_1 \vee \cdots \vee \varphi_n$, where each φ_i ($i \in [1, n]$) is a conjunction of Horn clauses. Given that the consistency problem for Horn clauses is solvable in linear time, we will conclude that our problem is solvable in quadratic time.

Let D be a DTD, Σ a set of functional dependencies over D and $S \cup \{p\} \subseteq paths(D)$. Recall that we assumed that each constraints in Σ is of the form $S' \rightarrow p'$, where $S' \cup \{p'\} \subseteq paths(D)$. We define $paths(\Sigma)$ as $\{s \mid$ there is $S' \rightarrow p' \in \Sigma$ such that $s \in S' \cup \{p'\}\}$. To define the propositional formula φ we view each path $s \in paths(\Sigma) \cup S \cup \{p\}$ as a propositional variable. Furthermore, for each path q which is a prefix of p we define a propositional formula φ_q as

$$\neg p \wedge \left(\bigwedge_{s \in P_q \cup S} s \right) \wedge \left(\bigwedge_{s \in N_q} \neg s \right) \wedge \bigwedge_{\psi \in \Gamma} \psi,$$

where P_q, N_q and Γ are set of propositional variables and formulas defined as follows.

—For each $s \in paths(\Sigma)$ such that s cannot be nullified from $S \cup \{p\}$ and q is not a prefix of s, s is included in P_q.
—For each $s \in paths(\Sigma)$ such that $s \in EPaths(D)$, s cannot be nullified from $S \cup \{p\}$ and q is a prefix of s, s is included in N_q.
—For each $S' \rightarrow p' \in \Sigma$, if there is no $q' \in S' \cup \{p'\}$ such that q' can be nullified from $S \cup \{p\}$, then $(\bigwedge_{s \in S'} s) \rightarrow p'$ is included in Γ

We note that φ_q is a conjunction of Horn clauses.

The propositional formula φ is defined as the disjunction of some of the formula φ_qs. The following lemma shows that in this disjunction we only need to consider qs such that $q = q'.\tau$, for some $\tau \in E$, and $P(last(q'))$ contains τ^* or τ^+.

LEMMA A.2. *Let $D = (E, A, P, R, r)$ be a simple DTD, Σ a set of functional dependencies over D and $S \cup \{p, q\} \subseteq paths(D)$ such that q is a prefix of p. If there is $\tau \in E$ such that $q = q'.\tau$ and $P(last(q'))$ contains τ^* or τ^+, then φ_q is satisfiable iff there is an XML tree T such that $T \models (D, \Sigma)$, $tuples_D(T) = \{t_1, t_2\}$, $t_1.S = t_2.S$, $t_1.S \neq \perp$, $t_1.p \neq t_2.p$, $t_1.p \neq \perp$, $t_2.p \neq \perp$, $t_1.q \neq t_2.q$ and*

—*For each* $s \in paths(D)$*, if* s *can be nullified from* $S \cup \{p\}$*, then* $t_1.s = t_2.s = \perp$.
—*For each* $s \in paths(D)$*, if* q *is not a prefix of* s *and* s *cannot be nullified from* $S \cup \{p\}$*, then* $t_1.s = t_2.s$ *and* $t_1.s \neq \perp$.

PROOF. ($\Rightarrow$) Let σ be a truth assignment satisfying φ_q. We define tuples t_1 and t_2 as follows. For each $s \in paths(D)$, if s can be nullified from $S \cup \{p\}$, then $t_1.s = t_2.s = \perp$. If s cannot be nullified from $S \cup \{p\}$ we consider two cases. If q is not a prefix of s, then $t_1.s = t_2.s$ and $t_1.s \neq \perp$. Otherwise, if $\sigma(s) = 1$, then $t_1.s = t_2.s$ and $t_1.s \neq \perp$, else $t_1.s \neq t_2.s$, $t_1.s \neq \perp$ and $t_2.s \neq \perp$.

It is straightforward to prove that there is an XML tree $T \in trees_D(\{t_1, t_2\})$ such that $T \models D$ and $tuples_D(T) = \{t_1, t_2\}$. Given that $\sigma \models \neg p \wedge \bigwedge_{s \in S} s$, $t_1.S = t_2.S$, $t_1.S \neq \perp$, $t_1.p \neq t_2.p$, $t_1.p \neq \perp$ and $t_2.p \neq \perp$. Besides, $t_1.q \neq t_2.q$, since $q \in N_q$ and $\sigma \models \bigwedge_{s \in N_q} \neg s$. Thus, to finish the proof we have to show that $T \models \Sigma$. Let $S' \rightarrow p' \in \Sigma$. If there is $q' \in S' \cup \{p'\}$ such that q' can be nullified from $S \cup \{p\}$, then T trivially satisfies $S' \rightarrow p'$ since $t_1.q' = t_2.q' = \perp$. Otherwise, suppose that $t_1.S' = t_2.S'$ and $t_1.S' \neq \perp$. Then, by considering that $\sigma \models \bigwedge_{s \in P_q} s$ and the definition of t_1 and t_2, we conclude that $\sigma \models \bigwedge_{s \in S'} s$. Thus, given that $\sigma \models (\bigwedge_{s \in S'} s) \rightarrow p'$, we conclude that $\sigma(p') = 1$, and, therefore, $t_1.p' = t_2.p'$.

($\Leftarrow$) Suppose that there is an XML tree T satisfying the conditions of the lemma. Define a truth assignment σ as follows. For each $s \in paths(\Sigma) \cup S \cup \{p\}$, if $t_1.s \neq t_2.s$ then $\sigma(s) = 0$. Otherwise, $\sigma(s) = 1$.

Given that $t_1.p \neq t_2.p$ and $t_1.S = t_2.S$, $\sigma(\neg p) = 1$ and $\sigma \models \bigwedge_{s \in S} s$. Let $s \in P_q$. By definition, s cannot be nullified from $S \cup \{p\}$ and q is not a prefix of s, and, therefore, $t_1.s = t_2.s$. Thus, $\sigma(s) = 1$. We conclude that $\sigma \models \bigwedge_{s \in P_q} s$. Let $s \in N_q$. By definition, s cannot be nullified from $S \cup \{p\}$, q is a prefix of s and $s \in EPaths(D)$. Hence, $t_1.s \neq t_2.s$ and $\sigma(s) = 0$. We conclude that $\sigma \models \bigwedge_{s \in N_q} \neg s$. Finally, let $(\bigwedge_{s \in S'} s) \rightarrow p' \in \Sigma_q$. If $\sigma \models \bigwedge_{s \in S'} s$, then by definition of σ and Σ_q, we conclude that $t_1.S' = t_2.S'$ and $t_1.S' \neq \perp$. Thus, given that $T \models \Sigma$, we conclude that $t_1.p' = t_2.p'$ and, therefore, $\sigma(p') = 1$. □

Combining Lemmas A.1 and A.2 we obtain:

LEMMA A.3. *Let* $D = (E, A, P, R, r)$ *be a simple DTD,* Σ *a set of functional dependencies over* D *and* $S \cup \{p\} \subseteq paths(D)$*. Assume that* $X = \{q \in paths(D) \mid q$ *is a prefix of* p *and there is* $\tau \in E$ *such that* $q = q'.\tau$ *and* $P(last(q'))$ *contains* τ^* *or* $\tau^+\}$*. Then,* $(D, \Sigma) \not\vdash S \rightarrow p$ *iff* $\varphi = \bigvee_{q \in X} \varphi_q$ *is satisfiable.*

Finally, we are ready to show that for a simple DTD D and a set of FDs $\Sigma \cup \{S \rightarrow p\}$ over D, checking whether $(D, \Sigma) \vdash S \rightarrow p$ can be done in quadratic time. The size of each formula φ_q in the previous Lemma is $O(\|\Sigma\| + \|S\| + \|p\|)$. Thus, it is possible to verify whether φ_q is satisfiable in time $O(\|\Sigma\| + \|S\| + \|p\|)$, since satisfiability of propositional Horn formulas can be checked in linear time [Dowling and Gallier 1984]. Hence, given that there are at most $\|p\|$ of these formulas, checking whether formula $\bigvee_{q \in X} \varphi_q$ in Lemma A.3 is satisfiable requires time $O(\|p\| \cdot (\|\Sigma\| + \|S\| + \|p\|))$. To construct this formula, first we

execute two steps:

(1) For every $s \in paths(\Sigma)$, find the longest common prefix of s and a path from $S \cup \{p\}$, which requires time $O(\|s\| \cdot (\|S\| + \|p\|))$. By using this prefix verify whether s can be nullified from $S \cup \{p\}$, which requires time $O(\|s\| \cdot \|D\|)$.
(2) For each $s \in paths(\Sigma)$ and for each prefix q of p, verify whether q is a prefix of s, which requires time $O(\|q\|)$.

The total time required by these steps is $O(\|\Sigma\| \cdot (\|D\| + \|S\| + \|p\|))$. Let k be the number of paths in Σ and l be the number of prefixes of p. The information generated by the first step is stored in a array with k entries, one for each path in Σ, indicating whether each of these paths can be nullified from $S \cup \{p\}$. Similarly, the information generated by the second step is stored in l arrays with k entries each. By using these data structures, the formula $\bigvee_{q \in X} \varphi_q$ in Lemma A.3 can be constructed in time $O(\|p\| \cdot (\|\Sigma\| + \|S\| + \|p\|))$. Thus, the total time of the algorithm is $O(\|p\| \cdot (\|\Sigma\| + \|S\| + \|p\|) + \|\Sigma\| \cdot (\|D\| + \|S\| + \|p\|))$. This completes the proof of Theorem 7.1.

A.2 Proof of Theorem 7.2

To prove this theorem first we prove two lemmas. Let $D = (E, A, P, R, r)$ be a disjunctive DTD and $\tau \in E$ such that $P(\tau) = s_1, \ldots, s_n$. Assume that for a fixed $k \in [1, n]$, $s_k = s_1' | s_2'$, where s_1', s_2' are simple disjunctions over alphabets A_1', A_2' and $A_1' \cap A_2' = \emptyset$. Assume that there is only one $p_\tau \in paths(D)$ such that $last(p_\tau) = \tau$. We define $paths_i(D)$ (for $i = 1, 2$) as the set of all paths q in D such that one of the following statement holds: (1) p_τ is not a proper prefix of q or (2) there is $\tau' \in E$ such that $p_\tau.\tau'$ is a prefix of q and τ' is in the alphabet of any of the regular expressions $s_1, \ldots, s_{k-1}, s_i', s_{k+1}, \ldots, s_n$. Then we define DTDs $D_i = (E_i, A_i, P_i, R_i, r)$ (for $i = 1, 2$) as follows. $E_i = \{\tau' \in E \mid \tau'$ is mentioned in some $q \in paths_i(D)\}$, $A_i = \{@l \mid$ there is $\tau' \in E_i$ such that $@l \in R(\tau')\}$, $P_i(\tau) = s_1, \ldots, s_{k-1}, s_i', s_{k+1}, \ldots, s_n$, $P_i(\tau') = P(\tau')$, for each $\tau' \in E_i - \{\tau\}$, and $R_i = R|_{E_i}$. Moreover, given a set of functional dependencies Σ over D, we define a set of functional dependencies Σ_i over D_i (for $i = 1, 2$) as follows. For each $S \to p \in \Sigma$, if $S \cup \{p\} \subseteq paths_i(D)$, then $S \to p$ is included in Σ_i.

LEMMA A.4. *Let D, Σ, τ, p_τ, D_i and Σ_i, for $i = 1, 2$ be as above and let $S \to p$ be a functional dependency over D. Then*

(a) *If $S \cup \{p\} \not\subseteq paths_i(D)$ for every $i \in [1, 2]$, then $(D, \Sigma) \vdash S \to p$.*
(b) *If $S \cup \{p\} \subseteq paths_1(D)$ and $S \cup \{p\} \not\subseteq paths_2(D)$, then $(D, \Sigma) \vdash S \to p$ iff $(D_1, \Sigma_1) \vdash S \to p$.*
(c) *If $S \cup \{p\} \subseteq paths_i(D)$ for every $i \in [1, 2]$, then $(D, \Sigma) \vdash S \to p$ iff for every $i \in [1, 2]$, $(D_i, \Sigma_i) \vdash S \to p$.*

PROOF. (a) Let $p_i \in paths_i(D)$ ($i \in [1, 2]$) such that $p_i \in S \cup \{p\}$, for every $i \in [1, 2]$, $p_1 \notin paths_2(D)$ and $p_2 \notin paths_1(D)$. Let T be an XML tree such that $T \models (D, \Sigma)$, and $t_1, t_2 \in tuples_D(T)$. Without loss of generality, assume that $p_1 \in S$. If $t_1.p_1 = t_2.p_1$ and $t_1.p_1 \neq \perp$, then $t_1.p_2 = t_2.p_2 = \perp$, and, therefore, $T \models S \to p$. Thus, we conclude that $(D, \Sigma) \vdash S \to p$.

(b) If $(D, \Sigma) \vdash S \rightarrow p$, we have to prove that $(D_1, \Sigma_1) \vdash S \rightarrow p$. Let T_1 be an XML such that $T_1 \models (D_1, \Sigma_1)$. This tree conforms to D and satisfies Σ, since each constraint $\varphi \in \Sigma - \Sigma_1$ contains at least one path q such that for every $t \in tuples_D(T_1)$, $t.q = \perp$. Hence, $T_1 \models S \rightarrow p$.

Suppose that $(D_1, \Sigma_1) \vdash S \rightarrow p$. We have to prove that $(D, \Sigma) \vdash S \rightarrow p$. Let T be an XML tree such that $T \models (D, \Sigma)$, and $t_1, t_2 \in tuples_D(T)$. Let $p_1 \in paths_1(D)$ such that $p_1 \in S \cup \{p\}$ and $p_1 \notin paths_2(D)$. By contradiction, suppose that $t_1.S = t_2.S$, $t_1.S \neq \perp$ and $t_1.p \neq t_2.p$. If $p_1 \in S$, then there is $T_1 \in trees_D(\{t_1, t_2\})$ such that $T_1 \models D_1$, since $t_1.p_1 \neq \perp$ and $t_2.p_1 \neq \perp$. Since $T \models \Sigma$, $T_1 \models \Sigma_1$, and, therefore $(D_1, \Sigma_1) \not\vdash S \rightarrow p$, a contradiction. If $p_1 = p$, without loss of generality, we can assume that $t_1.p_1 \neq \perp$. If $t_2.p_1 \neq \perp$, then there is $T_1 \in trees_D(\{t_1, t_2\})$ such that $T_1 \models D_1$. But, $T_1 \models \Sigma_1$, since $T \models \Sigma$, and, therefore $(D_1, \Sigma_1) \not\vdash S \rightarrow p$, a contradiction. Assume that $t_2.p_1 = \perp$. Define $t_2' \in \mathcal{T}(D_1)$ as follows. For each $w \in paths_1(D) \cap paths_2(D)$, $t_2'.w = t_2.w$, and for each $w \in paths_1(D) - paths_2(D)$, if $t_1.w = \perp$, then $t_2'.w = \perp$, otherwise $t_2'.w \neq t_1.w$. Given that $t_1.p_\tau \neq t_2.p_\tau$, since $t_1.p_1 \neq \perp$ and $t_2.p_1 = \perp$, we conclude that there is an XML tree $T_1 \in trees_D(\{t_1, t_2'\})$ such that T_1 conforms to D_1. But $T_1 \models \Sigma_1$, since $trees_D(\{t_1, t_2\}) \models \Sigma$. Thus, $(D_1, \Sigma_1) \not\vdash S \rightarrow p$, again a contradiction.

(c) We will only prove the "if" direction. The "only if" direction is analogous to the proof of this direction in (b). Assume that $(D, \Sigma) \not\vdash S \rightarrow p$. We will show that $(D_1, \Sigma_1) \not\vdash S \rightarrow p$ or $(D_2, \Sigma_2) \not\vdash S \rightarrow p$.

Given that every disjunctive DTD is a relational DTD (see Proposition 7.3), by Lemma A.6 we conclude that $(D, \Sigma) \not\vdash S \rightarrow p$ if and only if there is an XML tree T and a path q prefix of p such that $T \models (D, \Sigma)$, $tuples_D(T) = \{t_1, t_2\}$, $t_1.S = t_2.S$, $t_1.S \neq \perp$, $t_1.p \neq t_2.p$, $t_1.q \neq t_2.q$ and for each $s \in paths(D)$, if q is not a prefix of s, then $t_1.s = t_2.s$. We consider three cases.

(1) If q is not a prefix of p_τ. Then, there is $T' \in trees_D(\{t_1, t_2\})$ such that T' conforms to either D_1 or D_2. Without loss of generality, assume that $T' \models D_1$. In this case, $T' \models \Sigma_1$, since $T \models \Sigma$. Hence, $(D_1, \Sigma_1) \not\vdash S \rightarrow p$.

(2) If q is a prefix of p_τ and there exists $a_1' \in A_1'$ and $a_2' \in A_2'$ such that $t_1.p_\tau.a_1' \neq \perp$ and $t_2.p_\tau.a_2' \neq \perp$. In this case, we define $t_2' \in \mathcal{T}(D_1)$ as follows. For each $w \in paths_1(D) \cap paths_2(D)$, $t_2'.w = t_2.w$, and for each $w \in paths_1(D) - paths_2(D)$, if $t_1.w = \perp$, then $t_2'.w = \perp$, otherwise $t_2'.w \neq t_1.w$. Then, there exists $T' \in trees_{D_1}(\{t_1, t_2'\})$ such that $T' \models D_1$, $T' \models \Sigma_1$ and $T' \not\models S \rightarrow p$, since $T \models \Sigma$ and $T \not\models S \rightarrow p$. We conclude that $(D_1, \Sigma_1) \not\vdash S \rightarrow p$.

(3) If q is a prefix of p_τ and there are no $a_1' \in A_1'$ and $a_2' \in A_2'$ such that either $t_1.p_\tau.a_1' \neq \perp$ and $t_2.p_\tau.a_2' \neq \perp$ or $t_2.p_\tau.a_1' \neq \perp$ and $t_1.p_\tau.a_2' \neq \perp$. This case is analogous to the first one. □

Given a disjunctive DTD $D = (E, A, P, R, r)$, to apply the previous lemma we need to find an element type τ such that there is exactly one path in D whose last element is τ and $P(\tau) = s_1, \ldots, s_k, \ldots, s_n$, where $s_k = s_1'|s_2'$, s_1' and s_2' are simple disjunctions over alphabets A_1', A_2' and $A_1' \cap A_2' = \emptyset$. If there is no such an element type and D is not a simple DTD, it is possible to create it by using the following transformation. Pick τ satisfying the previous conditions except

for there is more than one path whose last element is τ. Pick $p \in paths(D)$ such that $last(p) = \tau$. Define a DTD $D_p = (E_p, A, P_p, R_p, r_p)$ as follows. $r_p = [r]$ and $E_p = (E - \{r\}) \cup \{[q] \mid q \in paths(D)$ and q is a prefix of $p\}$ (we use square brackets to distinguish between paths and element types). The functions P_p and R_p are defined as follows.

—For each $q \in paths(D)$ and $\tau' \in E$ such that $q.\tau'$ is a prefix of p, $P_p([q]) = f(P(last(q)))$, where f is a homomorphism defined as $f(\tau') = [q.\tau']$ and $f(\tau'') = \tau''$ for each $\tau'' \neq \tau'$. Moreover, $P_p([p]) = P(last(p))$ and $P_p(\tau') = P(\tau')$, for each $\tau' \in E - \{r\}$.

—For each $[q] \in E_p$, $R_p([q]) = R(last(q))$. Moreover, $R_p(\tau') = R(\tau')$, for each $\tau' \in E - \{r\}$.

Let $\Sigma \cup \{S \to q\}$ be a set of functional dependencies over D. We define a set of functional dependencies $\Sigma_p \cup \{S_p \to q_p\}$ over D_p as follows. For each path q' mentioned in $\Sigma \cup \{S \to q\}$, if $q' = q_1.q_2$, where q_1 is the longest common prefix of q' and p, then q' is replaced by $g(q_1).q_2$, where g is an homomorphism defined as $g([r]) = [r]$ and $g([w.\tau']) = g([w]).[w.\tau']$, for each $w.\tau'$ prefix of p. The following is straightforward.

LEMMA A.5. *Let D, $\Sigma \cup \{S \to q\}$, D_p and $\Sigma_p \cup \{S_p \to q_p\}$ be as above. Then, $(D, \Sigma) \vdash S \to q$ iff $(D_p, \Sigma_p) \vdash S_p \to q_p$.*

Theorem 7.2 now follows from Lemmas A.4 and A.5.

A.3 The Implication Problem for Relational DTDs is in coNP

To prove this theorem we start with the following lemma.

LEMMA A.6. *Given a relational DTD D, a set Σ of functional dependencies over D and $S \cup \{p\} \subseteq paths(D)$, $(D, \Sigma) \nvdash S \to p$ if and only if there is an XML tree T and a path q prefix of p such that T conforms to D, T satisfies Σ, $tuples_D(T) = \{t_1, t_2\}$, $t_1.S = t_2.S$, $t_1.S \neq \perp$, $t_1.p \neq t_2.p$, $t_1.q \neq t_2.q$ and for each $s \in paths(D)$, if q is not a prefix of s, then $t_1.s = t_2.s$.*

PROOF. We will prove only the "only if" direction, since the "if" direction is trivial.

Suppose that $(D, \Sigma) \nvdash S \to p$. There is an XML tree T' conforming to D and satisfying Σ such that $T' \not\models S \to p$. Then, there are tuples $t'_1, t'_2 \in tuples_D(T')$ such that $t'_1.S = t'_2.S$, $t'_1.S \neq \perp$ and $t'_1.p \neq t'_2.p$. Let q be the shortest prefix of p such that $t'_1.q \neq t'_2.q$. We define tree tuples t_1 and t_2 as follows. For each $s \in paths(D)$, if q is not a prefix of s, then $t_1.s = t'_1.s$ and $t_2.s = t'_1.s$. Otherwise, $t_1.s = t'_1.s$ and $t_2.s = t'_2.s$. Notice that $t_1, t_2 \in tuples_D(T')$.

Given that D is a relational DTD, it is possible to find $T \in trees_D(\{t_1, t_2\})$ such that $T \models D$. We need to prove that T satisfies the conditions of the lemma. By definition of t_1 and t_2, $tuples_D(T) = \{t_1, t_2\}$ and for each $s \in paths(D)$, if q is not a prefix of s, then $t_1.s = t_2.s$. Besides, $t_1.S = t_2.S$, $t_1.S \neq \perp$ and $t_1.p \neq t_2.p$, since $t'_1.S = t'_2.S$, $t'_1.S \neq \perp$, $t'_1.p \neq t'_2.p$ and q is a prefix of p. Finally, $t_1.q \neq t_2.q$, since $t'_1.q \neq t'_2.q$, and $T \models \Sigma$, since $T' \models \Sigma$ and $t_1, t_2 \in tuples_D(T')$. □

Now we are ready to prove that the implication problem for relational DTDs is in coNP. Let D be a relational DTD, Σ a set of functional dependencies over D and $S \cup \{p\} \subseteq paths(D)$. Let $prefix(\Sigma \cup \{S \rightarrow p\})$ be the set of all $p' \in paths(D)$ such that p' is a prefix of a path mentioned in $\Sigma \cup \{S \rightarrow p\}$. Notice that $\|prefix(\Sigma \cup \{S \rightarrow p\})\|$ is $O(\|\Sigma \cup \{S \rightarrow p\}\|^2)$.

To check whether $(D, \Sigma) \nvdash S \rightarrow p$, we use a nondeterministic algorithm that guesses the tuples t_1 and t_2 mentioned in Lemma A.6. This algorithm does not construct all the values in t_1 and t_2, it guesses only the values of these tuples that are necessary to verify whether $trees_D(\{t_1, t_2\}) \models \Sigma$. The algorithm works as follows. For each $s \in prefix(\Sigma \cup \{S \rightarrow p\})$, guess the values of $t_1.s$ and $t_2.s$. Verify whether it is possible to construct an XML tree conforming to D and containing t_1 and t_2. If this does not hold, then return "no". Otherwise, guess a prefix q of p. Verify whether $t_1.S = t_2.S$, $t_1.S \neq \perp$, $t_1.p \neq t_2.p$, $t_1.q \neq t_2.q$ and for each $s \in paths(\Sigma \cup \{S \rightarrow p\})$, if q is not a prefix of s, then $t_1.s = t_2.s$. If this does not hold, then return "no". Otherwise, check whether the values in t_1 and t_2 satisfy Σ. If this is the case, then return "yes", otherwise return "no".

The previous algorithm works in nondeterministic polynomial time, since $\|prefix(\Sigma \cup \{S \rightarrow p\})\|$ is $O(\|\Sigma \cup \{S \rightarrow p\}\|^2)$. Therefore, we conclude that the implication problem for relational DTDs is in coNP.

A.4 Proof of Proposition 7.7

We only need to prove the "if" direction. Suppose that for each nontrivial FD of the form $S \rightarrow p.@l$ or $S \rightarrow p.\mathrm{S}$ in Σ, $S \rightarrow p \in (D, \Sigma)^+$.

Assume that (D, Σ) is not in XNF. Without loss of generality, assume that there exists a nontrivial functional dependency $S' \rightarrow p'.@l'$ such that $S' \rightarrow p'.@l' \in (D, \Sigma)^+$ and $S' \rightarrow p' \notin (D, \Sigma)^+$. By Lemma A.6, there is an XML tree T and a path q prefix of p' such that T conforms to D, T satisfies Σ, $tuples_D(T) = \{t_1, t_2\}$, $t_1.S' = t_2.S'$, $t_1.S' \neq \perp$, $t_1.p' \neq t_2.p'$, $t_1.q \neq t_2.q$ and for each $s \in paths(D)$, if q is not a prefix of s, then $t_1.s = t_2.s$. If $t_1.p'.@l' \neq t_2.p'.@l'$, then $(D, \Sigma) \nvdash S' \rightarrow p'.@l'$, a contradiction. Thus, we can assume that $t_1.p'.@l' = t_2.p'.@l'$. We can also assume $t_1.p'.@l' \neq \perp$, since if $t_1.p'.@l' = t_2.p'.@l' = \perp$, then $t_1.p' = t_2.p' = \perp$ and, therefore, $T \models S' \rightarrow p'$. Define a new tree tuple t_1' as follows: $t_1'.w = t_1.w$, for each $w \neq p'.@l'$, $t_1'.p'.@l' \neq t_1.p'.@l'$ and $t_1'.p'.@l' \neq \perp$. Then, there is an XML tree $T' \in trees_D(\{t_1', t_2\})$ such that $T' \models D$ and $T' \not\models S' \rightarrow p'.@l'$, since $p'.@l' \notin S'$ ($S' \rightarrow p'.@l'$ is a nontrivial functional dependency). If $T' \models \Sigma$, then $(D, \Sigma) \nvdash S' \rightarrow p'.@l'$, a contradiction. Hence $T' \not\models \Sigma$ and, therefore, there is $S \rightarrow p'' \in \Sigma$ such that $T' \not\models S \rightarrow p''$. But p'' must be equal to $p'.@l'$, since $t_1, t_2 \in tuples_D(T)$ and $T \models \Sigma$. Therefore, $T \not\models S \rightarrow p'$, because $t_1.S = t_1'.S = t_2.S$, $t_1'.S \neq \perp$ and $t_1.p' \neq t_2.p'$. We conclude that $(D, \Sigma) \nvdash S \rightarrow p'$, which contradicts our initial assumption since $S \rightarrow p'.@l'$ is a nontrivial FD in Σ.