NAIST-IS-DD1361012

Doctoral Dissertation

Query Preservation for Tree-Structured Data

Kazuki Miyahara

March 14, 2016

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Kazuki Miyahara

Thesis Committee:

Professor Minoru Ito	(Supervisor)
Professor Yuji Matsumoto	(Co-supervisor)
Professor Hiroyuki Seki	(Co-supervisor)
Associate Professor Yuichi Kaji	(Co-supervisor)
Assistant Professor Kenji Hashimoto	(Nagoya University)

Query Preservation for Tree-Structured Data*

Kazuki Miyahara

Abstract

Query preservation is a notion for information preservation when the structure of the data is updated. This thesis discusses the decidability of several query preservation problems for tree-structured data.

First, we consider the problem of deciding whether a query can be preserved by a nondeterministic view. It is known that preservation is decidable if views are given by single-valued non-copying devices such as compositions of single-valued extended linear top-down tree transducers with regular look-ahead, and queries are given by deterministic MSO tree transducers (where MSO stands for Monadic Second-Order logic). In this thesis, we extend the result to the case in which views are given by nondeterministic devices that are not always single-valued. We define two variants of preservation: universal preservation and existential preservation, and discuss their decidability.

Second, we discuss the decidability of node query preservation problems. We assume a view given by a deterministic linear top-down data tree transducer (DLT^{V}) and an n-ary query based on runs of a tree automaton. We say that a DLT^{V} Tr strongly preserves a query Q if there is a query Q' such that for every tree t in the domain of Tr, the answer set of Q' for $Tr(t)$ is equal to the answer set of Q for t. We also say that Tr weakly preserves Q if there is a query Q' such that for every t in the domain of Tr, the answer set of Q' for $Tr(t)$ includes the answer set of Q for t. We show that the weak preservation problem is coNP-complete and the strong preservation problem is in 2-EXPTIME. We also show that the problems are decidable when a given transducer is a single-valued extended linear top-down data tree transducer with regular look-ahead, which is a more expressive transducer than DLT^{V}.

Keywords:

tree automata, tree transducers, query preservation

*Doctoral Dissertation, Graduate School of Information Science,
Nara Institute of Science and Technology, NAIST-IS-DD1361012, March 14, 2016.

Acknowledgments

I am indebted to my supervisor Hiroyuki Seki. This thesis would not have been written without his support, encouragement, advice, and help. It all started when he taught me formal language theory and the field of theoretical computer science, or, presumably, when I said "hello" naïvely in front of the door of his office on the open day of NAIST in May 2010.

I would like to express my deep appreciation to Kenji Hashimoto, who freely shared his profound knowledge, insights, and ideas about the theory of tree automata and tree transducers. He has also made numerous valuable suggestions on the papers I have written and an earlier draft of this thesis; his help is deeply appreciated.

I am grateful to Minoru Ito and Yuji Matsumoto for accepting to be the referees of this thesis. I am especially thankful to Yuichi Kaji for many fruitful discussions about topics including not only computer science but other fields such as literature, fine art, and food cultures.

I would like to thank all my colleagues and friends at NAIST, who made my life in Nara fruitful and enjoyable. Specifically, I would like to express my gratitude to Fumiyo Tagawa and her family; she was a staff at the Applied Algorithmics Laboratory at NAIST, in which I began the study of query preservation.

Last but not least, I would like to thank my family for their constant support during my undergraduate and graduate studies.

Nara, March 2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

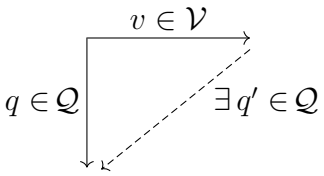
The cherry-trees are in flower but it is chilly in the garden. There is an early frost.
— *The Cherry Orchard, Act I* [11]

1.1 Research Motivation

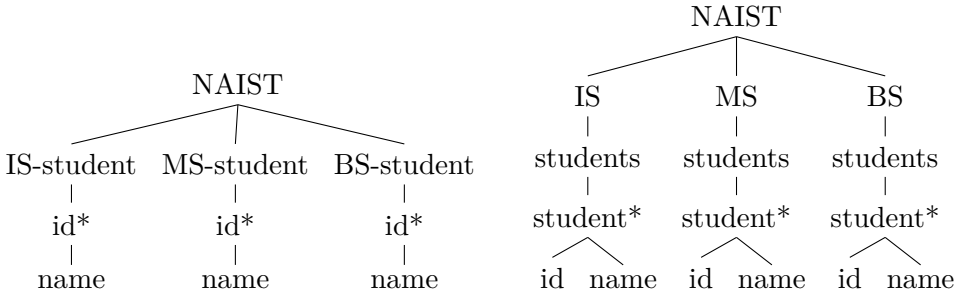
Almost every company or all kinds of organizations retrieve, handle, and store some information or data such as employee information, customer information, patent information, etc. These data are usually stored in databases, and maintained by using a database management system. Due to data exchanges and data updates, long-term databases are often required to transform the structure of its contents. When transforming the structure a fundamental problem comes up: *what kind of information is needed to be preserved?* The solution of the problem depends on what information is needed to be recovered from the transformed data. There are some formulation of such information preservation such as invertibility and *query preservation*. Intuitively, invertibility means that one can recover the source data from the transformed data, and query preservation means that information retrieved from source data by a (designated) query can be also retrieved from the transformed data, by some query that is determined by the transformation and the source query.

In database theory, some solutions for query preservation problem have been presented, where the problem is motivated by data integration and query optimization (see, e.g., surveys [31, 32]). Let $\mathcal{Q}$ be a class of queries and $\mathcal{V}$ be a class of transformations (or views). Query preservation for $\mathcal{Q}$ under $\mathcal{V}$ is the problem deciding whether, given a query $q \in \mathcal{Q}$ and a view $v \in \mathcal{V}$, a mapping $q' \in \mathcal{Q}$

can be constructed from q and v such that $q = v \circ q'$, where $v \circ q'$ denotes the composition of v and q' defined as $(v \circ q')(t) = q'(v(t))$ for every input t. If such a mapping q' exists, we say that v preserves q or, q can be rewritten in terms of v.



Various techniques of query preservation have been developed in relational database theory. On the other hand, query preservation for semi-structured data, especially, tree-structured data such as XML documents has received attention recently due to the enormous success of the model on the Web (see, e.g., [2, 4, 7, 23, 29, 33, 36, 40]).



(a) Source schema S_s (b) Target schema S_t

Figure 1.1: Example: source schema and target schema.

Example 1. (Schema update) Let $\mathcal{Q}$ be a class of queries and $\mathcal{V}$ be a class of transformations. Consider a tree-structured database D that stores information about students at NAIST such as their IDs and names. The schema S_s of the database is depicted in Figure 1.1a, which has the root labeled "NAIST" with (ordered) children labeled "{IS,MS,BS}-student" followed by some children labeled "id", which followed by a child labeled "name" (where "IS", "MS", and "BS" stand for (the department of) Information Science, Materials Science, and Biological Sciences, respectively, and "X*" means that there are zero or more nodes labeled 'X'). We also consider some applications that use a query $q \in \mathcal{Q}$ over instances on S_s. One day, a database administrator of D decided to change the schema of D from S_s to S_t depicted in Figure 1.1b. Let $v \in \mathcal{V}$ be a schema transformation from S_s to S_t. Query preservation is the problem deciding whether $q' \in \mathcal{Q}$ can be constructed from q and v such that $q = v \circ q'$. If such q' exists, we

can say that the information used by the applications is preserved through the transformation. ◊

The case when both views and queries are single-valued (or deterministic) tree transducers, which are a formal model of computer programs that transform each input tree-structured data to the output one, was studied in [4,33]. Contributions of [4,33] will be reviewed in Section 1.2.1 in some detail. These two studies opened a new research direction of query preservation for tree-structured data. Starting from [4, 33], this thesis aims at extending the study on query preservation in two ways, namely, *nondeterminism* and *hybrid approach*.

Example 2. (Transformation by tree transducer) The transformation v of Example 1 can be realized by a linear deterministic extended top-down tree transducer (see Section 2.2.2 for a formal definition). We assume for the sake of simplicity that there is an upper limit m on the number of information about students. The transducer is a tuple $E = (Q, \Sigma, \Delta, I, R)$ where $Q = \{q_r, q_{\mathrm{id}}\}$, $\Sigma = \{$NAIST,IS-student,MS-student,BS-student,id,name$\}$, $\Delta = \{$NAIST,IS,MS,BS,students,student,id,name$\}$, $I = \{q_r\}$, and R contains the following two rules:

$$
\begin{aligned}
r_1 :\quad & q_r(\mathrm{NAIST}(\mathrm{IS\text{-}student}(x_1, \ldots, x_m),\\
& \qquad \mathrm{MS\text{-}student}(x_{m+1}, \ldots, x_{2m}),\\
& \qquad \mathrm{BS\text{-}student}(x_{2m+1}, \ldots, x_{3m})))\\
& \qquad \to \mathrm{NAIST}(\mathrm{IS}(\mathrm{students}(q_{\mathrm{id}}(x_1), \ldots, q_{\mathrm{id}}(x_m))),\\
& \qquad\qquad \mathrm{MS}(\mathrm{students}(q_{\mathrm{id}}(x_{m+1}), \ldots, q_{\mathrm{id}}(x_{2m}))),\\
& \qquad\qquad \mathrm{BS}(\mathrm{students}(q_{\mathrm{id}}(x_{2m+1}), \ldots, q_{\mathrm{id}}(x_{3m})))),\\
r_2 :\quad & q_{\mathrm{id}}(\mathrm{id}(\mathrm{name})) \to \mathrm{student}(\mathrm{id},\mathrm{name}).
\end{aligned}
$$

These rules are depicted in Figure 1.2. The rule r_2 in R performs a *local rotation*; such rule is not expressible by ordinary top-down tree transducers that are not *extended*. ◊

Nondeterminism. Single-valued tree transducers output just one tree for each input tree (see [4,33,40]). However, to our knowledge, no previous work has considered the query preservation for nondeterministic tree transducers (that are not always single-valued) as views. Nondeterminism of tree transducers is required in some applications, e.g., probabilistic database (see a survey [14]) and natural language processing. In machine translation, each sentence in a source natural language can possibly be translated into more than one sentence in a target language (see, e.g., [35, 38, 39]). Thus we need nondeterminism in tree transducers that model syntax-based machine translations (see, e.g., [9, 46]). By the way,

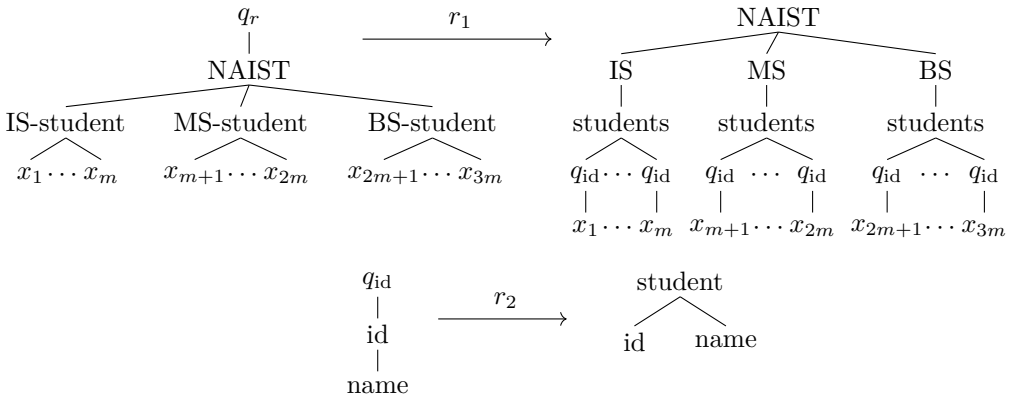


Figure 1.2: Example: rules of an extended top-down tree transducer.

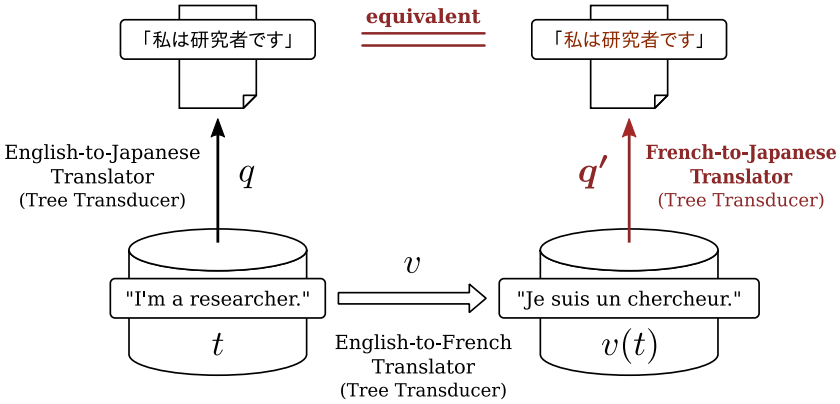


Figure 1.3: An application of query preservation to machine translations.

bilingual documents are essentially required to construct statistical syntax-based translators. The translation accuracy of statistical syntax-based translators depends heavily on quality and quantity of the documents that are used to construct the translators (see, e.g., [8]). However, preparing huge and high-quality bilingual documents requires many efforts and costs in general. Query preservation for nondeterministic tree transducers suggests a solution to this problem, which we call *rewriting-based construction* of machine translators. For instance, let q, v be English-to-Japanese and English-to-French machine translators, respectively, realized by nondeterministic tree transducers. One can construct by the rewriting-based construction French-to-Japanese translator q' from q and v, if q can be rewritten in terms of v (see Figure 1.3). Advantages of the rewriting-based construction of translators are: (1) the translation accuracy of q' is guaranteed to be almost the same as q, because $q(t) = q'(v(t))$ holds for every input t by the definition of query preservation, and (2) any bilingual documents are not re-

quired. In the above example, French-to-Japanese bilingual documents are not needed to construct the French-to-Japanese translator q'.

There are gaps in the expressive power of tree transducers between ones studied in the previous work [4, 33] for the query preservation and ones used to construct statistical syntax-based machine translations. To achieve the rewriting-based construction, the expressive power of tree transducers has to be extended from single-valued (or deterministic) models to nondeterministic ones.

Hybrid approach. By the way, query preservation for a tree query, which answers tree(s) retrieved from an input tree, requires that not only the existence of all nodes but also the tree structure in the query result should be completely preserved. This type of preservation is meaningful when we regard the tree structure itself as necessary information. In contrast, query preservation for *node* queries, which answer a set of nodes retrieved from an input tree, requires that a view should maintain the relationship between the nodes specified by a node query, rather than the tree structures.

For example, assume that a given view extracts some nodes and sorts them depending on the labels of their children, and then removes the children. Also assume that a given node query extracts exactly the same nodes. In this case, the original ordering between the extracted nodes cannot be identified from the transformed tree in general. Since the set of the extracted nodes is preserved, the query preservation holds for the node query. On the other hand, the query preservation does not hold for a tree query that returns an ordered tree in which the target nodes occur according to the original ordering, because the view loses the ordering as stated above.

Example 3. Consider a single-valued view Tr and a single-valued query Q such that their domains are a set of trees $\{\, r(\spadesuit(m), \clubsuit(n)), r(\clubsuit(m), \spadesuit(n)) \mid m, n \in \{1, 2\}\}$. Tr sorts subtrees whose root nodes are labeled by '$\spadesuit$' or '$\clubsuit$' depending on the labels ('1' or '2') of their children in ascending order from the left to the right, and then removes the children. Q just removes the nodes labeled by '1' or '2'. In this case, the original ordering between the extracted nodes cannot be identified from the transformed tree. For instance, let $t_1 = r(\spadesuit(1), \clubsuit(2))$ and $t_2 = r(\clubsuit(2), \spadesuit(1))$, which are depicted in Figure 1.4. Any mapping cannot identify the original ordering from $Tr(t_1) = Tr(t_2)$, because Tr loses the ordering. Hence the query preservation does not hold for the tree query Q. On the other hand, consider a node query $\bar{Q}$ that just extracts a set of nodes labeled by '$\spadesuit$' or '$\clubsuit$'. In the case, the query preservation holds, because the set of the extracted nodes is preserved. $\Diamond$

To our knowledge, no previous work has considered the query preservation for node queries with views realized by tree transducers.

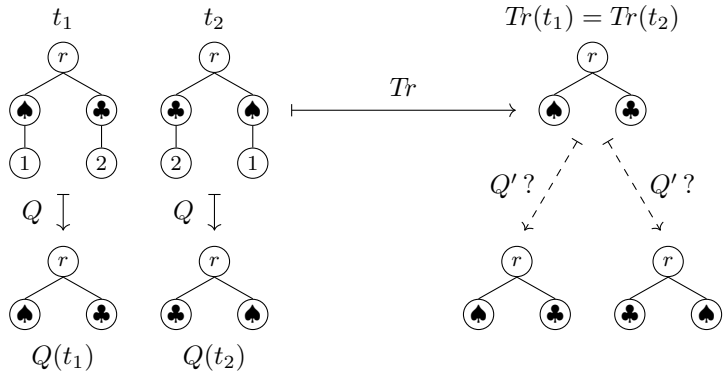


Figure 1.4: The original ordering between the extracted nodes cannot be identified from the transformed tree.

1.2 Research Contribution

This thesis extends the study of query preservation in two ways according to the two observations in the previous section. This section summarizes the two contributions. The details are presented in Chapter 2 and Chapter 3, respectively.

1.2.1 Query Preservation for Nondeterministic Tree Transducers

Our first study in Chapter 2 contributes to the rewriting-based construction by extending the previous work [4, 33] on the query preservation from single-valued models to nondeterministic ones. More specifically, Hashimoto et al. showed in [33] that the query preservation problem is decidable when views are realized by single-valued extended linear bottom-up tree transducers and queries by single-valued bottom-up tree transducers. Benedikt et al. generalized in [4] the results of [33]. They showed that the problem is decidable when views are realized by compositions of single-valued extended linear top-down tree transducers with regular look-ahead and queries by deterministic MSO tree transducers (where MSO stands for Monadic Second-Order logic). Note that the problem is undecidable even if the views can copy only once at each root of input trees [4]. Thus [4, 33] and we treat views that cannot copy*.

In Chapter 2, we first define two variants of information preservation, which are natural extensions of query preservation for nondeterministic views: universal preservation and, its relaxed version, existential preservation.

*Usually the non-copying property is called *linearity*.

Let $\mathcal{V}$, $\mathcal{Q}$ be classes of queries and views, respectively. Given a view $v \in \mathcal{V}$ and a query $q \in \mathcal{Q}$, v universally preserves q if there is a query $q' \in \mathcal{Q}$ such that for every input t to q and for every output $t' \in v(t)$, $q(t) = q'(t')$ holds; v existentially preserves q if there is a query q' such that for every input t to q there is an output $t' \in v(t)$ satisfying $q(t) = q'(t')$. An example of the universal and the existential preservation is depicted in Figure 1.5. Obviously, if v universally preserves q,

(a) Universal preservation (b) Existential preservation

Figure 1.5: v universally or existentially preserves q, where v is 2-valued.

then v existentially preserves q. Intuitively, if v universally preserves q then the result of q can be computed from any output of v. Whereas if v existentially preserves q then there exists at least one output of v from which the result of q can be computed. Existential preservation is useful in the case that the result of query is more important than that of view. For example, in the case of Figure 1.3, v does not need to universally preserve q to construct q'. The French-to-Japanese translator q' can be constructed if at least one result of v preserves the result of q, that is, if v existentially preserves q then q' can be constructed.

To obtain the decidability of universal preservation for nondeterministic views, we first extend slightly the results in [20] on the equivalence problem for deterministic MSO tree transducers (see Theorem 13). Namely, we show that for a deterministic MSO tree transducer q_1 and a *nondeterministic* one q_2, the equivalence of q_1 and q_2 is decidable. Note that the equivalence is undecidable if q_1 and q_2 are both nondeterministic ones. Then, by adopting the proof strategy introduced in [4] that uses "uniformizers" and reduces the query preservation problem to the equivalence problem for deterministic MSO tree transducers, we obtained the desired result (Theorem 15), that is, we show that the problem is decidable when views are realized by compositions of nondeterministic extended linear top-down tree transducers with regular look-ahead and queries by deterministic MSO tree transducers. Summary of decidability results on query preservation is shown in Table 1.1, where B, DT, DMSOT, ELB, ELT stand for the classes of all transductions realized by bottom-up, deterministic top-down, deterministic MSO, extended linear bottom-up, and extended linear top-down tree transducers, respectively. "s-" is an abbreviation of single-valued, and the transducers in the classes with superscript R have regular look-ahead. The composition closure

Table 1.1: Decidability results on query preservation. Our result is indicated in bold. Incomparability is denoted by $\bowtie$.

	single-valued			nondeterministic
Query \ View	s-ELB	$\subsetneq$	(s-ELT$^{\mathrm{R}}$)* $\subsetneq$	(ELT$^{\mathrm{R}}$)*
s-B	decidable [33]			
$\subsetneq$ DT$^{\mathrm{R}}$		decidable [4]		
$\bowtie$ DMSOT		decidable [4]		**decidable** (Thm.15)

Table 1.2: Summary of our decidability results, where $\forall$ and $\exists$ stand for universal and existential preservation, respectively, "part" stands for the preservation for nondeterministic queries (see Section 2.4), and "sound" means that we give a sound algorithm of the problem for the classes in the line.

Query	View	$\forall$ or $\exists$	Result
DMSOT	(ELT$^{\mathrm{R}}$)*	$\forall$	decidable (Thm.15)
DMSOT / DT$^{\mathrm{R}}$	finite-valued LB	$\exists$	sound (Thm.18)
finite-valued LB	(ELT$^{\mathrm{R}}$)*	$\forall$ part	sound (Thm.23)
finite-valued LB	finite-valued LB	$\exists$ part	sound (Cor.24)

of a class X is denoted by X^*. See Section 2.2 for the definitions of them. As depicted in Table 1.1 there are class hierarchies of tree transductions such that s-ELB $\subsetneq$ (s-ELT$^{\mathrm{R}}$)* $\subsetneq$ (ELT$^{\mathrm{R}}$)* and s-B $\subsetneq$ DT$^{\mathrm{R}}$. DMSOT is incomparable with DT$^{\mathrm{R}}$.

Furthermore, we give an algorithm that is sound for existential preservation (see Theorem 18). Also, we give other algorithms that are sound for the related situations in which queries are expressed by nondeterministic tree transducers, specifically *finite*-valued bottom-up tree transducers (see Theorem 23 and Corollary 24). To show the results, we use the decomposition theorem for such transducers [48] (see Theorem 17). These results are summarized in Table 1.2. Note that finite-valued LB is a proper subclass of (ELT$^{\mathrm{R}}$)* and incomparable with DMSOT, namely, DMSOT $\bowtie$ finite-valued LB $\subsetneq$ (ELT$^{\mathrm{R}}$)*. The sound algorithms are actually complete if a given view and a given query are single-valued transducers that are mentioned in the previous work.

The methods using the decomposition have a disadvantage that when decomposing a view v into $v_1, \ldots, v_K$, the domain of the resulting view v_i may be a proper subset of the domain of the original view v, which implies that universal or existential preservation cannot be guaranteed. We give a solution that reduces the disadvantage by constructing a union of views appropriately (see Theorem 22).

1.2.2 Hybrid Approach to Query Preservation

In Chapter 3, we examine query preservations of data tree transformations. We treat data trees as a data model, which is a ranked ordered tree where each node can have any nonnegative integer as a data value. We use deterministic linear top-down data tree transducers (abbreviated as DLT$^{\mathrm{V}}$s) and run-based n-ary queries [42] as classes of transformations (or views) and queries, respectively. Transformations induced by DLT$^{\mathrm{V}}$ include simple filterings, relabelings, insertions, and deletions of elements, depending on paths from the root, but do not include duplicate subtree copying. The transformation is determined independently of data values assigned to nodes, though some data values can be transferred from input to output without duplication. Run-based n-ary queries is a powerful class equivalent to MSO n-ary queries [49]. The query class can capture the core of navigation power of XPath [5] in the XML context, and XPath cannot express a run-based n-ary query with n greater than 1. The answer set of a query is the set of tuples of data values that are assigned to nodes selected by the query instead of the selected nodes themselves. In this sense, an n-ary query retrieves some tables consisting of n columns from a tree.

We define two types of query preservation in the above setting: weak and strong query preservation. We say that a DLT$^{\mathrm{V}}$ Tr strongly preserves a query Q if there is a query Q' such that for every tree 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, the answer set of Q' for $Tr(t)$ includes the answer set of Q for t.

Our contributions are as follows. 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 and EXPTIME-hard. If the tuple size n of queries is constant, the complexity becomes EXPTIME-complete. The decidability results of the two cases can be extended to the situation where the transformation is given by a single-valued extended linear top-down data tree transducer with regular look-ahead (abbreviated as s-ELT$^{\mathrm{VR}}$), which has more expressive power than DLT$^{\mathrm{V}}$.

1.3 Related Work

In this section, we review related work other than [4, 33] on query preservation. In [23,41], it was shown that the query preservation is undecidable for the class of views and queries that are able to simulate first-order logic (FO) queries and projection queries, respectively, and even for views and queries expressed as unions of conjunctive queries (that are much weaker than FO queries), in the relational case.

The problem was also shown to be undecidable in [23] for XSLT (or XQuery) as views and simple selection queries, in the XML context. In [29], views are defined as transformations that retrieve nodes selected by queries, such as Regular XPath and MSO queries in the context of unranked trees. Similarly, in [36] both queries and views are n-ary node queries represented by tree automata. The result in [24] is for deterministic tree transducers that require "origin" information.

Chapter 2

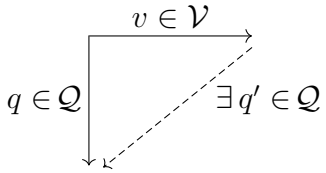
Query Preservation for Nondeterministic Tree Transducers

2.1 Introduction

On one side rise dark poplars,
behind them begins the cherry orchard.

— *Act II* [11]

Let $\mathcal{Q}$ be a class of queries and $\mathcal{V}$ be a class of views. *Query preservation* for $\mathcal{Q}$ under $\mathcal{V}$ is the problem deciding whether, given a query $q \in \mathcal{Q}$ and a view $v \in \mathcal{V}$, a mapping $q' \in \mathcal{Q}$ can be constructed from q and v such that $q = v \circ q'$. If such a mapping q' exists, we say that v preserves q.



As mentioned in Chapter 1, we generalize the results of [33] and [4] to that for nondeterministic views that are not always single-valued. We first define two variants of information preservation, which are natural extensions of query preservation for nondeterministic views: universal preservation and, its relaxed version, existential preservation.

Let $\mathcal{V}$, $\mathcal{Q}$ be classes of queries and views, respectively. Given a view $v \in \mathcal{V}$ and a query $q \in \mathcal{Q}$, v universally preserves q if there is a query $q' \in \mathcal{Q}$ such that for

every input t to q and for every output $t' \in v(t)$, $q(t) = q'(t')$ holds; v existentially preserves q if there is a query q' such that for every input t to q there is an output $t' \in v(t)$ satisfying $q(t) = q'(t')$. Obviously, if v universally preserves q, then v existentially preserves q. Intuitively, if v universally preserves q then the result of q can be computed from any output of v. Whereas if v existentially preserves q then there exists at least one output of v from which the result of q can be computed.

Main results in this chapter are as follows (see Section 1.2.1 for details). We prove that the universal preservation is decidable for compositions of extended linear top-down tree transducers with regular look-ahead as views and deterministic MSO tree transducers as queries (see Theorem 15). To obtain the result we slightly generalize the result [20] of the equivalence problem for deterministic MSO tree transducers (see Theorem 13). Moreover, we show an algorithm that is sound for the existential preservation for finite-valued linear bottom-up tree transducers as views and deterministic MSO tree transducers as queries (see Theorem 18), and also show some algorithms that are sound for the problem for nondeterministic queries realized by finite-valued (linear) bottom-up tree transducers (see Theorem 23 and Corollary 24).

Organization

This chapter is organized as follows: Section 2.2 provides terminology and definitions of graphs, trees, tree automata, various tree transducers with their class hierarchies, and uniformizers. In Section 2.3, we show additional class hierarchies of tree transducers, which are needed to prove our main results. Section 2.4 defines universal and existential preservation for nondeterministic views. Furthermore, for nondeterministic queries two variants of query preservation are provided, which are analogue of existential preservation. Our main result on the decidability of universal preservation is presented in Section 2.5. The result is proved using a decidability result on the equivalence for deterministic and nondeterministic MSO tree transducers, which is included in the same section. A sound algorithm for existential preservation with single-valued queries is presented in Section 2.6, and others for nondeterministic queries are provided in Section 2.7. These sound algorithms employ a method that decomposes finite-valued tree transducers into a finite union of single-valued ones, and then the algorithms apply decision procedures of query preservation for single-valued tree transducers presented in the previous work [4, 33] to decomposed transducers. Section 2.6 also includes a solution for the problem of how to extend the domain of a view, which may be useful if a view does not preserve a query because of the difference between the domains of the view and the query. Section 2.8 concludes

the chapter and outlines future work.

2.2 Preliminaries

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 Σ of symbols with a mapping $\mathrm{rk} : \Sigma \to \mathbb{N}$, and let $\Sigma^{(n)}$ be the set $\{\, \sigma \in \Sigma \mid \mathrm{rk}(\sigma) = n \,\}$. For every $k \geq 1$, let $\mathcal{X}_k = \{\, x_i \mid i \in [k] \,\}$ be the set of variables with $\mathrm{rk}(x_i) = 0$ for every $x_i \in \mathcal{X}_k$. We write $\mathcal{X} = \bigcup_{k \geq 1} \mathcal{X}_k$ and generally assume that $\Sigma \cap \mathcal{X} = \emptyset$. For a binary relation $R \subseteq A \times B$, let $\mathrm{dom}(R) = \{\, a \in A \mid (a, b) \in R \,\}$, $\mathrm{ran}(R) = \{\, b \in B \mid (a, b) \in R \,\}$, $R^{-1} = \{\, (b, a) \in B \times A \mid (a, b) \in R \,\}$, and $R|_{A'} = \{\, (a, b) \in R \mid a \in A' \,\}$ for $A' \subseteq A$. The composition of relations $R_1 : A \to B$ and $R_2 : B \to C$, denoted by $R_1 \circ R_2$, is the relation $A \to C$ defined by $R_1 \circ R_2 = \{\, (a, c) \mid (a, b) \in R_1 \text{ and } (b, c) \in R_2 \text{ for some } b \in B \,\}$. For classes of binary relations $\mathcal{R}$, $\mathcal{S}$, we write $\mathcal{R} \circ \mathcal{S} = \{\, R \circ S \mid R \in \mathcal{R},\, S \in \mathcal{S} \,\}$, $\mathcal{R}^* = \{\, R_1 \circ \cdots \circ R_n \mid n \geq 0,\, R_i \in \mathcal{R}\ (1 \leq i \leq n) \,\}$, and $\mathcal{R}^{-1} = \{\, R^{-1} \mid R \in \mathcal{R} \,\}$.

2.2.1 Graphs, Trees, Strings and MSO Graph Transducers

We basically follow the definitions of [20].

A graph alphabet is a pair (Σ, Γ) where Σ and Γ are ranked alphabets of node labels and edge labels, respectively. A *graph over* (Σ, Γ) is a tuple (V, E, lab), with V a finite set of nodes, $E \subseteq V \times \Gamma \times V$ the set of labeled edges, and $\mathrm{lab} : V \to \Sigma$ the node-labeling function. For a graph g, we write V_g, E_g, and lab_g to denote the set of nodes, the set of edges, and the node-labeling function of g, respectively. The set of graphs over (Σ, Γ) is denoted by $G(\Sigma, \Gamma)$.

We define MSO formula and MSO graph transducers concretely below. The reader may skip the concrete definitions because they are only used in the proof sketches of some lemmas cited from [20].

For alphabets Σ and Γ, the language $\mathrm{MSO}(\Sigma, \Gamma)$ of MSO formulas over (Σ, Γ) uses node variables (written with lower-case letters x, y, ...) and node-set variables (written with upper-case letters X, Y, ...). Atomic formulas in $\mathrm{MSO}(\Sigma, \Gamma)$ are equalities $x = y$; membership constraints $x \in X$; labels $\mathrm{lab}_\sigma(x)$ for $\sigma \in \Sigma$, denoting that x has label σ; and edge labels $\mathrm{edg}_\gamma(x, y)$ for every $\gamma \in \Gamma$, denoting that there is an edge labeled γ from x to y. Formulas are built from the atomic formulas using the logical connectives $\wedge$, $\vee$, $\neg$, $\Rightarrow$ and the quantifiers $\exists x$, $\forall x$ (quantification on node variables), and $\exists X$, $\forall X$ (quantification on node-set variables). For a closed formula $\psi \in \mathrm{MSO}(\Sigma, \Gamma)$ and a graph $g \in G(\Sigma, \Gamma)$, we write $g \models \psi$ if g satisfies ψ. Let u, v be nodes of graph g and assume ψ has a free

variable x or variables x, y, we write $(g, u) \models \psi$ or $(g, u, v) \models \psi$ if g satisfies ψ with $x = u$ or with $x = u$, $y = v$, respectively.

Let (Σ_1, Γ_1) and (Σ_2, Γ_2) be graph alphabets. A *deterministic MSO graph transducer* from (Σ_1, Γ_1) and (Σ_2, Γ_2) is a tuple $M = (C, \varphi_{\mathrm{dom}}, \Psi, X)$ where C is a finite set of *copy names*, $\varphi_{\mathrm{dom}} \in \mathrm{MSO}(\Sigma_1, \Gamma_1)$ is the closed *domain formula*, $\Psi = \{\psi_{c,\sigma}(x)\}_{c \in C, \sigma \in \Sigma_2}$ is a family of *node formulas*, and $X = \{\chi_{c,c',\gamma}(x, y)\}_{c,c' \in C, \gamma \in \Gamma_2}$ is a family of *edge formulas*. For a deterministic MSO graph transducers M, the *graph transduction* $[\![M]\!] : G(\Sigma_1, \Gamma_1) \to G(\Sigma_2, \Gamma_2)$ *realized by* M is defined as follows: For every graph $g \models \varphi_{\mathrm{dom}}$ over (Σ_1, Γ_1), $h = [\![M]\!](g)$ is the graph over (Σ_2, Γ_2) with $V_h = \{\, (c, u) \mid c \in C,\ u \in V_g,$ and there is exactly one $\sigma \in \Sigma_2$ s.t. $(g, u) \models \psi_{c,\sigma}(x) \,\}$, $E_h = \{\, ((c, u), \gamma, (c', u')) \mid (c, u), (c', u') \in V_h,\ \gamma \in \Gamma_2,$ and $(g, u, u') \models \chi_{c,c',\gamma}(x, y) \,\}$, and $\mathrm{lab}_h = \{\, ((c, u), \sigma) \mid (c, u) \in V_h,\ \sigma \in \Sigma_2,$ and $(g, u) \models \psi_{c,\sigma}(x) \,\}$. Instead of $[\![M]\!](g)$ we write $M(g)$ by identifying a transducer M with its transduction $[\![M]\!]$ (and similarly for other transducers).

A (nondeterministic) MSO graph transducer M' is obtained from a deterministic one by allowing all formulas to have fixed free node-set variables called parameters. The transducer binds each parameter to a set of nodes of the input graph g satisfying the domain formula, then for each set of nodes, the node formulas and the edge formulas define the output graph as the deterministic one does. Thus, the graph transduction realized by M' is (not always a function but) a relation $[\![M']\!] \subseteq G(\Sigma_1, \Gamma_1) \times G(\Sigma_2, \Gamma_2)$.

For an alphabet Δ and $a_1, \ldots, a_n \in \Delta$ $(n \geq 0)$, we identify the string $w = a_1, \ldots, a_n$ over Δ with the graph in $G(\{\#\}, \Delta)$ that has #-labeled nodes $v_1, \ldots, v_{n+1}$, and an a_i-labeled edge from v_i to v_{i+1} for $1 \leq i \leq n$.

Let Σ be a ranked alphabet and m be the maximal rank of symbols in Σ. A *tree* t (*over* Σ) is an acyclic, connected, and directed graph in $G(\Sigma, [m])$. Every tree t has the node called the *root* that has no incoming edges. A node of t except the root is called a *leaf* if the node has no outgoing edges. Each node of t is labeled with a symbol $\sigma \in \Sigma$ and has just $\mathrm{rk}(\sigma)$-outgoing edges, labeled 1, 2, …, $\mathrm{rk}(\sigma)$, respectively. The set of all trees over Σ is denoted by $\mathcal{T}_\Sigma$. A tree whose root is labeled with $\sigma \in \Sigma^{(k)}$ and has subtrees $t_1, \ldots, t_k$ from left to right is denoted by $\sigma(t_1, \ldots, t_k)$. For $\sigma \in \Sigma^{(0)}$ we write $\sigma()$ as σ for simplicity. For Σ and $\mathcal{X}' \subseteq \mathcal{X}$, we write $\mathcal{T}_\Sigma(\mathcal{X}')$ to mean $T_{\Sigma \cup \mathcal{X}'}$. A tree $t \in \mathcal{T}_\Sigma(\mathcal{X}')$ is *linear* if every variable in $\mathcal{X}'$ occurs at most once in t. For sets of trees $T_1, \ldots, T_k$, a tree t and its subtrees $t_1, \ldots, t_k$ that are not subtrees of others, we denote by $t[t_i \leftarrow T_i \mid i \in [k]]$ the set of trees obtained from t by replacing each occurrence of a subtree t_i of t by a tree in T_i. Notice that different occurrences of t_i need not to be replaced by the same tree. The set of *contexts* $\mathcal{C}(\Sigma, \mathcal{X}')$ (over Σ and $\mathcal{X}'$) is the set of all linear trees in $\mathcal{T}_\Sigma(\mathcal{X}')$. For $t \in \mathcal{C}(\Sigma, \mathcal{X}_k)$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$, $t[t_1, \ldots, t_k]$ is the tree obtained from t by replacing the variable $x_i \in \mathcal{X}_k$ in t with t_i for each $i \in [k]$.

An MSO graph transducer M is called an MSO graph-to-tree transducer if the range of M is a set of trees. For sets of graphs A and B, M is called an MSO A-to-B transducer if $\mathrm{dom}(M) \subseteq A$ and $\mathrm{ran}(M) \subseteq B$. In the case of $A = B$, M is called an MSO A transducer, e.g., an MSO tree transducer (abbreviated by MSOT transducer, and by DMSOT for deterministic one). The class of all transductions realized by MSOT transducers is denoted by MSOT, and similarly for other transducers.

The closure properties of MSO transducers under composition are shown, e.g., in Proposition 3.2(2) of [12] and Proposition 2 of [6].

Proposition 1. MSO graph transductions and (deterministic) MSO tree transductions are closed under composition.

2.2.2 Tree Automata and Tree Transducers

We define tree automata and various tree transducers concretely below. The reader may skip the concrete definitions of them, because the definitions are used few times in the rest of this chapter, and are less important than class hierarchies of tree transducers.

A *deterministic bottom-up tree automaton* (TA) is a tuple $A = (P, \Sigma, F, \delta)$ where P is a finite set of states, Σ is a ranked alphabet, $F \subseteq P$ is a set of final states, and $\delta = \{ \delta_\sigma \mid \sigma \in \Sigma \}$ is a finite set of transition rules with $\delta_\sigma : P^k \to P$ for every $\sigma \in \Sigma^{(k)}$. We generalize δ to a mapping $\bar{\delta} : \mathcal{T}_\Sigma \to P$ such that $\bar{\delta}(\sigma(t_1, \ldots, t_k)) = \delta_\sigma(\bar{\delta}(t_1), \ldots, \bar{\delta}(t_k))$, where $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$. For all $p \in P$, we denote by $L(A)_p$ the set $\{ t \in \mathcal{T}_\Sigma \mid \bar{\delta}(t) = p \}$. The tree language accepted by A is $L(A) = \bigcup_{p \in F} L(A)_p$. A tree language L is *regular* if there exists a TA A such that $L = L(A)$.

An *extended bottom-up tree transducer* (EB transducer) is a tuple $E = (P, \Sigma, \Delta, F, R)$ where P is a finite set of states, Σ and Δ are ranked alphabets of input and output symbols, respectively, $F \subseteq P$ is a set of final states, and R is a finite set of transduction rules of the form $C[p_1(x_1), \ldots, p_k(x_k)] \to p(r)$, where $C \neq x_i$ (for any $i \in [k]$) is a context in $\mathcal{C}(\Sigma, \mathcal{X}_k)$, $p_1, \ldots, p_k, p \in P$, $x_1, \ldots, x_k \in \mathcal{X}_k$, $k \in \mathbb{N}$, and $r \in \mathcal{T}_\Delta(\mathcal{X}_k)$. For all $p \in P$ and $t \in \mathcal{T}_\Sigma$, we denote by $[\![E]\!]_p(t)$ the smallest set of trees $T \subseteq \mathcal{T}_\Delta$ such that for every rule $C[p_1(x_1), \ldots, p_k(x_k)] \to p(r)$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$, if $t = C[t_1, \ldots, t_k]$ then T includes $r[x_i \leftarrow [\![E]\!]_{p_i}(t_i) \mid i \in [k]]$. We also denote by $[\![E]\!]$ the transduction realized by E that is the binary relation $\{ (t, t') \in \mathcal{T}_\Sigma \times \mathcal{T}_\Delta \mid t' \in \bigcup_{p \in F} [\![E]\!]_p(t) \}$. An EB transducer is *linear* if for every rule ρ, each variable in the left-hand side of ρ occurs at most once in the right-hand side of ρ. We write ELB transducers to denote EB ones that are linear. For example, an EB transducer that has a rule $a(b(p_1(x_1)), p_2(x_2)) \to p(c(x_1, x_1))$ is *not* linear. A *bottom-up tree transducer* (B transducer) is an EB transducer whose

transduction rules are of the form $\sigma(p_1(x_1), \ldots, p_k(x_k)) \to p(r)$ where $\sigma \in \Sigma^{(k)}$. We write LB transducers to denote linear B ones. Notice that some definitions of EB transducers allow a rule that has the left-hand side of the form $p_i(x_i)$, which is called an input-ε rule (see, e.g., [17] and [33] in which EB is denoted by XBOT$^{-\mathrm{e}}$).

An *extended top-down tree transducer with regular look-ahead* (ET$^{\mathrm{R}}$ transducer) is a tuple $E = (Q, \Sigma, \Delta, I, R, A)$ where Σ and Δ are the same as an EB transducer, Q is a finite set of states, $I \subseteq Q$ is a set of initial states, A is a tree automaton (P, Σ, F, δ), and R is a finite set of transduction rules. For $T \subseteq \mathcal{T}_\Delta(\mathcal{X})$, we denote by $Q(T)$ the set $\{\, q(t) \mid q \in Q, t \in T \,\}$. Every transduction rule of R is of the form $q(C) \to r\ \langle p_1, \ldots, p_k \rangle$, where $C \neq x$ is a context in $\mathcal{C}(\Sigma, \mathcal{X})$, $q \in Q$, $p_1, \ldots, p_k \in P$, and $r \in \mathcal{T}_\Delta(Q(\mathcal{X}))$. For example, $q(a(b(x_1), x_2)) \to c(q_1(x_2), d(d(q_2(x_2))))\ \langle p_1, p_2 \rangle$ is a rule of an ET$^{\mathrm{R}}$ transducer. For all $q \in Q$ and $t \in \mathcal{T}_\Sigma$, we denote by $[\![E]\!]_q(t)$ the smallest set of trees $T \subseteq \mathcal{T}_\Delta$ such that for every rule $q(C) \to r\ \langle p_1, \ldots, p_k \rangle$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$, if $t = C[t_1, \ldots, t_k]$ and $\bar{\delta}(t_i) = p_i$ for every $i \in [k]$, then T includes $r[q'(x_i) \leftarrow [\![E]\!]_{q'}(t_i) \mid i \in [k], q' \in Q]$. We also denote by $[\![E]\!]$ the transduction realized by E that is the binary relation $\{\, (t, t') \in \mathcal{T}_\Sigma \times \mathcal{T}_\Delta \mid t \in L(A), t' \in \bigcup_{q \in I} [\![E]\!]_q(t) \,\}$. An ET$^{\mathrm{R}}$ transducer is *linear* if for every rule ρ, each variable in the left-hand side of ρ occurs at most once in the right-hand side of ρ. We write ELT$^{\mathrm{R}}$ transducers to denote linear ET$^{\mathrm{R}}$ ones. A *top-down tree transducer with regular look-ahead* (T$^{\mathrm{R}}$ transducer) is an ET$^{\mathrm{R}}$ transducer whose transduction rules are of the form $q(\sigma(x_1, \ldots, x_k)) \to r\ \langle p_1, \ldots, p_k \rangle$ where $\sigma \in \Sigma^{(k)}$. A T$^{\mathrm{R}}$ transducer E has no regular look-ahead (denoted by a T transducer) if tree automaton A of E is a trivial one-state tree automaton such that $L(A) = \mathcal{T}_\Sigma$. T$^{\mathrm{R}}$ transducers are *deterministic* (denoted by DT$^{\mathrm{R}}$ transducers) if for each state $q \in Q$ and symbol $\sigma \in \Sigma$ there exists at most one transduction rule that contains both q and σ in its left-hand side. As with the EB transducers, some definitions of ET$^{\mathrm{R}}$ transducers allow input-ε rules that have the left-hand side of the form $q_i(x_i)$ (see, e.g., [3, 27, 37, 39], in which ET$^{\mathrm{R}}$ is denoted by XTOP$^{\mathrm{R}}_{\mathrm{ef}}$ or e-XTOP$^{\mathrm{R}}$).

For a tree transducer M and an input tree t, $\mathrm{val}_M(t) = |M(t)|$ denotes the number of different outputs of M for t, and let $\mathrm{val}(M) = \sup\{\, \mathrm{val}_M(t) \mid t \in \mathcal{T}_\Sigma \,\}$, which is called *valuedness* of M. M is *finite-valued* if $\mathrm{val}(M) < \infty$. For any tree transducer M, we say that M is *single-valued* (or *functional*) if $\mathrm{val}(M) \leq 1$, i.e., for each input t, there exists at most one output $M(t)$. We use the prefix 's' to denote the single-valued tree transducers of the class under consideration, e.g., a single-valued EB transducer is denoted by an s-EB transducer. We also say that M is *finite-copying* (denoted with the subscript 'fc') if each subtree of the input tree is transduced by a bounded number of times independent of the input (see [1, 21] for more formal definition of finite-copying). We denote by $[\![M]\!]$ the transduction realized by M, which is a binary relation on trees. For transducers M_1 and M_2, we say that M_1 and M_2 are *equivalent* if $[\![M_1]\!] = [\![M_2]\!]$.

We will use the abbreviation such as EB, B for a class of tree transducers to denote the class of transductions realized by that class of tree transducers. By the definitions above the relations $\text{B} \subseteq \text{EB}$, $\text{LB} \subseteq \text{ELB}$, and $\text{T} \subseteq \text{T}^{\text{R}} \subseteq \text{ET}^{\text{R}}$ hold. Additionally, as noted in Section 2.1, there are well-known class hierarchies of tree transducers such that $\text{s-ELB} \subsetneq (\text{s-ELT}^{\text{R}})^* \subsetneq (\text{ELT}^{\text{R}})^*$, $\text{s-B} \subsetneq \text{DT}^{\text{R}} \bowtie \text{DMSOT}$, and $\text{DMSOT} \bowtie \text{finite-valued LB} \subsetneq (\text{ELT}^{\text{R}})^*$, where incomparability is denoted by $\bowtie$.

2.2.3 Uniformizers

We define uniformizers following [4]: Let R be a binary relation. A function u is a uniformizer of R if $u \subseteq R$ and $\text{dom}(u) = \text{dom}(R)$. For classes $\mathcal{T}$, $\mathcal{U}$ of transductions, we say that $\mathcal{T}$ *has uniformizers in* $\mathcal{U}$ if for every $\tau \in \mathcal{T}$ we can construct a uniformizer u of τ such that $u \in \mathcal{U}$. Benedikt et al. [4] showed that it is decidable whether v preserves q, by reducing the problem to the equivalence of the query q and the composed mapping $v \circ u \circ q$, where u is a uniformizer of v^{-1}.

The following result is used later to prove our main result.

Theorem 2. [Theorem 11 of [4]] $((\text{ELT}^{\text{R}})^*)^{-1}$ has uniformizers in $\text{DT}^{\text{R}}_{\text{fc}}$.

2.3 Class Hierarchies of Tree Transducers

In the section, we show some class hierarchies of tree transducers that are needed in Section 2.5 and Section 2.7.

Theorem 3.

$$\begin{aligned}(\text{ELT}^{\text{R}})^* \circ \text{DT}^{\text{R}}_{\text{fc}} \circ \text{DMSOT} &\subseteq (\text{ELT}^{\text{R}})^* \circ \text{DMSOT} \\ &\subseteq (\text{T}^{\text{R}}_{\text{fc}})^* \circ \text{DMSOT} \\ &\subseteq \text{MSOT}.\end{aligned}$$

Proof. $\text{DT}^{\text{R}}_{\text{fc}} \subseteq \text{DMSOT}$ by Theorem 7.4 of [18], and $\text{ELT}^{\text{R}} \subseteq \text{T}^{\text{R}}_{\text{fc}}$ by the construction in the proof of Theorem 4.8 of [39], which states $\text{ET}^{\text{R}} = \text{T}^{\text{R}}$ (see also the paragraph under Corollary 18 of [37]), moreover,

$$\begin{aligned}\text{T}^{\text{R}}_{\text{fc}} &\subseteq \text{DBQREL} \circ \text{T}_{\text{fc}} && (2.1)\\ &\subseteq \text{DBQREL} \circ \text{HOM}_{\text{fc}} \circ \text{LT} && (2.2)\\ &\subseteq \text{MSOT} && (2.3)\end{aligned}$$

where DBQREL is the class of deterministic B transducers that can only relabel, and HOM_{fc} is the class of finite-copying tree homomorphisms. (2.1) follows by

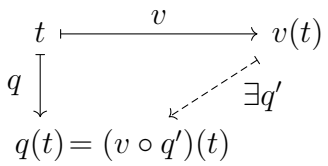
the fact that $\text{T}^{\text{R}} \subseteq \text{DBQREL} \circ \text{T}$ (Theorem 2.6 [16]), and (2.2) follows by the construction in the proof of Lemma 3.6 of [15], which states $\text{T} \subseteq \text{HOM} \circ \text{LT}$. It is not difficult to see that MSOT can simulate DBQREL, HOM_{fc}, and LT, then (3) follows by Proposition 1. □

Proposition 4. S-LB $\subsetneq$ DMSOT.

Proof. $\text{LB} \subseteq \text{ELB}$ by the definition of ELB, $\text{ELB} = \text{ELT}^{\text{R}}$ by Theorem 1 of [4], $\text{S-ELT}^{\text{R}} \subsetneq \text{DT}^{\text{R}}_{\text{fc}}$ by Corollary 13 of [4], and $\text{DT}^{\text{R}}_{\text{fc}} \subseteq \text{DMSOT}$ by Theorem 7.4 of [18]. □

2.4 Query Preservations

Let $\mathcal{V}$ and $\mathcal{Q}$ be a class of single-valued views and a class of single-valued queries for some tree-structured data. For a query $q \in \mathcal{Q}$ and a view $v \in \mathcal{V}$, we say that v *preserves* q if there exists a query $q' \in \mathcal{Q}$ such that $q(t) = (v \circ q')(t)$ for all tree t.



Hashimoto et al. [33] showed that the preservation problem is undecidable even queries and views are identity queries (ID) and tree homomorphisms (HOM), respectively.

Theorem 5. [Theorem 15 of [33]] For $v \in \text{HOM}$ as a view and $q \in \text{ID}$ as a query, it is undecidable whether v preserves q.

Proof. It was shown in [33] that the query preservation is decidable if and only if the injectivity for HOM is decidable. However, the injectivity is undecidable for HOM [25]. Hence the query preservation is also undecidable. □

Benedikt et al. [4] generalized the result for a total copy-once DT transducer. A DT transducer is *copy-once* if for every rule $p(\sigma(x_1, \ldots, x_k)) \to r$ the initial state p_I does not occur in r, and r is linear if $p_I \neq p$.

Theorem 6. [Theorem 5 of [4]] For a total copy-once DT transducer v as a view and $q \in \text{ID}$ as a query, it is undecidable whether v preserves q.

As mentioned in Section 2.1, previous studies on query preservation for tree transducers focus on single-valued (or deterministic) views. In contrast, nondeterministic views output a set of trees for each input tree. We consider two definitions

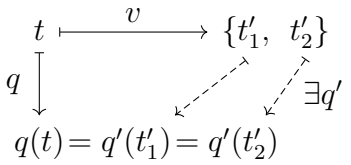
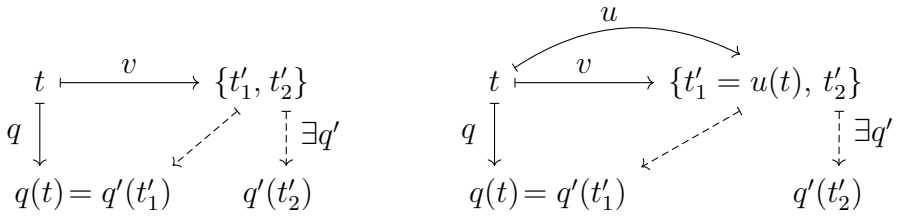


Figure 2.1: Example: v $\forall$-preserves q, where v is 2-valued.



(a) $q(t) = q'(t'_1)$ (b) $q(t) = (u \circ q')(t)$

Figure 2.2: Example: v $\exists$-preserves q, where v is 2-valued.

of query preservation for nondeterministic views that are not always single-valued: Let $\mathcal{V}$ be a class of nondeterministic views and $\mathcal{Q}$ be a class of single-valued queries. Given a view $v \in \mathcal{V}$ and a query $q \in \mathcal{Q}$ such that $\mathrm{dom}(q) \subseteq \mathrm{dom}(v)$, we say that v *universally preserves* q ($\forall$-preserves q for short) if

$$\exists q' \in \mathcal{Q}.\ \forall t \in \mathrm{dom}(q):\ q(t) = (v \circ q')(t) \qquad (\mathrm{ran}(v) \subseteq \mathrm{dom}(q')).$$

The above definition coincides with the definition of the query preservation if v is single-valued. An example of the universal preservation is depicted in Figure 2.1.

We also say that v *existentially preserves* q ($\exists$-preserves q for short) if

$$\exists q' \in \mathcal{Q}.\ \forall t \in \mathrm{dom}(q).\ \exists t' \in v(t):\ q(t) = q'(t').$$

This condition is equivalent to the following one: There exists a uniformizer u of v such that

$$\exists q' \in \mathcal{Q},\ \forall t \in \mathrm{dom}(q):\ q(t) = (u \circ q')(t).$$

By definition, if q is universally preserved by v, then q is also existentially preserved by v. Examples of the existential preservation are depicted in Figure 2.2.

Furthermore, for a class $\mathcal{Q}$ of nondeterministic queries we define additional two variants of query preservation, which are analogue of existential preservation. We say that v *$\forall$-preserves a part of* q if

$$\exists q' \in \mathcal{Q}.\ \forall t \in \mathrm{dom}(q).\ \exists t_q \in q(t):\ t_q = (v \circ q')(t) \qquad (\mathrm{ran}(v) \subseteq \mathrm{dom}(q')).$$

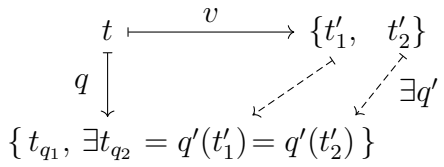
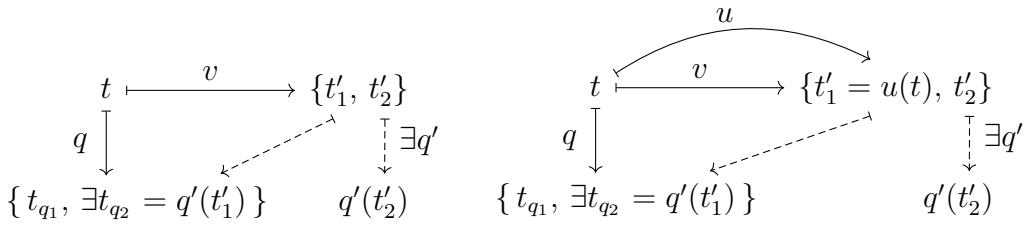


Figure 2.3: Example: v $\forall$-preserves a part of q, where v and q are 2-valued.



(a) $t_{q_2} = q'(t'_1)$ (b) $t_{q_2} = (u \circ q')(t)$

Figure 2.4: Example: v $\exists$-preserves a part of q, where v and q are 2-valued.

Similarly, we say that v *$\exists$-preserves a part of q* if

$$\exists q' \in \mathcal{Q}.\ \forall t \in \mathrm{dom}(q).\ \exists t_q \in q(t).\ \exists t' \in v(t):\ t_q = q'(t').$$

This condition is equivalent to the following one: There exists a uniformizer u of v such that

$$\exists q' \in \mathcal{Q}.\ \forall t \in \mathrm{dom}(q).\ \exists t_q \in q(t):\ t_q = (u \circ q')(t).$$

These two variants of query preservation restrict implicitly the class of q' to a single-valued one. Algorithms that are sound for the variants are presented in Section 2.7. Examples of the variants are depicted in Figures 2.3, 2.4.

2.5 Universal Preservation

The main result of this section is Theorem 15: For $v \in (\text{ELT}^{\text{R}})^*$ as a view and $q \in$ DMSOT as a query, it is decidable whether v $\forall$-preserves q. To obtain the result, we adopt the proof strategy taken by Benedikt et al. [4]. In [4], in order to show the decidability of the query preservation for views realized by (s-ELT$^{\text{R}}$)* transducers and queries by DMSOT transducers, Benedikt et al. reduced this problem to the equivalence problem of two DMSOT transducers, which is known to be decidable [20]. According to the strategy we reduce the problem to the equivalence of MSOT transducers M_1 and M_2, where M_1 is deterministic and M_2 is

nondeterministic. The previous result [20] is for the case that M_1 and M_2 are both deterministic MSOT transducers (see Theorem 12 below). We slightly extend the previous result (see Theorem 13). Note that if M_1 and M_2 are both nondeterministic MSOT transducers, the equivalence for them is known to be undecidable due to the negative result for nondeterministic word transducers [28], which are strictly less expressive than MSOT transducers.

To explain our result, let us summarize the decision procedure of [20] for the equivalence of DMSOT transducers. It can be shown that DMSOT transducers M_1 and M_2 are *not* equivalent if and only if there exist an input t and a position n such that the symbol at position n of $M_1(t)$ is different from the symbol at position n of $M_2(t)$. Hence, roughly speaking, the procedure tests whether a position n and distinct symbols a, b exist such that the pair (n, n) is contained in the set $S^{a,b}$ of all pairs (i, j) where a is the symbol at position i of $M_1(t)$ and b is the symbol at position j of $M_2(t)$, for some input t. In [20] the set $S^{a,b}$ is shown to be *semilinear* (defined below) and then the existence of a pair (n, n) in $S^{a,b}$ is decidable (as stated in Lemma 8). The set $S^{a,b}$ is constructed using *Parikh mapping* (also defined below).

Additional definitions are needed to describe the decision procedure precisely. For a string w, we denote by w/i the i-th letter of w. The Parikh mapping for graphs is the function $\mathrm{Par} : G(\Sigma, \Gamma) \to \mathbb{N}^k$ defined as $\mathrm{Par}(g) = (n_1, \ldots, n_k)$ where g is a graph over (Σ, Γ) with $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ and n_i is the number of σ_i-labeled nodes in g for $i \in [k]$. Similarly, the Parikh mapping for strings over $\Sigma = \{\sigma_1, ..., \sigma_k\}$ is the function $\mathrm{Par} : \Sigma^* \to \mathbb{N}^k$ defined as $\mathrm{Par}(w) = (n_1, \ldots, n_k)$ where n_i is the number of occurrences of σ_i in w for $i \in [k]$. A *discrete graph* (abbreviated as *dgraph*) is a graph that has no edges. Let dgr be a function $\Sigma^* \to G(\Sigma, \emptyset)$ such that $\mathrm{Par}(w) = \mathrm{Par}(\mathrm{dgr}(w))$ for any string $w \in \Sigma^*$. For a set G of graphs, we denote by $\mathrm{Par}(G)$ the set $\{\, \mathrm{Par}(g) \mid g \in G \,\}$. Similarly, for a string language L, let $\mathrm{Par}(L) = \{\, \mathrm{Par}(w) \mid w \in L \,\}$. A set $S \subseteq \mathbb{N}^k$ is *semilinear* if there exists a regular string language R such that $S = \mathrm{Par}(R)$. The set G is *Parikh* if $\mathrm{Par}(G)$ is semilinear. A set of graphs is *VR* if it is generated by a context-free vertex replacement graph grammar (or a C-edNCE or an S-HH grammar, see, e.g., [12, 13]). It should be noted that the set of all trees and the set of all strings are VR. The following two lemmas state useful properties of semilinear sets.

Lemma 7. [Theorem 7.1 of [12], Lemma 3 of [20]] The images of VR sets of graphs under MSO graph-to-dgraph transductions are Parikh.

Lemma 8. [Lemma 4 of [20]] It is decidable for a semilinear set $S \subseteq \mathbb{N}^2$ whether there exists an $n \in \mathbb{N}$ such that $(n, n) \in S$.

Let us refer to the important lemma of [20] (see Lemma 11 below), which is proved using the following two lemmas.

Lemma 9. [Lemma 5 of [20]] Let Δ be an alphabet and $a \in \Delta$. There exists an MSO string-to-dgraph transducer N^a_Δ such that for every $w \in \Delta^*$, $N^a_\Delta(w) = \{\, \mathrm{dgr}(a^n) \mid w/n = a \,\}$.

Proof. (Sketch) Construct an MSO string-to-dgraph transducer that selects nondeterministically a node v with an outgoing a-labeled edge, then copies v and nodes in the (sub)string of w that is from the first letter to a (notice that if a is the n-th letter of w, then the number of the nodes of the (sub)string is just n), and finally labels the nodes a. □

Lemma 10. [Lemma 6 of [20]] Let M_1, M_2 be MSO graph transducers. There exists an MSO graph transducer M such that for every graph g,

$$M(g) = \{\, g_1 \uplus g_2 \mid g_1 \in M_1(g),\ g_2 \in M_2(g) \,\},$$

where $g_1 \uplus g_2$ means the disjoint union of g_1 and g_2.

Proof. Let $M_1 = (C_1, \phi_1, \Psi_1, X_1)$ and $M_2 = (C_2, \phi_2, \Psi_2, X_2)$. We can assume w.l.o.g. the components of M_1 and M_2 are mutually disjoint. Then M can be defined as $(C_1 \cup C_2, \phi_1 \wedge \phi_2, \Psi_1 \cup \Psi_2, X_1 \cup X_2 \cup X)$, where X is the set of edge formulas $\{\, \chi_{c,c',\gamma} \equiv \mathrm{false} \mid (c, c') \in (C_1 \times C_2) \cup (C_2 \times C_1) \,\}$, which guarantees that there is no edges between nodes named $c \in C_1$ and $c' \in C_2$ by M. □

Lemma 11. [Lemma 7 of [20]] Let a, b be distinct symbols and let M_1, M_2 be MSO graph-to-string transducers. There exists an MSO graph-to-dgraph transducer $M^{a,b}$ such that for every graph g,

$$M^{a,b}(g) = \{\, \mathrm{dgr}(a^m b^n) \mid \exists w_1 \in M_1(g),\ w_2 \in M_2(g) : \\ w_1/m = a \ \text{ and } \ w_2/n = b \,\}.$$

Proof. (Sketch) By Proposition 1, Lemma 9 and Lemma 10, we can construct the desired transducer $M^{a,b} = (M_1 \circ N^a_{\Delta_1}) \uplus (M_2 \circ N^b_{\Delta_2})$. □

The decision procedure of [20] consists of three steps, in which the above three lemmas are used:

Step 1. Let M_1, M_2 be deterministic MSO tree transducers. Construct a deterministic MSO tree-to-string transducer W that "flattens" input trees to strings, then compose them with M_1, M_2, i.e., construct M'_1 and M'_2 with $[\![M'_1]\!] = [\![M_1]\!] \circ [\![W]\!]$ and $[\![M'_2]\!] = [\![M_2]\!] \circ [\![W]\!]$, respectively. Since DMSOT transductions

are closed under composition (see Proposition 1), M'_1 and M'_2 are obtained as DMSOT transducers.

Step 2. Let $N^{\$}$ be a deterministic MSO string transducer that reads a given input string $w \in \Delta_1 \cup \Delta_2$ and outputs the string $w\$$, where \$ is a symbol not in $\Delta_1 \cup \Delta_2$. For $i \in [2]$, let $M_i^{\$}$ be a deterministic MSO tree-to-string transducer with $[\![M_i^{\$}]\!] = [\![M'_i]\!] \circ [\![N^{\$}]\!]$. Now we can say that $[\![M_1]\!] \neq [\![M_2]\!]$ if and only if $[\![M_1^{\$}]\!] \neq [\![M_2^{\$}]\!]$ *.

Step 3. Let D_1 and D_2 be the domains of $M_1^{\$}$ and $M_2^{\$}$, respectively. Clearly, if $D_1 \neq D_2$ then $[\![M_1^{\$}]\!] \neq [\![M_2^{\$}]\!]$. It is decidable whether D_1 coincides with D_2 because the domain of every (D)MSOT transducer is regular (see, e.g., Theorem 5.82 of [13]). Then, assume $D_1 = D_2$ here and denote the domains of them by D. $[\![M_1^{\$}]\!] \neq [\![M_2^{\$}]\!]$ if and only if there exist $a \in \Delta_1 \cup \{\$\}$, $b \in \Delta_2 \cup \{\$\}$, $n \in \mathbb{N}$, and $t \in D$ such that $a \neq b$, $M_1^{\$}(t)/n = a$, and $M_2^{\$}(t)/n = b$. Let $M^{a,b}$ be the transducer of Lemma 11 for a, b, $M_1^{\$}$, and $M_2^{\$}$. $[\![M_1^{\$}]\!] \neq [\![M_2^{\$}]\!]$ iff $\mathrm{dgr}(a^n b^n)$ is in $M^{a,b}(D)$ for some $n \in \mathbb{N}$. Note that $\mathrm{dgr}(a^n b^n)$ is in $M^{a,b}(D)$ iff (n, n) is in $\mathrm{Par}(M^{a,b}(D))$. By Lemma 7 the set of vectors $\mathrm{Par}(M^{a,b}(D))$ is Parikh because D is regular and thus VR. By Lemma 8 it can be decided whether (n, n) is in $\mathrm{Par}(M^{a,b}(D))$ for some $n \in \mathbb{N}$. This proves the main result of [20].

Theorem 12. [Corollary 10 of [20]] It is decidable whether two DMSOT transducers are equivalent.

We now extend Theorem 12 slightly to show the decidability of the universal preservation for nondeterministic views.

Theorem 13. It is decidable whether a deterministic MSOT transducer M_1 and a nondeterministic MSOT transducer M_2 are equivalent.

Proof. Recall that the domain of every (D)MSOT is regular, so we can decide whether the domain of M_1 coincides with that of M_2. Let us assume $\mathrm{dom}(M_1) = \mathrm{dom}(M_2) = D$. Since Lemmas 7–11 described above are not restricted to deterministic MSOT transducers, the same idea for DMSOT can be applied to our case. Thus we can state that $M_1 \in$ DMSOT and $M_2 \in$ MSOT are not equivalent if and only if there exist $a \in \Delta_1 \cup \{\$\}$, $b \in \Delta_2 \cup \{\$\}$, $n \in \mathbb{N}$, $t \in D$, and $w \in M_2^{\$}(t)$ such that $a \neq b$, $M_1^{\$}(t)/n = a$, and $w/n = b$. The nondeterminism of M_2 makes differences in Steps 1–3, in which nondeterministic MSOT transducers M'_2 and $M_2^{\$}$ such that $[\![M'_2]\!] = [\![M_2]\!] \circ [\![W]\!]$ and $[\![M_2^{\$}]\!] = [\![M'_2]\!] \circ [\![N^{\$}]\!]$ are constructed instead of

* If we do not compose $N^{\$}$ with M'_i, then the following procedure answers "M_1 and M_2 are equivalent" incorrectly if one of them always outputs "prefix strings" of the other, i.e., for every input tree t and let $w = M'_1(t)$, there exists a string $z \neq \varepsilon$ such that $M'_2(t) = w \cdot z$, where $\cdot$ is concatenation of strings.

deterministic ones. Recall that MSOT transductions are closed under composition (Proposition 1), i.e., MSOT $\circ$ MSOT $\subseteq$ MSOT, hence MSOT $\circ$ DMSOT $\subseteq$ MSOT. For an input tree t, $a \in \Delta_1 \cup \{\$\}$, and $b \in \Delta_2 \cup \{\$\}$, assume $M_1^{\$}(t) = w_1$ and $w_2 \in M_2^{\$}(t)$ with $i \in \mathbb{N}$, $w_1/i = a$, $w_2/i = b$, $a \neq b$. In the case, obviously $M_1^{\$}$ and $M_2^{\$}$ are not equivalent. Let $M^{a,b}$ be an MSOT transducer in Step 3, which can be (effectively) constructed because Lemma 11 is not restricted to DMSOT transducers. By the assumption, $\mathrm{dgr}(a^i b^i) \in M^{a,b}(t)$, and so $\mathrm{Par}(M^{a,b}(t))$ contains $(i,i) \in \mathbb{N}^2$. Conversely, for distinct symbols $c \in \Delta_1 \cup \{\$\}$ and $d \in \Delta_2 \cup \{\$\}$, assume $\mathrm{Par}(M^{c,d}(D))$ includes $(j,j) \in \mathbb{N}^2$ with $j \in \mathbb{N}$. By the assumption, there exist a tree $t \in D$, $w_1 \in \Delta_1^*$, $w_2 \in \Delta_2^*$ such that $M_1^{\$}(t) = w_1\$$, $w_2\$ \in M_2^{\$}(t)$, $w_1\$/j = c$, and $w_2\$/j = d$. Hence M_1 and M_2 are not equivalent. It is decidable whether there exists $(i,i) \in \mathbb{N}^2$ in $\mathrm{Par}(M^{a,b}(t))$ because Lemmas 7 and 8 are not restricted to DMSOT transducers. □

We would like to mention that the discussion above can be applied to the equivalence of MSO graph-to-string or graph-to-tree transducers. Hence we can generalize the previous result (Theorem 8 of [20]) as follows, but the generalized result is not used in this thesis.

Corollary 14. For a deterministic MSO graph-to-tree or graph-to-string transducer M_1, a nondeterministic one M_2, and a VR set D of graphs, it is decidable whether $[\![M_1]\!]|_D = [\![M_2]\!]|_D$.

Proof. Let $D_i = \mathrm{dom}(M_i) \cap D$ for $i \in [2]$. As stated in the proof of Theorem 8 of [20], it is decidable whether $D_1 = D_2$. If $D_1 \neq D_2$, then $M_1 \neq M_2$. The rest of the proof of the case $D_1 = D_2$ is exactly the same as the proof of Theorem 13. □

We are now ready to describe our main result.

Theorem 15. For $v \in (\mathrm{ELT}^{\mathrm{R}})^*$ as a view and $q \in$ DMSOT as a query, it is decidable whether v $\forall$-preserves q.

Proof. By Theorem 2 we can construct a $\mathrm{DT}^{\mathrm{R}}_{\mathrm{fc}}$ transducer that realizes a uniformizer u of v^{-1}. Let $v \in (\mathrm{ELT}^{\mathrm{R}})^*$, $q \in$ DMSOT. We show that v $\forall$-preserves q if and only if $q = v \circ u \circ q$. The right-to-left direction is obvious. For the other direction, assume that v $\forall$-preserves q and let $\tilde{q}$ be a query in DMSOT such that $q = v \circ \tilde{q}$. Then, $v \circ u \circ q = v \circ u \circ v \circ \tilde{q} = v \circ \tilde{q} = q$ holds. Precisely, let $t \in \mathrm{dom}(q)$

and then,

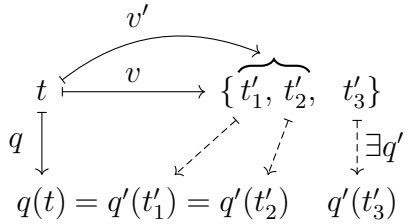
$$
\begin{aligned}
& q(u(v(t))) \\
&= \{\, q(t'') \mid t' \in v(t),\ t'' \in u(t') \,\} \\
&= \{\, \tilde{q}(v(t'')) \mid t' \in v(t),\ t'' \in u(t') \,\} \quad (\because q = v \circ \tilde{q}) \\
&= \{\, \tilde{q}(t') \mid t' \in v(t) \,\} \ (\because v \circ \tilde{q} \text{ is single-valued},\ t' \in v(t'')) \\
&= \tilde{q}(v(t)) \\
&= q(t).
\end{aligned}
$$

It follows that $u \circ q \in \text{DMSOT}$ by Proposition 1 and Theorem 7.4 of [18] that states $\text{DT}^{\text{R}}_{\text{fc}} \circ \text{DMSOT} \subseteq \text{DMSOT}$, and $v \circ u \circ q \in \text{MSOT}$ by Theorem 3. We can decide the equivalence of q and $v \circ u \circ q$ by Theorem 13. □

2.6 Existential Preservation

Even when v does not universally preserve q, it is still too early to give up on preserving information of q. In the case, we would like to know whether v existentially preserves q. There is a simple relation between the existential preservation and the universal preservation.

Proposition 16. Let v, v' be (nondeterministic) views and q be a single-valued query. If $v' \subseteq v$ and v' $\forall$-preserves q, then v $\exists$-preserves q.



Based on Proposition 16, we give a decidable sufficient condition of the existential preservation. We only focus on *finite*-valued views, because we use a decomposition theorem for finite-valued B transducers (recall that the results in the previous section focus not only on finite-valued views but also on "infinite"-valued ones). Note that it is decidable in deterministic polynomial time whether a B transducer is finite-valued or not (Theorem 6.9 of [47])†.

The following theorem states that every finite-valued B transducer can be effectively decomposed into a finite union of single-valued ones.

† We also note that for fixed $k > 1$ and a B transducer, it is decidable whether $\text{val}(M) \geq k$ in nondeterministic polynomial time (Theorem 2.2 of [48]). No deterministic polynomial time algorithm has been founded so far (to our knowledge) for the k-bounded finite-valuedness

Theorem 17. [Theorem 6.2 of [48]] For every finite-valued B transducer M, there exist single-valued ones $M_1, \ldots, M_K$ such that $[\![M]\!] = [\![M_1]\!] \cup \cdots \cup [\![M_K]\!]$, where $K \leq 2^{2^{P(|M|)}}$, $|M_j| \leq 2^{2^{P(|M|)}}$, $j \in [K]$, for some polynomial P independent of M.[‡]

By Proposition 16 and Theorem 17, we obtain a decidable sufficient condition of the existential preservation for views realized by finite-valued LB transducers and queries realized by DMSOT (or DT^{R}) transducers as follows. Notice that, in the following theorem, each v_j is single-valued, hence we just say "v_j preserves" instead of "v_j $\forall(\exists)$-preserves."

Theorem 18. Let v be a finite-valued LB transduction and q be a DMSOT (or DT^{R}) transduction. Let $v_1, \ldots, v_K$ be S-LB transductions such that $v = v_1 \cup \cdots \cup v_K$. Then, v $\exists$-preserves q if v_j preserves q for some $j \in [K]$. The latter condition is decidable.

Proof. By Proposition 16, v $\exists$-preserves q if v_j preserves q for some j. Every S-LB transduction is in $(\text{S-ELT}^{\text{R}})^*$. Thus the preservation is decidable by Theorem 15 of [4], which states that it is decidable for $\bar{v} \in (\text{S-ELT}^{\text{R}})^*$ and $\bar{q} \in$ DMSOT (or DT^{R}), whether $\bar{v}$ preserves $\bar{q}$ (see Table 1.1). □

Let $\exists$-Pres be the algorithm that decides the above sufficient condition.

Algorithm $\exists$-Pres

Input: Finite-valued LB transduction v, DMSOT (or DT^{R}) transduction q.

Output: If v_j preserves q for some $j \in [K]$, output "Yes," otherwise "No."

Step 1. Decompose v into S-LB transductions $v_1, \ldots, v_K$ by using Theorem 17.

Step 2. Decide v_j preserves q for each $j \in [K]$ by using Theorem 15 of [4], which states that it is decidable for a $(\text{S-ELT}^{\text{R}})^*$ transduction v' and a DMSOT (or DT^{R}) transduction q', whether v' preserves q'. Note that S-LB $\subsetneq (\text{S-ELT}^{\text{R}})^*$.

We are not sure whether $\exists$-Pres (and the other algorithms that are sound in the rest of this chapter) is complete for existential preservation, because even when

problem for B transducers. In the word case, for a word transducer there are deterministic polynomial time algorithms presented in [43,50] for the finite-valuedness, and for the k-bounded valuedness in [30, 43].

[‡]Here the size of a B transducer $M = (P, \Sigma, \Delta, F, R)$ is denoted by $|M|$, which is the sum of the sizes of all its rules:

$$|M| = \sum_{(\sigma(p_1(x_1),\ldots,p_k(x_k)) \to p(r)) \in R} (k + |r| + 2)$$

where $|r|$ is the size of a tree $r = \delta(t_1, \ldots, t_m) \in \mathcal{T}_\Delta(\mathcal{X}_m)$ defined by: $|r| = 0$ if $\delta \in \mathcal{X}_m$, $|r| = 1$ if $\delta \in \Delta^{(0)}$, and $|r| = 1 + \sum_{i=1}^{m} |t_i|$ if $\delta \in \Delta^{(m)}$.

$\exists$-Pres answers "no," another s-LB transduction v' may exist such that $v' \subseteq v$ and v' preserves q. The problem of deciding whether such v' exists seems to be hard, because one is required to prove a given query q is not preserved by v_i ($i \in [K]$) for *every* possible way of decomposing a finite-valued tree transduction v into single-valued ones $v_1, \ldots, v_K$.

Extending the domain of a view

Let v be a view given by a finite-valued LB transducer and q be a query given by a DMSOT transducer. When $\exists$-Pres answers "no," there remains a possibility that v existentially preserves q. When decomposing a view v into $v_1, \ldots, v_K$ by Theorem 17, the domain of the resulting transduction v_i may be a proper subset of the domain of the original transduction v. Consider the case when $\mathrm{dom}(q) \supsetneq \mathrm{dom}(v_i)$ for each $i \in [K]$. In the case, by the definition of query preservation, every v_i does not preserve q. Still, there may be $S \subseteq [K]$ such that the union v' of transductions $\forall$-preserves q where $v' = \bigcup_{j \in S} v_j$. If so, we can conclude that $\bigcup_{j \in [K]} v_j$ $\exists$-preserves q. Hence the following result holds as a corollary of Theorem 15. Note that every union of s-LB transductions is an LB transduction.

Corollary 19. Let $v_1, \ldots, v_K \in$ s-LB and $q \in$ DMSOT. It is decidable whether there is $S \subseteq [K]$ such that v' $\forall$-preserves q where $v' = \bigcup_{j \in S} v_j$. If such an S exists, $\bigcup_{j \in [K]} v_j$ $\exists$-preserves q (by Proposition 16).

By the way, let q be a query given by an s-B transducer. Since s-B is incomparable with DMSOT, we cannot apply Theorem 15 directly to obtain a decidable sufficient condition similar to the one stated in Corollary 19 for $q \in$ s-B. For the case, in order to construct an appropriate view $v' (\subseteq v)$ from $v_1, \ldots, v_K$ obtained by decomposing v by Theorem 17, each component v_j of v' is required not to be *joinable* with another. For views v_i, v_j and a query q, we say that v_i and v_j are joinable against q if there exists a pair of trees (t_1, t_2) such that $q(t_1) \neq q(t_2)$ and $v_i(t_1) = v_j(t_2)$. To show why transductions must not be joinable, let us suppose that v_i and v_j are joinable against q with a pair (t_1, t_2) and $v' = v_i \cup v_j$ (at least) existentially preserves q, then there exists q' in the same class as q's (s-B) such that for all $t \in \mathrm{dom}(q)$ there exists $t' \in v'(t)$ satisfying $q(t) = q'(t')$. However, $q'(t'_{1,2})$ for $t'_{1,2} = v_i(t_1) = v_j(t_2)$ has to be $\{q(t_1), q(t_2)\}$ but q' is single-valued, which is a contradiction. Thus, each component v_j of v' is required not to be joinable against q with another. We show that joinability is decidable by using the single-valuedness test for EB transducers with "grafting" presented in [33].

Lemma 20. Let $v_1, v_2 \in$ s-LB and $q \in$ s-B. It is decidable whether v_1 and v_2 are joinable against q.

Proof. Construct ELB transducers with grafting (ELB^{+g} for short) that realize v_1^{-1} and v_2^{-1}, respectively. The detailed construction is given in [33] (see Lemma 6 in it). Next, construct a TA A that accepts the intersection of the ranges of v_1 and v_2, i.e., $L(A) = \mathrm{ran}(v_1) \cap \mathrm{ran}(v_2)$. A can be effectively constructed by the fact that the range of every LB transducer is effectively regular (see, e.g., Corollary 3.11 of [15]) and by the construction of product automaton. After that, construct an ELB^{+g} transducer that realizes $w = v_1^{-1} \cup v_2^{-1}|_{L(A)}$, which is the union of v_1^{-1} and v_2^{-1}, and the domain of which is restricted to $L(A)$. Finally, decide whether $q' = w \circ q$ is single-valued or not, which is decidable by Lemmas 7–8 of [33]. It is not difficult to show that v_1 and v_2 are joinable against q if and only if q' is *not* single-valued. The left-to-right direction holds obviously by the definition of joinability. Conversely, if q' is not single-valued, there exists a tree $t \in \mathrm{dom}(q') \subseteq \mathrm{ran}(v_1) \cap \mathrm{ran}(v_2)$ such that $|q'(t)| \geq 2$. Hence, there exist t'_1, $t'_2 \in q'(t)$ with $t'_1 \neq t'_2$. Since q is single-valued, there exist t_1, t_2 ($t_1 \neq t_2$) such that $q(t_1) = t'_1 \neq t'_2 = q(t_2)$ and $v_1(t_1) = v_2(t_2)$. Thus v_1 and v_2 are joinable against q. □

Lemma 21. Let $v_1, \ldots, v_K \in$ s-LB and $q \in$ s-B. The union v' $\forall$-preserves q if and only if (1) each component v_j of v' is not joinable against q with another, (2) v_j preserves $q|_{\mathrm{dom}(v_j)}$, and (3) $\mathrm{dom}(q) \subseteq \mathrm{dom}(v')$.

Proof. As described in the beginning of this subsection, if (1) does not hold, v' does not $\exists$-preserve q, and hence v' also does not $\forall$-preserve q. On the other side, suppose (1), (2), and (3) hold. For simplicity, suppose $K = 2$. By (2), let q'_1, q'_2 be s-LB transductions such that $q|_{\mathrm{dom}(v_i)} = v_i \circ q'_i$ ($i \in [2]$). For any $t_{12} \in \mathrm{dom}(v_1) \cap \mathrm{dom}(v_2)$, $q(t_{12}) = (v_1 \circ q'_1)(t_{12}) = (v_2 \circ q'_2)(t_{12})$. Hence for any $t \in \mathrm{dom}(v_1) \cup \mathrm{dom}(v_2)$, $q(t) = (v_1 \cup v_2) \circ (q'_1 \cup q'_2)(t)$ holds. Also, $q'_1 \cup q'_2$ is single-valued by (1). Therefore, $v_1 \cup v_2$ $\forall$-preserves $q|_{\mathrm{dom}(v_1) \cup \mathrm{dom}(v_2)}$. Generally, $v_1 \cup \cdots \cup v_K$ $\forall$-preserves $q|_{\mathrm{dom}(v_1) \cup \cdots \cup \mathrm{dom}(v_K)}$, therefore v' $\forall$-preserves q by (3). □

By Lemmas 20, 21, we obtain the following result for queries realized by s-B transducers.

Theorem 22. Let $v_1, \ldots, v_K \in$ s-LB and $q \in$ s-B. It is decidable whether there is $S \subseteq [K]$ such that v' $\forall$-preserves q where $v' = \bigcup_{j \in S} v_j$. If such an S exists, $\bigcup_{j \in [K]} v_j$ $\exists$-preserves q (by Proposition 16).

Proof. It suffices to show that the three conditions (1)–(3) in Lemma 21 are decidable. (1) is decidable by Lemma 20 and (2) is also decidable [33] (see Table 1.1). (3) is decidable due to the regularity of $\mathrm{dom}(q)$ and $\mathrm{dom}(v')$. □

2.7 Nondeterministic Queries

Let v be a view and q be a nondeterministic query. In this section, we show two algorithms that are sound for the problem of deciding whether v universally or existentially preserves *a part of* q (see Section 2.4 for definition).

We can adopt the idea of Theorem 18 to obtain an algorithm called $\forall$-PresPart that is sound for query preservation for finite-valued queries. $\forall$-PresPart is almost the same as $\exists$-Pres (see the previous section), but it decomposes a given query q into $q_1, \ldots, q_K$ instead of a given view v, after that for each $i \in [K]$ it tests whether v $\forall$-preserves q_i.

Algorithm $\forall$-PresPart

Input: $(\text{ELT}^{\text{R}})^*$ transduction v, finite-valued LB transduction q.

Output: If v preserves q_j for some $j \in [K]$, output "Yes," otherwise "No."

Step 1. Decompose q into s-LB transductions $q_1, \ldots, q_K$ by using Theorem 17.

Step 2. Decide v $\forall$-preserves q_j for each $j \in [K]$. It is decidable by using Proposition 4 and Theorem 15.

Theorem 23. For $v \in (\text{ELT}^{\text{R}})^*$ as a view and a finite-valued LB transduction q as a query, the algorithm $\forall$-PresPart is sound for the problem of deciding whether v $\forall$-preserves a part of q.

By Theorems 18, 23, and by Theorem 21 of [33] that states query preservation is decidable for a view realized by an s-ELB transducer and a query realized by an s-B transducer (see Table 1.1), we also obtain an algorithm (called $\exists$-PresPart) for the case when views and queries are both finite-valued. We do not describe explicitly the algorithm $\exists$-PresPart here because it can easily be obtained by modifying slightly the algorithm $\forall$-PresPart.

Corollary 24. For a view v (resp. query q) realized by a finite-valued LB (resp. B) transducer, the algorithm $\exists$-PresPart is sound for the problem of deciding whether v $\exists$-preserves a part of q.

Weak condition. By the way, in [4] the "weak condition" is considered. For a view $v \in \mathcal{V}$ and a query $q \in \mathcal{Q}$, v ($\forall$- or $\exists$-) preserves q if there exists $q' \in \mathcal{Q}$ such that (1) $\text{dom}(v \circ q') = \text{dom}(q)$ and (2) $(v \circ q')(t) = q(t)$ for every $t \in \text{dom}(q)$. For some situation in practice, the condition (1) is weakened to $\text{dom}(v \circ q') \supseteq \text{dom}(q)$. The above results (Theorems, Lemmas, and Corollaries 15–24) can be applied to the weak condition by restricting $\text{dom}(v)$ to $\text{dom}(q) \cap \text{dom}(v)$[§].

[§] It is known that $\text{dom}(q)$ and $\text{dom}(v)$ are regular for every q realized by a DMSOT or an LB transducer and every v realized by $(\text{ELT}^{\text{R}})^*$, hence $\text{dom}(q) \cap \text{dom}(v)$ is also effectively regular.

Complexity. Since the complexity of the algorithm for deciding equivalence of DMSOT transducers is non-elementary [20], that of our procedure for deciding equivalence of DMSOT and MSOT transducers (Theorem 13) is also non-elementary. Hence the complexities of our algorithms that use the procedure are non-elementary as well. The complexities of the other algorithms (that do not use the procedure for the equivalence of (D)MSOT transducers) are dominated by that of the decomposition theorem (Theorem 17), which is in 2-EXPTIME [48].

2.8 Conclusion of the Chapter

We have defined two kinds of query preservation problem for nondeterministic views and queries on ranked trees: universal preservation and existential preservation. We have proved that the universal preservation is decidable for compositions of extended linear top-down tree transducers with regular look-ahead as views and deterministic MSO tree transducers as queries (see Theorem 15). To obtain the result we have slightly generalized the result [20] of the equivalence problem for deterministic MSO tree transducers (see Theorem 13). Moreover, we have shown an algorithm that is sound for the existential preservation for finite-valued linear bottom-up tree transducers as views and deterministic MSO tree transducers as queries (see Theorem 18), and also showed some algorithms that are sound for the problem for nondeterministic queries realized by finite-valued (linear) bottom-up tree transducers (see Theorem 23 and Corollary 24).

Chapter 3

Hybrid Approach to Query Preservation

3.1 Introduction

"You boldly settle all important questions, but tell me, dear,
isn't it because you're young, because you haven't had time to suffer
till you settled a single one of your questions?"

— *Act III* [11]

In this chapter, we focus on query preservation of a tree transducer for a node query where models for view(s) and queries are different.

As mentioned in Chapter 1, we examine query preservations of data tree transformations. We treat data trees as a data model, which is a ranked ordered tree where each node can have any nonnegative integer as a data value. We use deterministic linear top-down data tree transducers (abbreviated as DLT$^{\mathrm{V}}$s) and run-based n-ary queries [42] as classes of transformations and queries, respectively.

It is unknown whether the decision problem of the preservation for node queries discussed in this chapter can be reduced to the problem for a certain class of tree queries which is known to be decidable (mentioned in Chapter 2). For this reduction, we have to construct a tree query corresponding to a given node query, taking care of what kind of ordered tree is necessary and sufficient to express the relation of extracted nodes as well as the existence of the nodes. We conjecture that such simulation of run-based n-ary queries requires (a subclass of) DMSOT for the node selection ability, but, as mentioned in Chapter 2, the known way to decide the query preservation problem for (D)MSOT transducers requires deciding

the equivalence of DMSOT (and MSOT) transducers, which is non-elementary in general. In contrast, we show later that the query preservation problem for run-based n-ary queries is in 2-EXPTIME and EXPTIME-hard.

We define two types of query preservation in the above setting: weak and strong query preservation. We say that a DLT^{V} Tr strongly preserves a query Q if there is a query Q' such that for every tree 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, the answer set of Q' for $Tr(t)$ includes the answer set of Q for t.

Main results in this chapter are as follows. 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 and EXPTIME-hard. If the tuple size n of queries is constant, the complexity becomes EXPTIME-complete. The decidability results of the two cases can be extended to the situation where the view is given by a single-valued extended linear top-down data tree transducer with regular look-ahead (abbreviated as an s-ELT^{VR}), which has more expressive power than DLT^{V}.

Organization

This chapter is organized as follows: Section 3.2 provides terminology and definitions of data trees, tree automata, DLT^{V}, and run-based n-ary queries. Section 3.3 defines weak and strong query preservation problem and summarizes the decidability results of these problems, which are the main results of this chapter. Our decision algorithms for weak and strong query preservation problem are presented in Section 3.4 and Section 3.6, respectively. If a transducer Tr strongly (resp. weakly) preserves a query Q for every tree t, the algorithm we present in Section 3.5 can construct the query Q' for $Tr(t)$ that satisfies the conditions of the strong (resp. weak) query preservation. In Section 3.7, it is shown that the problems are decidable for s-ELT^{VR}. Section 3.8 concludes the chapter and outlines future work.

3.2 Preliminaries

Some notations and devises mentioned in Chapter 2 are redefined here for the sake of self-containment and readability.

3.2.1 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 Σ of symbols with a mapping $\mathrm{rk} : \Sigma \to \mathbb{N}$. Let $\Sigma_n = \{\, \sigma \in \Sigma \mid \mathrm{rk}(\sigma) = n \,\}$. 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 $\mathcal{T}_\Sigma^{(\mathbb{N})}$ of data trees over an alphabet Σ 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) $\mathrm{pos}(t)$ is defined in the usual way and let t/v denote the subtree of t at position $v \in \mathrm{pos}(t)$. The size $|t|$ of a data tree t is $|\mathrm{pos}(t)|$. A data tree t is *proper* if every symbol appearing in t has a value. If $t/v = \sigma^{(\nu)}(t_1, \ldots, t_n)$, we write $\mathrm{lab}(t, v) = \sigma$ and $\mathrm{val}(t, v) = \nu$. If $t/v = \sigma(t_1, \ldots, t_n)$, we write $\mathrm{lab}(t, v) = \sigma$ and $\mathrm{val}(t, v) = \mathit{nil}$. Let $t[v \leftarrow t']$ be the tree obtained from t by replacing t/v with t'. We say that a data tree t is *value-unduplicated* if $\mathrm{val}(t, v_1) \neq \mathrm{val}(t, v_2)$ for any two different positions $v_1, v_2 \in \mathrm{pos}(t)$.

A *tree* is a data tree that does not contain any value. Let $\mathcal{T}_\Sigma$ denote the set of all trees over Σ. 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 $\mathcal{X}_n = \{\, x_i \mid i \in [n] \,\}$ be a set of variables with $\mathrm{rk}(x_i) = 0$ for every $x_i \in \mathcal{X}_n$. A tree t is *linear* if each variable occurs at most once in t. A linear tree in $\mathcal{T}_{\Sigma \cup \mathcal{X}_n}$ is called an (n-ary) *context* over Σ. Let $\mathcal{C}(\Sigma, \mathcal{X}_n)$ denote the set of n-ary contexts over Σ. For a context $C \in \mathcal{C}(\Sigma, \mathcal{X}_n)$, let $C[t_1, \ldots, t_n]$ denote the tree obtained from C by replacing x_i with t_i for $i \in [n]$.

3.2.2 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, Σ is a ranked alphabet, $P_I \subseteq P$ is a set of initial states, and δ is a finite set of transition rules of the form $p \to \sigma(p_1, \ldots, p_n)$ where $p \in P$, $\sigma \in \Sigma_n$, and $p_1, \ldots, p_n \in P$. Let $\mathrm{state}(A) = P$. A TA A accepts a tree $t \in \mathcal{T}_\Sigma$ if there is a mapping $m : \mathrm{pos}(t) \to P$ such that (1) $m(\varepsilon) \in P_I$, and (2) for $v \in \mathrm{pos}(t)$ with $t/v = \sigma(t_1, \ldots, t_n)$, $m(v) \to \sigma(m(v1), \ldots, m(vn)) \in \delta$. The mapping m is called a *run* of A on t. The set of all runs of A on t is denoted by $\mathrm{run}(A, t)$. Let $L(A) = \{\, t \in \mathcal{T}_\Sigma \mid \mathrm{run}(A, t) \neq \emptyset \,\}$. A state of A is *useless* if it is not assigned to any position by any 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. For a TA $A = (P, \Sigma, P_I, \delta)$, let $|r| = \mathrm{rk}(\sigma) + 2$ be the size of a rule $r = p \to \sigma(p_1 \ldots, p_{\mathrm{rk}(\sigma)}) \in \delta$, and the size of A, denoted by $|A|$, is $|P| + \sum_{r \in \delta} |r|$.

A *linear top-down data tree transducer* (LT^{V}) is a transducer that can implement standard edit operations on trees such as insertion, deletion, relabeling, and moving data values. An LT^{V} is a tuple $Tr = (P, \Sigma, \Delta, P_I, \delta)$ where P is a finite set of states, Σ and Δ are ranked alphabets of input and output, respectively,

$P_I \subseteq P$ is a set of initial states, and δ is a finite set of transduction rules of the form

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

where $p, p_1, \ldots, p_n \in P$, $\sigma \in \Sigma_n$, $j \in \{\, v \mid v \in \mathrm{pos}(C),\ t/v \notin \mathcal{X}_n \,\}$, $x_1, \ldots, x_n \in \mathcal{X}_n$, $C \in \mathcal{C}(\Delta, \mathcal{X}_n)$, 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 $p(\sigma^{(z)}(x_1, \ldots, x_n)) \to C^{(j \leftarrow z)}[p_1(x_1), \ldots, p_n(x_n)] \in \delta$, $t_1, \ldots, t_n \in \mathcal{T}_\Sigma^{(\mathbb{N})}$ and $t/v = p(\sigma^{(\nu)}(t_1, \ldots, t_n))$ $(\nu \in \mathbb{N})$, then

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

where $C^{(j \leftarrow \nu)}$ is the context obtained from C by replacing $\mathrm{lab}(C, j)$ with $\mathrm{lab}(C, j)^{(\nu)}$. When the value position designation is missing in the rule, Tr does not transfer ν to any position of the output. Let $[\![Tr]\!] = \{\, (t, t') \mid p_I(t) \Rightarrow_{Tr}^* t',\ t \in \mathcal{T}_\Sigma^{(\mathbb{N})},\ t \text{ is proper},\ t' \in \mathcal{T}_\Delta^{(\mathbb{N})},\ p_I \in P_I \,\}$. The domain of Tr is defined as $\mathrm{dom}(Tr) = \{\, t \mid \exists t'.\ (t, t') \in [\![Tr]\!] \,\}$, and the range of Tr is defined as $\mathrm{rng}(Tr) = \{\, t' \mid \exists t.\ (t, t') \in [\![Tr]\!] \,\}$. We define the size $|Tr|$ of an LT^{V} Tr as with tree automata. Let $|r| = |\mathrm{pos}(C)| + 2$ be the size of a rule $r = p(\sigma^{(z)}(x_1, \ldots, x_n)) \to C^{(j \leftarrow z)}[p_1(x_1), \ldots, p_n(x_n)] \in \delta$, and the size $|Tr|$ is $|P| + \sum_{r \in \delta} |r|$.

An LT^{V} $Tr = (P, \Sigma, \Delta, P_I, \delta)$ is *deterministic* (denoted as a $\mathrm{DLT}^{\mathrm{V}}$) if (1) $|P_I| = 1$, and (2) for each $p \in P$ and $\sigma \in \Sigma$, there exists at most one transduction rule that contains both p and σ in its left-hand side. If Tr is deterministic, there is only one pair $(t, t') \in [\![Tr]\!]$ for each $t \in \mathrm{dom}(Tr)$. Thus, we write $Tr(t) = t'$ when $(t, t') \in [\![Tr]\!]$. For $L \subseteq \mathcal{T}_\Sigma^{(\mathbb{N})}$, 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 DLT 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 $p_1(\sigma^{(z)}(x_1, x_2)) \to \sigma'^{(z)}(p_2(x_2))$. A *value-erasing* rule is a rule that does not have the value position designation in its right-hand side.

Consider a $\mathrm{DLT}^{\mathrm{V}}$ $Tr = (P_T, \Sigma, \Delta, \{p_T^0\}, \delta_T)$ and $t \in \mathrm{dom}(Tr)$. For $v \in \mathrm{pos}(t)$, we say that Tr *ignores* v in t if $p_T^0(t[v \leftarrow x]) \Rightarrow_{Tr}^* t' \in \mathcal{T}_\Delta^{(\mathbb{N})}$ where x is a variable. We say that Tr *reaches* $p \in P_T$ at v in t if there exist $C' \in \mathcal{C}(\Delta, \{x\})$ containing x and $t'_v \in \mathcal{T}_\Delta^{(\mathbb{N})}$ such that $p_T^0(t\,[v \leftarrow x]) \Rightarrow_{Tr}^* C'[p(x)]$ and $p(t/v) \Rightarrow_{Tr}^* t'_v$. For $t \in \mathrm{dom}(Tr)$, consider a mapping $m_t^T : \mathrm{pos}(t) \to P_T \cup \{\bot\}$ such that:

- if Tr ignores v in t, then $m_t^T(v) = \bot$,
- if Tr reaches p at v in t, then $m_t^T(v) = p$.

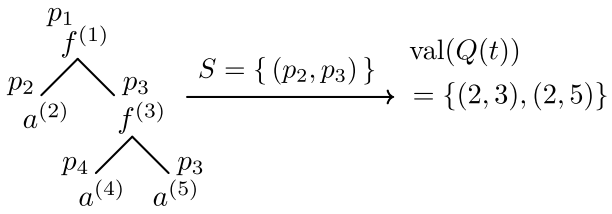


Figure 3.1: Example of 2-RQ.

The mapping m_t^T is determined uniquely by Tr and t because Tr is deterministic. We call m_t^T a *run* of Tr for t. Note that if $m_t^T(v) = p$ and $\mathrm{lab}(v) = \sigma$, then Tr applies to v a rule in δ_T with p and σ in its left-hand side.

3.2.3 Run-based n-ary Queries

A *run-based n-ary query* (n-RQ) [42] is a node selection query that retrieves sets of n-tuples of nodes from trees. An n-RQ is a pair (A, S) where $A = (P, \Sigma, P_I, \delta)$ is a TA and $S \subseteq P^n$. In this chapter, 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

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

where $Q(m,t) = \{ (v_1, \ldots, v_n) \mid (m(v_1), \ldots, m(v_n)) \in S, v_1, \ldots, v_n \in \mathrm{pos}(t) \}$. For an n-RQ Q and a data tree t, let $\mathrm{val}(Q(t)) = \{ (\mathrm{val}(t, v_1), \ldots, \mathrm{val}(t, v_n)) \mid (v_1, \ldots, v_n) \in Q(t), \mathrm{val}(t, v_i) \neq \mathit{nil},\ i \in [n] \}$. Note that if $t^- \notin L(A)$ then $Q(t) = \emptyset$. Two queries Q_1 and Q_2 are *equivalent* if $\mathrm{val}(Q_1(t)) = \mathrm{val}(Q_2(t))$ for each data tree t. We assume that for a given query $Q = (A, S)$, the TA A is reduced. We define that the size $|Q|$ of an n-RQ Q is $|A| + n\,|S|$.

Example 4. Consider the 2-RQ $Q = (A, S)$ defined by: $A = (P, \Sigma, P_I, \delta)$, $P = \{p_1, p_2, p_3, p_4\}$, $\Sigma_2 = \{f\}$, $\Sigma_0 = \{a\}$, $P_I = \{p_1\}$, $\delta = \{ p_1 \to f(p_2, p_3), p_2 \to a,\ p_3 \to f(p_4, p_3),\ p_3 \to a,\ p_4 \to a \}$, and $S = \{(p_2, p_3)\}$. Figure 3.1 shows that the result of the query on the data tree $t = f^{(1)}(a^{(2)}, f^{(3)}(a^{(4)}, a^{(5)}))$ is $\mathrm{val}(Q(t)) = \{(2,3),(2,5)\}$, where the numbers 1 to 5 are the data values on t. ◊

3.3 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

$$\forall t \in \mathrm{dom}(Tr).\ \mathrm{val}(Q'(Tr(t))) = \mathrm{val}(Q(t)). \tag{3.1}$$

We also define the *weak* query preservation. We say that the transducer Tr *weakly preserves* the query Q if there exists $Q' \in \mathcal{L}_Q$ such that

$$\forall t \in \mathrm{dom}(Tr).\ \mathrm{val}(Q'(Tr(t))) \supseteq \mathrm{val}(Q(t)). \tag{3.2}$$

Example 5. Let $Q = (A, \{p_1\})$ where $A = (\{p_0, p_1, p_2\}, \{f, g, a\}, \{p_0\}, \{p_0 \to f(p_1, p_2),\ p_0 \to g(p_2, p_1),\ p_1 \to a,\ p_2 \to a\})$. Let Tr be a DLT$^{\mathrm{V}}$ 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 $p_0 \to h(p_1, p_2), p_0 \to h(p_2, p_1)$ satisfies Eq. (3.2). On the other hand, Tr does not preserve Q because $\mathrm{val}(Q(t_1)) = \{4\} \neq \{5\} = \mathrm{val}(Q(t_2))$ while $Tr(t_1) = Tr(t_2)$, which imply that there is no Q' that satisfies Eq. (3.1). ◊

In this chapter, we primarily focus on the following decision problems.

PROBLEM:	n-WEAKQUERYPRES
INPUT:	DLT$^{\mathrm{V}}$ Tr, n-RQ Q
QUESTION:	Does Tr weakly preserve Q?

PROBLEM:	n-QUERYPRES
INPUT:	DLT$^{\mathrm{V}}$ Tr, n-RQ Q
QUESTION:	Does Tr preserve Q?

During the discussion about the problems, we assume that $L(A) \subseteq \mathrm{dom}(Tr)$ where A is a TA of an n-RQ $Q = (A, S)$. This assumption does not lose generality since we can easily construct a query $Q' = (A', S')$ satisfying the following conditions from a given Q and Tr:

- $L(A') = L(A) \cap \mathrm{dom}(Tr)$, and
- $\forall t \in \mathrm{dom}(Tr).\ \mathrm{val}(Q(t)) = \mathrm{val}(Q'(t))$.

Table 3.1: Summary of complexity results on node query preservations.

Parameter	n-WeakQueryPres	n-QueryPres
fixed n	in PTIME (Theorem 28)	EXPTIME-complete (Theorem 39)
unbounded n	coNP-complete (Theorem 28)	in 2-EXPTIME / EXPTIME-hard (Theorem 38)

Our main results are that n-WeakQueryPres is coNP-complete, and n-QueryPres is in 2-EXPTIME and EXPTIME-hard for unbounded n. If n is fixed, n-WeakQueryPres is in PTIME, and n-QueryPres is EXPTIME-complete. Summary of the complexity results is shown in Table 3.1.

The second result we obtained is that given an n-RQ Q and a DLT$^{\mathrm{V}}$ Tr, if Tr preserves Q then we can construct a query Q' satisfying the condition of the query preservation (Eq. (3.1)). If n is fixed, we can construct Q' in polynomial time (Theorem 33). We show the detail of our query construction algorithm and its correctness in Section 3.5.

3.4 Decision Algorithm for n-WeakQueryPres

3.4.1 Unary Queries

We first give a polynomial time algorithm (called 1-WQP) that decides 1-WeakQueryPres. In Section 3.4.2, we will give a decision algorithm for n-WeakQuery-Pres by using 1-WQP. We explain the idea of 1-WQP here, assuming for simplicity that $S = \{p\}$, i.e., $|S| = 1$, for a given query $Q = (A, S)$. Our algorithm for the weak query preservation decides if there exist a tree $t \in \mathrm{dom}(Tr)$ and a position $v \in \mathrm{pos}(t)$ satisfying the following conditions:

- There exists a run $m \in \mathrm{run}(A, t^-)$ such that $m(v) = p$.
- The data load at 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 v of t that satisfy the above conditions. Consider a value $\nu \in \mathbb{N}$ such that $\nu \neq \mathrm{val}(t, v')$ for any $v' \in \mathrm{pos}(t)$, and a tree $t^{(v \leftarrow \nu)}$, which is obtained from t by changing the value of the position v to ν. We then have $\nu \in \mathrm{val}(Q(t^{(v \leftarrow \nu)}))$ and $Tr(t^{(v \leftarrow \nu)}) = Tr(t)$. However, ν is not contained in $Tr(t^{(v \leftarrow \nu)})$ because the unique value ν at v on $t^{(v \leftarrow \nu)}$ is removed by Tr. Hence there is no Q' satisfying $\mathrm{val}(Q'(Tr(t^{(v \leftarrow \nu)}))) \supseteq \mathrm{val}(Q(t^{(v \leftarrow \nu)}))$, and

thus Tr does not weakly preserve Q. Conversely, if such t and v do not exist, we can specify the position of $Tr(t)$ corresponding to each position of t selected by Q, by a TA that simulates each run $m \in \mathrm{run}(A, t^-)$ on $Tr(t)$.

Algorithm 1-WQP to Decide 1-WeakQueryPres

Step 1 constructs a TA A_T such that $L(A_T) = \mathrm{dom}(Tr)$ and for every $t \in \mathrm{dom}(Tr)$ a run of A_T for t coincides with that of Tr for t. Step 2 and Step 3 construct a reduced product automaton A'' that satisfies $L(A'') = L(A) \cap L(A_T)$. Step 4 checks if there exists a state of A'' assigned to a position that will be selected by Q 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 $S \subseteq P_A$, $\mathrm{DLT}^{\mathrm{V}}$ $Tr = (P_T, \Sigma, \Delta, \{p_T^0\}, \delta_T)$.

Output: If Tr weakly preserves Q, output "Yes," otherwise "No."

Step 1. Construct the following TA $A_T = (P_T \cup \{\bot\}, \Sigma, \{p_T^0\}, \delta_T')$ from Tr where $\bot \notin P_T$ and δ_T' is the smallest set satisfying the following conditions.

- Let $p(\sigma^{(z)}(x_1, \ldots, x_d)) \to C^{(j \leftarrow z)}[p_1(x_1), \ldots, p_d(x_d)] \in \delta_T$ with $p, p_1, \ldots, p_d \in P_T$, $\sigma \in \Sigma_d$, and $C \in \mathcal{C}(\Delta, \mathcal{X}_d)$. For each $i \in [d]$, define $\widetilde{p}_i$ as follows. If C contains x_i, let $\widetilde{p}_i = p_i$. If C does not contain x_i, let $\widetilde{p}_i = \bot$. Then, $p \to \sigma(\widetilde{p}_1, \ldots, \widetilde{p}_d) \in \delta_T'$.
- For each $\sigma \in \Sigma$, $\bot \to \sigma(\bot, \ldots, \bot) \in \delta_T'$.

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 TA $A' = (P_A \times P_T', \Sigma, P_A^I \times P_T^I, \delta')$ from $Q = (A, S)$ and $A_T = (P_T', \Sigma, P_T^I, \delta_T')$: $(p_A, p_T) \to \sigma((p_A^1, p_T^1), \ldots, (p_A^d, p_T^d)) \in \delta'$ if and only if $p_A \to \sigma(p_A^1, \ldots, p_A^d) \in \delta_A$ and $p_T \to \sigma(p_T^1, \ldots, p_T^d) \in \delta_T'$.

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

Step 4. If the following subset $\mathcal{P}^{del} \subseteq P''$ is empty, output "Yes," otherwise "No."

$$\begin{aligned}\mathcal{P}^{del} = \{\, (p, p_T) \in P'' \mid\ & p \in S \text{ and } (p_T = \bot, \text{ or} \\ & \text{there are a rule } (p, p_T) \to \sigma((p_A^1, p_T^1), \ldots, (p_A^d, p_T^d)) \in \delta'' \\ & \text{and a value-erasing rule in } \delta_T \text{ that has } p_T \text{ and } \sigma \text{ in its left-hand side}) \,\}\end{aligned}$$

We now give a lemma for the correctness of our algorithm.

Lemma 25. Let Q be a 1-RQ and Tr be a DLT^{V}. Tr weakly preserves Q if and only if $\mathcal{P}^{del} = \emptyset$ in Step 4 of the algorithm 1-WQP.

Proof. ($\Rightarrow$) Suppose $\mathcal{P}^{del} \neq \emptyset$ and let $(p, p_T) \in \mathcal{P}^{del}$. We have $p \in S$, and either of the following two conditions holds: (i) $p_T = \bot$ or (ii) there is a symbol σ such that δ'' contains a rule $(p, p_T) \to \sigma((p_A^1, p_T^1), \ldots, (p_A^d, p_T^d))$ and δ_T contains a value-erasing rule with p_T and σ in its left-hand side. Since the TA A'' constructed in Step 3 of 1-WQP does not have useless states and rules, there exist a tree $t \in L(A'')$, a run $m \in \text{run}(A'', t)$, and a node $v \in \text{pos}(t)$ such that $m(v) = (p, p_T)$. In addition, if (ii) holds, we can suppose that $\text{lab}(t, v) = \sigma$. We now assume without loss of generality that t is value-unduplicated. Because A'' is a reduced product TA of A and A_T, we can see that $\pi_1 \circ m$ (resp. $\pi_2 \circ m$) is a run of A (resp. A_T) for t where π_i is a mapping that retrieves the i th component p_i from a state pair (p_1, p_2). Since $\pi_1 \circ m(v) = p \in S$, we have $\text{val}(t, v) \in \text{val}(Q(t))$. Since $\pi_2 \circ m$ is also a run of Tr for t and $\pi_2 \circ m(v) = p_T$, we see that v is ignored by Tr if (i) holds and $\text{val}(t, v)$ is erased by the value-erasing rule of Tr if (ii) holds. It follows from the uniqueness of $\text{val}(t, v)$ that $Tr(t)$ does not have $\text{val}(t, v)$ at any node. Hence, $\text{val}(t, v) \notin \text{val}(Q'(Tr(t)))$ holds for any 1-RQ Q'. Thus, Tr does not weakly preserve Q.
($\Leftarrow$) We give a proof for this direction in Section 3.5. More concretely, we show that if $\mathcal{P}^{del} = \emptyset$, the algorithm 1-WQC given in Section 3.5 can construct a query Q' satisfying Eq. (3.2). □

Theorem 26. 1-WeakQueryPres is in PTIME.

3.4.2 General Case

We sketch an algorithm for general case below. We will assume that $|S| = 1$ and let $s = (p_1, \ldots, p_n)$. The basic idea is to consider the 1-RQ $Q' = (A, \{p_1, \ldots, p_n\})$ instead of Q and test whether Tr weakly preserves Q'. However, this does not work in general because $Q(t)$ contains only a tuple $(v_1, \ldots, v_n)$ of positions such that there is a run $m \in \text{run}(A, t^-)$ satisfying $m(v_i) = p_i$ for each i ($i \in [n]$) simultaneously while $Q'(t)$ contains every position v such that there is a run $m \in \text{run}(A, t^-)$ satisfying $m(v) = p_i$ even if there is some p_j ($j \neq i$) such that for any $u \in \text{pos}(t)$, $m(u) \neq p_j$.

Example 6. Let $Q = (A, S)$ be the 3-RQ defined by

$$\begin{aligned}
&A = (P, \Sigma, P_I, \delta),\ P = \{p_1, p_2, p_3, p_\#\},\\
&\Sigma = \Sigma_2 \cup \{\#\},\ \Sigma_2 = \{A, B, C\},\ P_I = \{p_1\},\\
&\delta = \{\, p_1 \to A(p_2, p_\#),\ \ p_2 \to B(p_3, p_\#),\\
&\qquad p_2 \to C(p_\#, p_\#),\ p_3 \to C(p_\#, p_\#),\ p_\# \to \# \,\},\\
&S = \{(p_1, p_2, p_3)\}.
\end{aligned}$$

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

$$\begin{aligned}
\delta_T = \{\, & p_1(A^{(z)}(x_1, x_2)) \to A^{(z)}(p_2(x_1), p_\#(x_2)),\\
& p_2(B^{(z)}(x_1, x_2)) \to B^{(z)}(p_3(x_1), p_\#(x_2)),\\
& p_2(C^{(z)}(x_1, x_2)) \to \#,\\
& p_3(C^{(z)}(x_1, x_2)) \to C^{(z)}(p_\#(x_1), p_\#(x_2)),\\
& p_\#(\#) \to \# \,\},
\end{aligned}$$

where P and Σ are the same as A. Consider the following data trees:

$$\begin{aligned}
t_1 &= A^{(1)}(B^{(2)}(C^{(3)}(\#, \#), \#), \#),\\
t_2 &= A^{(1)}(C^{(3)}(\#, \#), \#).
\end{aligned}$$

Figure 3.2 shows $Q(t_1)$, $Q(t_2)$, $Tr(t_1)$ and $Tr(t_2)$, the results of Q and Tr to t_1 and t_2. In fact, for any data tree t, the TA A 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 the Tr deletes a subtree of t to whose root Tr assigns p_2. That is, the deletion of a subtree by Tr does not violate the weak query preservation for Q. However, if we consider the 1-RQ $Q' = (A, \{p_1, p_2, p_3\})$ instead of Q and apply 1-WQP to Q', then 1-WQP answers "No" (because Tr does not weakly preserve Q'). $\Diamond$

To overcome the above mentioned problem, we construct $Q^F = (A^F, S^F)$ from $Q = (A, S)$ such that $Q(t) \neq \emptyset$ if and only if $\text{run}(A^F, t^-) \neq \emptyset$ and $Q^F(t) = \{\, v_i \mid (v_1, \ldots, v_n) \in Q(t),\ i \in [n] \,\}$. This modification can be done by augmenting each state p with a subset P of $\{p_1, \ldots, p_n\}$.

Let $Q = (A, S)$ be a given n-RQ. We can see that a DLT^{V} Tr weakly preserves Q if and only if for every $s \in S$, Tr weakly preserves $Q_s = (A, \{s\})$. Also $Q(t) = \bigcup_{s \in S} Q_s(t)$. So, it suffices to give an algorithm that decides if Tr weakly preserves $Q = (A, S)$ where $|S| = 1$.

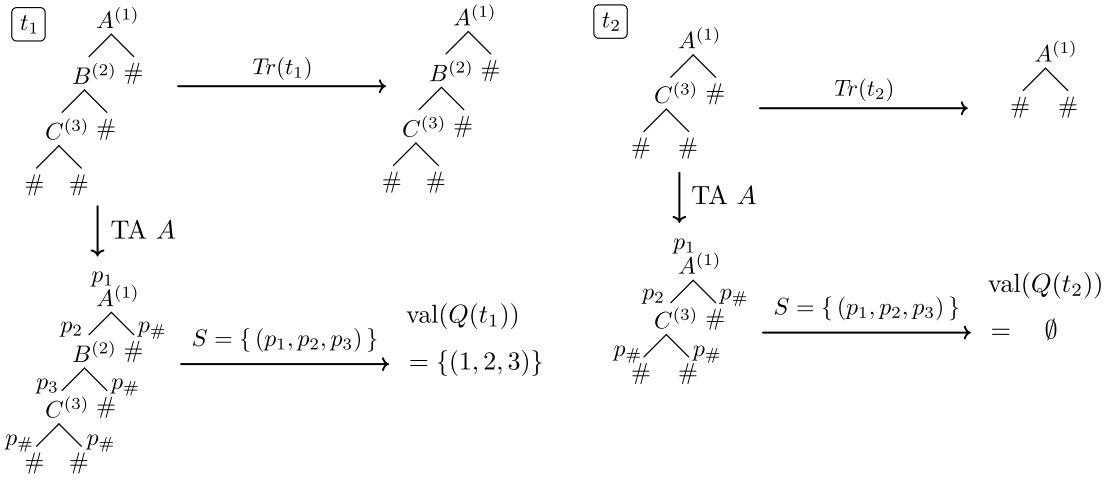


Figure 3.2: Results of Q and Tr on t_1 and t_2.

Algorithm n-WQP to Decide n-WeakQueryPres

Input: n-RQ $Q = (A, \{(p_1, \ldots, p_n)\})$, $A = (P_A,\ \Sigma,\ P_A^I,\ \delta_A)$ where $p_i \in P_A$ $(i \in [n])$, DLT$^{\text{V}}$ $Tr = (P_T, \Sigma, \Delta, \{p_T^0\}, \delta_T)$.

Output: If Tr weakly preserves Q, output "Yes," otherwise "No."

Step 1. Let $P_s = \{p_1, \ldots, p_n\} \subseteq P_A$. Construct $A^F = (P_A \times 2^{P_s}, \Sigma, P_A^I \times \{P_s\}, \delta^F)$ from A where δ^F is defined as follows: For $p \to a(p_A^1, \ldots, p_A^d) \in \delta_A$ and $P, P_1, \ldots, P_d \subseteq P_s$,

$$(p, P) \to a((p_A^1, P_1), \ldots, (p_A^d, P_d)) \in \delta^F$$

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

Step 2. Let $Q^F = (A^F, \{p_1, \ldots, p_n\} \times 2^{P_s})$. Decide whether Tr weakly preserves Q^F by 1-WQP.

Lemma 27. Given an n-RQ Q and a DLT$^{\text{V}}$ Tr, Tr weakly preserves Q if and only if Tr weakly preserves Q^F.

Proof (Sketch). Let $Q = (A, S)$ and $Q^F = (A^F, S^F)$. For $m \in \mathrm{run}(A^F, t^-)$, 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 itself). Especially, an initial state of A^F is a pair of an initial state of A and $\{p_1, \ldots, p_n\}$, meaning that each p_i ($i \in [n]$) should be used at least once in the input tree. Hence, $Q(t) \neq \emptyset$ if and only if $\mathrm{run}(A^F, t^-) \neq \emptyset$ and $Q^F(t) = \{\, v_i \mid (v_1, \ldots, v_n) \in Q(t), i \in [n] \,\}$. Thus, A^F is weakly preserved by Tr if and only if the original A is weakly preserved by Tr. □

If n is not fixed, the above algorithm will take exponential time in n. The following theorem gives the time-complexity for the weak query preservation problem. If the tuple size n of queries is constant, we can solve the weak query preservation problem for n-RQ under $\mathrm{DLT}^{\mathrm{V}}$ in polynomial time by using the algorithm for unary queries.

Theorem 28. n-WeakQueryPres is coNP-complete in general, in PTIME if n is fixed.

Proof. We show that the complement of n-WeakQueryPres is in NP. If Tr does not weakly preserve $Q = (A, S)$, there exist a tree t, positions $v_1, \ldots, v_n$, and a run $m \in \mathrm{run}(A, t^-)$ such that $(m(v_1), \ldots, m(v_n)) \in S$ and Tr deletes some v_i. We consider as a witness the n-paths from the root to $v_1, \ldots, v_n$ with partial runs of A and Tr on the paths. Given a witness, we can verify if some tree t includes the n-paths, the partial runs are consistent with A and Tr in t, and some v_i in t is deleted by Tr. The verification can be done in polynomial time if the size of a witness is polynomial. We show that there is a witness of polynomial size if Tr does not weakly preserve Q. We call the least common ancestor of v_i and v_j ($i \neq j$) their confluence position. A segment is the path between the adjacent confluent positions or between some v_i and the nearest confluent position. In Figure 3.3, a segment is depicted as a subpath between two bold nodes. We can choose a witness t_w such that the length of any segment is at most $(|Tr| + 1) \times |Q| + 1$ in the following way (see Figure 3.3): If the length of a segment is greater than $(|Tr|+1) \times |Q|+1$, there exist two different nodes u_1, u_2 on the segment such that at least one of u_1 and u_2 is not a confluence position, the same state is assigned to u_1 and u_2, and the same rule is applied at u_1 and u_2. Assume without loss of generality that u_1 is an ancestor of u_2. Then $t_w[u_1 \leftarrow t_w/u_2]$, that is, the tree obtained by short-cutting the subpath from u_1 to u_2 is also a witness. By iterating the above short-cut process, we can obtain a witness of which size is at most $2n \times ((|Tr| + 1) \times |Q| + 1) \times N$ where N is the maximal rank of symbols in Σ.

For the coNP-hardness, we show that 3UNSAT is polynomial time reducible to the weakly preservation problem. Let ϕ be an instance of 3UNSAT with

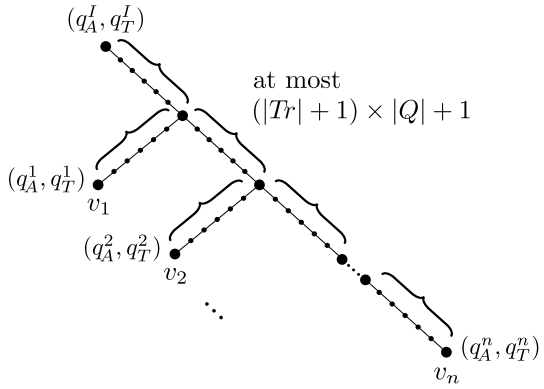


Figure 3.3: Witness t_w of the complement of n-WeakQueryPres.

k variables $x_1, \ldots, x_k$ and n clauses $C_1, \ldots, C_n$. First, we construct a query $Q = (A, S)$ as follows:

$$
\begin{aligned}
&Q = (A, S),\ A = (P, \Sigma, P_I, \delta),\ S = \{(P_{C_1}, \ldots, P_{C_n})\},\\
&P = \{\, P_r,\ P_{X_1}, \ldots, P_{X_k},\ P_{T_1}, \ldots, P_{T_k},\ P_{F_1}, \ldots, P_{F_k},\ P_{C_1}, \ldots, P_{C_n}\,\},\\
&\Sigma = \{\, r,\ X_1, \ldots, X_k,\ T_1, \ldots, T_k,\ F_1, \ldots, F_k,\ C_1, \ldots, C_n\,\},\\
&P_I = \{P_r\},
\end{aligned}
$$

where δ is the smallest set that satisfies the following conditions:

- $P_r \to r(P_{X_1}, \ldots, P_{X_k}) \in \delta$.
- For each $i \in [k]$, $P_{X_i} \to X_i(P_{T_i}) \in \delta$ and $P_{T_i} \to T_i \in \delta$, also $P_{X_i} \to X_i(P_{F_i}) \in \delta$ and $P_{F_i} \to F_i \in \delta$.
- $P_{T_i} \to T_i(P_{C_{t_1}}, \ldots, P_{C_{t_d}}) \in \delta$ where $\{C_{t_1}, \ldots, C_{t_d}\}$ is the set of all clauses containing x_i.
- $P_{F_i} \to F_i(P_{C_{f_1}}, \ldots, P_{C_{f_d}}) \in \delta$ where $\{C_{f_1}, \ldots, C_{f_d}\}$ is the set of all clauses containing $\overline{x_i}$.
- For each $j \in [n]$, $P_{C_j} \to C_j \in \delta$.

Figure 3.4 shows examples of a tree that is accepted by the TA A constructed from $\phi = (x_1 \vee \overline{x_2} \vee x_1) \wedge (\overline{x_1} \vee x_2 \vee x_4) \wedge (\overline{x_3} \vee x_4 \vee x_3)$.

Next, we construct a transducer Tr that deletes all nodes of a tree $t^- \in L(A)$ except the root node of t^-. For a given ϕ, Q and Tr can be constructed in linear time.

Figure 3.4: Examples of a tree accepted by the TA A constructed by $\phi = (x_1 \vee \overline{x_2} \vee x_1) \wedge (\overline{x_1} \vee x_2 \vee x_4) \wedge (\overline{x_3} \vee x_4 \vee x_3)$. (a) corresponds to $(x_1, x_2, x_3, x_4) =$ (T,T,T,T), and (b) corresponds to $(x_1, x_2, x_3, x_4) =$ (T,F,T,F), where T and F stand for the Boolean values true and false, respectively.

Lemma 29. ϕ is satisfiable if and only if there exists a tree $t \in L(A)$ such that for each clause $C_1, \ldots, C_n$ of ϕ, t contains the nodes $C_1, \ldots, C_n$ corresponding to the clauses.

Proof. ($\Rightarrow$) Suppose that ϕ has a satisfying assignment v. If a literal x_i on v makes a clause C_j true, the TA A constructed from ϕ accepts a tree containing $X_i(T_i(\ldots, C_j, \ldots))$ as its subtree. Whereas if a literal $\overline{x_i}$ on v makes a clause C_j true, the TA A accepts a tree containing $X_i(F_i(\ldots, C_j, \ldots))$ as its subtree. By the assumption, for each clause C_j there exists a literal on v that makes C_j true. Then, there exists a tree $t \in L(A)$ such that for each clause C_j, t contains $X_i(T_i(\ldots, C_j, \ldots))$ or $X_i(F_i(\ldots, C_j, \ldots))$ as its subtree. Therefore t contains the nodes $C_1, \ldots, C_n$ corresponding to the clauses $C_1, \ldots, C_n$ of ϕ.

($\Leftarrow$) Suppose that there exists a tree $t \in L(A)$ such that t contains the nodes $C_1, \ldots, C_n$ corresponding to the clauses $C_1, \ldots, C_n$ of ϕ. By the construction of A, each node C_j of t has a subtree $X_i(T_i(\ldots, C_j, \ldots))$ or $X_i(F_i(\ldots, C_j, \ldots))$ as its child. If C_j has $X_i(T_i(\ldots, C_j, \ldots))$, a clause C_j of ϕ is true when x_i is true, whereas if C_j has $X_i(F_i(\ldots, C_j, \ldots))$, a clause C_j of ϕ is true when x_i is false. By the assumption, each clause C_j is true under this assignment, and hence the assignment above satisfies ϕ. □

By applying Lemma 29 we show that ϕ is satisfiable if and only if Tr does not weakly preserve Q.

($\Rightarrow$) Suppose ϕ is satisfiable. By Lemma 29, there exists a tree $t \in L(A)$ that has nodes $C_1, \ldots, C_n$. Then a query result $Q(t)$ is not empty because $S = \{(P_{C_1}, \ldots, P_{C_n})\}$. Since Tr deletes all nodes except a root node, Tr does not weakly preserve Q if and only if $Q(t)$ is not empty. Thus, if ϕ is satisfiable, Tr does not weakly preserve Q.

($\Leftarrow$) By the assumption, $Q(t)$ is not empty since Tr does not preserve Q, i.e., a tree accepted by A has the nodes $C_1, \ldots, C_n$. Therefore ϕ is satisfiable by Lemma 29. □

3.5 Construction of Queries

If a transducer Tr weakly preserves a query Q, a query $Q' = (A', S')$ on target documents can be constructed by a type-inference algorithm that runs in polynomial time. The algorithm (called 1-WQC) works as follows: (1) Construct a TA A' from A such that $L(A') = T(L(A))$ where T is the DLT obtained from Tr by removing the manipulation of values, and (2) construct S' accordingly.

Algorithm 1-WQC to Construct Queries on Target Documents

Input: 1-RQ $Q = (A, S)$ where $A = (P_A, \Sigma, P_A^I, \delta_A)$, $S = \{p_1, \ldots, p_n\} \subseteq P_A$, DLT^{V} $Tr = (P_T, \Sigma, \Delta, \{p_T^0\}, \delta_T)$.

Output: A 1-RQ $Q' = (A'', S'')$ on target documents that satisfies Eq. (3.2).

Step 1. Construct a TA $A' = ((P_A \times P_T) \cup (\delta_A \times \delta_T \times C_{\text{all}}),\ \Delta,\ P_A^I \times \{p_T^0\},\ \delta')$ from Q and Tr where $C_{\text{all}} = \cup_{r \in \delta_T} \text{pos}(\text{rhs}(r))$, $\text{rhs}(r)$ is the right-hand side of $r \in \delta_T$, and δ' is defined as follows: For any rules $r_A = (p_A \to \sigma(p_A^1, \ldots, p_A^d)) \in \delta_A$ and $r_T = (p_T(\sigma^{(z)}(x_1, \ldots, x_d)) \to C^{(j \leftarrow z)}[p_T^1(x_1), \ldots, p_T^d(x_d)]) \in \delta_T$,

- for each $v \in \text{pos}(C)$ such that $\text{lab}(C, v) \in \Delta$,

$$M(v) \to \text{lab}(C, v)(M(v1), \ldots, M(vd_v)) \in \delta'$$

  where $d_v = \text{rk}(\text{lab}(C, v))$ and M is a mapping such that for each $v \in \text{pos}(C)$,
  - $M(v) = (p_A, p_T)$ if $v = \varepsilon$,
  - $M(v) = (p_A^i, p_T^i)$ if $\text{lab}(C, v) = x_i \in \mathcal{X}_d$, and
  - $M(v) = (r_A, r_T, v)$ otherwise;
- $(p_A, p_T) \to (p_A^i, p_T^i) \in \delta'$ if $C = x_i \in \mathcal{X}_d$.

Step 2. Construct a reduced TA without ε-rules equivalent to A'. Formally, let $\tilde{p} \Rightarrow_\varepsilon \tilde{p}'$ if and only if $\tilde{p} \to \tilde{p}' \in \delta'$, and $\Rightarrow_\varepsilon^*$ be the reflexive transitive closure of $\Rightarrow_\varepsilon$. For each rule $\tilde{p} \to \sigma(\tilde{p}_1, \ldots, \tilde{p}_d) \in \delta'$ and $\tilde{p}'_1, \ldots, \tilde{p}'_d \in \text{state}(A')$, add to δ' a new rule $\tilde{p} \to \sigma(\tilde{p}'_1, \ldots, \tilde{p}'_d)$ if for $i \in [d]$, $\tilde{p}_i \Rightarrow_\varepsilon^* \tilde{p}'_i$ and there is a rule with $\tilde{p}'_i$ in its left-hand side and some symbol in Δ in its right-hand side. Then, remove all

epsilon rules, useless states and transition rules of A'. Let A'' be the resulting TA.

Step 3. Compute $S'' = \bigcup_{i=1}^{n} S_{p_i}$ where S_{p_i} is the smallest subset of state(A'') satisfying the following conditions.

- $(r_A, r_T, v) \in S_{p_i}$ if r_A has p_i in its left-hand side, and the right-hand side of r_T is $C^{(j \leftarrow z)}[p_T^1(x_1), \ldots, p_T^d(x_d)])$ where $j \neq \varepsilon$ and $v = j$.
- $(p_i, p_T) \in S_{p_i}$ if $p_T(\sigma^{(z)}(x_1, \ldots, x_d)) \to C^{(j \leftarrow z)}[p_T^1(x_1), \ldots, p_T^d(x_d)]) \in \delta_T$ where $j = \varepsilon$.

We now prove the $\Leftarrow$ direction of Lemma 25 to show that Tr weakly preserves Q.

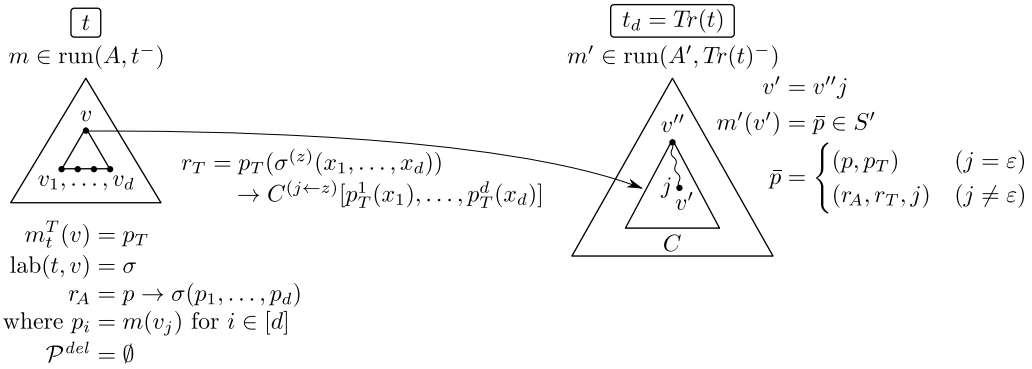


Figure 3.5: The relationship between an input tree t and a transformed tree t_d.

Proposition 30. Let Q be a 1-RQ and Tr be a DLT$^{\mathrm{V}}$. If $\mathcal{P}^{del} = \emptyset$ in Step 4 of the algorithm 1-WQP then Tr weakly preserves Q.

Proof (Sketch). Let $Q = (A, S)$ be the given query, and $Q' = (A', S')$ be the query constructed by 1-WQC from Q. Assume $\mathcal{P}^{del} = \emptyset$. We can say that for all $t \in \mathrm{dom}(Tr)$, $m \in \mathrm{run}(A, t^-)$, and $v \in Q(m, t)$, there exist $m' \in \mathrm{run}(A', Tr(t)^-)$ and $v' \in Q'(m', Tr(t))$ such that $\mathrm{val}(t, v) = \mathrm{val}(Tr(t), v')$. Let $t \in \mathrm{dom}(Tr)$ and $m \in \mathrm{run}(A, t^-)$. Any position v of t such that $m(v) \in S$ is not deleted by Tr because $\mathcal{P}^{del} = \emptyset$. From the construction of the query $Q' = (A', S')$ in 1-WQC, there is a run m' of Q' on $Tr(t)^-$ corresponding to m. As shown in Figure 3.5, for any position v of t such that $m(v) \in S$ and $m_t^T(v) = p_T$, a rule $r_T = p_T(\sigma^{(z)}(x_1, \ldots, x_d)) \to C^{(j \leftarrow z)}[p_T^1(x_1), \ldots, p_T^d(x_d)]$, which is not value-erasing, is applied to v. Then, the context C is output and the value of v is transferred to the position $v' = v''j$ where v'' is the root position of C in $Tr(t)$. Since the corresponding run m' assigns a state $\bar{p} \in S'$ to v'', Q' can retrieve the value of v from $Tr(t)$. Hence, we have $\mathrm{val}(Q(t)) \subseteq \mathrm{val}(Q'(Tr(t)))$. □

Lemma 31. If a DLT^{V} Tr weak preserves a 1-RQ Q, the query Q' satisfying Eq. (3.2) can be constructed by 1-WQC in polynomial time.

Proof. This lemma can be easily shown by Lemma 25 and Proposition 30. □

We can create n-ary queries on the transformed documents by using a natural variant of 1-WQC. Given a DLT^{V} Tr and an n-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 $S'' = \prod_{i=1}^{n} S_{p_i}$ in Step 3 while 1-WQC computes $S'' = \bigcup_{i=1}^{n} S_{p_i}$. The query Q' constructed by n-WQC is the minimum one as the following theorem claims.

Theorem 32. For any n-RQ Q'' such that Tr weakly preserves Q by Q'', and any $t \in \text{dom}(Tr)$, $\text{val}(Q'(Tr(t))) \subseteq \text{val}(Q''(Tr(t)))$.

Proof. We assume for contradiction that there are an n-RQ Q'' and a tree $t \in \text{dom}(Tr)$ such that $\text{val}(Q'(Tr(t))) \setminus \text{val}(Q''(Tr(t))) \neq \emptyset$. Let $Q = (A, S)$ and $Q' = (A', S')$. Let $\bar{\nu} = (\nu_1, \ldots, \nu_n) \in \text{val}(Q'(Tr(t))) \setminus \text{val}(Q''(Tr(t)))$. Since Tr weakly preserves Q by any of Q' and Q'', both $\text{val}(Q'(Tr(t)))$ and $\text{val}(Q''(Tr(t)))$ contains $\text{val}(Q(t))$, and thus $\bar{\nu} \notin \text{val}(Q(t))$. Because $\bar{\nu} \in \text{val}(Q'(Tr(t)))$, there exist a tuple $\bar{w} = (w_1, \ldots, w_n)$ of positions of $Tr(t)$ and a run $m_\nu \in \text{run}(A', Tr(t)^-)$ such that $(m_\nu(w_1), \ldots, m_\nu(w_n)) \in S'$ and $\text{val}(Tr(t), w_i) = \nu_i$ for each $i \in [n]$. By the definition of DLT^{V}, there is a tuple $\bar{v} = (v_1, \ldots, v_n)$ of positions of t such that for each $i \in [n]$, w_i is output when applying some rule of Tr to v_i and $\text{val}(t, v_i) = \text{val}(Tr(t), w_i)$. Since $\bar{\nu} \notin \text{val}(Q(t))$, however, there exists no run $m \in \text{run}(A, t^-)$ such that $(m(v_1), \ldots, m(v_n)) \in S$. The construction of Q' guarantees that for the run m_ν of A' for $Tr(t)^-$, there are a tree t_a, a run $m' \in \text{run}(A, t_a^-)$, a tuple $\bar{v}' = (v'_1, \ldots, v'_n)$ of positions of t_a such that $Tr(t_a)^- = Tr(t)^-$, $(m'(v'_1), \ldots, m'(v'_n)) \in S$, and for each $i \in [n]$, $\text{val}(t_a, v'_i) = \text{val}(Tr(t_a), w_i)$ and w_i is output when applying some rule of Tr to v'_i. Let $\bar{\nu}_a = (\text{val}(Tr(t_a), w_1), \ldots, \text{val}(Tr(t_a), w_n))$, and then $\bar{\nu}_a$ is in both $\text{val}(Q(t_a))$ and $\text{val}(Q'(Tr(t_a))$. We assume without loss of generality that each of t and t_a is value-unduplicated. Then, $\bar{w}$ is the only tuple of positions of $Tr(t)$ (resp. $Tr(t_a)$) from which Q' extracts $\bar{\nu}$ (resp. $\bar{\nu}_a$). It is also clear that $\bar{v}'$ is the only tuple of positions of t_a from which Q extracts $\bar{\nu}_a$. We have $\bar{\nu} \notin \text{val}(Q''(Tr(t)))$ and thus $\bar{w} \notin Q''(Tr(t))$. Since $Tr(t_a)^- = Tr(t)^-$, $\bar{w} \notin Q''(Tr(t_a))$. Thus, $\bar{\nu}_a \notin \text{val}(Q''(Tr(t_a)))$ but $\bar{\nu}_a \in \text{val}(Q(t_a))$. This contradicts the weak preservation by Q''. □

Now we can say that the query Q' satisfying the condition of the query preservation (Eq. (3.1)) can be constructed by the algorithm n-WQC if Tr preserves Q.

Theorem 33. If a DLT^{V} Tr preserves an n-RQ Q, the query Q' satisfying Eq. (3.1) can be constructed by n-WQC. If n is fixed, n-WQC runs in polynomial time.

Proof. Recall that the query preservation is a special case of the weak query preservation. By Theorem 32, the query Q' constructed by n-WQC is the minimum one that satisfies the condition of the weak query preservation (Eq. (3.2)). Thus the query Q' satisfies Eq. (3.1) if Tr preserves Q. □

3.6 Decidability of Query Preservation

In this section, we provide an algorithm n-QP that decides the (strong) query preservation.

3.6.1 One-by-Run Property

We define a property, called the One-by-Run property, that helps us to reduce the query preservation problem to the equivalence problem of TAs.

Definition 34. An n-RQ $Q = (A, S)$ has the *One-by-Run property* if (1) there exist pairwise disjoint subsets $S_1, \ldots, S_n$ of state(A) satisfying $S = S_1 \times \cdots \times S_n$, and (2) for every $t^- \in L(A)$, $m \in \mathrm{run}(A, t^-)$, and $i \in [n]$, there exists exactly one $v \in \mathrm{pos}(t^-)$ such that $m(v) \in S_i$.

Given $Q = (A, \{s\})$, we can construct a query that has the One-by-Run property and is equivalent to Q. Let $A = (P_A, \Sigma, P_A^I, \delta_A)$, $s = (p_1^s, \ldots, p_n^s)$, and $P_s = \{p_1^s, \ldots, p_n^s\}$. Construct an n-RQ $\widetilde{Q} = (\widetilde{A}, \widetilde{S})$ from A as follows. Let $\widetilde{A} = (\widetilde{P}, \Sigma, \widetilde{P_I}, \widetilde{\delta})$ where $\widetilde{P} = P_A \times 2^{P_s} \times ([n] \cup \{0\})$, $\widetilde{P_I} = P_A^I \times \{P_s\} \times ([n] \cup \{0\})$. $\widetilde{\delta}$ is the set of rules such that $(p, P, f) \to \sigma((p_1, P_1, f_1), \ldots, (p_d, P_d, f_d)) \in \widetilde{\delta}$ with $f, f_1, \ldots, f_d \in [n] \cup \{0\}$ if and only if there exists a rule $p \to \sigma(p_1, \ldots, p_d) \in \delta_A$ and for all $i \in [n]$, the following conditions hold:

- There exists at most one $j \in [d]$ satisfying $p_i^s \in P_j$.
- If $p_i^s \in P_j$ for some $j \in [d]$, then $p_i^s \in P$ and $f \neq i$.
- If $p_i^s \notin P_j$ for all $j \in [d]$ and $p = p_i^s$, then $p_i^s \in P$ and $f = i$, or $p_i^s \notin P$ and $f \neq i$.
- If $p_i^s \notin P_j$ for all $j \in [d]$ and $p \neq p_i^s$, then $p_i^s \notin P$ and $f \neq i$.

Finally, remove useless states and rules of $\widetilde{A}$ and let $\widetilde{S} = S_1 \times \cdots \times S_n$ where $S_i = \{\, (p_i^s, P_i, i) \mid P_i \subseteq P_s \,\}$.

Let $\widetilde{Q} = (\widetilde{A}, \widetilde{S})$ be the query obtained from $Q = (A, \{(p_1^s, \ldots, p_n^s)\})$ by the construction above. Let t be a data tree such that $t^- \in L(A)$ and v be a node of t^-. Let $\widetilde{m}$ be a run of $\widetilde{A}$ on t^-, $\widetilde{m}(v) = (p, P, f)$ and $\widetilde{Q}(\widetilde{m}, t) = \{(v_1, \ldots, v_n)\}$.

The first component p of $\widetilde{m}(v)$ is a state of A. The third component f of $\widetilde{m}(v)$ is i if and only if $v = v_i$. When $\widetilde{m}(v) = (p_i^s, P, i)$, p_i^s is said to be *selected* at v by $\widetilde{A}$ according to $\widetilde{m}$. If we regard $\widetilde{A}$ moves according to $\widetilde{m}$ in the bottom-up manner, the second component P is the set consisting of the states which have already been selected at some descendants of v. For example, consider a rule $r = (p_4^s, P_4, 4) \to \sigma((p_1^s, P_1, 1), (p_2, P_2, 0), (p_3^s, P_3, 0))$ of $\widetilde{A}$ and let $\widetilde{m}(v) = (p_4^s, P_4, 4)$. By the construction of $\widetilde{A}$, $p_1^s \in P_1$ and $P_4 = P_1 \cup P_2 \cup P_3 \cup \{p_4^s\}$. If the rule r is applied at v having children v_1, v_2 and v_3, p_1^s is selected at v_1 and p_4^s is selected at v. Now we prove the correctness of the above construction.

Lemma 35. Let $\widetilde{Q} = (\widetilde{A}, \widetilde{S})$ be the query constructed from $Q = (A, \{s\})$ by the construction above. The query $\widetilde{Q}$ has the One-by-Run property and $\widetilde{Q}$ is equivalent to Q.

Proof. First, we show that $\widetilde{Q} = (\widetilde{A}, \widetilde{S})$ has the One-by-Run property. Let $s = (p_1^s, \ldots, p_n^s)$ and $P_s = \{p_1^s, \ldots, p_n^s\}$. It is obvious that $\widetilde{S}$ satisfies the condition (1) of the One-by-Run property by the construction of $\widetilde{S}$. The second condition (2) can be derived from the following property (†), which can be shown by induction on $N = |\mathrm{pos}(t^-/v)|$:

$$\begin{aligned} & \widetilde{m}(v) = (p, P, f) \text{ with } p_i^s \in P \\ \iff \quad & \exists! w \geq v.\ \widetilde{m}(w) \in S_i = \{\, (p_i^s, P_i, i) \mid P_i \subseteq P_s \,\}. \end{aligned} \tag{†}$$

Because the second component of initial states of $\widetilde{A}$ is $P_s = \{p_1^s, \ldots, p_n^s\}$, the property (†) implies that there exists exactly one node v_i of t^- such that $\widetilde{m}(v_i) = (p_i^s, P, i)$ for any run $\widetilde{m}$ of $\widetilde{A}$, and $i \in [n]$. Hence, $\widetilde{Q}$ satisfies the condition (2) of the One-by-Run property.

Next, we show that $\mathrm{val}(\widetilde{Q}(t)) = \mathrm{val}(Q(t))$ for every data tree t such that $t^- \in L(A)$. It is obvious that $\mathrm{val}(\widetilde{Q}(t)) = \mathrm{val}(Q(t))$ holds if $\widetilde{Q}(t) = Q(t)$. Let $\widetilde{m}$ be a run of $\widetilde{A}$ on a tree $t^- \in L(A)$. For $i \in [n]$ and a node $v_i \in \mathrm{pos}(t)$ satisfying $\widetilde{m}(v_i) = (p_i^s, P_i, i) \in S_i$, there exists a run m of A such that $m(v_i) = p_i^s$ by the construction of $\widetilde{Q}$. Hence $\widetilde{Q}(\widetilde{m}, t) \subseteq Q(m, t)$. For each run $\widetilde{m}$ of $\widetilde{A}$, $\widetilde{Q}(\widetilde{m}, t) \subseteq Q(m, t)$ holds. Therefore $\widetilde{Q}(t) \subseteq Q(t)$. Next, let m be a run of A on t^- and assume $(v_1, \ldots, v_n) \in Q(m, t)$. Let $\widetilde{m}$ be the mapping obtained from m and $(v_1, \ldots, v_n)$ by $\widetilde{m}(v) = (p, P, f)$ where $p = m(v)$, $f = i$ if $v = v_i$, and $f = 0$ otherwise and P is determined in the bottom-up way. This $\widetilde{m}$ is a run of $\widetilde{A}$ on t^- by the construction of $\widetilde{A}$, and $\widetilde{Q}(\widetilde{m}, t) = \{(v_1, \ldots, v_n)\}$ by the condition (2) of the One-by-Run property. □

Thus, any n-RQ $Q = (A, \{s_1, \ldots, s_k\})$ can be represented as the union of n-RQs $\widetilde{Q}^1, \ldots, \widetilde{Q}^k$ such that each $\widetilde{Q}^i$ is equivalent to $(A, \{s_i\})$ and has the One-by-Run property.

3.6.2 Algorithm for Query Preservation

We explain the idea of our algorithm n-QP. Let Tr and $Q = (A, S)$ be a DLT$^{\mathrm{V}}$ and an n-RQ given as inputs. At the first step, n-QP turns Q into the union of n-RQs having the One-by-Run property. For simplicity to explain the idea, let us assume here that the obtained query is an n-RQ $\widetilde{Q} = (\widetilde{A}, \widetilde{S})$. The query can be represented as a set of "marked" trees. For a tree t and a run m of $\widetilde{A}$ on t^-, we say that τ is a marked tree for t corresponding to m if τ is obtained from t by replacing $\mathrm{lab}(t, v_i)$ with $(i, \mathrm{lab}(t, v_i))$ for each $i \in [n]$ where $(m(v_1), \ldots, m(v_n)) \in \widetilde{S}$. By the One-by-Run property, each marked tree for t represents exactly one tuple of positions selected by $\widetilde{Q}$. Let A_{mk} be a TA that recognizes the set of all the marked trees for $\widetilde{Q}$. The algorithm n-QP decides if $L(A_{\mathrm{mk}}) = T_{\mathrm{mk}}^{-1}(T_{\mathrm{mk}}(L(A_{\mathrm{mk}})) \cap L(A_{\mathrm{wp}})$ where T_{mk} is a marked version of Tr that ignores data values, and A_{wp} recognizes the set of candidate marked trees in which any marked position is deleted by Tr. Note that the set of trees in the right-hand side of the above equality can be captured by a TA, and so n-QP reduces the query preservation problem to the equivalence problem of TAs.

Algorithm n-QP to Decide n-QueryPres

Input: n-RQ $Q = (A, \{s_1, \ldots, s_k\})$, DLT$^{\mathrm{V}}$ $Tr = (P_T, \Sigma, \Delta, \{p_T^0\}, \delta_T)$.

Output: If Tr preserves Q, output "Yes," otherwise "No."

Step 1. For each s_i, construct $\widetilde{Q}^i$ equivalent to $(A, \{s_i\})$ having the One-by-Run property.

Step 2. For each $\widetilde{Q}^j = (A^j, S^j)$ where $A^j = (P^j, \Sigma, P_I^j, \delta^j)$ and $S^j = S_1^j \times \cdots \times S_n^j$, construct TA $A_{\mathrm{mk}}^j = (P^j, \Sigma \cup ([n] \times \Sigma), P_I^j, \delta_{Aj}^{\mathrm{mk}})$ from A^j where $\delta_{Aj}^{\mathrm{mk}}$ is defined as follows: For each rule of the form $p \to \sigma(p_1, \ldots, p_d) \in \delta^j$, if p is in S_i^j then $p \to (i, \sigma)(p_1, \ldots, p_d) \in \delta_{Aj}^{\mathrm{mk}}$, otherwise $p \to \sigma(p_1, \ldots, p_d) \in \delta_{Aj}^{\mathrm{mk}}$. Then, construct the TA A_{mk} as the union TA of $A_{\mathrm{mk}}^1, \ldots, A_{\mathrm{mk}}^n$.

Step 3. Construct the DLT $T_{\mathrm{mk}} = (P_T, \Sigma \cup ([n] \times \Sigma), \Delta \cup ([n] \times \Delta), \{p_T^0\}, \delta_T^{\mathrm{mk}})$ where δ_T^{mk} is the smallest set satisfying the following conditions: For each $i \in [n]$ and for each rule $p_T(\sigma^{(z)}(x_1, \ldots, x_d)) \to C^{(j \leftarrow z)}[p_T^1(x_1), \ldots, p_T^d(x_d)] \in \delta_T$ that is not a value-erasing rule, let $p_T(\sigma(x_1, \ldots, x_d)) \to C[p_T^1(x_1), \ldots, p_T^d(x_d)] \in \delta_T^{\mathrm{mk}}$, and $p_T((i, \sigma)(x_1, \ldots, x_d)) \to \overline{C}[p_T^1(x_1), \ldots, p_T^d(x_d)] \in \delta_T^{\mathrm{mk}}$ where $\mathrm{lab}(\overline{C}, j) = (i, \mathrm{lab}(C, j))$, and for each $v \in \mathrm{pos}(C)$ satisfying $v \neq j$, $\mathrm{lab}(\overline{C}, j) = \mathrm{lab}(C, j)$.

Step 4. Construct a TA A'_{mk} such that

$$L(A'_{\mathrm{mk}}) = T_{\mathrm{mk}}^{-1}(T_{\mathrm{mk}}(L(A_{\mathrm{mk}}))),$$

by type inference and inverse type inference [10], where $T_{\mathrm{mk}}^{-1}(L) = \{\, t \mid T_{\mathrm{mk}}(t) \in L \,\}$.

Step 5. Construct the TA $A_{\mathrm{wp}} = (P'', \Sigma \cup ([n] \times \Sigma), P_I', \delta'' \cup \delta_{\mathrm{wp}})$ from Tr and Q, as the marked version of $A'' = (P'', \Sigma, P_I'', \delta'')$ constructed in 1-WQP, where δ_{wp} is defined as follows: For each rule $(p, p_T) \to \sigma((p_A^1, p_T^1), \ldots, (p_A^d, p_T^d))$ of A'', if $p_T \neq \bot$ and Tr has no value-erasing rule with p_T in its left-hand side and σ in its right-hand side, then add the rule $(p, p_T) \to (i, \sigma)((p_A^1, p_T^1), \ldots, (p_A^d, p_T^d))$ to δ_{wp}.

Step 6. If $L(A_{\mathrm{mk}}) = L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})$, output "Yes," otherwise "No."

Example 7. Recall Q and Tr in Example 5. By steps 1–4 of the above algorithm n-QP,

$$\begin{aligned} L(A_{\mathrm{mk}}) &= \{f((1,a),a), g(a,(1,a))\}, \\ L(A'_{\mathrm{mk}}) &= \{f((1,a),a), g((1,a),a), f(a,(1,a)), g(a,(1,a))\} \\ &= L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}}). \end{aligned}$$

Hence, $L(A_{\mathrm{mk}}) \subsetneq L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})$ holds and the algorithm answers "No." ◊

3.6.3 Correctness

To show the correctness of n-QP, we prepare some functions for marked trees. We use the functions lab and val for any marked trees and their positions as natural extensions of the functions for data trees and their positions. By the assumption, if $\tau \in L(A_{\mathrm{mk}})$ then for each $i \in [n]$ there exist a unique $v_i \in \mathrm{pos}(\tau)$ and a unique $\nu_i \in \mathbb{N}$, such that $\mathrm{lab}(\tau, v_i) = (i, \sigma)$ and $\mathrm{val}(\tau, v_i) = \nu_i$ hold, and we denote $\mathrm{val}(\tau) = (\nu_1, \ldots, \nu_n)$. For instance, if $\tau = f^{(3)}((2,a)^{(4)}, (1,a)^{(5)})$ then $\mathrm{val}(\tau) = (5, 4)$. We denote the function that removes all marks from a marked tree τ as $I^- : \mathcal{T}^{(\mathbb{N})}_{\Sigma \cup ([n] \times \Sigma)} \to \mathcal{T}^{(\mathbb{N})}_{\Sigma}$. For any v of τ satisfying $\mathrm{lab}(\tau, v) = (i, \sigma)$, $\mathrm{lab}(I^-(\tau), v) = \sigma$.

Before showing the correctness of n-QP, we show the following lemma for properties of (marked) trees accepted by A_{mk}.

Lemma 36. Let $Q = (A, S)$ be an n-RQ. For any t where $t^- \in L(A)$, the following condition holds.

$$\begin{aligned} &(\nu_1, \ldots, \nu_n) \in \mathrm{val}(Q(t)) \Longleftrightarrow \\ &\exists \tau.\ (\, \tau^- \in L(A_{\mathrm{mk}}) \wedge I^-(\tau) = t \wedge \mathrm{val}(\tau) = (\nu_1, \ldots, \nu_n)\,). \end{aligned} \tag{3.3}$$

Proof. It is obvious by the construction of A^j_{mk}. □

We show the correctness of n-QP by using Lemma 36.

Lemma 37. A DLT$^{\mathrm{V}}$ Tr preserves an n-RQ Q if and only if $L(A_{\mathrm{mk}}) = L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})$ in Step 6 of the algorithm n-QP.

Proof. ($\Rightarrow$) We show that Tr does not preserve $Q = (A, S)$ if $L(A_{\mathrm{mk}}) \neq L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})$.

(1) Assume that $L(A_{\mathrm{mk}}) \setminus (L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})) \neq \emptyset$. Let τ be a tree such that $\tau^- \in L(A_{\mathrm{mk}}) \setminus (L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}}))$. We can assume that τ is value-unduplicated. By Lemma 36, we have $\mathrm{val}(\tau) \in Q(I^-(\tau))$. We also have $\tau^- \in L(A'_{\mathrm{mk}})$ because $\tau^- \in T^{-1}_{\mathrm{mk}}(T_{\mathrm{mk}}(\tau^-)) \subseteq L(A'_{\mathrm{mk}})$ whenever $\tau^- \in L(A_{\mathrm{mk}}) \subseteq \mathrm{dom}(T_{\mathrm{mk}})$. Thus, $\tau^- \notin L(A_{\mathrm{wp}})$. It follows from the construction of A_{wp} that there exists a position $v \in \mathrm{pos}(\tau)$ such that $\mathrm{lab}(\tau, v) = (i, \sigma)$ for some $i \in [n]$ and $\sigma \in \Sigma$, and Tr ignores v or applies a value-erasing rule to v in $I^-(\tau)$. In addition, since τ is value-unduplicated, any position of $I^-(\tau)$ other than v does not have $\mathrm{val}(\tau, v)$, and thus $\mathrm{val}(\tau, v)$ is not contained in $Tr(I^-(\tau))$. Hence, for any n-RQ Q', $Q'(Tr(I^-(\tau)))$ does not contain $\mathrm{val}(\tau)$, the i th element of which is $\mathrm{val}(\tau, v)$, and thus $Q(I^-(\tau)) \neq Q'(Tr(I^-(\tau)))$. Note that in this case, Tr does not weakly preserve Q.

(2) Assume that $(L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})) \setminus L(A_{\mathrm{mk}}) \neq \emptyset$. Let τ be a tree such that $\tau^- \in (L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})) \setminus L(A_{\mathrm{mk}})$. We can assume that τ is value-unduplicated. From the construction of A'_{mk}, there exists a tree τ_a such that $\tau_a^- \in L(A_{\mathrm{mk}})$ and $T_{\mathrm{mk}}(\tau_a^-) = T_{\mathrm{mk}}(\tau^-)$. We can assume without loss of generality $I^-(\tau_a) \neq I^-(\tau)$, $\mathrm{val}(\tau_a) = \mathrm{val}(\tau)$, and that τ_a is value-unduplicated. Then, we have $\mathrm{val}(\tau) \in Q(I^-(\tau_a))$ by Lemma 36. Since $\tau^- \notin L(A_{\mathrm{mk}})$ and τ is value-unduplicated, $\mathrm{val}(\tau) \neq \mathrm{val}(\widetilde{\tau})$ for any tree $\widetilde{\tau}$ such that $\widetilde{\tau}^- \in L(A_{\mathrm{mk}})$ and $I^-(\widetilde{\tau}) = I^-(\tau)$, and so we have $\mathrm{val}(\tau) \notin Q(I^-(\tau))$. Next, let us observe $Tr(I^-(\tau))$ and $Tr(I^-(\tau_a))$. We have $(Tr(I^-(\tau)))^- = (Tr(I^-(\tau_a)))^-$ because $T_{\mathrm{mk}}(\tau^-) = T_{\mathrm{mk}}(\tau_a^-)$. Let $\mathrm{val}(\tau) = (\nu_1, \ldots, \nu_n)$. There is one and only one n-tuple $\bar{v} = (v_1, \ldots, v_n)$ of positions of $Tr(I^-(\tau))$ such that $\mathrm{val}(Tr(I^-(\tau)), v_i) = \nu_i$ for each $i \in [n]$. Also in $Tr(I^-(\tau_a))$, $\bar{v}$ is the only one tuple of positions such that $\mathrm{val}(Tr(I^-(\tau_a)), v_i) = \nu_i$ for each $i \in [n]$. Thus, for any n-RQ Q', $\mathrm{val}(\tau) \in Q'(Tr(I^-(\tau)))$ if and only if $\mathrm{val}(\tau) \in Q'(Tr(I^-(\tau_a)))$. Here, we show that Tr does not preserve Q by any n-RQ. For contradiction, we assume that some Q' satisfies $\mathrm{val}(Q(t)) = \mathrm{val}(Q'(Tr(t)))$ for every $t \in \mathrm{dom}(Tr)$. We have $\mathrm{val}(\tau) \in Q'(Tr(I^-(\tau_a)))$ because $\mathrm{val}(\tau) \in Q(I^-(\tau_a))$, and then $\mathrm{val}(\tau) \in Q'(Tr(I^-(\tau)))$. However, $\mathrm{val}(\tau) \notin Q(I^-(\tau))$ as stated above. This is a contradiction.

($\Leftarrow$) We show that $L(A_{\mathrm{mk}}) \neq L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})$ if Tr does not preserve $Q = (A, S)$.

(1) Assume that Tr does not weakly preserve Q. There is a tree t and positions $v_1, \ldots, v_n$ of t such that $(v_1, \ldots, v_n) \in Q(t)$, and some v_i ($i \in [n]$) is ignored or is applied a value-erasing rule to by Tr. Consider the marked tree τ such

that $I^-(\tau) = t$ and $\mathrm{lab}(\tau, v_i) = (i, \mathrm{lab}(t, v_i))$ for each $i \in [n]$, and then $\tau^- \in L(A_{\mathrm{mk}}) \setminus L(A_{\mathrm{wp}})$. Hence $\tau^- \in L(A_{\mathrm{mk}}) \setminus (L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}}))$.

(2) Assume that Tr weakly preserves Q but does not strongly. We consider the query Q' constructed by n-WQC, by which Tr weakly preserves Q. Since Tr does not strongly preserve Q, there is a tree t such that $\mathrm{val}(Q(t)) \subsetneq \mathrm{val}(Q'(Tr(t)))$. Consider $\bar{\nu} \in \mathrm{val}(Q'(Tr(t))) \setminus \mathrm{val}(Q(t))$. Since $\bar{\nu} \in \mathrm{val}(Q'(Tr(t)))$, there are a run $m' \in \mathrm{run}(A', Tr(t)^-)$ and positions $w_1, \ldots, w_n$ of $Tr(t)$ such that $(m'(w_1), \ldots, m'(w_n)) \in S'$ and $(\mathrm{val}(Tr(t), w_1), \ldots, \mathrm{val}(Tr(t), w_n)) = \bar{\nu}$. By the definition of $\mathrm{DLT}^{\mathrm{V}}$, since $\mathrm{val}(Tr(t), v_i) \neq nil$ for all $i \in [n]$, there are positions $v_1, \ldots, v_n$ of t such that $(\mathrm{val}(t, v_1), \ldots, \mathrm{val}(t, v_n)) = \bar{\nu}$. Moreover, it follows that any v_i ($i \in [n]$) is not deleted by Tr, that is, v_i is neither ignored nor applied a value-erasing rule to by Tr. Since $\bar{\nu} \notin \mathrm{val}(Q(t))$, there is no run $m \in \mathrm{run}(A, t^-)$ such that $(m(v_1), \ldots, m(v_n)) \in S$. The construction of Q' guarantees that for the run m' of A' for $Tr(t)^-$, there are a tree t_a, a run $m' \in \mathrm{run}(A, t_a^-)$, positions $v'_1, \ldots, v'_n$ of t_a such that $Tr(t_a)^- = Tr(t)^-$, $(m'(v'_1), \ldots, m'(v'_n)) \in S$, and for each $i \in [n]$, w_i is output when applying some rule of Tr to v'_i, $\mathrm{val}(t_a, v'_i) = \mathrm{val}(Tr(t_a), w_i)$. Here, we consider the marked tree τ such that $I^-(\tau) = t$ and $\mathrm{lab}(\tau, v_i) = (i, \mathrm{lab}(t, v_i))$ for each $i \in [n]$. Similarly, let τ_a be the marked tree of t_a. Then, we have $\tau^- \notin L(A_{\mathrm{mk}})$ and $\tau_a^- \in L(A_{\mathrm{mk}})$ by Lemma 36. We also have $T_{\mathrm{mk}}(\tau^-) = T_{\mathrm{mk}}(\tau_a^-)$, and thus $\tau^- \in T_{\mathrm{mk}}^{-1}(T_{\mathrm{mk}}(\tau_a^-)) \subseteq L(A'_{\mathrm{mk}})$. Moreover, $\tau^- \in L(A_{\mathrm{wp}})$ because any v_i in $I^-(\tau)$ is not deleted by Tr. Therefore, $\tau^- \in (L(A'_{\mathrm{mk}}) \cap L(A_{\mathrm{wp}})) \setminus L(A_{\mathrm{mk}})$. □

Theorem 38. n-QueryPres is in 2-EXPTIME in general, EXPTIME-hard even if Q is a unary query.

Proof. By Lemma 35, any n-RQ $Q = (A, \{s_1, \ldots, s_n\})$ can be divided into n of n-RQ $\widetilde{Q}^1, \ldots, \widetilde{Q}^n$. Furthermore by applying Lemma 37, we can decide whether Tr preserves Q for any $\widetilde{Q}^j$ ($j \in [n]$) and Tr. Hence n-QueryPres for any n-RQ Q and Tr is also decidable. We can show n-QP runs in double-exponential time as follows. We decompose a given query in Step 1, which can be done in exponential time, and we test the equivalence for TAs in Step 6 for each query decomposed. It is known that the equivalence problem for TAs is EXPTIME-complete (see, e.g., [10]), so we can compute Step 6 in exponential time. Thus n-QueryPres is in 2-EXPTIME.

For the EXPTIME-hardness, we give a reduction from the inclusion problem for tree automata, which is known to be EXPTIME-complete. Consider two TAs $A_1 = (P_1, \Sigma, P_1^I, \delta_1)$, $A_2 = (P_2, \Sigma, P_2^I, \delta_2)$, and let $A'_1 = (P'_1, \Sigma, P_1'^I, \delta'_1)$ be a copy of A_1, where P_1, P_2, and P'_1 are disjoint sets. Let the query be the

1-RQ $Q = (A_q, P_1 \cup P_2)$ where

$$A_q = (P_1 \cup P_2 \cup P_1' \cup \{p_q\},\ \Sigma \cup \{a, b\},\ \{p_q\},\ \delta_1 \cup \delta_2 \cup \delta_1' \cup \delta_q),$$
$$\delta_q = \{\, p_q \to a(p_I) \mid p_I \in P_1^I \,\} \cup \{\, p_q \to b(p_I) \mid p_I \in P_1'^I \cup P_2^I \,\},$$

$a, b \notin \Sigma$, $\mathrm{rk}(a) = \mathrm{rk}(b) = 1$, and $p_q \notin P_1 \cup P_2 \cup P_1'$. We let Tr be a $\mathrm{DLT}^{\mathrm{V}}$ that only deletes any node labeled a or b, e.g., $Tr(a(b(c))) = c$. We show that Tr preserves Q if and only if $L(A_1) \subseteq L(A_2)$, i.e., any tree $t_1 \in L(A_1) \setminus L(A_2)$ does not exist. Note that the above Tr weakly preserves Q, because Tr does not deletes any node retrieved by Q. Consider the four cases below:

1. If $t_1 \in L(A_1) \setminus L(A_2)$: $\mathrm{val}(Q(a(t_1))) = \{\, \mathrm{val}(t, v) \mid v \in \mathrm{node}(t) \,\}$, $\mathrm{val}(Q(b(t_1))) = \emptyset$, and $Tr(a(t_1)) = Tr(b(t_1))$.
2. If $t_1 \in L(A_1) \cap L(A_2)$: $\mathrm{val}(Q(a(t_1))) = \mathrm{val}(Q(b(t_1)))$, and $Tr(a(t_1)) = Tr(b(t_1))$.
3. If $t_1 \in L(A_2) \setminus L(A_1)$: $a(t_1) \notin L(A)$, and any $t' \neq b(t_1)$ satisfying $Tr(b(t_1)) = Tr(t')$ does not exist in $L(A)$.
4. If $t_1 \notin L(A_1) \cup L(A_2)$: $a(t_1), b(t_1) \notin L(A)$.

Suppose Tr preserves Q. In the case 1, if there exists $t_1 \in L(A_1) \setminus L(A_2)$, we have $\mathrm{val}(Q(a(t_1))) \neq \mathrm{val}(Q(b(t_1)))$ and $Tr(a(t_1)) = Tr(b(t_1))$, which violate the condition (3.1) of the query preservation. Hence any tree $t_1 \in L(A_1) \setminus L(A_2)$ does not exist. The other cases do not violate the condition (3.1). Conversely, if any tree $t_1 \in L(A_1) \setminus L(A_2)$ does not exist, Tr does not preserve Q due to the same reason above. Thus Tr preserves Q if and only if $L(A_1) \subseteq L(A_2)$.

Note that the above Q is a 1-RQ, and hence we obtain EXPTIME-hardness even if Q is a unary query. □

Theorem 39. n-QueryPres is EXPTIME-complete if n is fixed.

Proof. By Theorem 38, n-QueryPres is EXPTIME-hard, and we can compute Step 1 of the algorithm n-QP in polynomial time if n is fixed. □

Note that the above algorithm can work for more powerful classes of transducers that preserve regularity under those transducers and inverse applications of them. We discuss an extension of views in Section 3.7.

3.7 Extension of Views

We have seen in the previous sections, the problem of deciding whether Tr preserves Q or not is decidable, where Tr is a deterministic linear top-down data tree transducer. In this section, we show that the class of tree transductions can be extended to a more expressive class L whose transductions are single-valued and preserve regularity. Note that a tree transduction $Tr \in L$ *preserves regularity* if $Tr(L(A))$ and $Tr^{-1}(L(A))$ are regular for any TA A, i.e., the sets are recognized by some TAs. Whereas Tr is *single-valued* (or functional) if Tr has exactly one output tree for each input tree in $\mathrm{dom}(Tr)$.

The class of single-valued extended linear top-down tree transducer with regular look-ahead (s-ELT^R) is one of the more expressive class than the class of DLT whose transductions are single-valued and preserve regularity [4, 26, 39]. The transduction of s-ELT^R has rules whose left-hand side has multiple symbols instead of a symbol, and the transduction has the ability to inspect a regular property for the subtree of an input tree. These are why s-ELT^R is more expressive than DLT$^{\mathrm{V}}$.

Our proof of the correctness of the algorithm n-QP (Theorem 38) requires only the single-valuedness and the regularity of tree transducers, and thus we can easily extend our decidability result to the s-ELT$^{\mathrm{VR}}$, which is a natural extension of s-ELT^R running on data trees.

Theorem 40. Given an n-RQ Q and an s-ELT$^{\mathrm{VR}}$ Tr, the problem of deciding whether Tr preserves Q is decidable.

3.8 Conclusion of the Chapter

We have studied the decidability problems of the weak query preservation and the strong query preservation for deterministic linear top-down data tree transducers and run-based n-ary queries. We have shown 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 have also shown the strong query preservation problem is in 2-EXPTIME in general, EXPTIME-hard even if n is fixed. This decidability results have been extended to the class of single-valued extended linear top-down data tree transducers, which is a more expressive class than the class of deterministic linear top-down data tree transducers. We have provided an efficient algorithm to construct the query on transformed trees from a given query on input trees and a given view.

Chapter 4

Conclusion

We have considered several query preservation problems for tree-structured data in this thesis.

In Chapter 2, we have defined two kinds of query preservation problems for nondeterministic views and queries on ranked trees: universal preservation and existential preservation. We have proved that the universal preservation problem is decidable for compositions of extended linear top-down tree transducers with regular look-ahead as views and deterministic MSO tree transducers as queries (see Theorem 15). To obtain the result we have slightly generalized the result [20] of the equivalence problem for deterministic MSO tree transducers (see Theorem 13). Moreover, we have shown an algorithm that is sound for the existential preservation for finite-valued linear bottom-up tree transducers as views and deterministic MSO tree transducers as queries (see Theorem 18), and also showed some algorithms that are sound for the problem for nondeterministic queries realized by finite-valued (linear) bottom-up tree transducers (see Theorem 23 and Corollary 24). We would like to know whether (1) a sound and *complete* algorithm exists for the existential preservation, and (2) our results can be extended to more expressive classes of tree transducers such as macro tree transducers (see, e.g., [18,19,22]). Obtaining a positive solution for the question (1) seems difficult, because one is required to prove a given query q is not preserved by v_i ($i \in [K]$) for *every* possible way of decomposing a finite-valued tree transduction v into single-valued ones $v_1, \ldots, v_K$.

As mentioned in Theorem 17, finite-valued bottom-up tree transducers can be effectively decomposed into a finite number of single-valued ones of *double*-exponential order of the size of the original transducers. Whereas, in the word case, k-valued (word) transducers can be effectively decomposed into k single-valued (unambiguous) ones [45, 51] of *single*-exponential size [44]. Can k-valued

tree transducers decomposed into k single-valued ones of single-exponential size? It is an important problem that remains open for twenty years.

In Chapter 3, we have studied the decidability problems of the weak query preservation and the strong query preservation for deterministic linear top-down data tree transducers and run-based n-ary queries. We have shown 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 have also shown the strong query preservation problem is in 2-EXPTIME in general, EXPTIME-hard even if n is fixed. This decidability results have been extended to the class of single-valued extended linear top-down data tree transducers, which is a more expressive class than the class of deterministic linear top-down data tree transducers. We have provided an efficient algorithm to construct the query on transformed trees from a given query on input trees and a given view.

We would like to know whether the weak and strong query preservation problems are decidable or not for a transduction model having a copy operation, because copying elements is one of the fundamental operations for trees. Note that, as mentioned in Chapter 2, (universal and existential) query preservation problem is undecidable for the case when a view and a query are realized by tree transducers and the view can copy.

"I will. I'll get there and show others the way."

— *Act IV* [11]

References

[1] A.V. Aho and J.D. Ullman. Translations on a context free grammar. *Information and Control*, vol.19, no.5, pp.439–475, 1971.

[2] M. Arenas and L. Libkin. XML data exchange: Consistency and query answering. *Journal of the ACM*, vol.55, no.2, pp.7:1–7:72, 2008.

[3] A. Arnold and M. Dauchet. Bi-transductions de forêts. *Proc. 3rd Conference on Automata, Languages, and Programming*, pp.74–86, 1976.

[4] M. Benedikt, J. Engelfriet, and S. Maneth. Determinacy and rewriting of top-down and MSO tree transformations. In *Mathematical Foundations of Computer Science*, ed. K. Chatterjee and J. Sgall, LNCS 8087, pp.146–158, 2013.

[5] A. Berglund et al. XML Path Language (XPath) 2.0 (Second Edition). *W3C Recommendation*, 2010. `http://www.w3.org/TR/xpath20/`.

[6] R. Bloem and J. Engelfriet. A comparison of tree transductions defined by monadic second order logic and by attribute grammars. *Journal of Computer and System Sciences*, vol.61, no.1, pp.1–50, 2000.

[7] M. Bojańczyk, L.A. Kołodziejczyk, and F. Mopturlak. Solutions in XML data exchange. *Journal of Computer and System Sciences*, vol.79, no.6, pp.785–815, 2013.

[8] T. Brants, A.C. Popat, P. Xu, F.J. Och, and J. Dean. Large language models in machine translation. *Proc. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp.858–867, 2007.

[9] F. Braune, N. Seemann, D. Quernheim, and A. Maletti. Shallow local multi bottom-up tree transducers in statistical machine translation. *Proc. 51st Annual Meeting of the Association for Computational Linguistics*, vol.1, pp.811–821, 2013.

[10] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications. Book Draft, 2007.

[11] A. Chekhov. The Cherry Orchard. *Plays by Chekhov, Second Series*, Translated by Julius West, 1916. `http://www.gutenberg.org/ebooks/7986`

[12] B. Courcelle. Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science*, vol.126, no.1, pp.53–75, 1994.

[13] B. Courcelle and J. Engelfriet. Graph Structure and Monadic Second-Order Logic, a Language-Theoretic Approach. *Encyclopedia of Mathematics and its applications*, vol.138, Cambridge University Press, 2012.

[14] N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt. *Communications of the ACM*, vol.52, no.7, pp.86–94, 2009.

[15] J. Engelfriet. Bottom-up and top-down tree transformations—a comparison. *Mathematical Systems Theory*, vol.9, no.2, pp.198–231, 1975.

[16] J. Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, vol.10, no.1, pp.289–303, 1976.

[17] J. Engelfriet, E. Lilin, and A. Maletti. Extended multi bottom-up tree transducers. *Acta Informatica*, vol.46, no.8, pp.561–590, 2009.

[18] J. Engelfriet and S. Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Information and Computation*, vol.154, no.1, pp.34–91, 1999.

[19] J. Engelfriet and S. Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM Journal on Computing*, vol.32, no.4, pp.950–1006, 2003.

[20] J. Engelfriet and S. Maneth. The equivalence problem for deterministic MSO tree transducers is decidable. *Information Processing Letters*, vol.100, no.5, pp.206–212, 2006.

[21] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, L systems, and two-way machines. *Journal of Computer and System Sciences*, vol.20, no.2, pp.150–202, 1980.

[22] J. Engelfriet and H. Vogler. Macro tree transducers. *Journal of Computer and System Sciences*, vol.31, no.1, pp.71–146, 1985.

[23] W. Fan and P. Bohannon. Information preserving XML schema embedding. *ACM Transactions on Database Systems*, vol.33, no.1, pp.4:1–4:44, 2008.

[24] E. Filiot, S. Maneth, P.A. Reynier, and J.M. Talbot. Decision problems of tree transducers with origin. In *Automata, Languages, and Programming*, ed. M.M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, LNCS 9135, pp.209–221, 2015.

[25] Z. Fülöp and P. Gyenizse. On injectivity of deterministic top-down tree transducers. *Information Processing Letters*, vol.48, no.4, pp.183–188, 1993.

[26] Z. Fülöp and A. Maletti. Composition closure of ε-free linear extended top-down tree transducers. In *Developments in Language Theory*, ed. M.P. Béal and O. Carton, LNCS 7907, pp.239–251, 2013.

[27] Z. Fülöp, A. Maletti, and H. Vogler. Weighted extended tree transducers. *Fundamenta Informaticae*, vol.111, no.2, pp.163–202, 2011.

[28] T.V. Griffiths. The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines. *Journal of the ACM*, vol.15, no.3, pp.409–413, 1968.

[29] B. Groz, S. Staworko, A.C. Caron, Y. Roos, and S. Tison. Static analysis of XML security views and query rewriting. *Information and Computation*, vol.238, pp.2–29, 2014.

[30] E.M. Gurari and O.H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Mathematical Systems Theory*, vol.16, no.1, pp.61–66, 1983.

[31] A.Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, vol.10, no.4, pp.270–294, 2001.

[32] A.Y. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. *Proc. 32nd International Conference on Very Large Data Bases*, pp.9–16, 2006.

[33] K. Hashimoto, R. Sawada, Y. Ishihara, H. Seki, and T. Fujiwara. Determinacy and subsumption for single-valued bottom-up tree transducers. In *Language and Automata Theory and Applications*, ed. A.H. Dediu, C. Martín-Vide, and B. Truthe, LNCS 7810, pp.335–346, 2013.

[34] M. Kay. XSL Transformations (XSLT) 2.0 (Second Edition). *W3C Recommendation*, 2007. `http://www.w3.org/TR/xslt20/`.

[35] K. Knight and J. Graehl. An overview of probabilistic tree transducers for natural language processing. In *Computational Linguistics and Intelligent Text Processing*, ed. A. Gelbukh, LNCS 3406, pp.1–24, 2005.

[36] L. Lakshmanan and A. Thomo. View-based tree-language rewritings for XML. In *Foundations of Information and Knowledge Systems*, ed. C. Beierle and C. Meghini, LNCS 8367, pp.270–289, 2014.

[37] A. Maletti. Compositions of extended top-down tree transducers. *Information and Computation*, vol.206, no.9-10, pp.1187–1196, 2008.

[38] A. Maletti. Survey: Weighted extended top-down tree transducers part II—application in machine translation. *Fundamenta Informaticae*, vol.112, no.2-3, pp.239–261, 2011.

[39] A. Maletti, J. Graehl, M. Hopkins, and K. Knight. The power of extended top-down tree transducers. *SIAM Journal on Computing*, vol.39, no.2, pp.410–430, 2009.

[40] K. Miyahara, K. Hashimoto, and H. Seki. Node query preservation for deterministic linear top-down tree transducers. *IEICE Transactions on Information and Systems*, vol.98, no.3, pp.512–523, 2015.

[41] A. Nash, L. Segoufin, and V. Vianu. Views and queries: Determinacy and rewriting. *ACM Transactions on Database Systems*, vol.35, no.3, pp.21:1–21:41, 2010.

[42] J. Niehren, L. Planque, J.M. Talbot, and S. Tison. N-ary queries by tree automata. In *Database Programming Languages*, ed. G. Bierman and C. Koch, LNCS 3774, pp.217–231, 2005.

[43] J. Sakarovitch and R. de Souza. On the decidability of bounded valuedness for transducers. In *Mathematical Foundations of Computer Science*, ed. E. Ochmański and J. Tyszkiewicz, LNCS 5162, pp.588–600, 2008.

[44] J. Sakarovitch and R. de Souza. Lexicographic decomposition of k-valued transducers. *Theory of Computing Systems*, vol.47, no.3, pp.758–785, 2010.

[45] M.P. Schützenberger. Sur les relations rationnelles entre monoïdes libres. *Theoretical Computer Science*, vol.3, no.2, pp.243–259, 1976.

[46] N. Seemann, F. Braune, and A. Maletti. String-to-tree multi bottom-up tree transducers. *Proc. 53rd Annual Meeting of the Association for Computational Linguistics*, vol.1, pp.815–824, 2015.

[47] H. Seidl. Single-valuedness of tree transducers is decidable in polynomial time. *Theoretical Computer Science*, vol.106, no.1, pp.135–181, 1992.

[48] H. Seidl. Equivalence of finite-valued tree transducers is decidable. *Mathematical Systems Theory*, vol.27, no.4, pp.285–346, 1994.

[49] J. Thatcher and J. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, vol.2, no.1, pp.57–81, 1968.

[50] A. Weber. On the valuedness of finite transducers. *Acta Informatica*, vol.27, no.8, pp.749–780, 1990.

[51] A. Weber. Decomposing a k-valued transducer into k unambiguous ones. *Theoretical Informatics and Applications*, vol.30, no.5, pp.379–413, 1996.

Publication List

Peer-reviewed Journal Papers

1. Kazuki Miyahara, Kenji Hashimoto, and Hiroyuki Seki. Query Rewriting for Nondeterministic Tree Transducers. *IEICE Transactions on Information and Systems*, to appear. (related to Chapter 2)

2. Kazuki Miyahara, Kenji Hashimoto, and Hiroyuki Seki. Node Query Preservation for Deterministic Linear Top-Down Tree Transducers. *IEICE Transactions on Information and Systems*, vol.98, no.3, pp.512–523, Mar. 2015. (related to Chapter 3)

Peer-reviewed International Conference Paper

1. Kazuki Miyahara, Kenji Hashimoto, and Hiroyuki Seki. Node Query Preservation for Deterministic Linear Top-Down Tree Transducers. *Proc. 2nd International Workshop on Trends in Tree Automata and Tree Transducers*, EPTCS 134, pp.27–37, Oct. 2013. (related to Chapter 3)

Workshop Papers

1. Kazuki Miyahara, Kenji Hashimoto, and Hiroyuki Seki. On the Query Preservation for Nondeterministic Tree Transducers. *IEICE Technical Report*, vol.115, no.508, SS2015-79, pp.19–24, Mar. 2016.
(related to Chapter 2)

2. Kazuki Miyahara, Kenji Hashimoto, and Hiroyuki Seki. Node Query Preservation for Deterministic Linear Top-Down Tree Transducers (in Japanese). *IEICE Technical Report*, vol.112, no.275, SS2012-38, pp.13–18, Nov. 2012. (related to Chapter 3)

Award

1. IEICE SIGSS Encouraging Prize, May 2013.