View-Based Tree-Language Rewritings for XML

Laks V.S. Lakshmanan[1] and Alex Thomo[2]

[1] University of British Columbia, Vancouver, BC, Canada, laks@cs.ubc.ca
[2] University of Victoria, Victoria, BC, Canada, thomo@cs.uvic.ca

Abstract. We study query rewriting using views (QRV) for XML. Our queries and views are regular tree languages (RTLs) represented by tree automata over marked alphabets, where the markers serve as "node selectors". We formally define query rewriting using views for RTLs and give an automata-based algorithm to compute the maximally contained rewriting. The formalism we use is equal in power with Monadic Second Order (MSO) logic, and our algorithm for computing QRV is the first to target this expressive class. Furthermore we prove a tight lower bound, thus showing that our algorithm is optimal. Another strength of our automata-based approach is that we are able to cast computing QRV into executing a sequence of intuitive operations on automata, thus rendering our approach practical as it can be easily implemented utilizing off-the-shelf automata toolboxes. Finally, we generalize our framework to account for more complex queries in the spirit of the `FOR` clause in XQuery. For this generalization as well, we give an optimal algorithm for computing the maximally contained rewriting of queries using views.

1 Introduction

Query rewriting using views (QRV) is a fundamental problem that finds wide applications in query optimization, data integration, data warehousing, security, and other critical database services. In this paper we study QRV for node-selecting queries and views, over XML trees. As there are several variants of views for answering queries we begin by illustrating the classical QRV problem we focus on in this paper.

Example 1. Suppose we have a very large collection of movies such as imdb.com, organized in a super-tree, with each movie being a sub-tree containing title, year, and characters. Suppose the character nodes branch out into actors playing the character. Sometimes, a movie contains characters played by more than one actor. We call these characters "multi-actor characters (MAC)". Consider a query which returns all the movie sub-trees having a MAC. Such movies are a small minority; their number is about 50. Assume that the result of this query is materialized into a view. Obviously this view is of tremendous help in answering some new queries, such as "find all the actors playing a MAC". We can rewrite this new query into "find all the actors playing a MAC in a movie having a MAC" and answer it on the view-extension instead of accessing the original database. The difference in performance is huge; using the view materialization takes 50

accesses, whereas using the original database takes hundreds of thousands of accesses (there are about 700,000 movies as per imdb.com). □

Due to its importance, QRV has been explored for fragments of XPath (cf. [5, 31, 16, 8, 2, 30]) and fragments of XQuery (cf. [32, 24, 4]). In this paper, we study the problem for queries and views represented by tree automata, which provide for a more general approach that can elegantly capture fine grained structure along vertical and horizontal axes. These queries can be easily specified using a DTD-like syntax, rendering them user-friendly devices for querying XML. While the techniques and tools introduced in the aforementioned works are interesting and well-founded for computing QRV for XPath and XQuery, they do not seem to extend to dealing with the more general setting of queries and views represented by tree automata, which have desirable properties with respect to expressivity in querying XML. In regard to expressivity, Neven and Schwentick [20, 21] and Schwentick [26] argue for formalisms as expressive as monadic second order logic (MSO) for specifying node-selecting queries. This is sometimes called a "golden standard" against which query formalisms should be measured up.

In this paper, we study the QRV problem for queries and views represented by automata equivalent in power to MSO. It is worth pointing out that being able to handle this target expressivity is one of the strengths of our approach to QRV. Another strength is the fact that computing QRV is cast into executing a sequence of intuitive operations on automata. This makes our solution quite practical, as it can be easily implemented on top of the readily available automata toolboxes (e.g. [12, 9]).

Our queries and views are first described as sets of tree-position pairs in order to facilitate the definitions and understanding of rewritings. In practice the queries and views are specified by finite tree automata over alphabets with marked symbols that serve the purpose of selecting nodes in XML trees. More specifically, for our constructions we use colors as markers, which makes the development easier. This formalism is equivalent to the formalism of querying trees using automata with boolean markings (ABM) [cf. [28], further developed in [23]].

Automata over colored alphabets provide us with critical advantages in computing QRV. The first advantage is the ability to express and construct a series of intermediate languages for obtaining query rewritings. Being able to use multiple colors for marking regions of interest in the trees of these intermediate languages is crucial to our approach. Our language constructions are also facilitated by the one-way nature of the automata we use. The second advantage is the ability to determinize the automata. This is key to our main construction for query rewriting.

Summarizing, we provide an algorithm for QRV for the general case where queries and views are implemented in an automata framework equal in power to MSO, to our knowledge, for the first time. Our algorithm runs in *singly-exponential time*. We show that QRV in our case is EXPTIME-hard, thus showing the optimality of our algorithm. Additionally, we show how to extend our results to reasoning about QRV for the more general case of queries returning

a forest of trees as an answer in the spirit of the `FOR` clause in XQuery. For this generalization as well, we provide an optimal algorithm to compute query rewritings using views.

We make the following contributions.

1. We define tree-pattern operators to cleanly express view-based rewriting as the solution to a view-query equation (Section 4).
2. Next, we present an algorithm for computing query rewritings using views. We define languages *over colored alphabets* and present a series of intermediate automata constructions, which we believe are of independent interest. We prove an EXPTIME lower bound for QRV, and show that our algorithm is optimal (Section 5).
3. We generalize our results to the case where queries select more than one subtree and produce a set of forests as output. We show that we are still able to compute view-based rewritings in singly-exponential time, thus being again optimal with respect to the above lower bound (Sections 6, 7, and 8).

2 Related Works

Automata theory has long been recognized as a useful tool for providing elegant solutions to challenging problems on XML (cf. [19, 21, 14, 22, 26, 17, 7]).

Two prominent automata-based approaches for querying XML data that have been proposed are (bottom-up) finite tree automata with selecting states (FTAS) introduced by Neven ([19], p. 128) and Frick et al. ([14]), and query automata (QA) introduced by Neven and Schwentick ([21]). Both formalisms are shown to capture precisely the queries definable in MSO and thus are the natural candidates for us to study the QRV problem in. However, two important issues prevent us from using either of these formalisms directly for our study of QRV: (1) Deterministic FTAS are too weak to express all queries expressible by nondeterministic FTAS; determinism is a key property that our techniques and algorithms rely on. (2) On the other hand, QA are deterministic, but two-way, and thus can go up and down a target tree multiple times, a feature that makes reasoning about QRV difficult.

Another nice standard for querying XML is Propositional Dynamic Logic (PDL) for trees ([1, 6]), which corresponds to Regular XPath. The latter was shown by ten Cate and Segoufin ([27]) to be not expressively complete for MSO. Regular XPath was extended later to μXPath by Calvanese et al. ([7]) to obtain the full power of MSO. It may be possible to use automata formalisms, such the one in [7] capturing μXPath, for QRV–this is an avenue we leave open for future exploration.

We close this section by briefly discussing two other works on query rewriting. Fan et al. ([11]) study rewritings of regular XPath queries defined over *virtual* views. This is a different problem from QRV that we consider here. The views in our case are materialized and the queries are over the original database. It would be interesting to see how our techniques could be used for the problem

studied by Fan et al., considering an alternate query formalism such as the one we propose that is complete for MSO.

Thomo and Venkatesh ([29]) use Visibly Pushdown Automata to model and rewrite (XML) schemas by using other schemas. Again, this problem is different from the problem we study in this paper.

3 Automata

We consider finite ordered trees – simply called *trees*. Also, we consider the trees to be *unranked*, which is to say that the nodes of the tree have an arbitrary (but finite) arity (cf. [10], p. 200). The nodes of the trees are labeled by symbols drawn from a fixed alphabet Σ. We denote by Υ the set of all trees over Σ. We use $a, b, \ldots, e$ to denote labels in Σ and $x, y, \ldots$, possibly with subscripts, to denote tree nodes. We denote by r_t the root of tree t, by σ_x the label of node x, and by N_t the set of nodes of a tree $t \in \Upsilon$.

Definition 1. *A non-deterministic, bottom-up,* finite tree automaton *(FTA) over Σ is a quadruple $\mathcal{A} = (S, \Sigma, F, \Delta)$, where S is a finite set of states, $F \subseteq S$ is a set of final states, and Δ is a finite set of transition rules of the form $H \xrightarrow{a} s$, where $H \subseteq S^*$ is a regular language over S, $a \in \Sigma$, and $s \in S$.*

Here, H is called a *horizontal* language. For simplicity, we blur the distinction between regular languages over S and the regular expressions used to specify them.

Definition 2. *A tree t is accepted by an FTA $\mathcal{A}$ if there exists a mapping $\mu : N_t \to S$ such that:*

1. *If $\mu(x) = s$, then there is a transition rule $H \xrightarrow{\sigma_x} s$ in Δ with $\mu(x_1) \ldots \mu(x_n) \in H$, where $x_1, \ldots, x_n$ are* all *the children of x in t in order.*
2. *If x is a leaf and $\mu(x) = s$, then there is a transition rule $H \xrightarrow{\sigma_x} s$ in Δ with $\epsilon \in H$.*
3. *$\mu(r_t) \in F$.*

A mapping μ as above specifies an *accepting run* of $\mathcal{A}$ on t. An accepting run can be considered to be a tree of the same shape as t whose nodes are labeled by states given by the mapping μ.

We denote by $L(\mathcal{A})$ the set (language) of trees accepted by $\mathcal{A}$. A tree language L is said to be accepted (or recognized) by an FTA $\mathcal{A}$ if $L = L(\mathcal{A})$. Tree languages recognized by FTAs are called *regular tree languages* (RTLs).

Example 2. Consider a collection of trees representing movies having, among other elements, one or more characters, each played by one or more actors. Let us use `m`, `t`, `y`, `c`, *and* `a` *to abbreviate* movie, title, year, character, *and* actor, *respectively.*

An FTA accepting movies having at least one character played by two or more actors is

$$\mathcal{A} = (\{s, s_{\mathtt{m}}, s_{\mathtt{c}}, s_{\mathtt{a}}\}, \Sigma, \{s_{\mathtt{m}}\}, \Delta)$$

where $\Sigma \supset \{\mathtt{m}, \mathtt{t}, \mathtt{y}, \mathtt{c}, \mathtt{a}\}$ *and* Δ *has the following transition rules:*

$$
\begin{aligned}
s^* s_{\mathtt{c}} s^* &\xrightarrow{\mathtt{m}} s_{\mathtt{m}} \\
s^* s_{\mathtt{a}} s_{\mathtt{a}} s^* &\xrightarrow{\mathtt{c}} s_{\mathtt{c}} \\
s^* &\xrightarrow{\mathtt{a}} s_{\mathtt{a}} \\
s^* &\xrightarrow{\Sigma} s.
\end{aligned}
$$

The last transition rule is a shorthand for saying that "we can go from s^* *to* s *on any symbol of* Σ*."*

Let now t *be the movie tree given in Figure 1, left. Clearly,* t *is accepted by* $\mathcal{A}$*. We assume that all the textual (unstructured data) have been identified (labeled) with a special symbol, say* $\mathtt{d} \in \Sigma$*. An accepting run of* $\mathcal{A}$ *on* t *is given in the same figure, right.*

We describe in Section 4 how to modify $\mathcal{A}$ *to* query *for those movies having a character played by two or more actors.* □

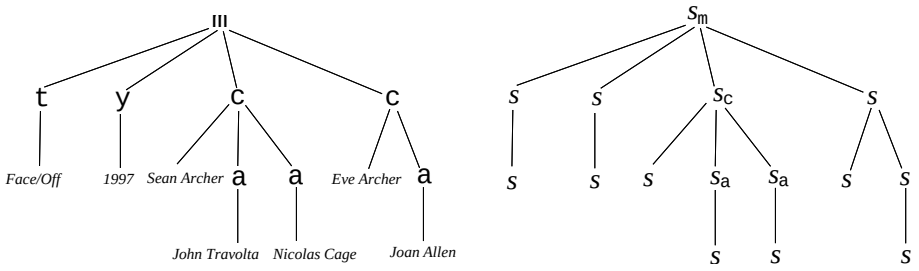

Fig. 1. A tree t and an accepting run of $\mathcal{A}$ on t.

It is worth noting that an FTA can be specified using extended DTDs (EDTDs), a syntax that may be more familiar to a user (cf. [10], p. 233).

Consider again Definition 1. The FTA $\mathcal{A}$ is called *deterministic* if $H_1 \cap H_2 = \emptyset$ for all transitions $H_1 \xrightarrow{a} s_1$ and $H_2 \xrightarrow{a} s_2$, where $s_1 \neq s_2$. In such a case, for any tree t, there can be at most one accepting run of $\mathcal{A}$ on t.

FTAs that never get "stuck" are called *complete*. Given an FTA, it can be verified that the determinization procedure of [10] (p. 204) produces a deterministic FTA that is complete. Given a tree t, a complete and deterministic FTA (CDFTA) $\mathcal{A}$ can read t in only one way, i.e., there is exactly one run of $\mathcal{A}$ on t.

An FTA is *normalized* if for each symbol-state pair (a, s), there is at most one transition $H \xrightarrow{a} s$. Each FTA can be transformed into a normalized FTA by unioning the left-hand sides of the transition rules labeled by the same symbol and having the same state on the right-hand side. If the FTA is deterministic, then it will remain so after this transformation.

Intersection. It is a well known fact that if L_1 and L_2 are RTLs, so is $L_1 \cap L_2$. However, the proof is typically done by encoding unranked trees into binary ones (cf. [10], p. 209). We can instead use a direct construction which preserves completeness and determinism. Given FTAs $\mathcal{A}_1$ and $\mathcal{A}_2$ for L_1 and L_2, respectively, we can construct an FTA $\mathcal{A}$ which recognizes $L_1 \cap L_2$. Furthermore, if $\mathcal{A}_1$ and $\mathcal{A}_2$ are complete and deterministic, so is $\mathcal{A}$ (see Appendix).

3.1 Targetedness

Here we introduce our notion of "targeted" FTAs. Targeted FTAs turn out to be important in computing rewritings of queries using views.

Definition 3. *A normalized FTA is* targeted *if each state is the target (right side) of at most one transition rule.*

Any normalized FTA $\mathcal{A} = (S, \Sigma, F, \Delta)$ can be transformed into a targeted one. For this, suppose that for a given state s, we have that

$$H_1 \xrightarrow{a_1} s, \ldots, H_k \xrightarrow{a_k} s$$

are the transition rules in Δ with target s. We introduce the operator $\uparrow_s$ applied on $\mathcal{A}$ with result $\mathcal{A} \uparrow_s$ which is another FTA $(S', \Sigma, F', \Delta')$ obtained from $\mathcal{A}$ with the addition of fresh states $s_1, \ldots, s_k \notin S$, and

$$\begin{aligned}
S' &= (S \setminus \{s\}) \cup \{s_1, \ldots, s_k\} \\
F' &= \begin{cases} (F \setminus \{s\}) \cup \{s_1, \ldots, s_k\} & \text{if } s \in F \\ F & \text{otherwise} \end{cases} \\
\Delta' &= \{\gamma_s(H_i) \xrightarrow{a_i} s_i : i \in [1,k]\} \cup \\
&\qquad \{\gamma_s(H) \xrightarrow{a} s' : H \xrightarrow{a} s' \text{ in } \Delta, \text{ and } s' \neq s\}
\end{aligned}$$

where γ_s is substitution (in regular languages, c.f. [15], p. 60) of s by $\{s_1, \ldots, s_k\}$. It can be verified that

Lemma 1. $L(\mathcal{A}) = L(\mathcal{A} \uparrow_s)$.

There is only a finite number of times–equal to $|S|$–we can apply $\uparrow$ on a given FTA $\mathcal{A}$ because the newly created states fulfill the targetness condition. At the end of this procedure we obtain a targeted FTA, which we denote by $\mathcal{A} \uparrow$. From this, and Lemma 1, we have that

Theorem 1. *For each FTA $\mathcal{A}$ we can obtain an equivalent targeted FTA $\mathcal{A} \uparrow$ in PTIME.*

Furthermore, applying $\uparrow$ on an FTA $\mathcal{A}$, preserves determinism, i.e. we have

Proposition 1. *If $\mathcal{A}$ is deterministic, then so is $\mathcal{A} \uparrow$.*

4 Queries, Views, and Rewritings

We consider here (tree node) positions given by Dewey-style strings in $\mathbb{N}^*$. If t is a tree, we denote by $pos(t)$ the set of all t's positions. $pos(t)$ is prefix-closed and contains ϵ as the root position. Given a position $x \in pos(t)$, we define t_x to be the subtree of t rooted at x.

Definition 4. *A* pattern *is a tree-position pair (p, x), where $x \in pos(p)$.*

Let $\Upsilon^{\mathtt{x}} = \{(p, x) \mid p \in \Upsilon, x \in pos(p)\}$.

Definition 5. *A* tree query (TQ) *Q is a subset of $\Upsilon^{\mathtt{x}}$.*

When a query has only one pattern (p, x), we will blur the distinction between $\{(p, x)\}$ and (p, x).

Recall that Υ is the set of all trees over Σ. We call them *target trees*. Let $t \in \Upsilon$ be a target tree.

Definition 6. *The answer to Q on $t \in \Upsilon$ is $ans(Q, t) = \{t_x : (t, x) \in Q\}$.*

For two queries Q_1, Q_2, we define containment and equivalence as follows.

Definition 7.

1. *$Q_1 \sqsubseteq Q_2$ if $ans(Q_1, t) \subseteq ans(Q_2, t)$ for each $t \in \Upsilon$.*
2. *$Q_1 \equiv Q_2$ if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$.*

We show that

Theorem 2. *If $Q_1 \subseteq Q_2$ then $Q_1 \sqsubseteq Q_2$.*

Proof. Let $t' \in ans(Q_1, t)$. Then there exists $(t, x) \in Q_1$ such that $t_x = t'$. Since $Q_1 \subseteq Q_2$, we have $(t, x) \in Q_2$ and thus $t_x \in ans(Q_2, t)$. □

Corollary 1. *If $Q_1 = Q_2$ then $Q_1 \equiv Q_2$.*

Regular TQs.

Let $\hat{\Sigma} = \{\hat{a} : a \in \Sigma\}$ be an alphabet of marked symbols. Given (p, x), consider $\hat{p}$ on $\Sigma \cup \hat{\Sigma}$ that is the same as p, but with the node at position x being marked by the corresponding symbol in $\hat{\Sigma}$. Now a query Q is a *regular tree query* (RTQ) if set $\{\hat{p} \mid (p, x) \in Q\}$ is regular, i.e. given by a tree automaton.

Example 3. Let us revisit Example 2. Assume that the collection of movie trees is organized into a super-tree t with a root labeled by $\mathtt{r}$, and the movie trees as sub-trees of the root. Now let Q be a query that asks for all the movie sub-trees having at least one character played by two or more actors. It can be verified that this query can be given by the following automaton.

$$\mathcal{A} = (\{s, s_{\mathtt{t}}, s_{\mathtt{m}}, s_{\mathtt{c}}, s_{\mathtt{a}}\}, \Sigma \cup \hat{\Sigma}, \{s_{\mathtt{t}}\}, \Delta)$$

where Δ has the following transition rules

$$
\begin{aligned}
s^* s_{\mathtt{m}} s^* &\xrightarrow{\mathtt{r}} s_{\mathtt{t}} \\
s^* s_{\mathtt{c}} s^* &\xrightarrow{\hat{\mathtt{m}}} s_{\mathtt{m}} \\
s^* s_{\mathtt{a}} s_{\mathtt{a}} s^* &\xrightarrow{\mathtt{c}} s_{\mathtt{c}} \\
s^* &\xrightarrow{\mathtt{a}} s_{\mathtt{a}} \\
s^* &\xrightarrow{\Sigma} s. \quad \square
\end{aligned}
$$

Views and Rewritings. First we give a simple example to build up the intuition.

Example 4. Consider a view definition containing only the pattern tree given in Figure 2 [left] and a query containing only the pattern tree given in the same figure [middle]. Clearly, the view is useful in answering the query. Intuitively, the pattern tree we need to use to extract the answer to the query using the view is shown in the same figure [right]. This is nothing else but the "rewriting" of the query using the view in this example.

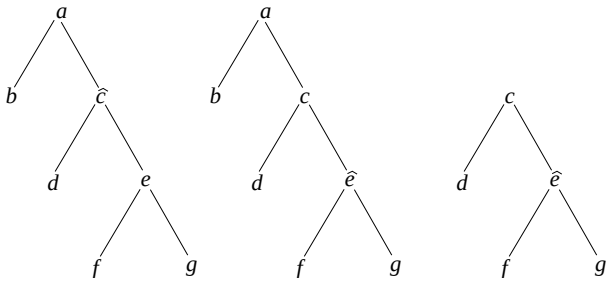

Fig. 2. A simple view, a simple query, and the rewriting. Pattern positions are given by $\hat{}$.

We now make precise a rewriting of a query Q using a view V.

Recall sets Υ, $\Upsilon^{\mathbf{x}}$. Furthermore, let $\Upsilon^{\mathbf{x},\mathbf{y}}$ be the set of all tree-position-position triples (p, x, y), such that $x, y \in pos(p)$, and x is a proper prefix of y, i.e. x is a proper ancestor of y. We restrict x to be a proper prefix of y because, as we show in 10.2, the queries and views can be transformed so that their specified positions (markings) never coincide in matching patterns.[3]

Let $(p, x), (q, y) \in \Upsilon^{\mathbf{x}}$. We define

$$
(p, x) \star (q, y) = \begin{cases} (p, xy) & \text{if } p_x = q \text{ and } y \neq \epsilon \\ \text{undefined} & \text{otherwise} \end{cases}
$$

[3] 10.2 should be read after being familiarized with the notation introduced in 5.

and

$$(p,x) \star (q,y) = \begin{cases} (p, x, xy) & \text{if } p_x = q \text{ and } y \neq \epsilon \\ \text{undefined} & \text{otherwise.} \end{cases}$$

where xy is the concatenation of x and y.

In simple words, for $(p,x) \star (q,y)$ and $(p,x) \star (q,y)$ to be defined, the subtree of p rooted at x must *be identical to* q structure-wise. Also, structure-wise, $(p,x) \star (q,y)$ and $(p,x) \star (q,y)$ are the same as (p,x).

We note that, when defined, $(p,x) \star (q,y) \in \Upsilon^{\mathbf{x}}$ and $(p,x) \star (q,y) \in \Upsilon^{\mathbf{x},\mathbf{y}}$. Referring to Figure 2 from the previous example, let the view be the tree on the left and the query be the tree in the middle. Then the tree on the right is a rewriting. If we regard the view on the left as (p,x) and the query in the middle as $(p,x) \star (q,y)$, then the rewriting needed to answer the query is (q,y), which is indeed the tree on the right. Therefore, finding the rewriting is solving a $\star$ equation with the rewriting as the unknown.

For two sets $L_1, L_2 \subseteq \Upsilon^{\mathbf{x}}$, we define $L_1 \star L_2$ and $L_1 \star L_2$ in the natural way. Also, we blur the distinction between a set of one element and the element itself. We note that $L_1 \star L_2 \subseteq \Upsilon^{\mathbf{x}}$ and $L_1 \star L_2 \subseteq \Upsilon^{\mathbf{x},\mathbf{y}}$.

We will use $\star$ to define the rewriting and $\star$ to reason about some steps in the construction for the rewriting.

In the following we use Greek letters υ and ξ to denote tree-position pairs.

Definition 8. *The* maximally contained rewriting (MCR) *of* $Q \subseteq \Upsilon^{\mathbf{x}}$ *using* $V \subseteq \Upsilon^{\mathbf{x}}$ *is*

$$R = \{\xi \in \Upsilon^{\mathbf{x}} : V \star \xi \subseteq Q\}.$$

We also define set X of "bad" patterns as

$$X = \{\xi \in \Upsilon^{\mathbf{x}} : \text{ there exists } \upsilon \in V \text{ such that } \upsilon \star \xi \in Q^c\}$$

where $Q^c = \Upsilon^{\mathbf{x}} \setminus Q$. Set X will be crucial in our construction. Observe that X is not equal to $\Upsilon^{\mathbf{x}} \setminus R$. The latter also contains patterns that cannot be $\star$-ed with V's patterns. Now let us define

$$Y = \{\xi \in \Upsilon^{\mathbf{x}} : \text{ there exists } \upsilon \in V \text{ such that } \upsilon \star \xi \in Q\}.$$

We have that

Proposition 2. $R = Y \setminus X$.

Example 5. Consider the following queries and views on chain trees, where the positions have been represented as $\hat{}$ *over the corresponding nodes. The chains should be read backwards to simulate a bottom-up processing.*

1. *Let* $Q = \{aac\hat{d}e,\ bbc\hat{d}e,\ ccc\hat{d}e\}$ *and* $V = \{aa\hat{c}de,\ bb\hat{c}de\}$. *We have* $c\hat{d}e \in R$ *because* $V \star c\hat{d}e = \{aac\hat{d}e,\ bbc\hat{d}e\} \subseteq Q$. *In fact* $c\hat{d}e$ *is the only chain in* R.

2. *Let* $Q = \{aac\hat{de}, ccc\hat{de}\}$ *and* $V = \{aa\hat{cde}, bb\hat{cde}\}$. *We have* $c\hat{de} \notin R$ *because* $bbc\hat{de} \in V \star c\hat{de}$, *but* $bbc\hat{de} \notin Q$. *Notice,* $c\hat{de}$ *is in* X *and* Y. □

Let $T_{V,t}$ be the *materialized answer* (MA) to V on $t \in \Upsilon$, i.e., $T_{V,t} = ans(V,t)$. We answer Q using V by computing

$$ans(R, T_{V,t}) = \bigcup_{v \in T_{V,t}} ans(R, v)$$

where R is the MCR of Q using V. We have that

Theorem 3. $ans(R, T_{V,t}) \subseteq ans(Q,t)$ *for each* $t \in \Upsilon$.

Proof. From Definition 8 and Theorem 2, we have $ans(V \star R, t) \subseteq ans(Q,t)$. Now the claim follows from this equality which we prove next

$$ans(V \star R, t) = ans(R, T_{V,t}).$$

"$\supseteq$": Let $z \in ans(R, T_{V,t})$. Then there exists $(q,y) \in R$ and $q \in T_{V,t}$ such that $q_y = z$. Since $q \in T_{V,t}$, there exists $(t,x) \in V$ such that $t_x = q$. Clearly, $q_y = z \in ans((t,x) \star (q,y), t)$. By $(t,x) \in V$ and $(q,y) \in R$ we obtain $z \in ans(V \star R, t)$ as required.

"$\subseteq$": Let $z \in ans(V \star R, t)$. There exists $(t, xy) = (t,x) \star (q,y) \in V \star R$, where $(t,x) \in V$ and $(q,y) \in R$, such that $ans((t,xy),t) = t_{xy} = z$. Consequently, $t_x \in T_{V,t}$ and $z \in ans(R, T_{V,t})$. □

5 Computing the MCR

The previous section refers to general queries and views. In this section we show that when they are RTQs, we can effectively compute QRV and show that it is an RTQ as well.

First we define the inverse of the $\star$ operation. Let $\upsilon, \upsilon' \in \Upsilon^{\mathbf{x}}$. We define

$$\upsilon \circledcirc \upsilon' = \begin{cases} \xi & \text{if } \upsilon' = \upsilon \star \xi \\ \text{undefined} & \text{otherwise} \end{cases}$$

This definition is lifted to subsets of $\Upsilon^{\mathbf{x}}$ in the natural way. Now, it can be verified that

Proposition 3. $X = V \circledcirc Q^c$ *and* $Y = V \circledcirc Q$.

Once we obtain X and Y, we can obtain R as $Y \setminus X$ (see Proposition 2). In the rest of the section, we present an automata-based solution for computing

$$K = J \circledcirc J'$$

when J and J' are RTQs.

Tree-position pair-sets facilitate the definitions of queries and rewritings. However, when we work with automata, we need to talk in terms of languages that the automata recognize. Here, we introduce colors for marking the positions in pattern trees. Colors are similar to Boolean markings of [28] and [23] (or to our previous ˆ marking), however, they make the presentation easier.

Thereto, we consider the "red" and "blue" alphabets $\Sigma^{\mathbf{r}} = \{a^{\mathbf{r}} : a \in \Sigma\}$ and $\Sigma^{\mathbf{b}} = \{a^{\mathbf{b}} : a \in \Sigma\}$ in addition to the alphabet Σ. We refer to the elements of Σ as being "black" and to Σ as the "black" alphabet. We refer to nodes as black, red, or blue nodes if their symbol is black, red, or blue.

We define $\mathbf{b}$ as a (unary) operator that given $(p, x) \in \Upsilon^{\mathbf{x}}$ returns a tree over $\Sigma \cup \Sigma^{\mathbf{b}}$ that is isomorphic to p with the node at position x colored blue.

Likewise, we define $\mathbf{r}$ as a (unary) operator that given $(p, y) \in \Upsilon^{\mathbf{y}}$ returns a tree over $\Sigma \cup \Sigma^{\mathbf{r}}$ that is isomorphic to p with the node at position y colored red. [$\Upsilon^{\mathbf{y}}$ is the same as $\Upsilon^{\mathbf{x}}$ and is used for notation parallelism.]

Furthermore, we define $\mathbf{br}$ as a (unary) operator that given $(p, x, y) \in \Upsilon^{\mathbf{x},\mathbf{y}}$ returns a tree over $\Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}$ that is isomorphic to p with the node at position x colored blue and the node at the position y colored red.

Based on these operators, we define

$$\begin{aligned}\Upsilon^{\mathbf{b}} &= \{\mathbf{b}(p, x) : (p, x) \in \Upsilon^{\mathbf{x}}\}\\ \Upsilon^{\mathbf{r}} &= \{\mathbf{r}(p, y) : (p, y) \in \Upsilon^{\mathbf{y}}\}\\ \Upsilon^{\mathbf{b},\mathbf{r}} &= \{\mathbf{br}(p, x, y) : (p, x, y) \in \Upsilon^{\mathbf{x},\mathbf{y}}\}.\end{aligned}$$

$\Upsilon^{\mathbf{b}}$, $\Upsilon^{\mathbf{r}}$, $\Upsilon^{\mathbf{b},\mathbf{r}}$ are *languages of trees* over $\Sigma \cup \Sigma^{\mathbf{b}}$, $\Sigma \cup \Sigma^{\mathbf{r}}$, $\Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}$, respectively.

Clearly, there is a one-to-one correspondence between the elements of $\Upsilon^{\mathbf{x}}$, $\Upsilon^{\mathbf{y}}$, $\Upsilon^{\mathbf{x},\mathbf{y}}$, and the elements of $\Upsilon^{\mathbf{b}}$, $\Upsilon^{\mathbf{r}}$, $\Upsilon^{\mathbf{b},\mathbf{r}}$, respectively. Therefore, we will blur the distinction between elements and subsets of $\Upsilon^{\mathbf{x}}$, $\Upsilon^{\mathbf{y}}$, $\Upsilon^{\mathbf{x},\mathbf{y}}$, and elements and subsets of $\Upsilon^{\mathbf{b}}$, $\Upsilon^{\mathbf{r}}$, $\Upsilon^{\mathbf{b},\mathbf{r}}$, respectively.

We will use (sans-serif) p, q to refer to colored patterns of $\Upsilon^{\mathbf{b}}$, $\Upsilon^{\mathbf{r}}$, $\Upsilon^{\mathbf{b},\mathbf{r}}$.

Let $\mathsf{p} \in \Upsilon^{\mathbf{b}}$ and $\mathsf{q} \in \Upsilon^{\mathbf{r}}$ with their corresponding $(p, x) \in \Upsilon^{\mathbf{x}}$ and $(q, y) \in \Upsilon^{\mathbf{y}}$. We define

$$\begin{aligned}\mathsf{p} \mathbin{\dot{\star}} \mathsf{q} &= \mathbf{r}((p, x) \mathbin{\dot{\star}} (q, y))\\ \mathsf{p} \mathbin{\star} \mathsf{q} &= \mathbf{br}((p, x) \mathbin{\star} (q, y)).\end{aligned}$$

We have $\mathsf{p} \mathbin{\dot{\star}} \mathsf{q} \in \Upsilon^{\mathbf{r}}$ and $\mathsf{p} \mathbin{\star} \mathsf{q} \in \Upsilon^{\mathbf{b},\mathbf{r}}$ (when they are defined). We extend $\mathbin{\dot{\star}}$ and $\mathbin{\star}$ to languages in the natural way.

In particular, from now on, we will consider $J \subseteq \Upsilon^{\mathbf{b}}$, $J' \subseteq \Upsilon^{\mathbf{r}}$, $K \subseteq \Upsilon^{\mathbf{r}}$, $J \mathbin{\dot{\star}} K \subseteq \Upsilon^{\mathbf{r}}$, and $J \mathbin{\star} K \subseteq \Upsilon^{\mathbf{b},\mathbf{r}}$.

In order to aid us in the development, we also define

$\Phi^{\mathbf{r}}$ to be the set of all trees having all nodes black, except for the root which is red, and

$\Phi^{\mathbf{b},\mathbf{r}}$ to be the set of all trees having all nodes black, except for the root which is blue and another node which is red.

It can be verified that the languages we defined so far, $\Upsilon^{\mathbf{r}}$, $\Upsilon^{\mathbf{b}}$, $\Upsilon^{\mathbf{b},\mathbf{r}}$, $\Phi^{\mathbf{r}}$, and $\Phi^{\mathbf{b},\mathbf{r}}$ are all RTLs. Also observe that from the definition of the $\pentagram$ operation, $K \subseteq \Upsilon^{\mathbf{r}} \setminus \Phi^{\mathbf{r}}$.

If p is a tree over $\Sigma \cup \Sigma^{\mathbf{r}} \cup \Sigma^{\mathbf{b}}$, we denote by $\mathsf{p}^{\neg\mathbf{b}}$ the tree over $\Sigma \cup \Sigma^{\mathbf{r}}$ that is the same as p, but with the blue nodes turned black. For a language L over $\Sigma \cup \Sigma^{\mathbf{r}} \cup \Sigma^{\mathbf{b}}$, we define

$$L^{\neg\mathbf{b}} = \{\mathsf{p}^{\neg\mathbf{b}} : \mathsf{p} \in L\}.$$

If L is an RTL, we can construct an FTA for L and then an FTA for $L^{\neg\mathbf{b}}$ by changing all the blue symbols in the transitions of the FTA for L to black. We define similarly $\mathsf{p}^{\neg\mathbf{r}}$ and $L^{\neg\mathbf{r}}$.

5.1 Auxiliaries

In our construction we will need the following items

$B_L = \{\mathsf{p} \in \Upsilon^{\mathbf{b},\mathbf{r}} : \mathsf{p}^{\neg\mathbf{b}} \in L\}$ for $L \subseteq \Upsilon^{\mathbf{r}}$
$B'_L = \{\mathsf{p} \in \Upsilon^{\mathbf{b},\mathbf{r}} : \mathsf{p}^{\neg\mathbf{r}} \in L\}$ for $L \subseteq \Upsilon^{\mathbf{b}}$
$C_L = \{\mathsf{p} \in \Phi^{\mathbf{b},\mathbf{r}} : \mathsf{p}^{\neg\mathbf{b}} \in L\}$ for $L \subseteq \Upsilon^{\mathbf{r}} \setminus \Phi^{\mathbf{r}}$.

These languages are easy to construct when L is RTL. We show how to do that for B_L. The other ones are similar. Let $\mathcal{A} = (S, \Sigma \cup \Sigma^{\mathbf{r}}, F, \Delta)$ be an FTA for L. We construct FTA $\mathcal{B} = (S, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}, F, \Delta_{\mathcal{B}})$, where

$$\Delta_{\mathcal{B}} = \Delta \cup \{H \xrightarrow{a^{\mathbf{b}}} s : H \xrightarrow{a} s \text{ in } \Delta \text{ and } a \in \Sigma\}.$$

Proposition 4. *Given an FTA for L, an FTA for B_L can be constructed in polynomial time. Furthermore, if the FTA for L is a complete and deterministic FTA (CDFTA), then a CDFTA for B_L can be constructed in polynomial time as well.*

Proof. The statements follow from the observation that $L(\mathcal{B}) \cap \Upsilon^{\mathbf{b},\mathbf{r}} = B_L$, and the facts that (1) $\mathcal{B}$ is constructed in polynomial time from $\mathcal{A}$ and is a CDFTA when $\mathcal{A}$ is such, (2) an intersection CDFTA is computable in polynomial time when supplied with CDFTAs as input. □

We also get exactly the same facts for B'_L and C_L as those stated for B_L in the above proposition.

5.2 The Algorithm

Here we construct an FTA for C_K [K, we are interested in, is $C_K^{\neg\mathbf{b}}$, easily computed once we have an automaton for C_K].

Consider B_J and $B'_{J'}$. From FTAs for B_J and $B'_{J'}$, we construct a FTA for $B_J \cap B'_{J'}$. Let $\mathcal{D} = (S, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}, F, \Delta)$ be this FTA. We also transform it to be targeted. Observe that $L(\mathcal{D}) \subseteq \Upsilon^{\mathbf{b},\mathbf{r}}$.

Now, we construct FTA $\mathcal{E} = (S, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}, F_{\mathcal{E}}, \Delta)$, where

$$F_{\mathcal{E}} = \{s \in S : \text{ there exists } H \xrightarrow{a^{\mathbf{b}}} s \text{ in } \Delta\}.$$

Clearly, $L(\mathcal{E}) \subseteq \Phi^{\mathbf{b},\mathbf{r}}$. This is true because $\mathcal{D}$ is targeted. We have

Theorem 4. $L(\mathcal{E}) = C_K$.

Proof. "$\subseteq$": Let $\mathsf{q} \in L(\mathcal{E}) \subseteq \Phi^{\mathbf{b},\mathbf{r}}$. By the construction of $\mathcal{E}$, there exists $\mathsf{p} \in B_J \cap B'_{J'} \subseteq \Upsilon^{\mathbf{b},\mathbf{r}}$, with the blue node at a position x, such that $\mathsf{p}_x = q$.

Since $\mathsf{p} \in B_J \cap B'_{J'}$, $\mathsf{p}^{\neg\mathbf{r}} \in J$ and $\mathsf{p}^{\neg\mathbf{b}} \in J'$. Also, since $\mathsf{p} \in \Upsilon^{\mathbf{b},\mathbf{r}}$, $(\mathsf{p}^{\neg\mathbf{b}})_x \in \Upsilon^{\mathbf{r}} \setminus \Phi^{\mathbf{r}}$. We have $\mathsf{p}^{\neg\mathbf{r}} \mathbin{\dot{\star}} (\mathsf{p}^{\neg\mathbf{b}})_x = \mathsf{p}^{\neg\mathbf{b}}$, i.e. $(\mathsf{p}^{\neg\mathbf{b}})_x \in J \mathbin{\dot{\star}} J' = K$, which means $\mathsf{p}_x = q \in C_K$.

"$\supseteq$": Let $\mathsf{q} \in C_K \subseteq \Phi^{\mathbf{b},\mathbf{r}}$. We have $\mathsf{q}^{\neg\mathbf{b}} \in K \subseteq \Upsilon^{\mathbf{r}} \setminus \Phi^{\mathbf{r}}$. By the definition of K, there exists $\mathsf{p} \in J$ such that $\mathsf{p} \mathbin{\dot{\star}} \mathsf{q}^{\neg\mathbf{b}}$ is defined and $\mathsf{p} \mathbin{\dot{\star}} \mathsf{q}^{\neg\mathbf{b}} \in J'$. This, coupled with the fact that $\mathsf{q}^{\neg\mathbf{b}} \in \Upsilon^{\mathbf{r}} \setminus \Phi^{\mathbf{r}}$, implies $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}} \in B_J$. By the definition of "$\star$" and $B'_{J'}$, $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}} \in B'_{J'}$, too, therefore, $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}} \in B_J \cap B'_{J'}$, i.e., $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}}$ is accepted by $\mathcal{D}$. By the construction of $\mathcal{E}$, we have that q is accepted by $\mathcal{E}$. □

5.3 Complexity

Let J and J' be given by non-deterministic FTAs. It can be verified that

Proposition 5. *$K = J \mathbin{\dot{\star}} J'$ can be computed in polynomial time.*

Therefore, $Y = V \mathbin{\dot{\star}} Q$ can be computed in polynomial time. However, we also need to compute $X = V \mathbin{\dot{\star}} Q^c$, and then X^c, in order to compute R as $Y \setminus X$. If we are not careful, we can incur a double-exponential penalty. Here we show that it can be done instead in single-exponential time.

For this let us refer to the steps of computing C_K. Suppose J and J' are given by CDFTAs. By Proposition 4 we can obtain CDFTAs for B_J and $B'_{J'}$ in polynomial time. From these CDFTAs we obtain a CDFTA $\mathcal{D}$ for $B_J \cap B'_{J'}$ in polynomial time as well. Automaton $\mathcal{E}$ (representing C_K) is the same as $\mathcal{D}$, but with a different set of final states, so $\mathcal{E}$ is a CDFTA as well.

Since $\mathcal{E}$ is a CDFTA we can compute $(C_K)^c$ from it in polynomial time.

Now, we have that for X and Y

1. $C_X, C_Y \subseteq \Phi^{\mathbf{b},\mathbf{r}}$
2. $Y \setminus X = (C_Y \setminus C_X)^{\neg\mathbf{b}} = (C_Y \cap (C_X)^c)^{\neg\mathbf{b}}$.

Clearly, $\cap$ and $\neg\mathbf{b}$ are polynomial.

Let a query Q and a view V be specified as non-deterministic FTAs. We have the following theorems.

Theorem 5. *The MCR R of Q using V can be computed in exponential time.*

Proof. The proof follows from the above reasoning and the fact that to construct CDFTAs for Q (and then Q^c), and V takes exponential time. □

Theorem 6. *Computing the MCR of Q using V is EXPTIME-hard.*

Proof. We present a reduction from the problem of inclusion for RTLs (represented by non-deterministic FTAs with horizontal languages represented by NFAs) which is EXPTIME-complete. Let L_1 and L_2 be such RTLs over some alphabet Σ. Let $\# \notin \Sigma$. We construct V from L_1 by adding to the rightmost

leaf of trees in L_1 a subtree $\#^{\mathbf{b}}(\#)$. Similarly, we construct Q from L_2 by adding to the rightmost leaf of trees in L_2 a subtree $\#(\#^{\mathbf{r}})$. Now, we have that $L_1 \subseteq L_2$ if and only if the MCR of Q using V contains the single node pattern $\#^{\mathbf{r}}$. The latter can be checked in time polynomial in the size of an FTA for the rewriting.

In conclusion, if computing the MCR of Q using V were not EXPTIME-hard, then we would be able to decide the language inclusion for RTLs in less than EXPTIME, which is not possible in the worst case. □

6 *k*-ary Queries

So far, our attention has been focused on unary queries. In this section, we consider k-ary queries, for $k \geq 1$. Miklau and Suciu [18] motivate the study of containment for k-ary tree pattern queries by observing that the results can be applied in the context of optimizing the `FOR` clause of XQuery expressions, where multiple variables can be defined using XPath variables. The idea is that they can be captured in a single tree pattern[4] whose distinguished variables correspond to the variables bound in the `FOR` clause. Inspired by this, we consider tree-position vector queries whose vector contains k positions, for some arbitrary, but *fixed* $k \geq 1$. Such queries return a tuple (forest) of sub-trees from a target tree.

On the theoretical side, it has been shown that k-ary queries can be encoded by unary queries (see [25] and [3]). However, this is done by going through MSO formulas. Therefore, going from a k-ary query represented by an automaton to an encoding by unary queries and then back to (combination of) automata would incur non elementary complexity. Even if the complexity were not so, we still would be left with a combination of unary queries for which we needed to compute QRV in terms of another combination of unary queries which corresponds to the view. This is something that does not follow from the rewriting mechanism we developed in the previous part of the paper. The latter comment also applies to other k-ary query formalisms, defined in terms of compositions of unary queries such as the formalism introduced in [13].

Definition 9. *A* k-pattern *is a tree-position vector pair* $(p, [x_1, \ldots, x_k])$*, where*

1. $x_1, \ldots, x_k \in pos(p)$
2. $x_1 < \ldots < x_k$
3. $\nexists i, j \in [1, k]$*, such that* x_i *is a prefix of* x_j*.*

We also write a pattern as $(p, \mathbf{x})$, where $\mathbf{x} = [x_1, \ldots, x_k]$. A position vector possessing the three properties in above definition is called *proper*. Whenever we refer to a position vector, we assume it to be proper. We let $\Upsilon^{k\mathbf{x}}$ denote the set of all k-patterns.

Given a tree t and $\mathbf{x} = [x_1, \ldots, x_k]$, we define $t_{\mathbf{x}} = (t_{x_1}, \ldots, t_{x_k})$ as the tuple (forest) of sub-trees of t rooted at the positions in $\mathbf{x}$. We call a forest of k trees a k-forest, and denote by Ξ^k the set of all k-forests.

[4] Tree patterns considered in [18] have child and descendant edges, wildcards, and no order.

Definition 10. *A k-tree query (k-TQ) is a subset of $\Upsilon^{k\mathbf{x}}$. If $\{\hat{p} : (p, \mathbf{x}) \in Q\}$*[5] *is in addition* regular*, we say that Q is a* regular tree query (RTQ).

When a query has only one pattern $(p, \mathbf{x})$, we blur the distinction between $\{(p, \mathbf{x})\}$ and $(p, \mathbf{x})$.

Definition 11. *The answer to a k-query Q on t is $ans(Q, t) = \{t_{\mathbf{x}} : (t, \mathbf{x}) \in Q\}$.*

As for unary queries, it can be shown that for k-tree queries, set containment implies query containment.

Example 6. Let us recall the movie example in Section 4. Now let Q be a query given by $\mathcal{A} = (\{s, s_{\mathtt{t}}, s_{\mathtt{m}}, s_{\mathtt{c}}, s_{\mathtt{a}}\}, \Sigma \cup \hat{\Sigma}, \{s_{\mathtt{t}}\}, \Delta)$, where Δ has the following transition rules

$$\begin{aligned} s^* s_{\mathtt{m}} s^* &\xrightarrow{\mathtt{r}} s_{\mathtt{t}} \\ s^* s_{\mathtt{c}} s^* &\xrightarrow{\mathtt{m}} s_{\mathtt{m}} \\ s^* s_{\mathtt{a}} s^* s_{\mathtt{a}} s^* &\xrightarrow{\mathtt{c}} s_{\mathtt{c}} \\ s^* &\xrightarrow{\hat{\mathtt{a}}} s_{\mathtt{a}} \\ s^* &\xrightarrow{\Sigma} s. \end{aligned}$$

Clearly, $Q \subseteq \Upsilon^{2\mathbf{x}}$, and $ans(Q, t)$ consists of all 2-forests of (sub)tree pairs for actors who have played the same character together in some movie. □

7 *k*-ary Views and Rewritings

We define a $k, m\mathbf{y}$-forest to be a tuple $((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k))$ of patterns, such that $|\mathbf{y}_1| + \ldots + |\mathbf{y}_k| = m$, where $|\mathbf{y}_i|$, for $i \in [1, k]$, is the dimension of vector $\mathbf{y}_i$. We let $\Xi^{k,m\mathbf{y}}$ be the set of all $k, m\mathbf{y}$-forests.

Furthermore, we define a $k\mathbf{x}, m\mathbf{y}$-pattern to be a tuple of the form

$$(p, [(x_1, \mathbf{y}_1), \ldots, (x_k, \mathbf{y}_k)])$$

where $|\mathbf{y}_1| + \ldots + |\mathbf{y}_k| = m$, and x_i is a proper[6] prefix of positions in $\mathbf{y}_i$, for $i \in [1, k]$. We let $\Upsilon^{k\mathbf{x},m\mathbf{y}}$ be the set of all $k\mathbf{x}, m\mathbf{y}$-patterns.

Now, let $(p, [x_1, \ldots, x_k]) \in \Upsilon^{k\mathbf{x}}$ and $((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k)) \in \Xi^{k,m\mathbf{y}}$. We define

$$(p, [x_1, \ldots, x_k]) \mathbin{\dot{\star}} ((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k)) =$$

$$\begin{cases} (p, [x_1\mathbf{y}_1, \ldots, x_k\mathbf{y}_k]) & \text{if } p_{x_i} = q_i \text{ and } y_i \neq \epsilon, \forall i \in [1, k] \\ \text{undefined} & \text{otherwise} \end{cases}$$

[5] Similarly as for unary queries, $\hat{p}$ denotes the pattern on $\Sigma \cup \hat{\Sigma}$ that is the same as p, but with the nodes at positions of $\mathbf{x}$ being marked by the corresponding symbols in $\hat{\Sigma}$.

[6] By assuming properly transformed query and view (as in 10.2), we do not need to consider the case when some x_i coincides with some position in $\mathbf{y}_i$.

and

$$(p, [x_1, \ldots, x_k]) \star ((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k)) =$$
$$\begin{cases} (p, [(x_1, x_1\mathbf{y}_1), \ldots, (x_k, x_k\mathbf{y}_k)]) \\ \qquad\qquad \text{if } p_{x_i} = q_i \text{ and } y_i \neq \epsilon, \forall i \in [1, k] \\ \text{undefined} \qquad \text{otherwise} \end{cases}$$

where $x_i\mathbf{y}_i$, for $i \in [1, k]$, is the tuple obtained from $\mathbf{y}_i$ by prepending x_i to each of its positions.

We note that, when defined

$$(p, [x_1, \ldots, x_k]) \star ((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k)) \in \Upsilon^{m\mathbf{y}}$$
$$(p, [x_1, \ldots, x_k]) \star ((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k)) \in \Upsilon^{k\mathbf{x}, m\mathbf{y}}.$$

[For ease of exposition we are using $\Upsilon^{m\mathbf{y}}$ instead of $\Upsilon^{m\mathbf{x}}$.]

For $L_1 \subseteq \Upsilon^{k\mathbf{x}}$, $L_2 \subseteq \Xi^{k,m\mathbf{y}}$ we define $L_1 \star L_2$ and $L_1 \star L_2$ in the natural way. Again we blur the distinction between a set of one element and the element itself.

Definition 12. *The* maximally contained rewriting (MCR) *of* $Q \subseteq \Upsilon^{m\mathbf{y}}$ *using* $V \subseteq \Upsilon^{k\mathbf{x}}$ *is*

$$R = \{\xi \in \Xi^{k,m\mathbf{y}} : V \star \xi \subseteq Q\}.$$

(ξ being in $\Xi^{k,m\mathbf{y}}$ is of the form $((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k))$.)
We also define set X of "bad" patterns as

$$X = \{\xi \in \Xi^{k,m\mathbf{y}} : \text{ there exists } \upsilon \in V \text{ s.t. } \upsilon \star \xi \in Q^c\}$$

where $Q^c = \Upsilon^{m\mathbf{y}} \setminus Q$. ($\upsilon$ being in $\Upsilon^{k\mathbf{x}}$ is of the form $(p, [x_1, \ldots, x_k])$.) Now we define

$$Y = \{\xi \in \Xi^{k,m\mathbf{y}} : \text{ there exists } \upsilon \in V \text{ s.t. } \upsilon \star \xi \in Q\}.$$

Notice that R, X, and Y are subsets of $\Xi^{k,m\mathbf{y}}$. We have that

Proposition 6. $R = Y \setminus X$.

Dummy roots. We turn forests into trees by introducing a "dummy" root labeled by a special symbol $\Diamond \notin \Sigma$.

Given $(t_1, \ldots, t_k) \in \Xi^k$, we define

$$\Diamond(t_1, \ldots, t_k)$$

to be the tree obtained by making the roots of $t_1, \ldots, t_k$ to be children of a dummy new root labeled by $\Diamond$.

Likewise, given $((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k)) \in \Xi^{k,m\mathbf{y}}$, we define

$$\Diamond((q_1, \mathbf{y}_1), \ldots, (q_k, \mathbf{y}_k))$$

to be the tree obtained by making the roots of $q_1, \ldots, q_k$ to be children of a dummy new root labeled by $\Diamond$.

If L is a subset of Ξ^k or $\Xi^{k,m\mathbf{y}}$, we define $\Diamond L$ in the natural way by turning each forest in L into a tree as described.

Let $F \subseteq \Xi^k$ and $L \subseteq \Xi^{k,m\mathbf{y}}$. We define

$$ans(L, F) = \bigcup_{v \in F} ans(\Diamond L, \Diamond v).$$

Let $F_{V,t} = ans(V, t) \subseteq \Xi^k$ be the materialization of V on $t \in \Upsilon$, We answer Q using V by $ans(R, F_{V,t})$. Now, by generalizing the reasoning in the proof of Theorem 3 it can be verified that

Theorem 7. *$ans(R, F_{V,t}) \subseteq ans(Q, t)$ for each $t \in \Upsilon$.*

8 Computing the MCR of an m-ary Query using a k-ary View

Analogus to the case of unary queries, we first define the inverse of the $\star$ operation.

Let $\upsilon \in \Upsilon^{k\mathbf{x}}$ and $\upsilon' \in \Upsilon^{m\mathbf{x}}$. We define

$$\upsilon \circledcirc \upsilon' = \begin{cases} \xi & \text{if } \upsilon' = \upsilon \star \xi \\ \text{undefined} & \text{otherwise} \end{cases}$$

This definition is lifted to subsets of $\Upsilon^{\mathbf{x}}$ in the natural way. Now, it can be verified that as in the case of unary queries

Proposition 7. *$X = V \circledcirc Q^c$ and $Y = V \circledcirc Q$.*

Once we obtain X and Y, we can obtain R as $Y \setminus X$. In the rest of the section, we present an automata-based solution for computing

$$K = J \circledcirc J'$$

when J and J' are RTQs.

Here we again use colors in order to work with languages and automata. We define $\mathbf{b}$ as a (unary) operator that given $(p, \mathbf{x}) \in \Upsilon^{k\mathbf{x}}$ returns a tree over $\Sigma \cup \Sigma^{\mathbf{b}}$ that is isomorphic to p with the nodes at the positions of $\mathbf{x}$ colored blue.

Likewise, we define $\mathbf{r}$ as a (unary) operator that given $(p, \mathbf{y}) \in \Upsilon^{m\mathbf{y}}$ returns a tree over $\Sigma \cup \Sigma^{\mathbf{r}}$ that is isomorphic to p with the nodes at the positions of $\mathbf{y}$ colored red.

Furthermore, we define $\mathbf{br}$ as a (unary) operator that given $(p, (x_1, \mathbf{y}_1), \ldots, (x_k, \mathbf{y}_k)) \in \Upsilon^{k\mathbf{x},m\mathbf{y}}$ returns a tree over $\Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}$ that is isomorphic to p with the nodes at positions $x_1, \ldots, x_k$ colored blue and the nodes at the positions of $\mathbf{y}_1, \ldots, \mathbf{y}_k$ colored red.

Based on these operators, we define

$$\begin{aligned}
\Upsilon^{k\mathbf{b}} &= \{\mathbf{b}(p,\mathbf{x}) : (p,\mathbf{x}) \in \Upsilon^{k\mathbf{x}}\}\\
\Upsilon^{m\mathbf{r}} &= \{\mathbf{r}(p,\mathbf{y}) : (p,\mathbf{x}) \in \Upsilon^{m\mathbf{y}}\}\\
\Upsilon^{k\mathbf{b},m\mathbf{r}} &= \{\mathbf{br}(p,(x_1,\mathbf{y}_1),\ldots,(x_k,\mathbf{y}_k)) :\\
&\qquad (p,(x_1,\mathbf{y}_1),\ldots,(x_k,\mathbf{y}_k)) \in \Upsilon^{k\mathbf{x},m\mathbf{y}}\}\\
\Xi^{k,m\mathbf{r}} &= \{(\mathbf{r}(q_1,\mathbf{y}_1),\ldots,\mathbf{r}(q_k,\mathbf{y}_k)) :\\
&\qquad ((q_1,\mathbf{y}_1),\ldots,(q_k,\mathbf{y}_k)) \in \Xi^{k,m\mathbf{y}}\}.
\end{aligned}$$

Clearly, there is a one-to-one correspondence between the elements of $\Upsilon^{k\mathbf{x}}$, $\Upsilon^{m\mathbf{y}}$, $\Upsilon^{k\mathbf{x},m\mathbf{y}}$, $\Xi^{k,m\mathbf{y}}$ and the elements of $\Upsilon^{k\mathbf{b}}$, $\Upsilon^{m\mathbf{r}}$, $\Upsilon^{k\mathbf{b},m\mathbf{r}}$, $\Xi^{k,m\mathbf{r}}$, respectively. Therefore, we will blur the distinction between elements and subsets of $\Upsilon^{k\mathbf{x}}$, $\Upsilon^{m\mathbf{y}}$, $\Upsilon^{k\mathbf{x},m\mathbf{y}}$, $\Xi^{k,m\mathbf{y}}$ and elements and subsets of $\Upsilon^{k\mathbf{b}}$, $\Upsilon^{m\mathbf{r}}$, $\Upsilon^{k\mathbf{b},m\mathbf{r}}$, $\Xi^{k,m\mathbf{r}}$, respectively.

We will use (sans-serif) p,q to refer to colored patterns of $\Upsilon^{k\mathbf{b}}$, $\Upsilon^{m\mathbf{r}}$, $\Upsilon^{k\mathbf{b},m\mathbf{r}}$, and $\mathbf{q} = (\mathsf{q}_1,\ldots,\mathsf{q}_k)$ to refer to colored forests of $\Xi^{k,m\mathbf{r}}$.

Let $\mathsf{p} \in \Upsilon^{k\mathbf{b}}$ and $\mathbf{q} = (\mathsf{q}_1,\ldots,\mathsf{q}_k) \in \Xi^{k,m\mathbf{r}}$ with their corresponding $(p,\mathbf{x}) \in \Upsilon^{k\mathbf{x}}$ and $((q_1,\mathbf{y_1}),\ldots,(q_k,\mathbf{y_k})) \in \Xi^{k,m\mathbf{y}}$. We define

$$\begin{aligned}
\mathsf{p} \mathbin{\dot{\star}} \mathbf{q} &= \mathbf{r}((p,\mathbf{x}) \mathbin{\dot{\star}} ((q_1,\mathbf{y_1}),\ldots,(q_k,\mathbf{y_k})))\\
\mathsf{p} \star \mathbf{q} &= \mathbf{br}((p,\mathbf{x}) \star ((q_1,\mathbf{y_1}),\ldots,(q_k,\mathbf{y_k}))).
\end{aligned}$$

We extend $\dot{\star}$ and $\star$ to languages and forests in the natural way.

In particular, from now on, we will consider $J \subseteq \Upsilon^{k\mathbf{b}}$, $J' \subseteq \Upsilon^{m\mathbf{r}}$, $K \subseteq \Xi^{k,m\mathbf{r}}$, $J \mathbin{\dot{\star}} K \subseteq \Upsilon^{m\mathbf{r}}$, and $J \star K \subseteq \Upsilon^{k\mathbf{b},m\mathbf{r}}$.

We further define a few more notions. Let

$\Phi^{k,m\mathbf{r}}$ be the subset of $\Xi^{k,m\mathbf{r}}$ forests having *at least one* tree that has its root red.

$\Phi^{k\mathbf{b},m\mathbf{r}}$ be the set of all k-forests over $\Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}$, such that: (1) the roots of all patterns in the forest are blue, (2) *exactly* m nodes in total are red, and (3) all the other nodes are black.

Observe that from the definition of the $\fullmoon$ operation, $K \subseteq \Xi^{k,m\mathbf{r}} \setminus \Phi^{k,m\mathbf{r}}$.

We define the operators $(.)^{\neg\mathbf{b}}$ and $(.)^{\neg\mathbf{r}}$, for patterns, sets of patterns, forests, and sets of forests, over $\Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}$, in a manner similar to Section 5.

8.1 Auxiliaries

In our construction we will need the following items

$$\begin{aligned}
B_L &= \{\mathsf{p} \in \Upsilon^{k\mathbf{b},m\mathbf{r}} : \mathsf{p}^{\neg\mathbf{b}} \in L\} \text{ for } L \subseteq \Upsilon^{m\mathbf{r}}\\
B'_L &= \{\mathsf{p} \in \Upsilon^{k\mathbf{b},m\mathbf{r}} : \mathsf{p}^{\neg\mathbf{r}} \in L\} \text{ for } L \subseteq \Upsilon^{k\mathbf{b}}\\
C_L &= \{\mathbf{q} \in \Phi^{k\mathbf{b},m\mathbf{r}} : \mathbf{q}^{\neg\mathbf{b}} \in L\} \text{ for } L \subseteq \Xi^{k,m\mathbf{r}} \setminus \Phi^{k,m\mathbf{r}}.
\end{aligned}$$

In plain language C_L is the set of all forests in $\Phi^{k\mathbf{b},m\mathbf{r}}$ obtained from the forests of L after making the root of their trees blue.

The above languages are easy to construct when L ($\Diamond L$ in the third definition) is RTL. For instance, for B_L, we first construct an automaton $\mathcal{B}$ exactly as in Subsection 5.1, and then compute B_L as $L(\mathcal{B}) \cap \Upsilon^{k\mathbf{b},m\mathbf{r}}$. Similar statements as those made in Proposition 4 can be made here as well.

8.2 The Algorithm

In order to be compatible with the framework of FTAs (which work on trees, not forests), here we construct an CDFTA for $\Diamond C_K$. Set $\Diamond K$ (we are interested in) is easily computed once we have an automaton for $\Diamond C_K$.

Consider B_J and $B'_{J'}$. From CDFTAs for B_J and $B'_{J'}$, we construct a CDFTA for $B_J \cap B'_{J'}$. Let $\mathcal{D} = (S_\mathcal{D}, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}, F_\mathcal{D}, \Delta_\mathcal{D})$ be this CDFTA. We also transform it to be targeted. Observe that $L(\mathcal{D}) \subseteq \Upsilon^{k\mathbf{b},m\mathbf{r}}$. Let

$$S^{\mathbf{b}} = \{s \in S_\mathcal{D} : \text{ there exists } H \xrightarrow{a^{\mathbf{b}}} s \text{ in } \Delta_\mathcal{D}\}.$$

We denote $(S^{\mathbf{b}})^k$ by $S^{k\mathbf{b}}$. Next, we construct FTA $\mathcal{E} = (S_\mathcal{E}, \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{b}} \cup \{\Diamond\}, F_\mathcal{E}, \Delta_\mathcal{E})$ where

$$\begin{aligned}
S_\mathcal{E} &= S_\mathcal{D} \cup \{s_{\text{final}}, s_{\text{garbage}}\} \\
F_\mathcal{E} &= \{s_{\text{final}}\} \\
\Delta_\mathcal{E} &= \Delta_\mathcal{D} \cup \{S^{k\mathbf{b}} \xrightarrow{\Diamond} s_{\text{final}}\} \cup \{S^*_\mathcal{E} \setminus S^{k\mathbf{b}} \xrightarrow{\Diamond} s_{\text{garbage}}\} \cup \\
&\quad \{(s_{\text{garbage}})^* \xrightarrow{a} s_{\text{garbage}} : a \in \Sigma \cup \Sigma^{\mathbf{b}} \cup \Sigma^{\mathbf{r}}\}.
\end{aligned}$$

Since k is fixed, an automaton for $S^*_\mathcal{E} \setminus S^{k\mathbf{b}}$ can be constructed in polynomial time in the size of $S^{k\mathbf{b}}$ and $S_\mathcal{E}$. We can show that the FTA $\mathcal{E}$ constructed above is a CDFTA.

Clearly, $L(\mathcal{E}) \subseteq \Diamond\Phi^{k\mathbf{r},m\mathbf{r}}$. This is true because $\mathcal{D}$ is targeted. Now, by generalizing the reasoning in the proof of Theorem 4, we can show that

Theorem 8. *$L(\mathcal{E}) = \Diamond C_K$. Furthermore, $\mathcal{E}$ is complete and deterministic.*

Proof. "$\subseteq$." Let $\Diamond\mathsf{q} \in L(\mathcal{E})$, where $\mathsf{q} \in \Phi^{k\mathbf{b},m\mathbf{r}}$. By the construction of $\mathcal{E}$, there exists $\mathsf{p} \in B_J \cap B'_{J'} \subseteq \Upsilon^{k\mathbf{b},m\mathbf{r}}$, with blue node position-tuple $\mathbf{x}$, such that $\mathsf{p}_\mathbf{x} = \mathsf{q}$.

Since $\mathsf{p} \in B_J \cap B'_{J'}$, $\mathsf{p}^{\neg\mathbf{r}} \in J$ and $\mathsf{p}^{\neg\mathbf{b}} \in J'$. Also, observe that $\mathsf{p}^{\neg\mathbf{r}} \mathbin{\dot{\star}} (\mathsf{p}^{\neg\mathbf{b}})_\mathbf{x} = \mathsf{p}^{\neg\mathbf{b}}$. These imply that $(\mathsf{p}^{\neg\mathbf{b}})_\mathbf{x} \in J \mathbin{\dot{\star}} J' = K$, which means $\mathsf{p}_\mathbf{x} = \mathsf{q} \in C_K$, i.e. $\Diamond\mathsf{q} \in \Diamond C_K$.

"$\supseteq$." Let $\Diamond\mathsf{q} \in \Diamond C_K \subseteq \Diamond\Phi^{k\mathbf{b},m\mathbf{r}}$, where $\mathsf{q} \in C_K \subseteq \Phi^{k\mathbf{b},m\mathbf{r}}$, i.e. $\mathsf{q}^{\neg\mathbf{b}} \in K$. By the definition of K, there exists $\mathsf{p} \in J$ such that $\mathsf{p} \mathbin{\dot{\star}} \mathsf{q}^{\neg\mathbf{b}}$ is defined and in J'. This implies $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}} \in B_J$. By the definition of "$\star$" and $B'_{J'}$, $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}} \in B'_{J'}$, too, therefore, $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}} \in B_J \cap B'_{J'}$, i.e. $\mathsf{p} \star \mathsf{q}^{\neg\mathbf{b}}$ is accepted by $\mathcal{D}$. Now, by the construction of $\mathcal{E}$, we have $\Diamond\mathsf{q}$ is accepted by $\mathcal{E}$. □

Complexity. It can be verified that, similarly as in the case of unary queries (Subsection 5.3), computing R can be done in singly-exponential time when starting with non-deterministic FTAs for for Q and V. By Theorem 6, which applies to the case of $k = m = 1$, it follows that our constructions are optimal.

9 Conclusions

In this paper, we studied the problem of rewriting queries using views for XML data, focusing on tree-selecting queries equal in power to MSO. Colored tree languages and automata provided us a critical ground for implementing rewriting queries using views. We developed a singly-exponential algorithm for deriving the maximally contained rewriting of a query using a view, and then extended it to an algorithm for the general case where the query can be an m-ary query and the view can be a k-ary view. The latter class of queries is useful in optimizing the `FOR` clause of XQuery expressions. We showed the problem is EXPTIME-hard, thus showing our algorithms are optimal. Allowing our query formalism to also involve selection, join and restructuring is an avenue worth exploring. Extending the reasoning about QRV for this wider class is an interesting open problem.

References

1. L. Afanasiev, P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. Marx, and M. de Rijke. PDL for ordered trees. *Journal of Applied Non-Classical Logics*, 15(2):115–135, 2005.
2. F. N. Afrati, R. Chirkova, M. Gergatsoulis, B. Kimelfeld, V. Pavlaki, and Y. Sagiv. On rewriting XPath queries using views. In *EDBT*, 2009.
3. M. Arenas, P. Barceló, and L. Libkin. Combining temporal logics for querying XML documents. In *ICDT*, 2007.
4. A. Arion, V. Benzaken, I. Manolescu, and Y. Papakonstantinou. Structured materialized views for XML queries. In *VLDB*, 2007.
5. A. Balmin, F. Özcan, K. S. Beyer, R. Cochrane, and H. Pirahesh. A framework for using materialized XPath views in XML query processing. In *VLDB*, 2004.
6. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. An automata-theoretic approach to regular XPath. In *DBPL*, 2009.
7. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. Node selection query languages for trees. In *AAAI*, 2010.
8. B. Cautis, A. Deutsch, and N. Onose. XPath rewriting using multiple views: Achieving completeness and efficiency. In *WebDB*, 2008.
9. P. Claves, D. Jansen, S. J. Holtrup, M. Mohr, A. Reis, M. Schatz, and I. Thesing. LETHAL: Library for working with finite tree and hedge automata. Available on: `http://lethal.sf.net`, 2009.
10. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2007.
11. W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Rewriting regular XPath queries on XML views. In *ICDE*, 2007.

12. E. Filiot. Ranked and unranked tree automata libraries (grappa). Available on: `http://www.grappa.univ-lille3.fr/~filiot/tata`.
13. E. Filiot, J. Niehren, J.-M. Talbot, and S. Tison. Composing monadic queries in trees. In *PLAN-X*, 2006.
14. M. Frick, M. Grohe, and C. Koch. Query evaluation on compressed trees (extended abstract). In *LICS*, 2003.
15. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
16. L. V. S. Lakshmanan, H. Wang, and Z. J. Zhao. Answering tree pattern queries using views. In *VLDB*, 2006.
17. L. Libkin and C. Sirangelo. Reasoning about XML with temporal logics and automata. *J. Applied Logic*, 8(2):210–232, 2010.
18. G. Miklau and D. Suciu. Containment and equivalence for a fragment of XPath. *JACM*, 51(1):2–45, 2004.
19. F. Neven. *Design and Analysis of Query Languages for Structured Documents–A Formal and Logical Approach.* PhD thesis. Limburgs Universitair Centrum, 1999.
20. F. Neven and T. Schwentick. Expressive and efficient pattern languages for tree-structured data. In *PODS*, 2000.
21. F. Neven and T. Schwentick. Query automata over finite trees. *TCS*, 275(1-2):633–674, 2002.
22. F. Neven and T. Schwentick. On the complexity of XPath containment in the presence of disjunction, DTDs, and variables. *Logical Methods in Computer Science*, 2(3), 2006.
23. J. Niehren, L. Planque, J.-M. Talbot, and S. Tison. N-ary queries by tree automata. In *DBPL*, 2005.
24. N. Onose, A. Deutsch, Y. Papakonstantinou, and E. Curtmola. Rewriting nested XML queries using nested views. In *SIGMOD Conf.*, 2006.
25. T. Schwentick. On diving in trees. In *MFCS*, 2000.
26. T. Schwentick. Automata for XML - a survey. *J. Comput. Syst. Sci.*, 73(3):289–315, 2007.
27. B. ten Cate and L. Segoufin. XPath, transitive closure logic, and nested tree walking automata. In *PODS*, 2008.
28. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.
29. A. Thomo and S. Venkatesh. Rewriting of VPLs for XML data integration. In *CIKM*, 2008.
30. J. Wang, J. Li, and J. X. Yu. Answering tree pattern queries using views: a revisit. In *EDBT*, 2011.
31. W. Xu and Z. M. Özsoyoglu. Rewriting XPath queries using materialized views. In *VLDB*, 2005.
32. C. Yu and L. Popa. Constraint-based XML query rewriting for data integration. In *SIGMOD*, 2004.

10 Appendix

10.1 Intersection

Given FTAs $\mathcal{A}_1 = (S_1, \Sigma, F_1, \Delta_1)$ and $\mathcal{A}_2 = (S_2, \Sigma, F_2, \Delta_2)$ for L_1 and L_2, respectively, we construct FTA

$$\mathcal{A} = (S_1 \times S_2, \Sigma, F_1 \times F_2, \Delta_1 \otimes \Delta_2)$$

where $\otimes$ is defined as follows.

We denote $(s_1, s_2) \in S_1 \times S_2$ by s_{12}. Let H_1 and H_2 be two regular languages over S_1 and S_2, respectively, We define

$$H_1 \otimes H_2 = \{s_{i_1 j_1} \dots s_{i_n j_n} : s_{i_1} \dots s_{i_n} \in H_1, \\ s_{j_1} \dots s_{j_n} \in H_2, \text{ and } n \in \mathbb{N}\}.$$

Also, we complete $H_1 \otimes H_2$ by adding ϵ to it, if $\epsilon \in H_1$ and $\epsilon \in H_2$. Given ϵ-free finite state automata (FSAs) for H_1 and H_2 one can easily build an FSA for $H_1 \otimes H_2$ in polynomial time. Now, we define

$$\Delta_1 \otimes \Delta_2 = \{H_1 \otimes H_2 \xrightarrow{a} s_{12} : H_1 \xrightarrow{a} s_1 \text{ in } \Delta_1, \text{ and} \\ H_2 \xrightarrow{a} s_2 \text{ in } \Delta_2\}.$$

Proposition 8. *(1) $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. (2) Furthermore, (i) if $\mathcal{A}_1$ and $\mathcal{A}_2$ are complete, so is $\mathcal{A}$; (ii) if $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic, so is $\mathcal{A}$.*

Proof. Let ρ be a accepting run of $\mathcal{A}$ on a tree t. Notice ρ is a tree with the same shape as t, but with nodes labeled by states $s_{ij} \in S_1 \times S_2$. Consider now ρ_1 obtained by changing each s_{ij} in ρ to s_i, and ρ_2 obtained by changing each s_{ij} in ρ to s_j. It is easy to see ρ_1 and ρ_2 are accepting runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ on t, respectively, and so $L(\mathcal{A}) \subseteq L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Conversely, let ρ_1 and ρ_2 be accepting runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ on a tree t. ρ_1 and ρ_2 are trees with the same shape as t, but with nodes labeled by states in S_1 and S_2, respectively. Now consider the tree ρ with the same shape as t, but with nodes labeled by elements $s_{ij} \in S_1 \times S_2$, such that s_i and s_j label the corresponding nodes in ρ_1 and ρ_2, respectively. Clearly, ρ is a accepting run of $\mathcal{A}$ on t and so, $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \subseteq L(\mathcal{A})$. This proves claim (1).

Furthermore, it is clear from the construction of $\mathcal{A}$ and from that of runs ρ, ρ_1, ρ_2 above that whenever $\mathcal{A}_1$ and $\mathcal{A}_2$ are complete, so is $\mathcal{A}$. Suppose now $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic. From the construction, and since deterministic FTAs have at most one run on any tree data t, it follows that $\mathcal{A}$ is deterministic as well. □

10.2 Avoiding the coincidence of query and view markings

The transformation. Let $\# \notin \Sigma$. If $\mathsf{p} \in \Upsilon^{\mathbf{r}}$, we define $\mathsf{p}\#$ to be the pattern obtained from p by introducing a dummy node ξ_{xy} for each parent-child pair

x, y of nodes in p, and making ξ_{xy} child of x and parent of y, and labeling ξ_{xy} by $\#$. For an example, see Figure 3, top. Furthermore, if p has the root red, then we create a new root labeled by $\#$ with the old root as a child.

Let $\#^{\mathbf{b}}$ be the blue variant of $\#$. If $\mathsf{p} \in \Upsilon^{\mathbf{b}}$, we define $\mathsf{p}\#$ to be the pattern obtained from p by introducing as before a dummy node ξ_{xy}, for each parent-child pair x, y of nodes in p, and making ξ_{xy} child of x and parent of y. If y is black, ξ_{xy} is labeled by $\#$. Otherwise, i.e., if y is blue, ξ_{xy} is labeled by $\#^{\mathbf{b}}$, and y changes color to black. For an example, see Figure 3, bottom. Furthermore, if p has the root blue, then we create a new root labeled by $\#^{\mathbf{b}}$ with the old root as a child. The old root is changed in this case to black.

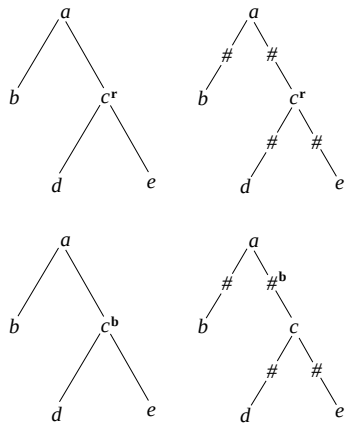

Fig. 3. Transforming a pattern in $\Upsilon^{\mathbf{r}}$ (top), and a pattern in $\Upsilon^{\mathbf{b}}$ (bottom).

Let $Q \subseteq \Upsilon^{\mathbf{r}}$ and $V \subseteq \Upsilon^{\mathbf{b}}$. We define

$$Q' = \{\mathsf{p}\# : \mathsf{p} \in Q\} \text{ and } V' = \{\mathsf{p}\# : \mathsf{p} \in V\}.$$

Clearly:

> The red node in a Q' pattern will never coincide with the blue node in a matching V' pattern.

Obtaining FTAs for Q' and V'. Let $\mathcal{A} = (S_{\mathcal{A}}, \Sigma \cup \Sigma^{\mathbf{r}}, F_{\mathcal{A}}, \Delta_{\mathcal{A}}\})$ be a targeted FTA for Q, where $S_{\mathcal{A}} = \{s_1, \ldots, s_{n_{\mathcal{A}}}\}$. Let $\hat{S}_{\mathcal{A}} = \{\hat{s}_1, \ldots, \hat{s}_{n_{\mathcal{A}}}\}$ be a set of new states, each corresponding to a state in $S_{\mathcal{A}}$.

We construct $\mathcal{A}' = (S_{\mathcal{A}'}, \Sigma \cup \Sigma^{\mathbf{r}} \cup \{\#\}, F_{\mathcal{A}'}, \Delta_{\mathcal{A}'}\})$, where

$$
\begin{aligned}
S_{\mathcal{A}'} &= S_{\mathcal{A}} \cup \hat{S}_{\mathcal{A}} \\
F_{\mathcal{A}'} &= \{\hat{s}_i : s_i \in F_{\mathcal{A}}\} \cup \\
&\qquad \{s_i : s_i \in F_{\mathcal{A}} \text{ and there exists } H \xrightarrow{a^{\mathbf{r}}} s_i \text{ in } \Delta_{\mathcal{A}}\} \\
\Delta_{\mathcal{A}'} &= \{H \xrightarrow{a} \hat{s}_i \text{ and } \hat{s}_i \xrightarrow{\#} s_i : H \xrightarrow{a} s_i \text{ in } \Delta_{\mathcal{A}}\} \cup \\
&\qquad \{H \xrightarrow{a^{\mathbf{r}}} \hat{s}_i \text{ and } \hat{s}_i \xrightarrow{\#} s_i : H \xrightarrow{a^{\mathbf{r}}} s_i \text{ in } \Delta_{\mathcal{A}}\}.
\end{aligned}
$$

We have that

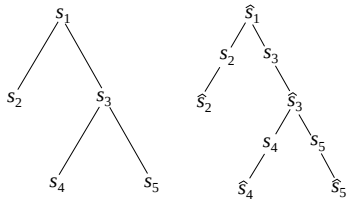

Fig. 4. Corresponding runs of $\mathcal{A}$ and $\mathcal{A}'$.

Proposition 9. *$\mathcal{A}'$ recognizes Q'.*

Proof. The claim follows from the observation that there is a one-to-one correspondence between the runs of $\mathcal{A}$ and the runs of $\mathcal{A}'$. An example is given in Figure 4. Let ρ, ϱ be corresponding runs. If ρ is a run of $\mathcal{A}$ on a pattern p, then ϱ is a run $\mathcal{A}'$ on $p\#$, and vice-versa. □

Let $\mathcal{B} = (S_{\mathcal{B}}, \Sigma \cup \Sigma^{\mathbf{b}}, F_{\mathcal{B}}, \Delta_{\mathcal{B}}\})$ be a targeted FTA for V, where $S_{\mathcal{B}} = \{s_1, \ldots, s_{n_{\mathcal{B}}}\}$. Let $\hat{S}_{\mathcal{B}} = \{\hat{s}_1, \ldots, \hat{s}_{n_{\mathcal{B}}}\}$ be a set of new states, each corresponding to a state in $S_{\mathcal{B}}$. We construct $\mathcal{B}' = (S_{\mathcal{B}'}, \Sigma \cup \Sigma^{\mathbf{b}} \cup \{\#, \#^{\mathbf{b}}\}, F_{\mathcal{B}'}, \Delta_{\mathcal{B}'}\})$, where

$$
\begin{aligned}
S_{\mathcal{B}'} &= S_{\mathcal{B}} \cup \hat{S}_{\mathcal{B}} \\
F_{\mathcal{B}'} &= \{\hat{s}_i : s_i \in F_{\mathcal{B}}\} \cup \\
&\qquad \{s_i : s_i \in F_{\mathcal{B}} \text{ and there exists } H \xrightarrow{a^{\mathbf{b}}} s_i \text{ in } \Delta_{\mathcal{B}}\} \\
\Delta_{\mathcal{B}'} &= \{H \xrightarrow{a} \hat{s}_i \text{ and } \hat{s}_i \xrightarrow{\#} s_i : H \xrightarrow{a} s_i \text{ in } \Delta_{\mathcal{B}}\} \cup \\
&\qquad \{H \xrightarrow{a} \hat{s}_i \text{ and } \hat{s}_i \xrightarrow{\#^{\mathbf{b}}} s_i : H \xrightarrow{a^{\mathbf{b}}} s_i \text{ in } \Delta_{\mathcal{B}}\}.
\end{aligned}
$$

We can show, similarly as for Proposition 9, that

Proposition 10. *$\mathcal{B}'$ recognizes V'.*

Rewriting. Let R' be the MCR of Q' using V'. The automaton for R' should be transformed into an automaton over the original $\Sigma \cup \Sigma^{\mathbf{r}}$ in order to be useful to extract the answer from the view extension. For this, we will make use of "ϵ"-transition rules (ϵ-TRs) of the form $s' \longrightarrow s$.

Let $\mathcal{C}' = (S_{\mathcal{C}'}, \Sigma \cup \Sigma^{\mathbf{r}} \cup \{\#\}, F_{\mathcal{C}'}, \Delta_{\mathcal{C}'})$ be a targeted FTA for R'. By the definition of R', each transition rule $H \xrightarrow{\#} s$ of $\mathcal{C}'$ should be such that $H \subseteq S_{\mathcal{C}'}$, i.e., the words of H are singleton states. Therefore, we can replace each $H \xrightarrow{\#} s$ transition rule by a set of $s' \xrightarrow{\#} s$ transition rules, for $s' \in H$.

Now we finally produce $\mathcal{C}$ that is the same as $\mathcal{C}'$, but with the $s' \xrightarrow{\#} s$ transition rules replaced by corresponding $s' \longrightarrow s$ ϵ-transition rules. $\mathcal{C}$ is an FTA over $\Sigma \cup \Sigma^{\mathbf{r}}$. It can be shown that FTAs with ϵ-TRs can be converted into equivalent FTAs without ϵ-TRs (cf. [10]).