Node Query Preservation for Deterministic Linear Top-Down Tree Transducers

Kazuki Miyahara
Nara Institute of Science and Technology
Nara, Japan
kazuki-mi@is.naist.jp

Kenji Hashimoto
Nagoya University
Nagoya, Japan
{k-hasimt,seki}@is.nagoya-u.ac.jp

Hiroyuki Seki

This paper discusses the decidability of node query preservation problems for XML document transformations. We assume a transformation given by a deterministic linear top-down data tree transducer (abbreviated as DLT\(^V\)) and an \(n\)-ary query based on runs of a tree automaton. We say that a DLT\(^V\) \(T_r\) strongly preserves a query \(Q\) if there is a query \(Q'\) such that for every document \(t\), the answer set of \(Q'\) for \(T_r(t)\) is equal to the answer set of \(Q\) for \(t\). Also we say that \(T_r\) weakly preserves \(Q\) if there is a query \(Q'\) such that for every \(t_d\) in the range of \(T\), the answer set of \(Q'\) for \(t_d\) is equal to the union of the answer set of \(Q\) for \(t\) such that \(t_d = T_r(t)\). We show that the weak preservation problem is coNP-complete and the strong preservation problem is in 2-EXPTIME.

1 Introduction

Due to data exchanges and schema updates, long-term databases often require XML document transformations. A fundamental concern with these XML document transformations is that the information contained in each source document should be preserved in the target document obtained by transforming the source document. Query preservation [2] is one of the formulations of such information preservation. A transformation preserves a query \(Q\) over the source documents if there is a query \(Q'\) on the target documents such that the answer of \(Q\) on a source document is equivalent to the answer of \(Q'\) on the target document.

The query preservation problem was shown in [2] to be undecidable for the class of transformations and queries which can simulate first-order logic queries and projection queries, respectively, in the relational data model. In [1, 4], query preservation has been studied in a setting such that transformations and queries are both modeled by tree transducers. In [4], the query preservation was shown to be decidable for compositions of functional linear extended top-down tree transducers (with regular look-ahead) as transformations, and for either deterministic top-down tree transducers (with regular look-ahead) or deterministic MSO definable tree transducers as queries. Also, query preservation (or, determinacy) was considered in [3] as relations between XML views in the context of unranked trees. In [3], views are defined as transformations that leave nodes selected by node queries, such as Regular XPath queries and MSO queries. In this paper, we focus on query preservation of a tree transducer for a node query model, that is, types of models for transformations and queries are different. We consider \(n\)-ary queries, which extract sets of \(n\)-tuples of nodes. Query preservation for tree-to-tree queries requires that the tree structure of the source query result should be restored by some query to the transformed data. In our setting, query preservation requires that a tree transducer should maintain the possibility to extract the relationship between (the values of) the nodes specified by a node query, rather than the tree structures.
In this paper, we model an XML document by a data tree, which is a ranked ordered tree where each node can have any nonnegative integer as a data value. We assume a transformation given by a deterministic linear top-down data tree transducer (abbreviated as $DLT^V$) and a run-based $n$-ary query \cite{5} (equivalent with an MSO $n$-ary query \cite{6}). As defined in the next section, the transformation is determined independently of data values assigned to nodes, though some data values can be transferred from input to output without duplication. The answer set of a query is the set of tuples of data values which are assigned to nodes selected by the query instead of the selected nodes themselves. We say that a $DLT^V$ $Tr$ strongly preserves a query $Q$ if there is a query $Q'$ such that for every document $t$, the answer set of $Q'$ for $Tr(t)$ is equal to the answer set of $Q$ for $t$. Also we say that $Tr$ weakly preserves $Q$ if there is a query $Q'$ such that for every $t_d$ in the range of $Tr$, the answer set of $Q'$ for $t_d$ is equal to the union of the answer set of $Q$ for $t$ such that $t_d = Tr(t)$. We show that the weak query preservation problem is coNP-complete. If the tuple size $n$ of queries is a constant, the complexity becomes PTIME. We also show that the strong preservation problem is in 2-EXPTIME. The decidability result of the two cases can be extended to the situation where the transformation is given by a $DLT^V$ with regular look-ahead.

2 Preliminaries

2.1 Data Trees

We denote the set of all nonnegative integers by $\mathbb{N}$. For $n \in \mathbb{N}$, the set $\{1, \ldots, n\}$ is denoted by $[n]$. A (ranked) alphabet is a finite set $\Sigma$ of symbols with a mapping $rk : \Sigma \to \mathbb{N}$. Let $\Sigma_n = \{ \sigma \in \Sigma \mid rk(\sigma) = n \}$. A data tree is a tree such that each symbol of the tree can have a nonnegative integer as a data value. Formally, the set $T^{(n)}_{\Sigma}$ of data trees over an alphabet $\Sigma$ is the smallest set $T$ such that $\sigma(t_1, \ldots, t_n) \in T$ and $\sigma(\nu)(t_1, \ldots, t_n) \in T$ for every $\sigma \in \Sigma_n$, $t_1, \ldots, t_n \in T$ and $\nu \in \mathbb{N}$. For a data tree $t$, the set of positions (nodes) $pos(t)$ is defined in the usual way and let $t/v$ denote the subtree of $t$ at position $v \in pos(t)$. If $t/v = \sigma(\nu)(t_1, \ldots, t_n)$, we write $lab(t, v) = \sigma$ and $val(t, v) = \nu$. If $t/v = \sigma(t_1, \ldots, t_n)$, we write $lab(t, v) = \sigma$. Let $t[v \leftarrow t']$ be the tree obtained from $t$ by replacing $t/v$ with $t'$. A data tree $t$ is proper if every symbol appearing in $t$ has a value. A tree is a data tree that does not contain any value. Let $T_{\Sigma}$ denote the set of all trees over $\Sigma$. For a data tree $t$, let $t^-$ denote the tree obtained from $t$ by removing all the values in $t$. For every $n \geq 1$, let $X_n = \{ x_i \mid i \in [n] \}$ be a set of variables with $rk(x_i) = 0$ for every $x_i \in X_n$. A tree $t$ is linear if each variable occurs at most once in $t$. A linear tree in $T_{\Sigma \cup X_n}$ is called an (n-ary) context over $\Sigma$. Let $C(\Sigma, X_n)$ denote the set of $n$-ary contexts over $\Sigma$. For a context $C \in C(\Sigma, X_n)$, let $C[t_1, \ldots, t_n]$ denote the tree obtained from $C$ by replacing $x_i$ with $t_i$ for $1 \leq i \leq n$.

2.2 Tree Automata and Tree Transducers

A tree automaton (TA) is a tuple $A = (P, \Sigma, P_I, \delta)$ where $P$ is a finite set of states, $\Sigma$ is a ranked alphabet, $P_I \subseteq P$ is a set of initial states, and $\delta$ is a finite set of transition rules of the form

$$p \rightarrow \sigma(p_1, \ldots, p_n)$$

where $p \in P$, $\sigma \in \Sigma_n$, and $p_1, \ldots, p_n \in P$. Let state($A$) = $P$. TA $A$ accepts a tree $t \in T_{\Sigma}$ if there is a mapping $m : pos(t) \rightarrow P$ such that (1) $m(\varepsilon) \in P_I$, and (2) for $v \in pos(t)$ with $t/v = \sigma(t_1, \ldots, t_n)$, $m(v) \rightarrow \sigma(m(v_1), \ldots, m(vn)) \in \delta$. The mapping $m$ is called an accepting run of $A$ on $t$. The set
of all accepting runs of $A$ on $t$ is denoted by $\text{run}(A,t)$. Let $L(A) = \{ t \in T_\Sigma \mid \text{run}(A,t) \neq \emptyset \}$. We simply write a run to mean an accepting run. A state of $A$ is useless if it is not assigned to any position by any accepting run of $A$, and a rule is useless if it has a useless state. A TA $A$ is said to be reduced, if $A$ has no useless states and transition rules.

A linear top-down data tree transducer (LT$^V$) is a tuple $Tr = (P, \Sigma, \Delta, P_I, \delta)$ where $P$ is a finite set of states, $\Sigma$ and $\Delta$ are ranked alphabets of input and output, respectively, $P_I \subseteq P$ is a set of initial states, and $\delta$ is a finite set of transition rules of the form

$$p(\sigma^{(z)}(x_1, \ldots, x_n)) \rightarrow C^{(j+\nu)}[p_1(x_1), \ldots, p_n(x_n)],$$

where $p, p_1, \ldots, p_n \in P$, $\sigma \in \Sigma_n$, $j \in \{ v \mid v \in \text{pos}(V), t/v \notin X_n \}$, $x_1, \ldots, x_n \in X_n$, $C \in C(\Delta, X_n)$, and $(j \leftarrow z)$ is optional. We call $(j \leftarrow z)$ the value position designation of the rule. The move relation $\Rightarrow_{Tr}$ of an LT$^V$ $Tr = (P, \Sigma, \Delta, P_I, \delta)$ is defined as follows: If $p(\sigma^{(z)}(x_1, \ldots, x_n)) \rightarrow C^{(j+\nu)}[p_1(x_1), \ldots, p_n(x_n)] \in \delta$, $t_1, \ldots, t_n \in T_\Sigma^{(n)}$ and $t/v = p(\sigma^{(v)}(x_1, \ldots, x_n))$ ($\nu \in \mathbb{N}$), then

$$t \Rightarrow t[v \leftarrow C^{(j+\nu)}[p_1(t_1), \ldots, p_n(t_n)]]$$

where $C^{(j+\nu)}$ is the context obtained from $C$ by replacing $\text{lab}(C,j)$ with $\text{lab}(C,j)^{(\nu)}$. When the value position designation is missing in the rule, we do not copy $\nu$ to any position of the output. Let $\llbracket Tr \rrbracket = \{ (t, t') \mid p_I(t) \Rightarrow_{Tr}^* t', t, t' \in T_{\Sigma}^{(n)}, t \text{ proper}, t' \in T_{\Delta}^{(n)}, p_I \subseteq P_I \}$. The domain of $Tr$ is defined as $\text{dom}(Tr) = \{ t \mid \exists t'. (t, t') \in \llbracket Tr \rrbracket \}$, and the range of $Tr$ is defined as $\text{rng}(Tr) = \{ t' \mid \exists t. (t, t') \in \llbracket Tr \rrbracket \}$. An LT$^V$ $Tr = (P, \Sigma, \Delta, P_I, \delta)$ is deterministic (denoted as DLT$^V$) if (1) $|P_I| = 1$, and (2) for each $p \in P$ and $\sigma \in \Sigma$, there exists at most one transition rule that contains both $p$ and $\sigma$ in its left-hand side. If $Tr$ is deterministic, there is only one pair $(t, t') \in \llbracket Tr \rrbracket$ for each $t \in \text{dom}(Tr)$. Thus, we write $Tr(t) = t'$ when $(t, t') \in \llbracket Tr \rrbracket$. For $L \subseteq T_{\Sigma}^{(n)}$, we write $Tr(L) = \{ Tr(t) \mid t \in L \}$. We denote by $Tr^{-1}$ the inverse of $Tr$, i.e., $Tr^{-1}(t') = \{ t \mid Tr(t) = t' \}$. Let DL$^V$ be the class of ordinary deterministic linear top-down tree transducers over trees containing no data values.

A subtree-deleting rule is a rule such that at least one variable in its left-hand side does not occur in its right-hand side as $p_1(\sigma^{(z)}(x_1, x_2)) \rightarrow \sigma^{(z)}(p_2(x_2))$. A value-erasing rule is a rule that does not have the value position designation in its right-hand side.

### 2.3 Run-based $n$-ary Queries

A run-based $n$-ary query ($n$-RQ) [5] is a pair $(A, S)$ where $A = (P, \Sigma, P_I, \delta)$ is a TA and $S \subseteq P^n$. In this paper, we assume that each $s \in S$ consists of $n$ different states. We simply call a run-based $n$-ary query a query. For a data tree $t$ and a query $Q = (A, S)$, define

$$Q(t) = \bigcup_{m \in \text{run}(A,t^{-})} Q(m, t),$$

where $Q(m, t) = \{ (v_1, \ldots, v_n) \mid (m(v_1), \ldots, m(v_n)) \in S, v_1, \ldots, v_n \in \text{pos}(t) \}$. For an $n$-RQ $Q$ and a data tree $t$, let $\text{val}(Q(t)) = \{ \text{val}(t, v_1), \ldots, \text{val}(t, v_n) \} \mid (v_1, \ldots, v_n) \in Q(t) \}$.

We assume that for all query $Q = (A, S)$, the TA $A$ is reduced.

**Example 1.** Consider 2-RQ $Q = (A, S)$ defined by: $A = (P, \Sigma, P_I, \delta)$, $P = \{ p_1, p_2, p_3, p_4 \}$, $\Sigma_0 = \{ a \}$, $P_I = \{ p_1 \}$, $\delta = \{ p_1 \rightarrow f(p_2, p_3), p_2 \rightarrow a, p_3 \rightarrow f(p_4, p_3), p_4 \rightarrow a, p_4 \rightarrow a \}$, and $S = \{ (p_2, p_3) \}$. Figure 1 shows that the result of the query on the data tree $t = f^{(1)}(a^{(2)}, f^{(3)}(a^{(4)}, a^{(5)})))$
is \( \text{val}(Q(t)) = \{(2, 3), (2, 5)\} \), where the numbers 1 to 5 are the data values on \( t \).

\[
\begin{array}{c}
p_1 \quad f^{(1)} \\
p_2 \quad a^{(2)} \\
p_3 \quad f^{(3)} \\
p_4 \quad a^{(4)} \\
p_5 \quad a^{(5)} \\
S = \{ (p_2, p_3) \} \quad \text{val}(Q(t))
\end{array}
\]

\[
\text{val}(Q(t)) = \{(2, 3), (2, 5)\}
\]

Figure 1: Example of 2-RQ.

### 2.4 Query Preservation

Let \( \mathcal{L}_T \) and \( \mathcal{L}_Q \) be a class of tree transducers and a class of queries, respectively. Given a query \( Q \in \mathcal{L}_Q \) and a tree transducer \( Tr \in \mathcal{L}_T \), we say that \( Tr \) (strongly) preserves \( Q \) if there exists \( Q' \in \mathcal{L}_Q \) that satisfies

\[
\text{val}(Q'(Tr(t))) = \text{val}(Q(t)) \tag{1}
\]

for any \( t \in \text{dom}(Tr) \). We also define the weak query preservation to simplify the discussions about the strong query preservation and its decision algorithm (see Section 4). We say that the transducer \( Tr \) weakly preserves the query \( Q \) if there exists \( Q' \in \mathcal{L}_Q \) such that for any \( t_d \in \text{rng}(Tr) \),

\[
\text{val}(Q'(t_d)) = \bigcup_{t \in Tr^{-1}(t_d)} \text{val}(Q(t)). \tag{2}
\]

By definition, we can see that \( Tr \) preserves \( Q \) if and only if (1) \( Tr \) weakly preserves \( Q \) and (2) for each \( t_d \in \text{rng}(Tr) \) and any two \( t_1, t_2 \in Tr^{-1}(t_d) \), \( \text{val}(Q(t_1)) = \text{val}(Q(t_2)) \) holds.

**Example 2.** Let \( Q = (A, \{p_1\}) \) where \( A = \{(p_0, p_1, p_2), \{f, g, a\}, \{p_0\}, \{p_0 \rightarrow f(p_1, p_2), \; p_0 \rightarrow g(p_2, p_1), \; p_1 \rightarrow a, \; p_2 \rightarrow a\}\} \). Let \( Tr \) be a DLT\(^{V} \) defined by the homomorphism that maps \( f, g, a \) to \( h, h, a \), respectively (and moves each data value as well). We can see that \( L(A) = \{f(a, a), g(a, a)\} \).

Let \( t_1 = f^{(3)}(a^{(4)}, a^{(5)}) \) and \( t_2 = g^{(3)}(a^{(4)}, a^{(5)}) \). Then \( Tr(t_1) = Tr(t_2) = h^{(3)}(a^{(4)}, a^{(5)}) \). In this example, \( Tr \) weakly preserves \( Q \). In fact, \( Q' \) obtained from \( Q \) by replacing the first two rules of \( A \) with \( p_0 \rightarrow h(p_1, p_2), \; p_0 \rightarrow h(p_2, p_1) \) satisfies Equation (2). On the other hand, \( Tr \) does not preserve \( Q \) because \( \text{val}(Q(t_1)) = \{4\} \neq \{5\} = \text{val}(Q(t_2)) \) while \( Tr(t_1) = Tr(t_2) \), which imply that there is no \( Q' \) that satisfies Equation (1).
3 Decidability of Weak Query Preservation

3.1 Unary Queries

3.1.1 Decidability

We show an algorithm that decides if a DLT\(^V\) \(Tr\) weakly preserves a 1-RQ \(Q\). We also prove that if \(Tr\) weakly preserves \(Q\), we can effectively construct a 1-RQ \(Q'\) that satisfies Equation (2) (the condition of the weak query preservation). Assume \(Q = (A, \{p\})\) with \(p \in \text{state}(A)\) for a while. Our algorithm for weak query preservation decides if there exist a tree \(t \in \text{dom}(Tr)\) and a position \(\tilde{v} \in \text{pos}(t)\) satisfying the next conditions:

- There exists a run \(m \in \text{run}(A, t^-)\) such that \(m(\tilde{v}) = p\).
- The data load at \(\tilde{v}\) on \(t\) is “removed” by a subtree-deleting rule or a value-erasing rule of \(Tr\).

Assume there exist a data tree \(t\) and a position \(\tilde{v}\) of \(t\) that satisfy the above conditions. Let \(t_d = Tr(t)\), then,

\[
\bigcup_{t' \in Tr^{-1}(t_d)} \text{val}(Q(t')) = \mathbb{N},
\]

because \(t(\tilde{v} - \nu) \in Tr^{-1}(t_d)\) and \(\nu \in \text{val}(Q(t'(\tilde{v} - \nu)))\) for any \(\nu \in \mathbb{N}\). However, there is no \(Q'\) satisfying \(\text{val}(Q'(t_d)) = \mathbb{N}\) because the left-hand side is a finite set. Thus, there exists no 1-RQ \(Q\) that satisfies Equation (2). Conversely, if such \(t\) and \(\tilde{v}\) do not exist, we can specify the position of \(Tr(t)\) that has the data value queried by \(Q'\) on \(Tr(t)\), by constructing a tree automaton that simulates each accepting run \(m \in \text{run}(A, t^-)\) on \(Tr(t)\).

Algorithm 1-WQP to Decide Weak Query Preservation

Step 1 constructs a TA \(A_T\) that specifies positions of \(t\) that will be deleted by a transducer \(Tr\).

Step 2 constructs a product automaton \(A'\) that satisfies \(L(A') = L(A) \cap L(A_T)\) to find positions that will be in \(Q(t)\) and be deleted by \(Tr\).

**Input:** 1-RQ \(Q = (A,S)\) where \(A = (P_A, \Sigma, P_A^I, \delta_A)\) is a TA and \(S \subseteq P_A\), DLT\(^V\) \(Tr = (P_T, \Sigma, \Delta, \{p_0^T\}, \delta_T)\).

**Output:** If \(Tr\) weakly preserves \(Q\), output “Yes,” otherwise “No.”

**Step 1.** Construct the following TA \(A_T = (P_T \cup \{\perp\}, \Sigma, \{p_0^T\}, \delta_T)\) from \(Tr\) where \(\perp \notin P_T\) and \(\delta_T\) is the smallest set satisfying the following conditions.

- Let \(p(\sigma^z(x_1, \ldots, x_n)) \rightarrow C^{(i \leftarrow z)}[p_1(x_1), \ldots, p_n(x_n)] \in \delta_T\) with \(p, p_1, \ldots, p_n \in P_T\), \(\sigma \in \Sigma_n\), and \(C \subseteq \Delta. A'_n\). For each \(i \in [n]\), define \(\tilde{p}_i\) as follows. If \(C\) contains \(x_i\), let \(\tilde{p}_i = p_i\). If \(C\) does not contain \(x_i\), let \(\tilde{p}_i = \perp\). Then, \(p \rightarrow \sigma(\tilde{p}_1, \ldots, \tilde{p}_n) \in \delta_T\).
- For each \(\sigma \in \Sigma, \perp \rightarrow \sigma(\perp, \ldots, \perp) \in \delta_T\).

**Step 2.** Construct a product TA \(A'\) of \(A\) and \(A_T\) that satisfies \(L(A') = L(A) \cap L(A_T)\). More specifically, construct the following tree automaton \(A' = (P_A \times P_T, \Sigma, P_A^I \times P_T, \delta')\) from \(Q = (A,S)\) and \(A_T = (P_T, \Sigma, P_T^I, \delta_T)\): \((p_A, p_T) \rightarrow \sigma ((p_A^1, p_T^1), \ldots, (p_A^k, p_T^k)) \in \delta'\) if and only if \(p_A \rightarrow \sigma(p_A^1, \ldots, p_A^k) \in \delta_A\) and \(p_T \rightarrow \sigma(p_T^1, \ldots, p_T^k) \in \delta_T\).

**Step 3.** Remove useless states and rules in \(A'\). Let \(A'' = (P'', \Sigma, P''_T, \delta'')\) be the resulting TA.
Construct a TA

Step 1. Output:  

Step 3. Compute if and only if  

Step 4. If the following subset  is empty, output “Yes,” otherwise “No.”  

Lemma 1. Let  be a 1-RQ and  be a DLTV.  weakly preserves  if and only if  in step 4 of the algorithm 1-WQCP.

3.1.2 Construction of Queries

If a transducer  weakly preserves a query , a query  on target documents can be constructed by a type-inference algorithm. The algorithm (called 1-WQC) works as follows: (1) Construct an automaton  from  such that , where  is the DLT obtained from  by removing the manipulation of values, and (2) construct  accordingly.

Algorithm 1-WQC to Construct Queries on Target Documents

Input: 1-RQ  where  is the DLTV  Tr = ( , , , { , } , ).  

Output: A 1-RQ  on target documents that satisfies Equation (2).

Step 1. Construct a TA  from  and  where  is defined as follows: For any rules  and  ,  where  , ,  . Let  .

- For each  such that , where  , and  is a mapping such that for each  ,
  
- if  
  
- if  
  
Step 2. Construct a reduced TA without epsilon rules equivalent with  . Formally, let if and only if  if for each rule  , , ,  , and there is a rule with  in its left-hand side and some symbol in  in its right-hand side. Then, remove all epsilon rules, useless states and transition rules of . Let  be the resulting TA.

Step 3. Compute  where  is the smallest subset of  satisfying the following conditions.
Given a K. Miyahara, K. Hashimoto & H. Seki

or not. Furthermore, if it preserves, the query \( Q \) by 1\( -n \) for

If the tuple size

3.2 General Case

However, this does not work in general because

Also

Example 3. Q

\[ A = (P, \Sigma, \{p_1\}) \]

\[ P = \{p_1, p_2, p_3, p_\#\} \]

\[ \Sigma = \Sigma_2 \cup \{\#\} \]

\[ \Sigma_2 = \{A, B, C\} \]

\[ P_I = \{p_1\} \]

\[ \delta = \{p_1 \rightarrow A(p_2, p_\#), p_2 \rightarrow B(p_3, p_\#), p_3 \rightarrow C(p_\#, p_\#), p_\# \rightarrow \#\} \]

\[ S = \{(p_1, p_2, p_3)\} \]

Also let \( Tr = (P, \Sigma, \{p_1\}, \delta_T) \) be DLT\(^V\) defined by

\[ \delta_T = \{p_1(A^{(\#)}(x_1, x_2)) \rightarrow A^{(\varepsilon \rightarrow \#)}(p_2(x_1), p_\#(x_2)), p_2(B^{(\#)}(x_1, x_2)) \rightarrow B^{(\varepsilon \rightarrow \#)}(p_3(x_1), p_\#(x_2)), p_3(C^{(\#)}(x_1, x_2)) \rightarrow \#, p_\#(\#) \rightarrow \#\} \]

where \( P \) and \( \Sigma_2 \) are the same as \( A_s \). Consider the following data trees:

\[ t_1 = A^{(1)}(B^{(2)}(C^{(3)}(\#, \#), \#), \#), \#) \]

\[ t_2 = A^{(1)}(C^{(3)}(\#, \#), \#) \]

Figure 2 shows \( Q(t_1), Q(t_2), Tr(t_1) \) and \( Tr(t_2) \), the application results of \( Q \) and \( Tr \) to \( t_1 \) and \( t_2 \). In fact, for any data tree \( t \), TA \( A_s \) never assigns \( p_3 \) to any position of \( t \) (and thus \( Q(t) = \emptyset \) because \( p_1, p_2, p_3 \) contains \( p_3 \)) if and only if DLT\(^V\) \( Tr \) deletes a subtree of \( t \) such that \( Tr \) assigns
Step 2. query preservation for

\[ n \]

Input: \( n \)

Algorithm preserved by

be done by augmenting each state

only if for every \( \delta \)

once in the input tree.

Let \( \text{pos}(t) \) be the reason described above.

\[ \text{If } \text{Tr}(t) \text{ to the root of the subtree. That is, the deletion of a subtree by } \text{Tr} \text{ does not violate the weak query preservation for } Q. \text{ However, if we consider } 1\text{-RQ } Q' = (A_s, \{p_1, p_2, p_3\}) \text{ instead of } Q \text{ and apply 1-WQP to } Q', \text{ then 1-WQP answers "No" (because } \text{Tr} \text{ does not weakly preserve } Q'). \]

To overcome the above mentioned problem, we modify \( A_s \) as \( A^F \) so that \( m \in \text{run}(A^F, t^-) \) only if for every \( p_j \) \((1 \leq j \leq n)\), there is \( u_j \in \text{pos}(t) \) such that \( m(u_j) = p_j \). This modification can be done by augmenting each state \( p \) with a subset \( P \) of \( \{p_1, \ldots, p_n\} \). For \( m \in \text{run}(A^F, t^-) \), if \( m \) assigns \((p, P)\) to a position \( v \), each state in \( P \) should be used at least once as the first component of a state in the subtree rooted at \( v \) (including \( v \)). Especially, an initial state of \( A^F \) is a pair of an initial state of \( A_s \) and \( \{p_1, \ldots, p_n\} \), meaning that each \( p_i \) \((1 \leq i \leq n)\) should be used at least once in the input tree. \( A^F \) is weakly preserved by \( \text{Tr} \) if and only if the original \( A_s \) is weakly preserved by \( \text{Tr} \).

**Algorithm n-WQP to Decide Weak Query Preservation**

**Input:** \( n\text{-RQ } Q = (A, S), A = (P_A, \Sigma, P^I_A, \delta_A), S \subseteq P^n_A \). DLT \( V \) \( \text{Tr} = (P_T, \Sigma, \Delta, \{p^0_T\}, \delta_T) \). We can assume that the set \( S \) of tuples of states has just one element \((p_1, \ldots, p_n)\), i.e., \( S = \{(p_1, \ldots, p_n)\} \), by the reason described above.

**Output:** If \( \text{Tr} \) weakly preserves \( Q \), output “Yes,” otherwise “No.”

**Step 1.** Let \( P_s = \{p_1, \ldots, p_n\} \subseteq P_A \). Construct \( A^F = (P_A \times 2^P, \Sigma, P^I_A \times \{P_s\}, \delta^F) \) from \( A \) where \( \delta^F \) is defined as follows: For \( p \rightarrow a(p^1_A, \ldots, p^n_A) \in \delta_A \) and \( P, P_1, \ldots, P_m \subseteq P_s \),

\[
(p, P) \rightarrow a((p^1_A, P_1), \ldots, (p^n_A, P_m)) \in \delta^F
\]

if and only if for each \( i \in [n] \),

- if \( p_i \in P \) and \( p = p_i \), then \( p_i \notin P_j \) for all \( j \in [m] \),
- if \( p_i \in P \) and \( p \neq p_i \), then there exists exactly one \( j \in [m] \) satisfying \( p_i \in P_j \), and
- if \( p_i \notin P \), then \( p_i \notin P_j \) for all \( j \in [m] \).

**Step 2.** Let \( Q^F = (A^F, \{p_1, \ldots, p_n\} \times 2^P) \). Decide whether \( \text{Tr} \) weakly preserves \( Q^F \) by 1-WQP.
We have that $Tr$ weakly preserves $Q$ if and only if $Tr$ weakly preserves $Q'$. If $n$ is not fixed, however, the above algorithm will take exponential time in $n$. The following theorem gives a lower bound for the weak query preservation problem.

**Theorem 2.** Given an $n$-RQ $Q$ and a DLT $Tr$, the problem of deciding whether $Tr$ weakly preserves $Q$ is coNP-complete. If it preserves, a query $Q'$ satisfying Equation (2) can be constructed.

We can create $n$-ary queries on the transformed documents by using a natural variant of $1$-$WQC$. Given a DLT $Tr$ and $1$-RQ $Q = (A, \{(p_1, \ldots, p_n)\})$, the variant (called $n$-$WQC$) works in almost the same way as $1$-$WQC$ except that it computes $S'' = \prod_{i=1}^{n} S_{p_i}$ at step 3 while $1$-$WQC$ computes $S'' = \bigcup_{i=1}^{n} S_{p_i}$.

## 4 Decidability of Query Preservation

We will provide an algorithm $n$-$Q\mathcal{P}$ that decides query preservation. To begin with, we give the following lemma.

**Lemma 2.** Let $Q = (A, \{s\})$ be an $n$-RQ. We can construct an $n$-RQ $\tilde{Q} = (\tilde{A}, \tilde{S})$ equivalent with $Q$ such that (1) there exist pairwise disjoint subsets $S_1, \ldots, S_n$ of state($\tilde{A}$) satisfying $\tilde{S} = S_1 \times \cdots \times S_n$, and (2) for every $t^- \in L(\tilde{A})$, $m \in \text{run}(\tilde{A}, t^-)$, and $i$ ($1 \leq i \leq n$), there exists exactly one $v \in \text{pos}(t)$ such that $m(v) \in S_i$.

Thus, any $n$-RQ $Q = (A, \{s_1, \ldots, s_k\})$ can be represented as the union of $n$-RQS $\tilde{Q}_1, \ldots, \tilde{Q}_k$ such that each $\tilde{Q}_i$ is equivalent with $(A, \{s_i\})$ and satisfies the two conditions in Lemma 2.

We explain the idea of our algorithm $n$-$Q\mathcal{P}$. Assume we are given a DLT $Tr$ and an $n$-RQ $Q = (A, S)$ where $S = S_1 \times \cdots \times S_n$ satisfying the conditions (1) and (2) in Lemma 2. As shown in Example 2, if $Tr$ weakly preserves $Q$ but $Tr$ does not strongly preserve $Q$, there are two different data trees $t_1$ and $t_2$ such that $Tr(t_1) = Tr(t_2)(= t')$, runs $m_1 \in \text{run}(A, t_1^-)$, $m_2 \in \text{run}(A, t_2^-)$, $m_1(v_1), m_2(v_2) \in S_i$, positions $v_1 \in \text{pos}(t_1)$, and $v_2 \in \text{pos}(t_2)$ that have different values.

To test whether this situation happens, the algorithm $n$-$Q\mathcal{P}$ introduces a “marked” input symbol $(i, \sigma)$ for each $i \in [n]$ and $\sigma \in \Sigma$; $n$-$Q\mathcal{P}$ first constructs a TA $A_{mk}$ that simulates $A$ except that for each $i \in [n]$ if $A$ assigns a state in $S_i$ when reading $\sigma$ at some position $v$ (which is unique by the conditions in Lemma 2), $A_{mk}$ assigns a state in $S_i$ when reading $(i, \sigma)$ at $v$. Then, $n$-$Q\mathcal{P}$ constructs $A'_{mk}$ such that $L(A'_{mk}) = T_{mk}^{-1}(T_{mk}(L(A_{mk})))$, where $T_{mk}$ is a “marked” version of $Tr$. This construction is possible because $LT^V$ and its inverse preserve regularity. Also, it constructs $A_{idx}$, which simulates $A$ while identifying a marked symbol $(i, \sigma)$ with $\sigma$. By the reason stated above, we can prove that $Tr$ preserves $Q$ if and only if $L(A_{mk}) = L(A_{idx}) \cap L(A'_{mk})$.

### Algorithm $n$-$Q\mathcal{P}$ to Decide Query Preservation

**Input:** $n$-RQ $Q = (A, \{s_1, \ldots, s_k\})$, DLT $Tr = (P_T, \Sigma, \Delta, \{p_T^0\}, \delta_T)$. We assume that $Tr$ weakly preserves $Q$ (which is a necessary condition that can be decided as shown in Theorem 2).

**Output:** If $Tr$ preserves $Q$, output “Yes,” otherwise “No.”

**Step 1.** For each $s_i$, construct $\tilde{Q}_i$ equivalent with $(A, \{s_i\})$ satisfying the conditions (1) and (2) in Lemma 2.
Step 2. For each $\tilde{Q}^i = (A^i, S^i)$ where $A^i = (P^i, \Sigma, P_I^i, \delta^i)$ and $S^i = S_1^i \times \cdots \times S_n^i$, construct TA $A_{mk}^i = (P^i, \Sigma \cup ([n] \times \Sigma), P_I^i, \delta_{mk}^i)$ from $A^i$ where $\delta_{mk}^i$ is defined as follows. For each rule of the form $p \rightarrow \sigma(p_1, \ldots, p_m) \in \delta^i$, if $p$ is in $S_1^i$ then $p \rightarrow (i, \sigma)(p_1, \ldots, p_m) \in \delta_{mk}^i$; otherwise $p \rightarrow \sigma(p_1, \ldots, p_m) \in \delta_{mk}^i$. Then, construct TA $A_{mk}$ as the union $A_{mk} = A_{mk}^1 \times \cdots \times A_{mk}^n$.

Step 3. Construct DLT $T_{mk} = (P_T, \Sigma \cup ([n] \times \Sigma), \Delta \cup ([n] \times \Delta), \{p_T^0, p_T^m\}, \delta_{mk}^T)$ where $\delta_{mk}^T$ is the smallest set satisfying the following conditions: For each $i \in [n]$ and for each rule $p_T(\sigma(z)(x_1, \ldots, x_m)) \rightarrow C(j \leftarrow z)[p_T^1(x_1), \ldots, p_T^m(x_m)] \in \delta_T^m$, let

$$p_T(\sigma(x_1, \ldots, x_m)) \rightarrow C[p_T^1(x_1), \ldots, p_T^m(x_m)] \in \delta_{mk}^T,$$

and

$$p_T((i, \sigma)(x_1, \ldots, x_m)) \rightarrow C[p_T^1(x_1), \ldots, p_T^m(x_m)] \in \delta_{mk}^T,$$

where $\text{lab}(\tilde{C}, j) = (i, \text{lab}(C, j))$, and for each $v \in \text{pos}(C)$ satisfying $v \neq j$, $\text{lab}(\tilde{C}, j) = \text{lab}(C, j)$.

Step 4. Construct TA $A_{mk}'$ such that

$$L(A_{mk}') = T_{mk}^{-1}(T_{mk}(L(A_{mk})))$$

by type inference and inverse type inference.

Step 5. Construct TA $A_{idx} = (P, \Sigma \cup ([n] \times \Sigma), P_I \cup ([n] \times P_I), \delta \cup \delta_{idx})$ from $A$ where $\delta_{idx}$ is defined as follows. For each rule of the form $p \rightarrow \sigma(p_1, \ldots, p_m) \in \delta$ and for each $i \in [n]$, let $p \rightarrow (i, \sigma)(p_1, \ldots, p_m) \in \delta_{idx}$.

Step 6. If $L(A_{mk}) = L(A_{idx}) \cap L(A_{mk}')$, output “Yes,” otherwise “No.”

Example 4. Recall $Q$ and $Tr$ in Example 2. By steps 1–4 of the above algorithm $n$-$QP$,

$$L(A_{mk}) = \{f((1, a), a), g((1, 1), a)\},$$

$$L(A_{mk}') = \{f((1, a), a), g((1, 1), a), f((1, a), a), g((1, a), a)\} = L(A_{idx}) \cap L(A_{mk}').$$

Hence, $L(A_{mk}) \not\subseteq L(A_{idx}) \cap L(A_{mk}')$ holds and the algorithm answers “No.”

Lemma 3. If $DLT^V Tr$ weakly preserves $n$-$RQ$ $Q$, then $Tr$ preserves $Q$ if and only if $L(A_{mk}) = L(A_{idx}) \cap L(A_{mk}')$ in step 5 of the algorithm $n$-$QP$.

Theorem 3. Given an $n$-$RQ$ $Q$ and a DLT$^V$ Tr, the problem of deciding whether Tr preserves $Q$ is in 2-EXPTIME. If it preserves, a query $Q'$ constructed by $n$-$WQC$ satisfies Equation (1).

5 Conclusion

We have studied the decidability problems of the weak query preservation and the strong query preservation. We have modeled an XML document by a data tree, a document transformation by a deterministic linear top-down data tree transducer, and a query to the tree by a run-based $n$-ary query. We showed the weak query preservation problem is coNP-complete for $n$-ary queries where $n$ is not fixed, and the problem becomes PTIME if $n$ is a constant. We also showed the strong query preservation problem is in 2-EXPTIME.
Future Work

Our model of tree transformations does not allow copying of elements on trees. Copying elements is one of the fundamental operations for XML, so it would be important to know whether the query preservation problems are decidable or not for a transformation model having a copy operation.

Acknowledgements

We would like to thank two anonymous referees for their helpful comments, which greatly improved the paper.

References


