SCIENCE CHINA
Information Sciences

• LETTER •

September 2016, Vol. 59 099102:1–099102:3
doi: 10.1007/s11432-016-5603-z

Single-view determinacy and rewriting completeness for a fragment of XPath queries

Lixiao ZHENG[1,2], Shuai MA[3*], Xiangyu LUO[1] & Tiejun MA[4]

[1]*College of Computer Science and Technology, Huaqiao University, Xiamen* 361021*, China;*
[2]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100190*, China;*
[3]*State Key Laboratory of Software Development Environment, Beihang University, Beijing* 100191*, China;*
[4]*Business School, University of Southampton, Southampton,* SO17 1BJ*, UK*

Received October 9, 2015; accepted February 3, 2016; published online August 23, 2016

Citation Zheng L X, Ma S, Luo X Y, et al. Single-view determinacy and rewriting completeness for a fragment of XPath queries. Sci China Inf Sci, 2016, 59(9): 099102, doi: 10.1007/s11432-016-5603-z

Dear editor,

The problem of answering queries using views, where a view is a set of predefined queries, arises in a variety of data management applications. To formalize the fact that a set of views $\boldsymbol{V}$ contains enough information for answering a specific query Q, Segoufin et al. [1] proposed the notion of determinacy: $\boldsymbol{V}$ *determines* Q iff $\boldsymbol{V}(D_1) = \boldsymbol{V}(D_2)$ implies $Q(D_1) = Q(D_2)$ for all database instances D_1 and D_2. Another formalization comes from a syntactic perspective, using the notion of rewriting: Q can be (equivalently) *rewritten* in terms of $\boldsymbol{V}$ using a rewriting language $\mathcal{L}_R$ iff there exists a query $R \in \mathcal{L}_R$ such that $Q(D) = R(\boldsymbol{V}(D))$ for all database instance D. A rewriting language $\mathcal{L}_R$ is said to be *complete* for $\mathcal{L}_V$-to-$\mathcal{L}_Q$ rewriting, where $\mathcal{L}_V$ is a view language and $\mathcal{L}_Q$ is a query language, if whenever a set of views $\boldsymbol{V} \in \mathcal{L}_V$ determines a query $Q \in \mathcal{L}_Q$ then there exists a rewriting $R \in \mathcal{L}_R$ of Q in terms of $\boldsymbol{V}$.

Determinacy has been well studied on relational databases for languages such as Datalog and conjunctive queries [1–3], and recently on graph databases for path queries [4]. However, little work has been reported in the context of XML databases, in which an XML document is modelled as an unordered, rooted and labeled tree (*tree* for short) t over an infinite alphabet Σ. In this letter, we consider determinacy in an XML context when queries and views are both defined in $\mathrm{XP}^{\{*,//,[]\}}$, a fragment of XPath queries constructed with wildcard ($*$), descendant edges ($//$) and branches ($[]$), together with its three sub-fragments $\mathrm{XP}^{\{//,[]\}}$, $\mathrm{XP}^{\{*,[]\}}$ and $\mathrm{XP}^{\{*,//\}}$ obtained by disallowing constructs $*$, $//$ and $[]$, respectively. We will focus on the single-view case, in which a view consists of only a single query, and the query language, view language and rewriting language are all the same. We also use the symbol V to refer to a single view and the symbol $\mathcal{L}$ to denote $\mathrm{XP}^{\{*,//,[]\}}$ or one of its sub-fragments, respectively, in the following.

We first analyze the complexity of deciding determinacy for $\mathrm{XP}^{\{*,//,[]\}}$. We notice that for a Boolean query Q and a Boolean view V, V determines P iff V contains Q. Query containment for Boolean $\mathrm{XP}^{\{*,//,[]\}}$ is known to be coNP-complete. This implies that determinacy for Boolean $\mathrm{XP}^{\{*,//,[]\}}$ is also coNP-complete. Since this is a special case of determinacy for $\mathrm{XP}^{\{*,//,[]\}}$, we get a lower bound.

Theorem 1. The determinacy problem for queries and views in $\mathrm{XP}^{\{*,//,[]\}}$ is coNP-hard.

We then show by counterexamples that even though an $\mathrm{XP}^{\{*,//,[]\}}$ view determines an $\mathrm{XP}^{\{*,//,[]\}}$

* Corresponding author (email: mashuai@buaa.edu.cn)
The authors declare that they have no conflict of interest.

query, there may not exist an $\mathrm{XP}^{\{*,//,[]\}}$ rewriting of the query using the view. That is, $\mathrm{XP}^{\{*,//,[]\}}$ is not complete for rewriting. For the two sub-fragments $\mathrm{XP}^{\{//,[]\}}$ and $\mathrm{XP}^{\{*,//\}}$, we obtain a similar result.

Theorem 2. $\mathcal{L}$ is not complete for $\mathcal{L}$-to-$\mathcal{L}$ rewriting when $\mathcal{L}$ is $\mathrm{XP}^{\{*,//,[]\}}$, $\mathrm{XP}^{\{//,[]\}}$ or $\mathrm{XP}^{\{*,//\}}$.

To cope with these negative results, we provide a set of necessary conditions for a view V determining a query Q, from which we know that determinacy does not hold if the properties do not hold. We first explain some concepts and notations.

$\mathrm{XP}^{\{*,//,[]\}}$ queries are also known as *tree patterns*. A tree pattern P is a tree with a set of nodes labeled with $*$ or symbols from an alphabet Σ ($* \notin \Sigma$), two types of edges (*child* edges and *descendant* edges) and a distinguished node called the output node $out(P)$. Each $\mathrm{XP}^{\{*,//,[]\}}$ query can be translated into a tree pattern with the same semantics and vice versa [5]. In light of this, we will use *pattern* instead of *query* in the following.

For a pattern P, we denote by $\widehat{P}$ the Boolean version of P without specifying its output node. Suppose that c_j is a child of the root of P; we denote by $P_{[j]}$ the branch of P connected from c_j to the root. If a branch contains the output node, we refer to this unique branch as $P_{[o]}$. We denote by $\mathcal{B}(P)$ the set of all the branches of P. The following set of necessary conditions concerns the Boolean versions and branches of P and V.

Proposition 1. If a view V determines a pattern P, then the following hold: (1) $\widehat{P} \subseteq \widehat{V}$; (2) for each branch $\widehat{P}_{[i]} \in \mathcal{B}(\widehat{P})$, $\widehat{P}_{[i]} \subseteq \widehat{V}_{[1]} \cup \cdots \cup \widehat{V}_{[m]}$ where $\widehat{V}_{[1]}, \ldots, \widehat{V}_{[m]} \in \mathcal{B}(\widehat{V})$ and $m = |\mathcal{B}(\widehat{V})|$; and (3) $\widehat{P}_{[o]} \subseteq \widehat{V}_{[o]}$.

Define the *height* of a node n of pattern P to be the number of edges on the path from the root to n. The *height* and *depth* of pattern P, denoted by $\mathrm{height}(P)$ and $\mathrm{depth}(P)$, are the maximal height of nodes of P and the height of the output node of P, respectively. We use $\Sigma(P)$ to denote the set of labels of Σ appearing in P. Note that the wildcard $*$ may appear in P, but not in $\Sigma(P)$. By the semantic conditions developed above, we further derive a set of syntactic conditions for determinacy.

Proposition 2. If a view V determines a pattern P, then the following hold: (1) $\Sigma(V) \subseteq \Sigma(P)$; (2) $\mathrm{label}(\mathrm{root}(V)) = \mathrm{label}(\mathrm{root}(P))$ or $\mathrm{label}(\mathrm{root}(V)) = *$; (3) $\mathrm{height}(V) \leqslant \mathrm{height}(P)$; and (4) $\mathrm{depth}(V) \leqslant \mathrm{depth}(P)$.

According to Proposition 1(1), if a pattern P is determined by a view V, then $\widehat{P} \subseteq \widehat{V}$. For the three sub-fragments of $\mathrm{XP}^{\{*,//,[]\}}$, the inclusion of $\widehat{P}$ into $\widehat{V}$ implies the existence of homomorphisms from $\widehat{V}$ to $\widehat{P}$ [5]. A homomorphism from $\widehat{V}$ to $\widehat{P}$ is a function h mapping the nodes of $\widehat{V}$ to the nodes of $\widehat{P}$ and satisfying the following conditions: $h(\mathrm{root}(\widehat{V})) = \mathrm{root}(\widehat{P})$; for each node n of $\widehat{V}$, either $\mathrm{label}(n) = *$ or $\mathrm{label}(n) = \mathrm{label}(h(n))$; and for each child edge (n_1, n_2) of $\widehat{V}$, $(h(n_1), h(n_2))$ is also a child edge, and for each descendant edge (n_1, n_2) of $\widehat{V}$, $h(n_2)$ is a descendant of $h(n_1)$ in $\widehat{P}$. One can easily verify that each homomorphism from $\widehat{V}$ to $\widehat{P}$ induces a subpattern of P that computes a superset of $P(t)$ when evaluated on $V(t)$ for any tree t. By taking the intersection of those supersets, we obtain the exact result of $P(t)$.

The above analysis leads to the following algorithm for checking determinacy. Notice that the intersection of a pattern P with a Boolean pattern B is defined as follows: Given tree t, if $B(t) \neq \emptyset$ then $P \cap B\ (t) = P(t)$, otherwise $P \cap B\ (t) = \emptyset$.

Algorithm 1: CheckDeterminacy(P,V)

Input: A pattern P and a view V
Output: *'Yes'* if V determines P and *'No'*, otherwise
(1) Find all the homomorphisms from $\widehat{V}$ to $\widehat{P}$, denoted by $H = \{h_1, \ldots, h_m\}$.
(2) If $H = \emptyset$, then return *'No'*.
(3) For each homomorphism $h_i \in H$: (a) Let n_i be the node $h_i(\mathrm{out}(V))$ and P_i be the subpattern of P rooted at n_i. (b) Compute the composition pattern $R_i := P_i \circ V$.
(4) Let $R := R_1 \cap \cdots \cap R_m$.
(5) If $R \equiv P$, then return *'Yes'*, else return *'No'*.

If the algorithm returns *Yes*, it means that we can compute the result of pattern P from the result of view V and thus V determines P. Therefore, the soundness holds. Clearly, the algorithm is also sound for the whole fragment $\mathrm{XP}^{\{*,//,[]\}}$. However, it may return more *false-negative* answers because in this case the existence of a homomorphism is no longer a necessary condition for containment, and thus, it is very likely that the set H in Step (1) is empty even though V determines P.

We now analyze the time complexity of this algorithm. The main computational cost in Algorithm 1 is Step (5): testing equivalence between a tree pattern and an intersection of a set of tree patterns. This has been shown to run in the worst-case exponential time. The other main computational cost is computing homomorphisms that can be done in polynomial time. Thus, in total, Algorithm 1 has exponential time complexity.

Claim 1. Algorithm 1 is sound and takes exponential time in the size of pattern P and view V.

However, we observe that, for $\mathrm{XP}^{\{*,[]\}}$, Algorithm 1 takes only polynomial time. Observe that $\mathrm{XP}^{\{*,[]\}}$ patterns contain no descendant edges. It can be verified that there is only one subpattern in Step (3)(a) of Algorithm 1 that contains the output node of P and all the other subpatterns are Boolean. We, hence, infer that the pattern R in Step (4) is still a tree pattern in $\mathrm{XP}^{\{*,[]\}}$, which leads to PTIME equivalence testing in Step (5). Thus, Algorithm 1 takes only polynomial time if patterns and views are in $\mathrm{XP}^{\{*,[]\}}$. Besides the PTIME complexity, we furthermore show that, for this fragment, the *No* answer from Algorithm 1 implies that determinacy does not hold.

Proposition 3. Consider a minimal pattern P and a minimal view V in $\mathrm{XP}^{\{*,[]\}}$. Let R be the intersection of patterns constructed from P and V as described in Algorithm 1. If $R \not\equiv P$, then V does not determine P.

By now, we can claim the following result.

Claim 2. Algorithm 1 is complete and runs in polynomial time for $\mathrm{XP}^{\{*,[]\}}$.

In Proposition 3, we assume that patterns and views are minimal. Minimizing $\mathrm{XP}^{\{*,[]\}}$ patterns can be easily done in polynomial time. Thus, given a pattern P and a view V defined in $\mathrm{XP}^{\{*,[]\}}$, we can check whether V determines P by first minimizing P and V and then applying Algorithm 1. This process takes in total polynomial time with the size of P and V. Moreover, Algorithm 1 provides a method for computing the result of P from the result of V if determinacy holds. Note that in Algorithm 1 Step (3)(a), if V determines P, then there is only one subpattern of P that contains the output node and all the others are Boolean. Let P_o be the subpattern containing the output node. One can verify that, given a tree t, if all the Boolean patterns are satisfied by some of the subtrees of $V(t)$, then $P_o(V(t))$ is equal to $P(t)$. In fact, we can express the above computation with only one $\mathrm{XP}^{\{*,[]\}}$ pattern by slightly reorganizing the subtrees in $V(t)$, as described as follows.

Algorithm 2: ANSWERPATTERN(P, V, ST)

Input: A pattern $P \in \mathrm{XP}^{\{*,[]\}}$, a view $V \in \mathrm{XP}^{\{*,[]\}}$ and a set of subtrees $ST = V(t)$ for some tree t
Output: The answer of pattern P on tree t
(1) Find all the homomorphisms from $\widehat{V}$ to $\widehat{P}$, denoted by $H = \{h_1, \ldots, h_m\}$.
(2) Find all the subpatterns $P = \{P_1, \ldots, P_m\}$ where P_i is the subpattern rooted at n_i and n_i is the node $h_i(\mathrm{out}(V))$ of P, for each $i \in [1, m]$.
(3) Merge the subpatterns in P into one pattern R by introducing a common root labeled by any symbol $l \in \Sigma$, and merge the subtrees in ST into one tree t_V by introducing a common root with the same label l.
(4) Evaluate R on tree t_V. Return the result $R(t_V)$.

Indeed, if all the Boolean subpatterns are satisfied by some of the subtrees of $V(t)$, then by construction, we can verify that $R(t_V)$ is equal to $P_o(V(t))$ where P_o is the unique subpattern of P that contains the output node. Note that pattern R is still in $\mathrm{XP}^{\{*,[]\}}$ and the combination of ST into one tree t_V is gained without loss of generality. This means that, whenever a view $V \in \mathrm{XP}^{\{*,[]\}}$ determines a pattern $P \in \mathrm{XP}^{\{*,[]\}}$, we can find a pattern $R \in \mathrm{XP}^{\{*,[]\}}$ to answer the pattern using the view. In this sense, we say that $\mathrm{XP}^{\{*,[]\}}$ is complete for $\mathrm{XP}^{\{*,[]\}}$-to-$\mathrm{XP}^{\{*,[]\}}$ rewriting.

Theorem 3. (1) The determinacy problem for patterns and views in $\mathrm{XP}^{\{*,[]\}}$ is decidable in PTIME. (2) $\mathrm{XP}^{\{*,[]\}}$ is complete for $\mathrm{XP}^{\{*,[]\}}$-to-$\mathrm{XP}^{\{*,[]\}}$ rewriting.

Conclusion. We have investigated the single-view determinacy and rewriting completeness problems for a widely used fragment of XPath queries constructed by wildcard labels, descendant edges and branches. We have proven that this fragment is not complete for rewriting and that deciding whether a view determines a query is coNP-hard. We have also provided a set of necessary conditions, from both semantic and syntactic aspects, for a view determining a query. Further, we have developed a sound algorithm for checking determinacy and identified a well-behaved sub-fragment for which determinacy is tractable in PTIME.

Acknowledgements Zheng is supported in part by NSFC (61472405&61502184) and NSF of Fujian Province (2015J01259). Ma is supported in part by 973 Program (2014CB340300) and NSFC (61322207&61421003). Luo is supported in part by NSFC (61170028) and Promotion Program for Young and Middle-aged Teachers in Science and Technology Research of Huaqiao University (ZQN-YX109).

Supporting information The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Nash A, Segoufin L, Vianu V. Views and queries: determinacy and rewriting. ACM Trans Database Syst, 2010, 35: 21
2 Fan W, Geerts F, Zheng L X. View determinacy for preserving selected information in data transformations. Inform Syst, 2012, 37: 1–12
3 Gogacz T, Marcinkowski J. The hunt for a red spider: conjunctive query determinacy is undecidable. In: Proceedings of the Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), Kyoto, 2015. 281–292
4 Francis N. Asymptotic determinacy of path queries using union-of-paths views. In: Proceedings of the International Conference on Database Theory (ICDT), Brussels, 2015. 44–59
5 Miklau G, Suciu D. Containment and equivalence for a fragment of XPath. J ACM, 2004, 51: 2–45