SCIENCE CHINA
Information Sciences

• Supplementary File •

Single-View Determinacy and Rewriting Completeness for a Fragment of XPath Queries

ZHENG LiXiao[1,2], MA Shuai[3*], LUO XiangYu[1] & Ma TieJun[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*

Appendix A Preliminaries

Appendix A.1 Trees and Tree Patterns

Trees. We model an XML document as an unordered, rooted and labeled tree (*tree* for short) t over an infinite alphabet Σ. We denote by $\mathcal{N}(t)$, $\mathcal{E}(t)$ and $root(t)$ the set of nodes, the set of edges and the root of t respectively. Each node $n \in \mathcal{N}(t)$ has a label drawn from Σ, denoted by $label(n)$. A tree t' is said to be a *subtree* of the tree t if $\mathcal{N}(t') \subseteq \mathcal{N}(t)$ and $\mathcal{E}(t') \subseteq \mathcal{E}(t)$. Given a node n of t, we denote by $(t)^n_{sub}$ the subtree of t that is rooted at n and induced by all the descendants of n. The subtree $t-n$ is obtained by removing $(t)^n_{sub}$ from t. We denote by ε the empty tree and T_Σ the set of all trees over Σ.

Given two trees t_1 and t_2, if there exists a node n in t_2 such that t_1 is equal to the subtree of t_2 rooted at n, we say that t_1 is *contained* in t_2, denoted by $t_1 \subseteq t_2$. Furthermore, we define the *combination* of a set of trees $t_1, ..., t_m$, denoted by $t_1 + ... + t_m$, as follows: if the roots of $t_1, ..., t_m$ have the same label, then merge them into a single node and take it as the root of the combination tree; otherwise the combination is the empty tree ε.

A fragment of XPath queries. We study a fragment of XPath queries that has been widely investigated in literature [**?**, **?**, **?**, **?**]. This fragment, denoted as $\mathrm{XP}^{\{*,//,[]\}}$, consists of label tests, child axes ($/$), wildcard ($*$), descendant axes ($//$) and branches ($[]$). A query Q of $\mathrm{XP}^{\{*,//,[]\}}$ is recursively defined as follows:

$$Q \rightarrow l \mid * \mid Q/Q \mid Q//Q \mid Q[Q]$$

where l is a node label drawn from Σ. We also study sub-fragments of $\mathrm{XP}^{\{*,//,[]\}}$, denoted by listing the constructs supported: $\mathrm{XP}^{\{//,[]\}}$, $\mathrm{XP}^{\{*,[]\}}$ and $\mathrm{XP}^{\{*,//\}}$. For instance, $\mathrm{XP}^{\{//,[]\}}$ is the sub-fragment of $\mathrm{XP}^{\{*,//,[]\}}$ consists of the $//$ and $[]$ constructs.

Let $Q_1, ..., Q_m$ be $\mathrm{XP}^{\{*,//,[]\}}$ queries. We use $Q_1 \cup ... \cup Q_m$ and $Q_1 \cap ... \cap Q_m$ to represent the *union* ($\cup$) and *intersection* ($\cap$) of $Q_1, ..., Q_m$, denoted by $\mathrm{XP}^{\{*,//,[],\cup\}}$ and $\mathrm{XP}^{\{*,//,[],\cap\}}$ respectively. When Q_i is restricted to the sub-fragment $\mathrm{XP}^{\{//,[]\}}$, we denote the union and intersection by $\mathrm{XP}^{\{//,[],\cup\}}$ and $\mathrm{XP}^{\{//,[],\cap\}}$. Similarly by $\mathrm{XP}^{\{*,[],\cup\}}$, $\mathrm{XP}^{\{*,[],\cap\}}$ and $\mathrm{XP}^{\{*,//,\cup\}}$, $\mathrm{XP}^{\{*,//,\cap\}}$ when restricted to sub-fragments $\mathrm{XP}^{\{*,[]\}}$ and $\mathrm{XP}^{\{*,//\}}$ respectively.

Tree patterns. A *tree pattern* (*pattern* for short) P is a non-empty tree with a set of nodes $\mathcal{N}(P)$ labeled with symbols from $\Sigma \cup \{*\}$ ($* \notin \Sigma$), two types of edges: *child* edges $\mathcal{E}_{/}(P)$ and *descendant* edges $\mathcal{E}_{//}(P)$, and a distinguished node called the output node $out(P)$. The root of P is denoted as $root(P)$. We also say that a pattern without specifying its output node is a *boolean* pattern.

If a pattern P' satisfies $\mathcal{N}(P') \subseteq \mathcal{N}(P)$ and $\mathcal{E}(P') \subseteq \mathcal{E}(P)$, then we say that P' is a *subpattern* of P. Given a node n of P, we denote by $(P)^n_{sub}$ the subpattern rooted at n and by $P-n$ the subpattern obtained by removing $(P)^n_{sub}$ from P. The *selection path* of a pattern is the path from the root to the output node. The nodes on the selection path are called *selection nodes*. The *height* of a node n of P, denoted by $height(n)$, is the number of edges on the path from the root to n. The *height* of a pattern P, denoted by $height(P)$, is the maximal height of nodes of P. The *depth* of a pattern P, denoted by $depth(P)$, is the height of the output node of P. We use $\Sigma(P)$ to denote the set of labels of Σ that appear in P. Note that the wildcard $*$ may appear in P, but not in $\Sigma(P)$.

Remarks. Figure **??** shows an example of tree and an example of pattern, in which we use double and single lines to represent descendant and child edges respectively, and use a circle to denote the output node. The height and depth of this pattern is 2 and 1 respectively. This pattern is equivalent to the XPath query $a[b]//*[c][//d]$. As stated in [**?**], each $\mathrm{XP}^{\{*,//,[]\}}$ query can be translated into a tree pattern with the same semantics and vice versa. In light of this, we will use the term *pattern* instead of the term *query* in the sequel.

* Corresponding author (email: mashuai@buaa.edu.cn)

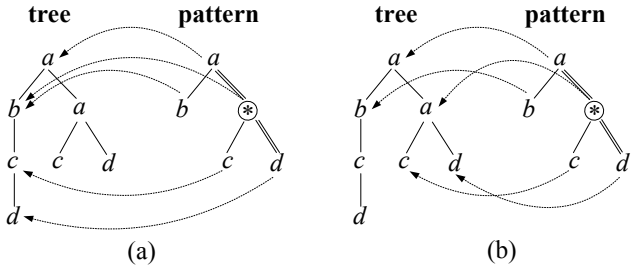


Figure A1 Example of embeddings from patterns to trees

We now define a notion of *embedding* to capture the semantics of tree patterns.

Embedding. Given a tree t and a pattern P, an embedding from P to t is a function $e:\mathcal{N}(P)\rightarrow\mathcal{N}(t)$ that satisfies the following conditions:

(1) Root preserving, *i.e.,* $e(root(P))=root(t)$;

(2) Label preserving, *i.e.,* for each node $n\in\mathcal{N}(P)$, either $label(n)=*$ or $label(n)=label(e(n))$;

(3) Structure preserving, *i.e.,* for each edge $(n_1,n_2)\in\mathcal{E}_{/}(P)$, we have that $(e(n_1),e(n_2))\in\mathcal{E}(t)$, and for each edge $(n_1,n_2)\in\mathcal{E}_{//}(P)$, $e(n_2)$ is a descendant of $e(n_1)$ in t.

Note that an embedding maps the output node of P to a node of t. We next define the evaluation of patterns.

Evaluation of patterns on trees. Given a pattern P and a tree t. Let e be an embedding from P to t and n be the node of t mapped by the output node of P through embedding e, *i.e.,* $n=e(out(P))$. We say that the subtree $(t)^n_{sub}$ of t is the *result* of this embedding. The evaluation of P on t is the set of results of all embeddings from P to t. That is, $P(t)=\{\ (t)^n_{sub}\mid n=e(out(P))$ where e is an embedding from P to $t\ \}$.

As an example, Figure **??** shows two embeddings from pattern P to tree t. Actually, there exist no other embeddings from P to t in this example. Thus the result of $P(t)$ consists of two subtrees rooted at the two child nodes of $root(t)$ labeled with a and b respectively.

The evaluation of a pattern P over a set of trees T is defined as: $P(T)=\bigcup_{t\in T}P(t)$. Note that, without loss of generality, a set of trees T can always be combined into a single tree t by introducing a common root r, and, hence, evaluating pattern P over T simply reduces to evaluating pattern $P'=r/P$ over the single tree t. The meaning of union and intersection is as usual: for any set of trees T, $(P_1\cup...\cup P_m)(T)=P_1(T)\cup...\cup P_m(T)$, and $(P_1\cap...\cap P_m)(T)=P_1(T)\cap...\cap P_m(T)$.

For a boolean pattern B, the result $B(t)$ is either $\{()\}$ or $\emptyset$, depending on whether there is an embedding from B to t. Note that $\{()\}$ is interpreted as *true* while $\emptyset$ stands for *false*. Thus, we write $t\models B$ if there exists an embedding from B to t, and $t\not\models B$, otherwise. For a pattern P, we denote by $\widehat{P}$ the boolean version of P without specifying its output node. Clearly $P(t)\neq\emptyset$ iff $t\models\widehat{P}$ for any pattern P and tree t.

Appendix A.2 Containment and Equivalence of Tree Patterns

Containment and equivalence. We say that a pattern P_1 is contained in another pattern P_2, denoted by $P_1\subseteq P_2$, if for any tree t, $P_1(t)\subseteq P_2(t)$. Two patterns are equivalent, denoted by $P_1\equiv P_2$, if both $P_1\subseteq P_2$ and $P_2\subseteq P_1$ hold. For boolean patterns, the containment problem reduces to the traditional implication problem: for boolean patterns B_1 and B_2, $B_1\subseteq B_2$ if $t\models B_1$ implies $t\models B_2$ for any tree t.

Canonical models. One technique to reason about the containment of tree patterns is using canonical models. Let B be a boolean pattern. A *model* of B is a tree t such that $t\models B$. A *canonical model* of B is a tree t obtained from B by applying the following two steps. First, each occurrence of the label $*$ is replaced with a symbol z of Σ that does not appear in any patterns elsewhere[1)]. Second, each descendant edge is replaced with a path of k ($k\geqslant 1$) edges, where all the internal nodes are labeled with z. We use $Mod(B)$ and $CMod(B)$ to denote the set of all models and all canonical models of B, respectively. The following result holds [**?**].

Lemma 1 ([**?**]). For any boolean patterns B_1 and B_2, $B_1\subseteq B_2$ iff $Mod(B_1)\subseteq Mod(B_2)$ iff $CMod(B_1)\subseteq Mod(B_2)$.

As shown in [**?**], the containment problems of patterns and boolean patterns are PTIME reducible. That is, two patterns P_1 and P_2 can always be translated into two boolean patterns B_1 and B_2 such that $P_1\subseteq P_2$ iff $B_1\subseteq B_2$.

Another technique to reason about containment is using the notion of *homomorphism*.

Homomorphism. Given two patterns P_1 and P_2, a homomorphism from P_2 to P_1 is a function $h:\mathcal{N}(P_2)\rightarrow\mathcal{N}(P_1)$ that satisfies the following conditions:

(1) Root preserving, *i.e.,* $h(root(P_2))=root(P_1)$;

(2) Output preserving, *i.e.,* $h(out(P_2))=out(P_1)$;

(3) Label preserving, *i.e.,* for each node $n\in\mathcal{N}(P_2)$, either $label(n)=*$ or $label(n)=label(h(n))$;

(4) Structure preserving, *i.e.,* for each edge $(n_1,n_2)\in\mathcal{E}_{/}(P_2)$, we have that $(h(n_1),h(n_2))\in\mathcal{E}_{/}(P_1)$, and for each edge $(n_1,n_2)\in\mathcal{E}_{//}(P_2)$, $h(n_2)$ is a descendant of $h(n_1)$ in P_1.

As an example, Figure **??** shows a homomorphism from $P_2=a[b]//*[c][//d]$ to $P_1=a[b//*]/*[c//d]$. Note that for boolean patterns, the notion of homomorphism only needs to satisfy conditions (1), (3) and (4), without the *output preserving* condition.

The existence of a homomorphism from one pattern to another is a sufficient condition for the containment problem. That is, if there exists a homomorphism from P_2 to P_1, then $P_1\subseteq P_2$. The converse, however, does not hold. For patterns either in $\mathrm{XP}^{\{//,[]\}}$ or $\mathrm{XP}^{\{*,[]\}}$, the existence of a homomorphism constitutes both a necessary and sufficient condition for deciding containment of single patterns [**?**, **?**, **?**]. This also holds for patterns in $\mathrm{XP}^{\{*,//\}}$after a standardization procedure which can be done in linear time [**?**].

Lemma 2 ([**?**, **?**, **?**]). For any patterns P_1 and P_2 in the three sub-fragments of $\mathrm{XP}^{\{*,//,[]\}}$, $P_1\subseteq P_2$ iff there exists a homomorphism from P_2 to P_1.

1) Such a symbol always exists as the alphabet Σ is infinite.

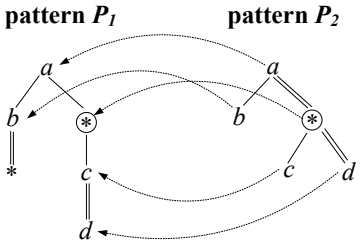


Figure A2 An example of homomorphism from pattern P_2 to P_1

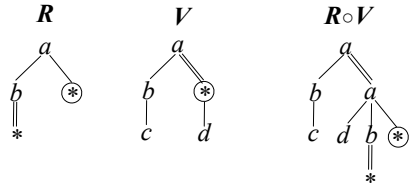


Figure A3 An example of pattern composition

Table A1 Summary of the symbols for notations

Notations	Default Symbols
Pattern	$P, P_1, P_2, \ldots$
Boolean Pattern	$B, B_1, B_2, \ldots$
View	$V, V_1, V_2, \ldots$
Tree	$t, t_1, t_2, \ldots$
Node	n, u, v, w
Label	$l, a, b, c, \ldots$
Embedding	$e, e_1, e_2, \ldots$
Homomorphism	$h, h_1, h_2, \ldots$

For convenience, we summarize the symbols used for all the key notations in Table **??**.

Appendix B Proof of Theorem 1

We first show that for any pattern P and any view V with roots being output nodes, $P \subseteq V$ iff V determines P.

• Assume first that $P \subseteq V$, then we show that P is a rewriting of P using V and thus determinacy holds. Since $P \subseteq V$, $label(out(V))$ (*i.e.,* $label(root(V))$) is either $*$ or the same with $label(root(P))$ (*i.e.,* $label(out(P))$). Thus $P \circ V$ is not empty. We next show that $P \circ V \equiv P$. Observe that the result of evaluating a pattern whose output node is root over a tree t is either $\{t\}$ or $\emptyset$, depending on whether an embedding exists. Therefore to show equivalence (or containment), we only need to focus on the existence of embeddings. For any tree t, if there is an embedding e from $P \circ V$ to t, then we can obtain an embedding e' from P to t by dropping the V part of $P \circ V$. Hence $P \circ V \subseteq P$. On the other hand, if there exists an embedding e_1 from P to t, by $P \subseteq V$, there also exists an embedding e_2 from V to t. Putting e_1 and e_2 together we can construct an embedding from $P \circ V$ to t and thus $P \subseteq P \circ V$. Therefore $P \circ V \equiv P$.

• Conversely, assume that V determines P, and we show $P \subseteq V$ by proof-by-contradiction. If $P \not\subseteq V$, then there exists a non-empty tree t such that $P(t) = \{t\}$ while $V(t) = \emptyset$. Let $t' = \varepsilon$. We have that $P(t') = V(t') = \emptyset$. That is, $V(t) = V(t')$ but $P(t) \neq P(t')$, contradicting with determinacy. Hence $P \subseteq V$.

Consider a pattern P whose root is output node and its boolean version $\widehat{P}$. For any tree t, if there exists an embedding from P (thus $\widehat{P}$) to t, then $P(t) = \{t\}$ (similarly $\widehat{P}(t) = \{()\}$), and $P(t) = \emptyset$ (similarly $\widehat{P}(t) = \emptyset$), otherwise. This means that patterns with roots being output nodes have the similar behavior as boolean patterns. Therefore the containment problem of the former reduces to that of the latter. Specifically, $P_1 \subseteq P_2$ iff $\widehat{P_1} \subseteq \widehat{P_2}$ for any patterns P_1 and P_2 with roots being output nodes. The containment problem for boolean patterns is coNP-complete [**?**]. Thus the containment problem for patterns with roots being output nodes is also coNP-complete. We thus have that the determinacy for patterns with roots being outputs nodes is coNP-complete. Since this is a special case of the determinacy for $\mathrm{XP}^{\{*,//,[]\}}$, we conclude that determinacy for $\mathrm{XP}^{\{*,//,[]\}}$ is coNP-hard.

Appendix C Proof of Theorem 2

We then show that for patterns and views in $\mathrm{XP}^{\{*,//,[]\}}$, determinacy does not necessarily imply the existence of a rewriting. That is, $\mathrm{XP}^{\{*,//,[]\}}$ is not complete for rewriting. In fact, we will show a stronger result: even for its sub-fragments, determinacy does not always imply the existence of a rewriting either. Here we consider the three sub-fragments, namely $\mathrm{XP}^{\{//,[]\}}$, $\mathrm{XP}^{\{*,[]\}}$ and $\mathrm{XP}^{\{*,//\}}$. Note that a pattern in $\mathrm{XP}^{\{//,[]\}}$ does not have $*$ labeled nodes, a pattern in $\mathrm{XP}^{\{*,[]\}}$ does not have descendant edges, and a pattern in $\mathrm{XP}^{\{*,//\}}$ does not have branches.

Recall that the rewriting existence problem is to check , given a pattern $P \in \mathcal{L}$ and a view $V \in \mathcal{L}$, whether there exists a pattern $R \in \mathcal{L}$ such that $R \circ V \equiv P$. Although the exact complexity of rewriting existence problem when $\mathcal{L}$ is $\mathrm{XP}^{\{*,//,[]\}}$ remains open, Xu and Mözsoyoglu [**?**] have shown that when $\mathcal{L}$ is one of the three sub-fragments of $\mathrm{XP}^{\{*,//,[]\}}$, the problem is easily checkable in PTIME. The checking process is described as follows.

(1) Let n be the node on the selection path of P with the same height as the output node of V;
(2) Let R be the subpattern of P rooted at node n of pattern P;
(3) Check whether $R \circ V \equiv P$? If yes, there exists a rewriting; otherwise, no rewriting exists.

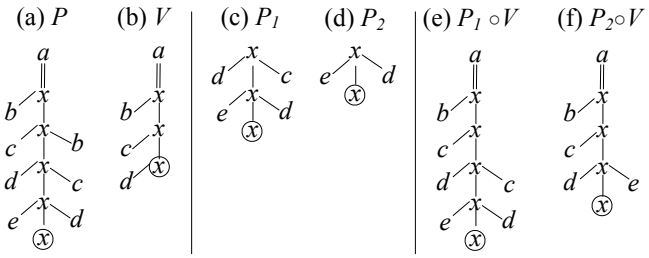


Figure C1 Example of the $\mathrm{XP}^{\{//,[]\}}$ case

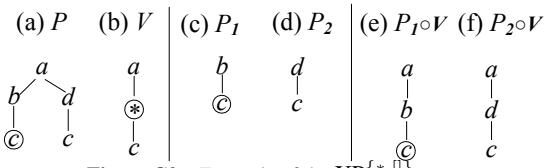


Figure C2 Example of the $\mathrm{XP}^{\{*,[]\}}$ case

Consider pattern $P=a//*//*//b/c$ and view $V=a//*//b$. The height of $out(V)$ on the selection path of V is 2. We then find the subpattern R of P rooted at the selection node with the same height, *i.e.,* $R=*//b/c$. Since the composition $R\circ V=a//*//b//b/c$ does not equal to P, there exists no rewriting of P using V.

We now show that, even for the three sub-fragments of $\mathrm{XP}^{\{*,//,[]\}}$, determinacy does not always imply the existence of a rewriting. We demonstrate this by the following three counter examples, each of which is for one of the three sub-fragments. The first example for $\mathrm{XP}^{\{//,[]\}}$ is taken from [**?**] with slight modifications.

Example 1 ($\mathrm{XP}^{\{//,[]\}}$ Case)**.** Consider the pattern P and view V shown in Figure **??** (a) and (b) respectively. One can verify that P has no rewriting using V: the composition of the subpattern $(P)^{n_4}_{sub}$, rooted at the fourth-node on the selection path of P, and V is not equivalent to P. However, for any tree t, we can always find $P(t)$ using the result of $V(t)$, which implies that V determines P. Consider the patterns P_1 and P_2 shown in Figure **??** (c) and (d) respectively. We evaluate the intersection $P_1\cap P_2$ over the subtrees in $V(t)$. It can be verified that $(P_1\cap P_2)(V(t))=P_1(V(t))\cap P_2(V(t))=P(t)$. Indeed, consider the composition $P_1\circ V$ and $P_2\circ V$ shown in Figure **??** (e) and (f) respectively. One can verify that $P(t)\equiv P_1\circ V(t)\cap P_2\circ V(t)$ for any tree t.

Example 2 ($\mathrm{XP}^{\{*,[]\}}$ Case)**.** Consider the pattern $P=a[d/c]/b/c$ and view $V=a/*[c]$ shown in Figure **??** (a) and (b) respectively. Again, P has no rewriting using V according to Xu and Mözsoyoglu's checking process. But given any tree t, we can still compute $P(t)$ from $V(t)$ as follows. We first check the boolean pattern P_2 (Figure **??** (d)) over the subtrees in $V(t)$. If P_2 is satisfied by at least one subtree in $V(t)$, we then evaluate $P_1=b/c$ (Figure **??** (c)) over the subtrees in $V(t)$ and the result of $P_1(V(t))$ is exactly the same with $P(t)$.

Example 3 ($\mathrm{XP}^{\{*,//\}}$ Case)**.** Consider the case that $P=a/*/b$ and $V=a//*$. Again, there exists no rewriting of P using V. Observe that for any tree t whose root is labeled with symbol a, V returns all the subtrees of t except t itself. Thus if $V(t)\neq\emptyset$, we can restore the original tree t from $V(t)$ and then evaluate P on t to get $P(t)$. More specifically, the original tree t can be restored from $V(t)$ as follows: (1) apply the following first-order logic formula $\phi(x):\neg\exists y\ Descendant(root(x),root(y))$ on $V(t)$ where $Descendant(u,v)$ is a predicate denoting that node u is a descendant of node v, and (2) then combine all the subtrees $x\in V(t)$ such that $\phi(x)$ holds in $V(t)$ by introducing a common root labeled with symbol a. The combined tree is exactly the original tree t.

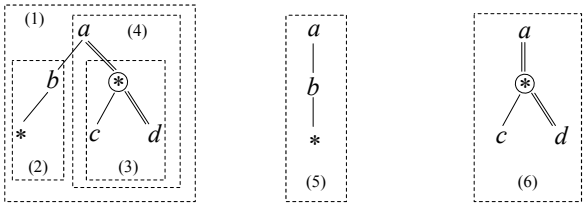
Those examples imply that we may need a more powerful language to answer a pattern using a view, even when the patterns and views are defined in a language less powerful than $\mathrm{XP}^{\{*,//,[]\}}$. Consider Example **??**. As explained in this example, according to the rewriting existence checking process described above, there exists no rewriting $R\in\mathrm{XP}^{\{//,[]\}}$ such that $R\circ V\equiv P$, but we can find a query $Q\in\mathrm{XP}^{\{//,[],\cap\}}$ to answer the pattern. Furthermore, it has been shown that $\mathrm{XP}^{\{//,[]\}}$ is properly contained in $\mathrm{XP}^{\{//,[],\cap\}}$. Consequently, $\mathrm{XP}^{\{//,[]\}}$ is not complete for $\mathrm{XP}^{\{//,[]\}}$-to-$\mathrm{XP}^{\{//,[]\}}$ rewriting. Now consider Example **??**. We analyze the expressive power of languages needed to answer the pattern in terms of logics. The key point is to find those subtrees st whose roots are directly connected to the root of tree t. As shown in the example, this can be expressed by a logic formula: $\phi(x):\neg\exists y\ Descendant(root(x),root(y))$. Thus the expressive power needed at least contains *negation* $\neg$. However, Benedikt et al. [**?**] have shown that the whole fragment $\mathrm{XP}^{\{*,//,[],\cup\}}$, which properly contains $\mathrm{XP}^{\{*,//,[]\}}$ and thus $\mathrm{XP}^{\{*,//\}}$, is equivalent in expressive power to a version of *positive-existential first-order logic* consisting of *conjunction* $\wedge$, *disjunction* $\vee$ and *existential quantification* $\exists$, without *negation* $\neg$. That is, $\mathrm{XP}^{\{*,//\}}$, even $\mathrm{XP}^{\{*,//,[]\}}$, cannot express *negation* $\neg$. Hence we can conclude that $\mathrm{XP}^{\{*,//\}}$, even $\mathrm{XP}^{\{*,//,[]\}}$, is not complete for $\mathrm{XP}^{\{*,//\}}$-to-$\mathrm{XP}^{\{*,//\}}$ rewriting. Since $\mathrm{XP}^{\{*,//\}}$ is a special case of $\mathrm{XP}^{\{*,//,[]\}}$, we thus have that $\mathrm{XP}^{\{*,//,[]\}}$ is not complete for $\mathrm{XP}^{\{*,//,[]\}}$-to-$\mathrm{XP}^{\{*,//,[]\}}$ rewriting, either.

Notice that in Example **??**, we need two patterns P_1 and P_2 to answer pattern P using the result of view V, but only P_1 contains an output node and P_2 is a boolean pattern. In fact, we will show that, for $\mathrm{XP}^{\{*,[]\}}$ case, even there exists no $\mathrm{XP}^{\{*,[]\}}$ rewriting of P using V according to Xu and Özsoyoglu's constructing method [**?**], we can still find an $\mathrm{XP}^{\{*,[]\}}$ pattern to compute the pattern P from the view V if, without loss of generality, we combine the view result into a single tree by simply introducing a common root. We will further explain this in Appendix F.

Appendix D Proof of Proposition 1

We first give an analysis of tree patterns to help identify necessary conditions. As we have already discussed patterns and views with roots being their output nodes in Appendix B, we assume that in the following, the output nodes of patterns and views are all below their roots.

For a pattern P whose root has m children $c_1,...,c_m$, we denote by $SP_1,...,SP_m$ the subpatterns $(P)^{c_1}_{sub},...,(P)^{c_m}_{sub}$, respectively. We denote by $P_{[j]}$ the subpattern of P obtained from SP_j by adding the edge $(root(P),c_j)$ and call it a $branch$ of P. Especially, for a branch



(1) P; (2) SP_1; (3) SP_2; (4) P-$P_{[1]}$; (5) $P_{[1]}$; (6) $P_{[2]}$

Figure D1 Examples of notations on patterns

$P_{[j]}$ of P, we denote by $P-P_{[j]}$ the subpattern obtained by removing SP_j from P. If a subpattern $P_{[j]}$ $(1\leqslant j\leqslant m)$ contains the output node of P, we refer to this unique branch as $P_{[o]}$. Furthermore, we denote by $\mathcal{B}(P)$ the set of all the branches of P. Note that all branches except $P_{[o]}$ are actually boolean patterns. We occasionally call $P_{[o]}$ the *output branch* and the others the *boolean branches*. We illustrate those notations with examples shown in Figure **??**. For the homomorphism between two patterns and their branches, we have the following.

Lemma 3. There exists a homomorphism from pattern P_2 to pattern P_1 iff (i) for each boolean branch $P_{2[i]}\in\mathcal{B}(P_2)$, there exists a branch $P_{1[j]}\in\mathcal{B}(P_1)$ such that a homomorphism exists from $\widehat{P}_{2[i]}$ to $\widehat{P}_{1[j]}$; and, moreover, (ii) there exists a homomorphism from $P_{2[o]}$ to $P_{1[o]}$.

We define branches on trees similarly to those on patterns. From the definition of embedding from patterns to trees, it is easy to verify the following results.

Lemma 4. Let P be a pattern. (i) For any tree t, $P(t)\neq\emptyset$ iff for each branch $P_{[i]}\in\mathcal{B}(P)$ there exists a branch $t_{[j]}\in\mathcal{B}(t)$ such that an embedding exists from $P_{[i]}$ to $t_{[j]}$. (ii) For any trees t_1 and t_2, if $t_1\subseteq t_2$, then $P(t_1)\subseteq P(t_2)$.

A pattern P is *minimal* if there exists no proper subpattern P' of P such that $P'\equiv P$ [**?**]. We assume in this paper that all patterns are minimal. An important result on pattern minimality is stated by the following Lemma.

Lemma 5 ([**?**])**.** If a boolean pattern B is minimal, then (i) each branch $B_{[i]}\in\mathcal{B}(B)$ is minimal, and (ii) for any branches $B_{[i]}, B_{[j]}\in\mathcal{B}(B)$ with $i\neq j$, $B_{[j]}\not\subseteq B_{[i]}$.

Intuitively, Lemma **??** (ii) means that if branch $B_{[i]}$ contains branch $B_{[j]}$, then $B_{[i]}$ is redundant and can be eliminated. Note that the above lemma focus only on boolean patterns. Based on this, we can readily derive the following result for patterns.

Lemma 6. If a pattern P is minimal, then (i) each branch $P_{[i]}\in\mathcal{B}(P)$ is minimal, and (ii) for any branches $P_{[i]}, P_{[j]}\in\mathcal{B}(P)$ such that $i\neq j$ and $P_{[i]}\neq P_{[o]}$, $\widehat{P}_{[j]}\not\subseteq\widehat{P}_{[i]}$.

We are now ready to proof Proposition 1. We show by contradiction that if one of the conditions does not hold, then V does not determine P.

(1) Suppose that $\widehat{P}\not\subseteq\widehat{V}$. Then there exists a non-empty tree t such that $t\models\widehat{P}$ but $t\not\models\widehat{V}$. Hence we have that $P(t)\neq\emptyset$ but $V(t)=\emptyset$. Let $t'=\varepsilon$. Then t and t' satisfy that $V(t)=V(t')$ but $P(t)\neq P(t')$, contradicting with determinacy.

(2) Suppose that there exists a branch $\widehat{P}_{[i]}\in\mathcal{B}(\widehat{P})$ such that $\widehat{P}_{[i]}\not\subseteq\widehat{V}_{[1]}\cup...\cup\widehat{V}_{[m]}$. We construct two trees t and t' satisfying $V(t)=V(t')$ but $P(t)\neq P(t')$. Here we do not consider the case that $P_{[i]}=P_{[o]}$ since we will prove later in (3) that $\widehat{P}_{[o]}\subseteq\widehat{V}_{[o]}$ which definitely ensures $\widehat{P}_{[o]}\subseteq\widehat{V}_{[1]}\cup...\cup\widehat{V}_{[m]}$.

As we assume that P is minimal, by Lemma **??** (ii), $\widehat{P}_{[j]}\not\subseteq\widehat{P}_{[i]}$ for any branch $\widehat{P}_{[j]}\in\mathcal{B}(\widehat{P})$ with $j\neq i$. Furthermore, by Lemma **??**, there exists a tree $t_j\in CMod(\widehat{P}_{[j]})$ such that $t_j\models\widehat{P}_{[j]}$ but $t_j\not\models\widehat{P}_{[i]}$ for each branch $\widehat{P}_{[j]}\in\mathcal{B}(\widehat{P})$ with $j\neq i$. Let t be the combination of such t_js (See Figure **??** (a)). It is easy to verify that $t\neq\varepsilon$: each t_j is from $CMod(\widehat{P}_{[j]})$ and all $\widehat{P}_{[j]}$s have a common root, which ensures that those t_js have the same root label. By the assumption that $\widehat{P}_{[i]}\not\subseteq\widehat{V}_{[1]}\cup...\cup\widehat{V}_{[m]}$, there exists a tree $t_i\in Mod(\widehat{P}_{[i]})$ satisfying $t_i\models\widehat{P}_{[i]}$ but $t_i\not\models\widehat{V}_{[1]}\cup...\cup\widehat{V}_{[m]}$. Let $t'=t+t_i$. If $label(root(P))=a$ where $a\in\Sigma$, then obviously $label(root(t))=label(root(t_i))=a$. If $label(root(P))=*$, then by the definition of canonical models (Recall Section **??**), $label(root(t))=z$. In this case we replace $label(root(t_i))$ with z if $label(root(t_i))\neq z$ (Such replacement does not effect the property of t_i, *i.e.,* t_i is still from $Mod(\widehat{P}_{[i]})$ and satisfies $t_i\models\widehat{P}_{[i]}$ but $t_i\not\models\widehat{V}_{[1]}\cup...\cup\widehat{V}_{[m]}$). In this way we can assure that $t'\neq\varepsilon$. We next verify that $V(t)=V(t')$ but $P(t)\neq P(t')$.

- Clearly $t\subseteq t'$. By Lemma **??**(ii), $V(t)\subseteq V(t')$. We now show $V(t')\subseteq V(t)$. Because $t_i\not\models\widehat{V}_{[1]}\cup...\cup\widehat{V}_{[m]}$, for each branch $\widehat{V}_{[k]}\in\mathcal{B}(\widehat{V})$, $k=1,...,m$, $t_i\not\models\widehat{V}_{[k]}$. That is, there exists no embedding from $\widehat{V}_{[k]}$ to t_i for every branch $\widehat{V}_{[k]}\in\mathcal{B}(\widehat{V})$. Hence we can infer that any embedding e from V to t' producing subtree $(t')^n_{sub}$ is also an embedding from V to t producing the same subtree. Therefore $V(t')\subseteq V(t)$. We can thus conclude that $V(t)=V(t')$, as desired.
- Consider $P(t')$. By construction, t' is the combination of t_i and t_js with $t_i\in Mod(\widehat{P}_{[i]})$ and $t_j\in CMod(\widehat{P}_{[j]})$ for each $j\neq i$. We can infer that for each branch in $\mathcal{B}(\widehat{P})$, there is a branch of t' to which an embedding exists. Then by Lemma **??**(i), $P(t')\neq\emptyset$. Now consider $P(t)$. Note that t is the combination of those t_js and for each t_j, $t_j\not\models\widehat{P}_{[i]}$. In other words, there exists no embedding from $\widehat{P}_{[i]}\in\mathcal{B}(\widehat{P})$ to any branch of t. Again by Lemma **??**(i), $P(t)=\emptyset$. Hence we have that $P(t)\neq P(t')$.

(3) Along the same lines as above, we assume by contradiction that $\widehat{P}_{[o]}\not\subseteq\widehat{V}_{[o]}$. We then construct two trees to show that determinacy does not hold.

From $\widehat{P}_{[o]}\not\subseteq\widehat{V}_{[o]}$ along with Lemma **??**, we know that there is a tree $t_o\in CMod(\widehat{P}_{[o]})$ such that $t_o\models\widehat{P}_{[o]}$ but $t_o\not\models\widehat{V}_{[o]}$. Let $t_{P\_}$ and t_V be any trees from $CMod(\widehat{P}-\widehat{P}_{[o]})$ and $CMod(\widehat{V})$ respectively. Let $t=t_V+t_{P\_}$ and let $t'=t+t_o$ (See Figure **??** (b)). One can easily verify that $t\neq t'\neq\varepsilon$.

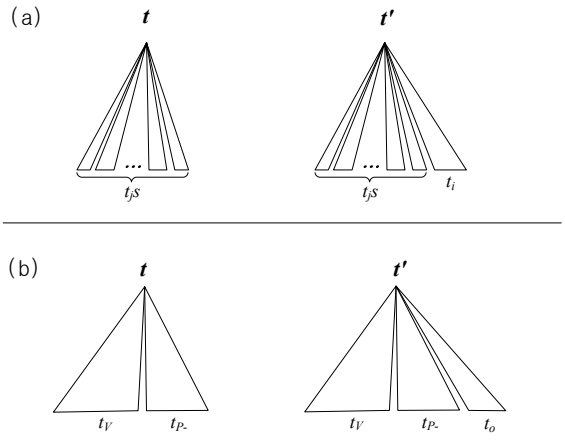


Figure D2 Constructions of trees in the proof of Proposition 1

• We first show that $V(t)=V(t')$. Clearly the existence of t_V in both t and t' ensures non-empty of $V(t)$ and $V(t')$. Moreover, $t\subseteq t'$ and by Lemma **??**(ii), $V(t)\subseteq V(t')$. We next verify the other direction $V(t')\subseteq V(t)$. Suppose that e' is an embedding from V to t' producing the subtree $(t')^n_{sub}$. Based on e' we construct an embedding e from V to t producing the same subtree as follows. For each branch $V_{[k]}\in\mathcal{B}(V)$, if the branch of t' mapped by $V_{[k]}$ via e' is not t_o, then let $e=e'$; otherwise let e map $V_{[k]}$ to its corresponding branch in the t_V part of t'. This branch always exists because $t_V\in CMod(V)$. Observe that $t_o\not\models\widehat{V}_{[o]}$ and thus there exists no embedding from $V_{[o]}$ to t_o. By construction, embedding $e=e'$ for nodes in branch $V_{[o]}$ which contains the output node of V, and therefore e produces the same subtree as e'. We hence have that $V(t')\subseteq V(t)$.

• To show $P(t)\neq P(t')$, it suffices to find a subtree st such that $st\notin P(t)$ but $st\in P(t')$. Notice that $t'=t+t_o=t_V+t_{P\_}+t_o$ with $t_{P\_}\models\widehat{P}-\widehat{P}_{[o]}$ and $t_o\models\widehat{P}_{[o]}$. One can readily verify that there is an embedding from P to t' which maps subpattern $P-P_{[o]}$ to $t_{P\_}$ of t' and subpattern $P_{[o]}$ to t_o. Let n_o be the node of t_o mapped by the output node of P. Then we have the subtree $(t_o)^{n_o}_{sub}\in P(t')$. However since t_o does not appear in tree t, the subtree rooted at node n_o of t_o could not be contained in $P(t)$.

This completes the proof of Proposition 1.

Appendix E Proof of Proposition 2

Let $t=\tau(P)$. It is easy to verify that $\Sigma(t)\setminus\{z\}=\Sigma(P)$ (recall that in the construction of $\tau(P)$, occurrences of $*$ are replaced with a new symbol z), $label(root(t))=label(root(P))$ if $label(root(P))\neq *$, and $height(t)=height(P)$. If V determines P, then from Proposition 1 (1), it follows that $\widehat{P}\subseteq\widehat{V}$. We then have $t\models\widehat{P}$ and thus $t\models\widehat{V}$. The existence of an embedding from $\widehat{V}$ (thus V) to t implies that: (1) for any symbol $l\in\Sigma(V)$ (note that $l\neq *$), $l\in\Sigma(t)\setminus\{z\}$, and thus $l\in\Sigma(P)$; (2) if $root(P)$ has a label $l\neq *$, then $root(t)$ has the same label and thus so does $root(V)$ or $root(V)$ is labeled with $*$; (3) any path of length k from the root to a leaf of V is mapped to a path of length $k'\geqslant k$ in t. We therefore conclude that (1) $\Sigma(V)\subseteq\Sigma(P)$; (2) $label(root(V))=label(root(P))$ or $label(root(V))=*$; and (3) $height(V)\leqslant height(P)$.

Let $t=\tau(P)$ and let n be the node of t that corresponds to the output node of P. By the construction of t, the height of n in t is exactly $depth(P)$.If V determines P, then $V(t)$ contains node n. Thus there must exist an embedding from V to t that maps the output node $out(V)$ of V to a node u on the path from root to n in t. It then follows that $height(u)\leqslant height(n)$. Notice that every embedding from a pattern to a tree maps a node with height h to a node with height no less than h. That is, $height(out(V))\leqslant height(u)$. Together we have that $height(out(V))\leqslant height(n)$. Since $height(out(V))=depth(V)$, $height(n)=depth(P)$, we then have that $depth(V)\leqslant depth(P)$.

Appendix F Proof of Proposition 3

We first prove some Lemmas.

Lemma 7. Consider two patterns P_1 and P_2, and a homomorphism h from $\widehat{P}_2$ to $\widehat{P}_1$. Let n be the node of P_1 mapped by the output node of P_2 via h. Then $\widehat{P}_1\subseteq(P_1)^n_{sub}\circ P_2$, and moreover, $P_1\subseteq(P_1)^n_{sub}\circ P_2$ if $(P_1)^n_{sub}$ contains the output node of P_1.

Proof. First, observe that the composition pattern $(P_1)^n_{sub}\circ P_2$ is not empty. Second, based on h, we can construct a homomorphism h' from $(P_1)^n_{sub}\circ P_2$ to $\widehat{P}_1$ as follows: h' maps each node from $(\widehat{P}_1)^n_{sub}$ part of $(P_1)^n_{sub}\circ P_2$ to the same node from $(\widehat{P}_1)^n_{sub}$ part of $\widehat{P}_1$, and maps the rest nodes from P_2 part of $(P_1)^n_{sub}\circ P_2$ to the corresponding nodes of P_1 as h does. Hence, we have that $\widehat{P}_1\subseteq(P_1)^n_{sub}\circ P_2$. If $(P_1)^n_{sub}$ contains the output node of P_1, then h' is also a homomorphism from $(P_1)^n_{sub}\circ P_2$ to P_1 and thus $P_1\subseteq(P_1)^n_{sub}\circ P_2$. □

Lemma 8. For any boolean patterns B_1 and B_2 in $\mathrm{XP}^{\{*,[]\}}$, if B_2 can be embedded into $\tau(B_1)$, then $B_1\subseteq B_2$.

Proof. By the definitions of embedding and homomorphism, and the fact that there are no descendant edges in B_1 and B_2, we can easily infer that if B_2 can be embedded into $\tau(B_1)$ then there must exist a homomorphism from B_2 to B_1. Thus by Lemma **??**, $B_1\subseteq B_2$.

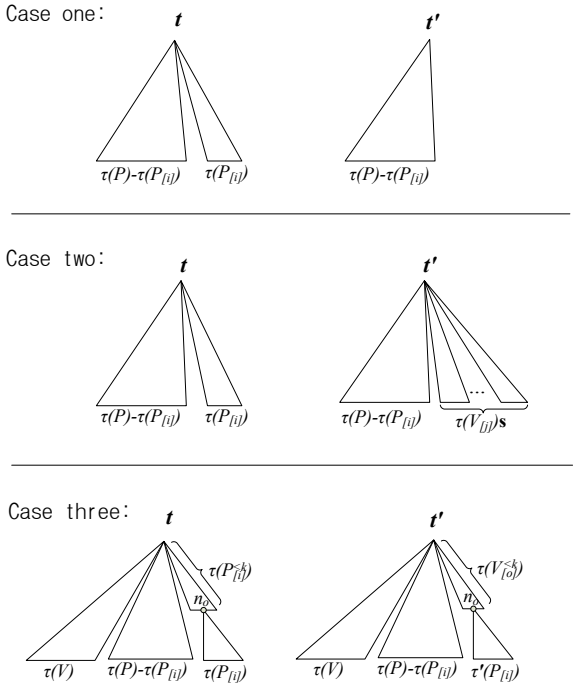


Figure F1 Constructions of trees in the proof of Proposition 3

Lemma 9. Let P be a minimal pattern in $\mathrm{XP}^{\{*,[]\}}$. (i) For any branch $P_{[i]} \in \mathcal{B}(P)$ and any branch $\tau(P_{[j]}) \in \mathcal{B}(\tau(P))$ such that $i \neq j$ and $P_{[i]} \neq P_{[o]}$, there exists no embedding from $P_{[i]}$ to $\tau(P_{[j]})$; (ii) For any branch $P_{[i]} \in \mathcal{B}(P)$, we have that $P(t) \neq P(t')$ where tree $t = \tau(P)$ and tree $t' = \tau(P) - \tau(P_{[i]})$.

Proof. (i) Suppose by contradiction that there is a branch $P_{[i]} \in \mathcal{B}(P)$ with $P_{[i]} \neq P_{[o]}$ and a branch $\tau(P_{[j]}) \in \mathcal{B}(\tau(P))$ with $i \neq j$ such that an embedding exists from the former to the later. Then by Lemma **??**, $\widehat{P}_{[j]} \subseteq \widehat{P}_{[i]}$. This, however, contradicts with Lemma **??** (ii) which states that if a pattern is minimal then no inclusion relation exists between any two branches (except the one containing the output node) of this pattern.

(ii) Clearly $P(t) \neq \emptyset$ and it contains the subtree of $\tau(P)$ rooted at $\tau(out(P))$ (recall that $\tau(out(P))$ is the corresponding node of $out(P)$ in $\tau(P)$ where $out(P)$ is the output node of P). If $P_{[i]} \neq P_{[o]}$, then from Lemma **??**(i) and Lemma **??**(i) we have that $P(t') = \emptyset$ and hence $P(t) \neq P(t')$. If $P_{[i]} = P_{[o]}$, then by construction, t' does not have the branch $\tau(P_{[o]})$, which implies that the subtree of $\tau(P)$ rooted at $\tau(out(P))$ could not be contained in $P(t')$. Thus we still have that $P(t) \neq P(t')$. □

We are now ready to prove Proposition 3.

As explained before, if P and V are in $\mathrm{XP}^{\{*,[]\}}$, then the DAG pattern R is still a tree pattern in $\mathrm{XP}^{\{*,[]\}}$. Furthermore, by Lemma **??**, $P \subseteq R$. Thus if $R \not\equiv P$, we must have that $R \not\subseteq P$. Then according to Lemma **??**, no homomorphism exists from P to R. This means that, by Lemma **??**, (1) there exists a boolean branch $P_{[i]} \in \mathcal{B}(P)$ such that $\widehat{P}_{[i]}$ could not be mapped to any branch $\widehat{R}_{[k]} \in \mathcal{B}(\widehat{R})$ via a homomorphism; or (2) $P_{[o]}$ could not be mapped to $R_{[o]}$ via a homomorphism. We next show that in either of these two cases, V does not determine P.

One can easily verify that if the root of V is its output node, then V determines P iff $\widehat{P} \subseteq \widehat{V}$. In this case, Algorithm 1 is clearly sound and complete. If the root of P is its output node, then P behaves as a boolean pattern. In this case we can assume that there is no output branch $P_{[o]}$ and the proof coincides with Part (1) (See below). If both the output nodes of V and P are below their roots, then one can verify that, in this case, V determines P only if their roots have the same label. Thus in the following we consider that both the output nodes of V and P are descendants of their roots and assume that the root of V has the same label with that of P. The purpose of this assumption is to ensure non-emptiness of the trees constructed in the following proof.

(1) Assume that $P_{[i]}$ is the boolean branch of P such that for any branch $R_{[k]}$ of R, no homomorphism exists from $\widehat{P}_{[i]}$ to $\widehat{R}_{[k]}$. Recall the construction of R in Algorithm 1. If the output node $out(V)$ of V could be mapped to m nodes $n_1, ..., n_m$ of P via homomorphisms from $\widehat{V}$ to $\widehat{P}$, then R consists of all the boolean branches $V_{[j]}$ of V and the other m branches, each of which is obtained from $V_{[o]}$ by composed with the subpattern $(P)^{n_i}_{sub}$ of P. We can easily verify that if there is no homomorphism from $\widehat{P}_{[i]}$ to any branch $\widehat{R}_{[k]}$ of $\widehat{R}$, then for any branch $\widehat{V}_{[j]}$ of $\widehat{V}$, no homomorphism exists from $\widehat{P}_{[i]}$ either.

Now consider the homomorphisms from $\widehat{V}$ to $\widehat{P}$. There are three cases concerning the boolean branch $P_{[i]}$ of P and the homomorphisms from $\widehat{V}$ to $\widehat{P}$.

Case one: $\widehat{P}_{[i]}$ could not be mapped by any branch of $\widehat{V}$ via any homomorphism from $\widehat{V}$ to $\widehat{P}$. By Lemma **??**, no embedding exists from any branch of V to $\tau(P_{[i]})$. We can infer that eliminating branch $\tau(P_{[i]})$ from tree $\tau(P)$ does not effect the evaluation result of V. That is, let tree $t = \tau(P)$ and tree $t' = \tau(P) - \tau(P_{[i]})$ (See Figure **??** Case one), and we have $V(t) = V(t')$. However, by Lemma **??**(ii), $P(t) \neq P(t')$. Thus V does not determine P.

Case two: $\widehat{P}_{[i]}$ could be mapped by some boolean branch $\widehat{V}_{[j]}$ (note that $V_{[j]} \neq V_{[o]}$) of $\widehat{V}$ via some homomorphism from $\widehat{V}$ to $\widehat{P}$. But for each of such $\widehat{V}_{[j]}$, as by the assumption and as analyzed before, $V_{[j]}$ could not be mapped by $\widehat{P}_{[i]}$. That is, by Lemma **??**, $\widehat{V}_{[j]} \not\subseteq \widehat{P}_{[i]}$ and furthermore by Lemma **??**, no embedding exists from $\widehat{P}_{[i]}$ to tree $\tau(V_{[j]})$. Let t_V be the combination of such $\tau(V_{[j]})$s. Let $t = \tau(P)$ and let t' be the tree obtained from t by replacing the branch $\tau(P_{[i]})$ by tree t_V (See Figure **??** Case two). We next show that $V(t) = V(t')$ but $P(t) \neq P(t')$.

- Clearly $P(t) \neq \emptyset$. Now consider $P(t')$. On one hand, by construction, branch $P_{[i]}$ could not be embedded into the t_V part of t'; on the other hand, by Lemma **??**(i), $P_{[i]}$ could not be embedded into the $\tau(P) - \tau(P_{[i]})$ part of t' either. In all, we have that $P_{[i]}$ could not be embedded into any branch of tree t'. Along with Lemma **??**(i), we conclude that $P(t') = \emptyset$. Thus $P(t) \neq P(t')$.
- Now consider the evaluation of V. By construction, the t_V part of t' ensures that if V can be embedded into t then it can also be embedded into t', and vice versa. By the assumption, there is no homomorphism from branch $\widehat{V}_{[o]}$ to branch $\widehat{P}_{[i]}$. Thus by Lemma **??**, $\widehat{V}_{[o]}$ can not be embedded into branch $\tau(P_{[i]})$ of t. Along the same lines we can verify that $\widehat{V}_{[o]}$ can not be embedded into any branch of the t_V part of t'. Because if it could, then it could be mapped to some branch $V_{[j]}$ via a homomorphism, and furthermore could be mapped to $P_{[i]}$, contradicting with the assumption. Notice that $V_{[o]}$ contains the output node of V and that $t - \tau(P_{[i]}) = t' - t_V$. Hence we conclude that $V(t) = V(t')$.

Case three: $\widehat{P}_{[i]}$ could be mapped by the branch $\widehat{V}_{[o]}$ of $\widehat{V}$ via some homomorphism from $\widehat{V}$ to $\widehat{P}$. We assume first that there is only one node of P (also of $P_{[i]}$) that is mapped by the output node of V via all the homomorphisms from $\widehat{V}$ to $\widehat{P}$, and let n_o be this node. By the construction of R, we have that the composition $(P)^{n_o}_{sub} \circ V_{[o]}$ is a branch of R. Suppose that the height of the output node V_o is k (recall that the height of a node is the number of edges on the path from the root to that node). Since there are no descendant edges, the height of node n_o is also k. Let $V^{<k}_{[o]}$ and $P^{<k}_{[i]}$ be the subpatterns obtained from $V_{[o]}$ and $P_{[i]}$ by pruning the subpatterns rooted at $out(V)$ and n_o, respectively. We can infer that there exists no homomorphism from $P^{<k}_{[i]}$ to $V^{<k}_{[o]}$. Because, if exists, there must exist a homomorphism from branch $P_{[i]}$ of P to branch $(P)^{n_o}_{sub} \circ V_{[o]}$ of R, contradicting with the assumption that $P_{[i]}$ could not be mapped to any branch of R via a homomorphism. Thus by Lemma **??**, $P^{<k}_{[i]}$ could not be embedded into $\tau(V^{<k}_{[o]})$. Consider the canonical tree $\tau(P_{[i]})$. Let $\tau'(P_{[i]})$ be obtained from $\tau(P_{[i]})$ by replacing the $\tau(P^{<k}_{[i]})$ part by $\tau(V^{<k}_{[o]})$. One can then easily infer that $P_{[i]}$ could not be embedded into $\tau'(P_{[i]})$ either.

We now construct two trees t and t' to show that determinacy does not hold. Let t be the combination of $\tau(V)$, $\tau(P) - \tau(P_{[i]})$ and $\tau(P_{[i]})$. Let t' be the combination of $\tau(V)$, $\tau(P) - \tau(P_{[i]})$ and $\tau'(P_{[i]})$ (See Figure **??** Case three).

- Clearly $P(t) \neq \emptyset$. Now consider $P(t')$. As analyzed before, branch $P_{[i]}$ could not be embedded into branch $\tau'(P_{[i]})$ of t'. As we assume that P is minimal and that $P_{[i]} \neq P_{[o]}$, by Lemma **??**(i), $P_{[i]}$ could not be embedded into any branch of the $\tau(P) - \tau(P_{[i]})$ part of t'. Furthermore by assumption that there is no homomorphism from $P_{[i]}$ to any branch of R and by the construction of R from V, we can infer that $P_{[i]}$ could not be embedded into the $\tau(V)$ part of t' either. Along with Lemma **??**(i), we have that $P(t') = \emptyset$ and thus $P(t) \neq P(t')$.
- Consider the evaluation of V. Note that the $\tau(V)$ part in both t and t' ensures that $V(t) \neq \emptyset$ and $V(t') \neq \emptyset$. Furthermore, the node n_o which is mapped by the output node of V induces identical subtrees in t and t'. We can infer that $V(t) = V(t')$.

If the output node of V could be mapped to more than one node of $P_{[i]}$ via homomorphisms from $\widehat{V}$ to $\widehat{P}$, then for each such node, we construct a branch $\tau'(P_{[i]})$ from $\tau(P_{[i]})$ along the same lines as above, and combine the constructed branch with t'. One can verify that in this way we can still assure that $V(t) = V(t')$ but $P(t) \neq P(t')$.

(2) Assume that $P_{[o]}$ could not be mapped to $R_{[o]}$ via a homomorphism. By Proposition 1 (iii), if V determines P then $\widehat{P}_{[o]} \subseteq \widehat{V}_{[o]}$. Furthermore by Lemma **??**, there must exist a homomorphism from $\widehat{V}_{[o]}$ to $\widehat{P}_{[o]}$. We assume that n_o is the only node of $P_{[o]}$ that is mapped by the output node $out(V)$ of $V_{[o]}$ via the homomorphisms from $\widehat{V}_{[o]}$ to $\widehat{P}_{[o]}$. We can infer that node n_o must be on the path from the root of P to the output node. Thus, by construction, branch $R_{[o]}$ is exactly the composition $(P)^{n_o}_{sub} \circ V_{[o]}$. Along the same lines as the proof of Case three in Part (1), we can construct two trees to show that if no homomorphism exists from $P_{[o]}$ to $R_{[o]}$, then V does not determine P. If the output node $out(V)$ of $V_{[o]}$ could be mapped to more than one node of $P_{[o]}$, then again, along the same lines as above, we can still construct two trees to show that determinacy does not hold. We omit the details here to avoid repetition.

This completes the proof of Proposition 3.

References

1 Miklau G, Suciu D. Containment and equivalence for a fragment of XPath. Journal of the ACM, 2004, 51 (1):2–45
2 Xu W, MÖzsoyoglu Z. Rewriting XPath queries using materialized views. In: Proceedings of the International Conference on Very Large Databases (VLDB). Trondheim: ACM, 2005. 121–132
3 Flesca S,Furfaro F, Masciari E. On the minimization of XPath queries. Journal of the ACM, 2008, 55 (1):2:1–2:46
4 Afrati F, Chirkova R, Gergatsoulis M, et al. On rewriting XPath queries using views. In: Proceedings of the International Conference on Extending Database Technology (EDBT). Saint Petersburg: ACM, 2009. 168–179
5 Ameryahia S, Cho S, Lakshmanan L V, Srivastava D. Minimization of tree pattern queries. In: Proceedings of the ACM Symposium on Management of Data (SIGMOD). Santa Barbara: ACM, 2001. 497–508
6 Wood P T. Minimising simple XPath expressions. In: Proceedings of the International Workshop on the Web and Databases (WebDB). 2001. 13–18
7 Wang J, Yu J X, Liu C. Independence of containing patterns property and its application in tree pattern query rewriting using views. World Wide Web, 2009, 12 (1):87–105
8 Flesca S, Furfaro F, Masciari E. On the minimization of XPath queries. In: Proceedings of the International Conference on Very Large Databases (VLDB). Berlin: ACM, 2003. 153–164

9 Benedikt M, Fan W, Kuper G, Structural properties of xpath fragments. In: Proceedings of the International Conference on Database Theory (ICDT). Siena: ACM, 2003. 79–95