

PH.D. DISSERTATION

Distance and Conjugacy of Word Transducers

SAINA SUNNY

19231102

*A dissertation submitted in partial fulfillment of
the requirements
for the degree of Doctor of Philosophy
in*

School of Mathematics and Computer Science



INDIAN INSTITUTE OF TECHNOLOGY GOA

April, 2025

Certificate of Approval

The Ph.D. Defence Committee recommends accepting the dissertation entitled **Distance and Conjugacy of Word Transducers**, submitted by **SAINA SUNNY** to the *Indian Institute of Technology Goa*, for the partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Computer Science**. The student has successfully defended the Ph.D. Viva-Voce Examination held in 25 July 2025.

Advisor:

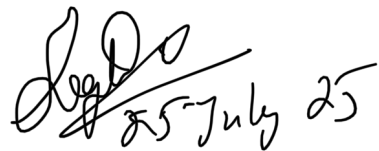


Dr. Amaldev Manuel

Ph.D. Defence Committee Members:



Chairperson: Prof. Krishna S



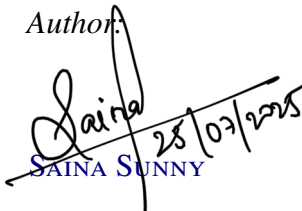
Internal Examiner: Dr. Sreejith A.V.

Declaration

I, SAINA SUNNY, hereby testify that the work embodied in this dissertation titled “Distance and Conjugacy of Word Transducers”, carried out under the supervision of Dr. AMALDEV MANUEL, is the result of bonafide research undertaken during the period 2019-2025 at Indian Institute of Technology Goa.

I declare that to the best of my knowledge, no part of this dissertation has been submitted earlier for the award of any degree, diploma, associateship, fellowship, or any other similar title of recognition from any other university or institution.

Author:


SAINA SUNNY

Advisor:


Dr. AMALDEV MANUEL

Dedication

To my Family.

Acknowledgements

I would like to convey my heartfelt gratitude to my supervisor Dr. Amaldev Manuel. His patience, encouragement, insightful discussions—both academic and personal—and immense knowledge have been instrumental in shaping my research. I am especially grateful for his trust in me, for instilling confidence, and for helping me refine my writing skills at every step. I am equally indebted to Dr. C. Aiswarya for her invaluable mentorship, from my coursework at CMI to our collaborations and to countless moments of guidance and support.

I sincerely thank all my other collaborators, including Prof. Emmanuel Filiot, Dr. Ismaël Jecker, and Dr. Khushraj Madnani, for their contributions to my research. A special note of appreciation to Prof. Emmanuel for the opportunity to visit ULB. I am also grateful to my doctoral committee members for their insightful guidance and direction.

I deeply appreciate the anonymous reviewers who have reviewed my research papers over the years. In particular, I am truly thankful to Prof. Anca Muscholl and Prof. Krishna S for kindly serving as reviewers of my thesis.

My heartfelt thanks to CMI for providing me the opportunity to take coursework and for hosting my visits. My time there was enriched by inspiring lectures, group study sessions, and engaging discussions. Special thanks to my CMI friends for making my stay truly memorable.

I am deeply thankful to IIT Goa for supporting my research, providing financial assistance, and facilitating my travel to conferences. I am grateful to all the faculty members at IIT Goa, particularly those in the SMCS department, for fostering a wonderful academic environment. The friendships I have formed within and beyond my department have been truly special—especially my Malayalee friends, department colleagues, and badminton companions.

Finally, I dedicate this thesis to my family. I am forever grateful to my parents, Mr. T U Sunny and Ms. Jolly Sunny, for their unwavering support and belief in me. I also thank my in-laws, Mr. Mathew Kurian and Ms. Jayamol T U, who have always encouraged me to pursue my dreams. I thank my brother, brother in-law and sisters in-law for their support

and encouragement. Above all, I am profoundly grateful to my partner and colleague, Mr. Prince Mathew, whose presence throughout my PhD journey has been irreplaceable. His unconditional love and constant support have been my greatest motivation to keep going on — always and forever.

Abstract

We present a quantitative approach to distinguishing between two word-to-word functions defined by transducers, moving beyond the binary concept of equivalence. Transducers, such as spell checkers and grammar correction tools, are machines that map input words to output words, defining relations between them, called rational relations. While testing the equivalence of functions defined by transducers is decidable, it alone does not suffice for many applications. For example, when comparing two grammar correction tools, can we ascertain which one aligns more closely with a benchmark based on their outputs? To address this, we define a distance between transducers based on output word metrics, such as edit distance. Specifically, the distance between two transducers is defined as the supremum of the distances between their respective outputs over all inputs. Two transducers are *close* if their distance is finite. We prove the computability of the distance of functional transducers for various classical edit distances.

Using the notion of distance between transducers, we define approximate versions of class membership problems for transducers, namely, *approximate functionality* (whether a transducer is close to a functional) and *approximate determinisation* (whether a functional transducer is close to an input-deterministic transducer). We show that these approximate problems are decidable, extending the known results for exact versions of these problems.

The computability of the above mentioned problems relies on a combinatorial result: checking whether a rational relation is conjugate – consisting of only pairs that are cyclic shifts of each other. We show that the conjugacy of a rational relation is decidable. Towards our decision procedure, we establish a new result that is a nontrivial generalisation of a characterisation given by Lyndon and Schützenberger in 1962, extending it from conjugacy of a pair of words to set of pairs of words.

Keywords: Finite state transducers, Edit distances, Combinatorics on words, Conjugacy, Determinisation, Functionality.

Author:

Advisor:

SAINA SUNNY

Dr. AMALDEV MANUEL

List of Publications

1. C. Aiswarya, Amaldev Manuel, and Saina Sunny. Edit Distance of Finite State Transducers. In 51st International Colloquium on Automata, Languages, and Programming (ICALP 2024). Volume 297 of LIPIcs, pp. 125:1-125:20, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2024). doi.org/10.4230/LIPIcs.ICALP.2024.125.
2. C. Aiswarya, Amaldev Manuel and Saina Sunny. Deciding Conjugacy of a Rational Relation (Extended Abstract). In 28th International Conference on Developments in Language Theory (DLT 2024). Lecture Notes in Computer Science, Vol 14791, Springer, Cham. doi.org/10.1007/978-3-031-66159-4_4 (invited for journal special issue).
3. Emmanuel Filiot, Ismaël Jecker, Khushraj Madnani and Saina Sunny. Approximate Problems for Finite Transducers. In 52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025). Volume 334 of LIPIcs, pp. 155:1–155:19, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2024). doi.org/10.4230/LIPICS.ICALP.2025.155.

Contents

| | |
|--|-------------|
| Acknowledgements | ix |
| Abstract | xi |
| List of Publications | xiii |
| List of Tables | xvi |
| List of Figures | xvii |
| 1 Introduction | 1 |
| 1.1 Overview | 1 |
| 1.2 Organisation | 6 |
| 1.3 Notes | 7 |
| 2 Transducers | 9 |
| 2.1 Finite State Transducers | 9 |
| 2.2 Expressiveness and Decidability | 13 |
| 2.3 Rational Expressions for Transducers | 16 |
| 3 Words and their Combinatorics | 23 |
| 3.1 Combinatorial Tools on Words | 23 |
| 3.2 Metric on Words | 35 |
| 3.3 Notes | 39 |
| 4 Comparing Word Transducers | 41 |
| 4.1 Introduction | 41 |
| 4.2 Metric over Transducers | 42 |

| | | |
|----------|--|------------|
| 4.3 | Diameter of a Relation | 48 |
| 4.4 | Index of a Relation in a Composition Closure of a Relation | 50 |
| 4.5 | Conclusion | 56 |
| 4.6 | Notes | 56 |
| 5 | Deciding Conjugacy of a Rational Relation | 59 |
| 5.1 | Introduction | 60 |
| 5.2 | Conjugacy of a Rational Relation | 62 |
| 5.3 | Common Witness Theorems | 66 |
| 5.4 | Existence of Common Witness for Kleene Closure | 78 |
| 5.5 | Existence of Common Witness for Monoid Closure | 85 |
| 5.6 | Computing Witness of a Sumfree Expression | 106 |
| 5.7 | Conclusion | 111 |
| 5.8 | Notes | 111 |
| 6 | Computing Distance of Rational Functions | 113 |
| 6.1 | Introduction | 113 |
| 6.2 | Deciding Closeness for Edit Distances | 115 |
| 6.3 | Deciding k -closeness for Edit distances | 130 |
| 6.4 | Conclusion | 133 |
| 7 | Approximate Problems on Transducers | 135 |
| 7.1 | Introduction | 135 |
| 7.2 | Twinning Properties | 137 |
| 7.3 | Approximate Determinisation of Rational Functions | 146 |
| 7.4 | Approximate Problems for Rational Relations | 155 |
| 7.5 | Conclusion | 160 |
| 8 | Summary and Future Work | 163 |
| 8.1 | Summary of Contributions | 163 |
| 8.2 | Future Work | 164 |
| | References | 167 |
| | Index | 177 |

List of Tables

| | | |
|-----|---|-----|
| 1.1 | Edit distances. | 3 |
| 4.1 | Problems regarding distance between two transducers w.r.t. the metric d | 45 |
| 4.2 | Problems regarding diameter of a relation w.r.t. the metric d | 48 |
| 4.3 | Problems regarding the index of a relation in the composition closure of another. | 51 |
| 8.1 | Summary of contributions. | 163 |

List of Figures

| | | |
|-----|--|-----|
| 1.1 | \mathcal{T}_1 (left) outputs letters at the odd positions, \mathcal{T}_2 (middle) outputs letters at the even positions and \mathcal{T}_3 (right) outputs only a 's. | 2 |
| 2.1 | SNF tree for the SNF expression $E_1 + E_2 + E_3$ | 18 |
| 3.1 | The boundedness preorder of edit distances. | 38 |
| 5.1 | v as infix of uu | 64 |
| 5.2 | When there are at least two common inner witnesses z_1, z_2 | 71 |
| 5.3 | When there is 1 common inner witness z_1 and 1 common outer witness z_2 | 71 |
| 6.1 | An edit in the interior of u and v | 123 |
| 6.2 | Words that require arbitrarily large number of edits. | 124 |
| 7.1 | A nondeterministic transducer \mathcal{T} (left), along with a sequential approximations \mathcal{D} (right) w.r.t. the Levenshtein family of distance. | 149 |
| 7.2 | Decomposition of an output produced by \mathcal{D} | 151 |

CHAPTER 1

Introduction

Contents

| | |
|----------------------------|---|
| 1.1 Overview | 1 |
| 1.2 Organisation | 6 |
| 1.3 Notes | 7 |

1.1 Overview

A finite state transducer is a computational model that reads an input word and produces an output word using finite memory. The literature on finite state transducers is rich and dates back to the early days of computer science, where they were called generalised sequential machines (Raney, 1958; Ginsburg and Rose, 1968). They are widely used in computer arithmetic (Frougny, 1986, 1992), natural language processing (Mohri et al., 2008) and programs analysis (Cohen and Collard, 1998). While automata accept a set of input words, transducers extend this by associating output words to each input word, thereby defining a relation between input and output words. Formally, if A is the input alphabet and B is the output alphabet, a transducer defines a relation over $A^* \times B^*$, known as a *rational relation*. If the relation is a graph of a function, then it is called a *rational function*. A rational function is *sequential* if it is recognised by an input-deterministic finite state transducer. They form a strict subclass of rational functions.

Given two transducers, a natural question is: how similar are they? The traditional approach is testing equivalence of transducers that determines whether the relations defined by the transducers are identical. Equivalence checking is a well-studied problem in the literature and it is known to be undecidable for transducers in general (Fischer and Rosenberg, 1968). But for transducers realising functions, checking equivalence is decidable (Blattner and Head, 1977) and PSPACE-complete (Gurari and Ibarra, 1983).

Our aim is to meaningfully compare two transducers beyond equivalence. Consider the functions (or transductions) given by the transducers in Figure 1.1. The transducers \mathcal{T}_1 and \mathcal{T}_2 output the letters at the odd and even positions respectively, while the transducer \mathcal{T}_3 erases b 's in the input. If we were to find the odd one among these three functions, arguably \mathcal{T}_3 will be picked, with the length of the respective output on any input deviating significantly from that of the others. Our purpose is to define a measure that quantifies such distances.

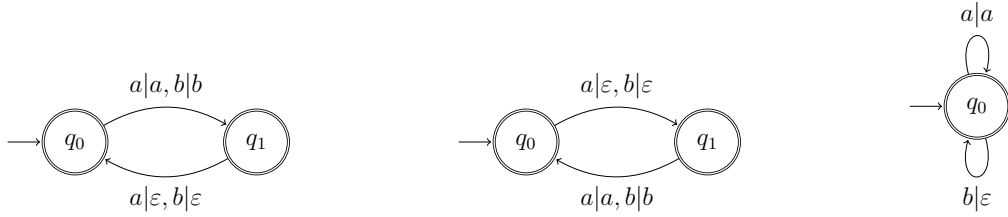


Fig. 1.1. \mathcal{T}_1 (left) outputs letters at the odd positions, \mathcal{T}_2 (middle) outputs letters at the even positions and \mathcal{T}_3 (right) outputs only a 's.

We generalise the boolean comparison to a quantitative setting by assigning a value to a pair of transducers that indicate how different they are from each other. We lift metrics over output words to the class of functional transducers and define a distance between transductions as follows. The distance between two transductions is the least upper bound of the distances between their respective outputs on any input word. We assume that their domains are the same, and we set the distance to infinity if this is not the case.

An obvious question is whether we can compute the distance for relevant metrics. A notable class of metrics is edit distances. The *edit distance* between two words is the minimum number of edit operations — such as insertion, deletion, or substitution — required to transform one word to another. This is particularly useful for comparing transducers with text-based output, such as spell checkers. We prove that the distance

| Edit Distance | Permissible Operations |
|----------------------------|--|
| Hamming | letter-to-letter substitutions |
| Conjugacy | left and right cyclic shifts |
| Transposition | swapping adjacent letters |
| Longest Common Subsequence | insertions and deletions |
| Levenshtein | insertions, deletions, and substitutions |
| Damerau-Levenshtein | insertions, deletions, substitutions and adjacent transpositions |

Table 1.1. Edit distances.

between rational functions with respect to the edit distances given in Table 1.1 are computable (Theorem 4.11).

Further, we introduce two generalisations of the notion of distance between functions, namely, *diameter* and *index* of a relation.

The diameter of a relation w.r.t. a metric is defined to be the supremum of the distance of every pair in the relation. This is studied for rational relations when distance over words is measured by their length difference (Frougny and Sakarovitch, 1991). We extend the result to edit distances and show that the diameter of a rational relation w.r.t. the edit distances given in Table 1.1 are computable (Theorem 4.15).

The *index* of a relation R in the composition closure of a relation S is defined to be the smallest integer k such that R is contained in the k -fold composition of S . The index of R in S can be thought of as the diameter of R w.r.t. the abstract edit S . If $k < \infty$, then R is said to have the *finite index property* in the composition closure of S . We prove that the finite index property is undecidable for arbitrary rational relations (Lemma 4.19). A rational relation is *metrizable* w.r.t. a metric d if the graph of the relation defines a distance equivalent to d up to boundedness. We show that the index of a rational relation in the composition closure of a *metrizable* relation w.r.t. the edit distances given in Table 1.1 are computable (Theorem 4.22).

The computability of distance, diameter and index relies on a combinatorial result: checking whether a rational relation is conjugate. Loosely speaking, the computability rests on deciding the conjugacy of a rational relation defined by the strongly connected

components of the cartesian product of the two transducers.

A pair of words is conjugate if they are cyclic shifts of each other. For instance, the pair of words $(listen, enlist)$ is conjugate, while $(loop, pool)$ is not. A relation is conjugate if every pair in the relation is conjugate. We address the decidability of conjugacy of rational relations: Given a rational relation, does it contain only conjugate pairs? Interestingly, the dual question of whether a given rational relation contains a conjugate pair is undecidable (Finkel et al., 2023). We prove that it is decidable to determine whether every pair in a rational relation is conjugate (Theorem 5.2).

Towards this, we assume that the rational relation is given as a rational expression over pairs of words. Every rational expression is effectively equivalent to a sum of sumfree expressions, possibly with an exponential size blow-up. The general problem reduces to determining the conjugacy of sumfree rational expressions. To solve this specific case, we generalise the classical theorem of Lyndon and Schützenberger (1962) from word combinatorics that equates conjugacy of a pair of words (u, v) and the existence of a word z (called a *witness*) such that $uz = zv$ (or $zu = vz$). We give two generalisations of this result. We say that a set of conjugate pairs has a *common witness* if there is a word that is a witness for every pair in the set. The generalisations are as follows:

1. If G is an arbitrary set of pairs of words, then G^* is conjugate if and only if there is a common witness for G . Moreover, a word is a common witness for G^* if and only if it is a common witness for G .
2. If $G_1^*, \dots, G_k^*, k > 0$ are arbitrary sets of pairs of words and $(\alpha_0, \beta_0), \dots, (\alpha_k, \beta_k)$ are arbitrary pairs of words, then the set of words

$$G = (\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1) \cdots (\alpha_{k-1}, \beta_{k-1})G_k^*(\alpha_k, \beta_k)$$

is conjugate if and only if it has a common witness. Moreover, the common witnesses of G are computable in polynomial time from the common witnesses of G_1^*, \dots, G_k^* .

A consequence of the above theorems is that a set of pairs generated by a sumfree rational expression is conjugate if and only if it has a common witness. Further, the set of common witnesses is computable by repeated applications of the above two results. This yields a polynomial time algorithm for checking the conjugacy of a sumfree expression and an exponential time procedure for the general problem.

Finally, using the notion of distance between transducers, we introduce approximate versions of some classical problems on transducers, including determinisation and functionality. Unlike finite automata, nondeterminism brings some extra expressive power when it comes to finite transducers. On the other hand, nondeterminism also yields some inefficiency issues when the input is received sequentially as a stream, because the whole input may have to be stored in memory until the first output symbol can be produced. This motivates the class of *sequential functions*, as the rational functions recognised by *input-deterministic* finite transducers, and the *determinisation problem*: given an arbitrary finite transducer, does it recognise a sequential function? In other words, can it be (input-) determinised? [Choffrut \(1977\)](#) gave a decidable characterisation for determinisation based on a structural property of transducer, known as *twinning property*. Later, [Weber and Klemm \(1995\)](#) proved that this property, and consequently the determinisation problem, is decidable in polynomial time.

The determinisation problem is a central problem in automata theory. It turns out that there are functions that are not exactly sequential but almost sequential, in the sense that it is “close to” some sequential functions. “Close to” can be defined using the distance between functions that we defined earlier. This raises a natural and fundamental problem, called *approximate determinisation* problem w.r.t. a metric d : given a finite transducer recognising a function f , does there exists a sequential function s such that their distance is finite? Computing whether two functions are close can be seen as the verification variant of approximate determinisation, while approximate determinisation is rather a synthesis problem, for which only f is given, and which asks to generate s if it exists. We show the decidability of approximate determinisation of finite state transducers w.r.t. Levenshtein, longest common subsequence and Damerau-Levenshtein distances ([Theorem 7.11](#)). Towards this, we introduce decidable structural properties of transducers related to the twinning property that is used to characterise exact determinisation.

We extend the study of approximate problems to rational relations, considering two related problems — approximate functionality and approximate determinisation. For this, we define metric over relations, defining distance between (not necessarily functional) transducers. The *approximate functionality* of a rational relation R asks whether a rational function f exists that stays within a bounded distance of R . We prove that this is decidable for all the metrics given in [Table 1.1](#) ([Theorem 7.22](#)). Similarly, the *approximate determinisation* of a rational relation R asks whether there is a sequential

function within a bounded distance of R . We show that the criterion for the approximate determinisation of a rational relation coincides with that of rational functions, and hence decidable ([Theorem 7.24](#)).

1.2 Organisation

The remaining chapters of this thesis are structured as follows.

Chapter 2: Transducers presents standard classes of finite state transducers and recalls relevant decidability results in the literature. Finally, discusses the formalism of expressions for relations definable by transducers.

Chapter 3: Words and their Combinatorics recalls some standard tools from combinatorics on words. Further, define metrics on words and discuss classical edit distances and their relation.

Chapter 4: Comparing Word Transducers formally introduces the notion of distance between transducers defining rational functions, explores related works and questions, and presents decidability results proven in [Chapter 6](#) using results from [Chapter 5](#). Further, it introduces diameter and index problems, highlighting their connections to the distance problem for specific instances and addressing related questions.

Chapter 5: Deciding Conjugacy of a Rational Relation studies a combinatorial result of whether every pair of words in a rational relation is conjugate (i.e., cyclic shifts of each other). It extends the characterisation of conjugacy from individual word pairs to certain sets of word pairs. Finally, it provides a decision algorithm for checking the conjugacy of rational relations.

Chapter 6: Computing Distance of Rational Functions computes the distance between transducers with respect to various metrics as listed in [Table 1.1](#) by addressing two key problems: boundedness (whether the distance is finite) and upper boundedness (whether the distance is at most a given value).

Chapter 7: Approximate Problems on Transducers introduces approximate determinisation and approximate functionality problems for transducers and shows their

decidability results for rational functions and rational relations. Towards this, it identifies certain decidable structural properties of transducers and provides characterisation based on these properties.

Chapter 8: Summary and Future Work summarises the contributions and results presented in the earlier chapters and discusses some future research directions.

1.3 Notes

This document utilizes Thomas Colcombet's [knowledge](#) package which allows key terms to be defined and later referenced through internal hyperlinks pointing to their original definitions. However, hyperlinks are not applied to every occurrence of a term. Instead, they appear at the beginning of each logical unit within a chapter, where each section is treated as a distinct unit. For theorems, lemmas, propositions, corollaries, claims, definitions, and tables, hyperlinks are provided for all nontrivial terms, ensuring clarity and ease of navigation.

The proofs of well-known standard results in the literature are omitted, but appropriate references are provided wherever they are mentioned.

CHAPTER 2

Transducers

Contents

| | | |
|-----|--|----|
| 2.1 | Finite State Transducers | 9 |
| 2.2 | Expressiveness and Decidability | 13 |
| 2.3 | Rational Expressions for Transducers | 16 |

2.1 Finite State Transducers

In this chapter, we recall the existing automaton models and the relations defined by them.

Sets and Words. Let \mathbb{N} (*resp.* \mathbb{N}^+) denote the set of *nonnegative integers* including (*resp.* excluding) zero, and \mathbb{R} denote the set of *real numbers*. For $n \in \mathbb{N}$, the notation $[n]$ represents the set $\{1, \dots, n\}$. Let A or B denote finite *alphabets*, which are finite sets of letters. A *word* is a sequence of letters. The *empty word* is denoted by ε . The set of all finite words over A is denoted by A^* , and let $A^+ = A^* \setminus \{\varepsilon\}$. The *length* of a word w is denoted by $|w|$, and in particular $|\varepsilon| = 0$. A word u is called a *factor* (*resp.* *prefix*, *suffix*) of a word v , if there exist words $x, y \in A^*$ such that $v = xuy$ (*resp.* $v = uy$, $v = xu$).

Relations and Functions. A binary *relation* R between two sets U and V is a subset of the cartesian product $U \times V$. The *domain* of R is defined as $\text{dom}(R) = \{u \mid (u, v) \in R\}$.

For $u \in U$, the set of elements related to u under R is denoted as $R(u) = \{v \mid (u, v) \in R\}$. The *identity relation* $id \subseteq U \times U$ is $\{(u, u) \mid u \in U\}$. A *function* $f : U \rightarrow V$ is a relation over $U \times V$ where $f(u)$ is a singleton set for every $u \in \text{dom}(f)$. For instance, the identity relation is a function. The function f is *partial* if $\text{dom}(f) \subsetneq U$. Given two relations R over $U \times V$ and S over $V \times W$, their *composition*, denoted by $S \circ R$, over $U \times W$ is defined by $(S \circ R)(u) = S(R(u)) = \bigcup_{v \in R(u)} S(v)$.

Finite State Automata. Automata is a fundamental model of computation that accepts a set of words using finite memory.

Definition 2.1 (Finite State Automaton)

A (*nondeterministic*) *finite state automaton* \mathcal{A} over the *alphabet* A is a tuple (Q, I, Δ, F) where Q is a finite set of states, $\Delta \subseteq Q \times A \cup \{\varepsilon\} \times Q$ is the finite set of transitions, $I \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states.

We write $p \xrightarrow{a} q$ whenever $(p, a, q) \in \Delta$ where $a \in A \cup \{\varepsilon\}$. A *run* of \mathcal{A} on an *input* word $a_1 \cdots a_n$, where $a_i \in A \cup \{\varepsilon\}$ for $i \in [n]$, is a sequence $q_0 \xrightarrow{a_1} q_1 \cdots \xrightarrow{a_n} q_n$. The run is *accepting* if $q_0 \in I$ and $q_n \in F$. The *language* of \mathcal{A} , denoted by $L(\mathcal{A})$, is the set of words w such that \mathcal{A} has an accepting run on w . The class of languages computed by finite state automaton are well known as *regular languages*.

An automaton is *deterministic* if I is a singleton set and the set of transitions is a (partial) function $Q \times A \rightarrow Q$. An automaton is *unambiguous* if on each input word the automaton has at most one run. The expressive power of finite state automata remains the same irrespective of whether they are nondeterministic or deterministic or unambiguous.

Two automata \mathcal{A}_1 and \mathcal{A}_2 are *equivalent* if $L(\mathcal{A}_1) = L(\mathcal{A}_2)$. The problem of checking equivalence is PSPACE-complete for nondeterministic- (Stockmeyer and Meyer, 1973), polynomial time for unambiguous- (Stearns and Hunt III, 1985), and NLOGSPACE for deterministic finite state automata.

Finite State Transducers and Rational Relations. Transducers extend finite state automata by labelling each transition and each final state by a possibly empty output word.

Definition 2.2 (Finite State Transducer)

A (*nondeterministic*) *finite state transducer* \mathcal{T} with input alphabet A and output alphabet B is a tuple $(Q, I, \Delta, F, \lambda, o)$ where (Q, I, Δ, F) is a *nondeterministic finite state automaton* over A , the labelling function $\lambda : \Delta \rightarrow B^*$ maps transitions to output words over B , and $o : F \rightarrow B^*$ maps final states to output words over B .

We write $p \xrightarrow{a|v}_{\mathcal{T}} q$, or simply $p \xrightarrow{a|v} q$ whenever $(p, a, q) \in \Delta$ and $v = \lambda(p, a, q)$. A *run* of \mathcal{T} labelled by an *input* word $a_1 \cdots a_n$, where $a_i \in A \cup \{\varepsilon\}$ for $i \in [n]$, is a sequence $q_0 \xrightarrow{a_1|v_1} q_1 \cdots \xrightarrow{a_n|v_n} q_n$. It is denoted more generally by $q_0 \xrightarrow{a_1 \cdots a_n | v_1 \cdots v_n} q_n$. The *output* word along the run is $v_1 \cdots v_n \cdot o(q_n)$ if $q_n \in F$, otherwise $v_1 \cdots v_n$. The run is *accepting* if $q_0 \in I$ and $q_n \in F$.

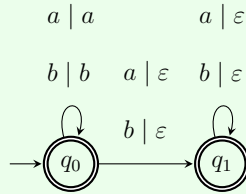
The relation *defined/recognised* by \mathcal{T} , denoted as $\llbracket \mathcal{T} \rrbracket \subseteq A^* \times B^*$, is defined as

$$\llbracket \mathcal{T} \rrbracket = \{(u, v) \mid v \in B^* \text{ is output along some accepting run labelled by } u\}. \quad (2.1)$$

Relations defined by transducers are called *rational relations*. For convenience, we use $\mathcal{T}(u)$ for $\llbracket \mathcal{T} \rrbracket(u)$, and $\text{dom}(\mathcal{T})$ to denote $\text{dom}(\llbracket \mathcal{T} \rrbracket)$, referred to as the *domain* of the transducer \mathcal{T} .

Example 2.3

Let $R \subseteq \{a, b\}^* \times \{a, b\}^*$ be a relation defined as $R = \{(u, v) \mid v \text{ is a prefix of } u\}$. The relation R is rational and the transducer below defines it.



We say a transducer is *real-time* if it has *no* transitions of the form $p \xrightarrow{\varepsilon|v} q$. A transducer \mathcal{T} is said to be *trim* if, for any state q , there exists at least one accepting run visiting q . We use interchangeably the following two equivalent *representations* of a transducer $\mathcal{T} = (Q, \Delta, I, F, \lambda, o)$.

1. $\mathcal{T} = (\mathcal{A}, \lambda, o)$ where $\mathcal{A} = (Q, I, \Delta, F)$ is called the *underlying* nondeterministic finite state automaton of \mathcal{T} .

2. $\mathcal{T} = (Q, I, \Delta', F, o)$ where $\Delta' \subseteq Q \times A \cup \{\varepsilon\} \times Q \times B^*$ is a set of transitions along with output words, i.e., $(p, a, q, v) \in \Delta'$ if $(p, a, q) \in \Delta$ and $\lambda(p, a, q) = v$. In this representation, the transducer can be seen as a finite state automaton over finite subsets of $A^* \times B^*$.

We define the cartesian product of two transducers using representation 2. Given two transducers $\mathcal{T}_i = (Q_i, I_i, \Delta_i, F_i, o_i), i \in [2]$, their *cartesian product*, denoted by $\mathcal{T}_1 \times \mathcal{T}_2$, is the transducer $(Q_1 \times Q_2, I_1 \times I_2, \Delta, F_1 \times F_2, o)$ where $((p_1, p_2), a, (q_1, q_2), (v_1, v_2)) \in \Delta$ if $(p_i, a, q_i, v_i) \in \Delta_i$ for $i \in [2]$, and, $o(p_1, p_2) = (o_1(p_1), o_2(p_2))$ for $(p_1, p_2) \in F_1 \times F_2$.

For transducers \mathcal{T}_1 and \mathcal{T}_2 with the same domain, ignoring the input word in $\mathcal{T}_1 \times \mathcal{T}_2$ results in a transducer that defines the following relation consisting of pairs of output words of \mathcal{T}_1 and \mathcal{T}_2 over any input,

$$R_o = \{(\mathcal{T}_1(w), \mathcal{T}_2(w)) \mid w \in \text{dom}(\mathcal{T}_1)\}. \quad (2.2)$$

One can construct a transducer recognising R_o by replacing each transition $p \xrightarrow{a|(v_1, v_2)} q$ in $\mathcal{T}_1 \times \mathcal{T}_2$ with $p \xrightarrow{v_1|v_2} q$ if $|v_1| \leq 1$; otherwise

$$p \xrightarrow{a_1|v_2} q_1 \xrightarrow{a_2|\varepsilon} q_2 \xrightarrow{a_3|\varepsilon} \dots q_{n-1} \xrightarrow{a_n|\varepsilon} q \quad (2.3)$$

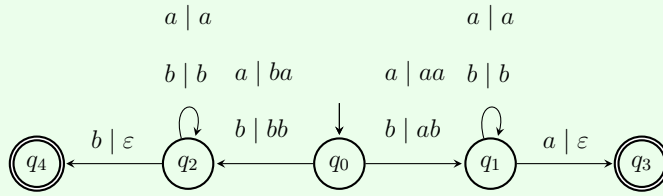
whenever $v_1 = a_1 \dots a_n$ for $a_i \in A, i \in [n]$ with $n \geq 2$ where q_1, \dots, q_{n-1} are dummy states. Such a transducer is referred to as the *output product* transducer of \mathcal{T}_1 and \mathcal{T}_2 .

Functional Transducers and Rational Functions. A transducer is *functional* if it defines a (partial) function from A^* to B^* , and the corresponding functions are called *rational functions*. Trivially, rational functions form a strict subclass of rational relations as witnessed by Example 2.3. For a functional transducer \mathcal{T} , by a slight abuse of notation, we denote by $\mathcal{T}(u)$ the unique output word produced by \mathcal{T} on $u \in \text{dom}(\mathcal{T})$. Example 2.4 illustrates a functional transducer that uses nondeterminism in the underlying automata to guess the last symbol. We can restrict the nondeterminism in the underlying automaton of the transducer and compute all rational functions. A transducer is *unambiguous* if its underlying automaton is unambiguous. Note that an unambiguous transducer is functional. The converse does not hold, but it is well-known that a function is recognised by a transducer iff it is recognised by a real-time transducer (Berstel, 1979) iff it is recognised by an unambiguous transducer (Eilenberg, 1974).

A natural extension of functional transducers is k -valued transducers. A transducer is *k -valued* (*resp. exactly k -valued*) if it produces at most (*resp. exactly*) k outputs for each input in the domain of the transducer. Functional and unambiguous transducers are exactly 1-valued.

Example 2.4

Consider the function $f_{last} : \{a, b\}^* \rightarrow \{a, b\}^*$ which moves the last symbol upfront, i.e., $f_{last}(u\sigma) = \sigma u$ where $\sigma \in \{a, b\}$ and $u \in \{a, b\}^*$. The function f_{last} is rational defined by the following transducer.



Sequential Transducers and Sequential Functions. A transducer is *sequential* if its underlying automaton is deterministic, and they define functions known as *sequential functions*. These were originally termed subsequential in the literature by Schützenberger (1977). The transducers given in Figure 1.1 are all sequential. Rational functions are strictly more expressive than sequential functions as nondeterminism allows for local transformations that depend on future properties of the input word. For instance, the transducer depicted in Example 2.4 is functional but not sequential (details in Section 2.2). Intuitively, the nondeterminism in this case is essential to guess initially the last letter of the input word.

2.2 Expressiveness and Decidability

Counter to the case of automata, various classes of transducers have different expressive power in general. As mentioned before, rational relations trivially extend rational functions, and rational functions strictly contain sequential functions. Hence, we have the following inclusion.

$$\text{sequential functions} \subsetneq \text{rational functions} \subsetneq \text{rational relations}$$

The first systematic study of rational relations was given by [Elgot and Mezei \(1965\)](#). Recent surveys by [Filiot and Reynier \(2016\)](#); [Muscholl and Puppis \(2019\)](#) provide further insights and references therein. We now discuss some classical problems on transducers.

Functionality. The *functionality* problem asks whether there exists an equivalent functional transducer for a given finite state transducer. It was first shown to be decidable by [Schützenberger \(1975\)](#), in which a word that violates functionality was shown to be of bounded length and hence gave a decision procedure of exponential complexity. A similar algorithm for functionality (or, 1-valuedness) was also given by [Blattner and Head \(1977\)](#) with exponential computing time. [Gurari and Ibarra \(1983\)](#) gave a polynomial time algorithm for deciding whether or not a transducer is *k-valued* for a fixed $k \in \mathbb{N}$, making functionality (when $k = 1$) decidable in polynomial time. Later, [Béal et al. \(2003\)](#) proposed a polynomial time decision procedure for functionality based on a natural construction performed on the *cartesian product* of the transducer by itself. The precise complexity of functionality is shown to be NLOGSPACE-complete by a reduction to the emptiness of one-counter automata ([Muscholl and Puppis, 2019](#)).

Equivalence. The *equivalence* problem asks whether the given two transducers \mathcal{T}_1 and \mathcal{T}_2 define the same relation, i.e., $\text{dom}(\mathcal{T}_1) = \text{dom}(\mathcal{T}_2)$ and $\mathcal{T}_1(w) = \mathcal{T}_2(w)$ for all $w \in \text{dom}(\mathcal{T}_1)$. This problem is fundamental and has been widely studied by several authors. Unfortunately, it is undecidable in general ([Fischer and Rosenberg, 1968](#)) and even for a *real-time* transducer ([Griffiths, 1968](#)). In fact, the unsolvability persists when the input (or output) *alphabet* is restricted to one letter ([Ibarra, 1978](#)). On the positive side, the equivalence problem is decidable for functional transducers ([Blattner and Head, 1977](#)). Specifically, it has been shown to be PSPACE-complete by [Gurari and Ibarra \(1983\)](#). The approach involves two main steps: (1) checking whether the transducers have the same *domain*, which is equivalent to deciding the equivalence of their underlying automata, and (2) determining whether the disjoint union of the two transducers is functional. The PSPACE-completeness follows from the PSPACE-completeness of the equivalence problem for finite state automata, combined with the fact that functionality can be checked in polynomial time. So, if the transducers are already known to have the same domain, then their equivalence is decidable in polynomial time. This result is indeed true for two *exactly k-valued* transducers (Theorem 4 of [Gurari and Ibarra \(1983\)](#)).

In the specific case of **unambiguous** transducers, which are 1-valued, the equivalence problem is decidable in polynomial time (Theorem 7 of Gurari and Ibarra (1983)). This is because the equivalence of the underlying unambiguous automata can be determined efficiently (Stearns and Hunt III, 1985). For **sequential** transducers, the equivalence problem is also shown to be NLOGSPACE-complete (Muscholl and Puppis, 2019).

Determinisation / Sequentiality. The *determinisation* (or *sequentiality*) problem asks does there exists an equivalent sequential transducer, i.e., deterministic in the input, for a given functional transducer. The idea for determinising a given transducer, as in the case of finite state automata, is performing an automata subset construction where on each transition, it will produce the longest common prefix of the outputs so far. The remaining delay between the outputs needs to be stored in the states, where the delay between output words u and v , denoted by $\text{delay}(u, v)$, is the pair (u', v') such that $u = \ell u'$ and $v = \ell v'$ where ℓ is the longest common prefix of u and v . The construction terminates only if the delays are finite. The finiteness of delay is characterised by a decidable structural property of transducers called the *twinning property* by Choffrut (1977). We recall a version of the twinning property of a transducer.

A trim transducer \mathcal{T} satisfies *twinning property* if for all initial states $p_0, q_0 \in \mathcal{T}$, for all states $p_1, q_1 \in \mathcal{T}$, for all words $u, v \in A^*$ and $u_1, u_2, v_1, v_2 \in B^*$, if

$$\begin{aligned} p_0 &\xrightarrow{u|u_1} p_1 \xrightarrow{v|v_1} p_1, \\ q_0 &\xrightarrow{u|u_2} q_1 \xrightarrow{v|v_2} q_1, \end{aligned}$$

then $\text{delay}(u_1, u_2) = \text{delay}(u_1 v_1, u_2 v_2)$.

The transducer depicted in Example 2.4, defining function f_{last} , does not satisfy the twinning property witnessed by the runs $q_0 \xrightarrow{a|aa} q_1 \xrightarrow{a|a} q_1$ and $q_0 \xrightarrow{a|ba} q_2 \xrightarrow{a|a} q_2$ in the transducer since $\text{delay}(aa, ba) \neq \text{delay}(aaa, baa)$. Choffrut (1977) showed that a function defined by a **trim** transducer \mathcal{T} is sequential if and only if \mathcal{T} has the twinning property. Weber and Klemm (1995) first proved that this property can be decided in polynomial time. Another polynomial time algorithm for deciding twinning is given by Béal and Carton (2002). Later, Béal et al. (2003) gave an algorithm using a polynomial time computation on the cartesian product of the transducer and itself. The complexity of sequentiality is shown to be NLOGSPACE-complete (Muscholl and Puppis, 2019).

2.3 Rational Expressions for Transducers

A **rational relation** defined by a **transducer** can be expressed by a *rational expression* over $A^* \times B^*$ generated by the alphabet $(A \times \{\varepsilon\}) \cup (\{\varepsilon\} \times B)$, where concatenation is component-wise. In this section, we formally define the rational expressions over a monoid.

A **monoid** $\mathbf{M} = (M, \cdot, 1)$ is a set M with an associative binary operation that has an identity. For convenience, we speak of the monoid operation as a product (\cdot) and denote the identity by 1. For example, A^* forms a monoid with concatenation as the operation and the empty word ε as the identity. The *product* operation can be extended to subsets of M as

$$X \cdot Y = \{x \cdot y \mid x \in X, y \in Y\}. \quad (2.4)$$

Since the operation is associative, we can define X^i without any ambiguity. For instance, defined inductively, $X^0 = \{1\}$, and $X^i = X^{i-1} \cdot X$, for $i > 0$. Similarly, the *Kleene closure* of X , denoted as X^* , is the closure of X under finite products, i.e.,

$$X^* = X^0 \cup X^1 \cup \dots \quad (2.5)$$

Definition 2.5 (Rational Subset)

The family of **rational subsets** of a **monoid** $\mathbf{M} = (M, \cdot, 1)$ is the smallest class containing all the finite subsets of M and closed under union, product and Kleene closure.

A rational relation over $A^* \times B^*$ is a rational subset of the product monoid $A^* \times B^*$. A natural way to present a rational subset is as an expression.

Definition 2.6 (Rational Expression)

A **rational expression** over the **monoid** $\mathbf{M} = (M, \cdot, 1)$ is defined recursively: $\emptyset, m \in M$ are rational expressions, and if E_1, E_2 are rational expressions then $E_1 \cdot E_2$, $E_1 + E_2$, and E_1^* are also rational expressions.

The **language** of a rational expression E , denoted as $L(E) \subseteq M$, is defined as follows: $L(\emptyset) = \emptyset$, $L(m) = \{m\}$, and

$$L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2), \quad L(E_1 + E_2) = L(E_1) \cup L(E_2), \quad L(E_1^*) = L(E_1)^*.$$

The *length* of a rational expression E over monoid M , denoted by $|E|$, is the total sum of the elements $m \in M$ and operators $(+, \cdot, *)$ used in the expression. The rational expressions define precisely the class of rational subsets of M . Two rational expressions are *equivalent* if they define the same sets. Rational relations can be expressed using rational expression over monoid $A^* \times B^*$.

Example 2.7

The expression $(\varepsilon, a)(ab, ba)^*(a, \varepsilon)$ represents the relation $\{((ab)^n a, a(ba)^n) \mid n \geq 0\}$. The expression $((a, aa) + (b, \varepsilon))^*$ represents $\{(u, v) \mid v \text{ is obtained from } u \text{ by duplicating } a\text{'s and discarding } b\text{'s}\}$.

A rational expression is *sumfree* if it does not use the union (i.e., $+$). The set of sumfree expressions is formally defined as a hierarchy. Fix a monoid $M = (M, \cdot, 1)$. Given a class \mathcal{C} of expressions over M , the *Kleene closure* of \mathcal{C} , denoted as $\mathcal{K}\mathcal{C}$, is the class of expressions

$$\mathcal{K}\mathcal{C} = \mathcal{C} \cup \{E^* \mid E \in \mathcal{C}\}. \quad (2.6)$$

The *monoid closure* of \mathcal{C} , denoted as $\mathcal{M}\mathcal{C}$, is the class of expressions

$$\mathcal{M}\mathcal{C} = \mathcal{C} \cup \{E_1 \cdots E_k \mid E_i \in \mathcal{C} \text{ for each } 1 \leq i \leq k \text{ and } k \in \mathbb{N}\}. \quad (2.7)$$

Definition 2.8 (Sumfree Expression)

The family \mathcal{F} of *sumfree expressions* over the monoid $M = (M, \cdot, 1)$ is defined inductively. Let $\mathcal{F}_0 = M \cup \{\emptyset\}$ and $\mathcal{F}_{i+1} = \mathcal{M}\mathcal{K}\mathcal{F}_i$ for each $i \geq 0$. We define

$$\mathcal{F} = \bigcup_{i \geq 0} \mathcal{F}_i.$$

The *star height* of an expression E is the smallest $k \in \mathbb{N}$ such that E belongs to \mathcal{F}_k .

Over the free monoid A^* , the set of expressions \mathcal{F}_0 is $A^* \cup \{\emptyset\}$ and $\mathcal{K}\mathcal{F}_0$ is the set of expressions $\mathcal{F}_0 \cup \{w^* \mid w \in A^*\}$ (for convenience we assume that \emptyset is not used in any other expression other than \emptyset itself). It is not difficult to see that $\mathcal{M}\mathcal{K}\mathcal{F}_0$ is the set of expressions $\mathcal{K}\mathcal{F}_0 \cup \{u_1 v_1^* u_2 v_2^* \cdots u_k v_k^* u_{k+1} \mid u_i, v_i \in A^*, k \in \mathbb{N}\}$.

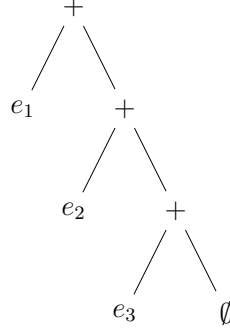


Fig. 2.1. SNF tree for the SNF expression $E_1 + E_2 + E_3$.

Every rational expression is effectively equivalent to a sum of sumfree expressions (called *sumfree normal form* (SNF)), by inductively rewriting the expression using the identities $(a + b)^* = (a^*b^*)^*$ and $(a + b) \cdot (c + d) = ac + ad + bc + bd$, as shown below.

Lemma 2.9

Every rational expression E can be converted to one in sumfree normal form E' in exponential time. Moreover, $|E'| \leq 2^{2^{|E|}}$.

Proof. Let E be a rational expression over the monoid \mathbf{M} . We assume that the rational expression E is given as a tree e . We take the size of e , denoted as $|e|$, to be the number of nodes in the tree. We inductively define a tree e' that has the same language as the sumfree normal form of the expression E and furthermore, as shown in Figure 2.1, it is in the shape of a right-comb with the internal nodes of the spine labelled with $+$'s (and the leaf of the spine is labelled with \emptyset) and the pendant left subtrees attached to the spine are sumfree. We call e' as the SNF tree of e .

We obtain an equivalent sumfree normal form expression and its expression tree e' by induction on the structure of E . We prove the following invariant along with the construction of e' .

Claim 2.10. $|e'| \leq 2^{2^{|e|}}$

The following definition is used in the analysis below. Let $N(e')$ denote the number of summands in e' , i.e., $N(e')$ is the number of nonempty pendant subtrees in the spine of the comb, or equivalently, the number of nodes labelled with $+$ in e' . Hence $N(e') \leq |e'|$.

Base Case: When E is \emptyset or $m \in M$, then E is already sumfree. The tree e corresponds to a tree with a single node. We take e' to be the tree with 3 nodes in SNF with the left subtree of the root being e . Hence $|e'| = 3$ and the claim holds.

Inductive Case: Assume that G and H are rational expressions with expression trees g and h respectively. Let g' and h' denote their SNF trees. By induction hypothesis, $G \equiv \alpha_1 + \dots + \alpha_k$ and $F \equiv \beta_1 + \dots + \beta_\ell$ such that α_i, β_j , $1 \leq i \leq k$, $1 \leq j \leq \ell$, are sumfree expressions that corresponds to each pendant subtrees attached to the spine of g' and h' respectively. Hence, there are k -many (*resp.* ℓ -many) nonempty pendant subtrees attached to the spine of g' (*resp.* h'). Also, $|g'| \leq 2^{2|g|}$ and $|h'| \leq 2^{2|h|}$.

1. If $E = G + H$, then by substituting for G and H , we get an equivalent expression of the desired form. This step takes constant time. To obtain e' we replace the leaf of the spine of g' with the root of h' . Clearly,

$$\begin{aligned}
 |e'| &= |g'| + |h'| - 1 \\
 &\leq 2^{2|g|} + 2^{2|h|} - 1 \quad (\text{Since } |g'| \leq 2^{2|g|}, |h'| \leq 2^{2|h|} \text{ by I.H.}) \\
 &\leq 2^{2(|g|+|h|+1)} \\
 &= 2^{2|e|}.
 \end{aligned}$$

2. If $E = G \cdot H$, then by substituting G and H we get $E \equiv (\alpha_1 + \dots + \alpha_k) \cdot (\beta_1 + \dots + \beta_\ell)$. Distributing the monoid operation over the union, we get $E \equiv (\alpha_1\beta_1 + \dots + \alpha_1\beta_\ell) + \dots + (\alpha_k\beta_1 + \dots + \alpha_k\beta_\ell)$, that is in the required form. This step takes time quadratic in the maximum among the length of the SNF expressions G and H .

The tree e' is a right-comb with $k\ell$ -many nonempty pendant subtrees where each subtree is obtained by the pairwise concatenation of nonempty pendant subtrees from g' and h' respectively. Clearly, $N(e') = N(g')N(h')$. Thus, there are $N(g')N(h')$ -many + nodes in the spine of e' , and $N(g')N(h')$ -many intermediate concatenation nodes in e' for pairwise concatenation of pendant subtrees of g' and h' , and finally the size of the total subtrees of e' is at most $|g'|N(h') + |h'|N(g')$.

Therefore,

$$\begin{aligned}
 |e'| &\leq N(g')N(h') + N(g')N(h') + |g'|N(h') + |h'|N(g') \\
 &\leq 4|g'||h'| && \text{(Since } N(g') \leq |g'|, N(h') \leq |h'|) \\
 &\leq 4 \cdot 2^{2|g|} 2^{2|h|} && \text{(Since } |g'| \leq 2^{2|g|}, |h'| \leq 2^{2|h|} \text{ by I.H.)} \\
 &\leq 2^{2(|g|+|h|+1)} \\
 &= 2^{2|e|}.
 \end{aligned}$$

3. Finally, if $E = G^*$, then by repeatedly applying the rational identity $(X + Y)^* = (X^*Y^*)^*$, where X, Y are rational expressions, we get $E = G^* \equiv (\alpha_1 + \alpha_2 + \dots + \alpha_k)^* = (\alpha_1^* \alpha_2^* \dots \alpha_k^*)^*$. This step takes linear time w.r.t. the length of the SNF expression G .

We obtain the tree g' corresponding to g , and construct a new tree h from g' as follows.

- Add an intermediate node labelled with $*$ between each pendant subtree and the spine of g' .
- Replace each $+$ labelled nodes in the spine with concatenation.
- Replace \emptyset in the leaf of spine with ε .
- Add a new root node labelled with $*$.

Now, e' is obtained by attaching h as the left subtree of a right-comb in the desired form. Clearly $N(e') = 1$.

$$\begin{aligned}
 |e'| &\leq |g'| + N(g') + 3 \\
 &\leq |g'| + |g'| + 3 && \text{(Since } N(g') \leq |g'|) \\
 &\leq 2 \cdot 2^{2|g|} + 3 && \text{(Since } |g'| \leq 2^{2|g|} \text{ by I.H.)} \\
 &= 2^{2|g|+1} + 3 \\
 &\leq 2 \cdot 2^{2|g|+1} && \text{(Since } |g| \geq 1, 2^{2|g|+1} \geq 8) \\
 &= 2^{2(|g|+1)} \\
 &= 2^{2|e|}.
 \end{aligned}$$

Hence, the upper bound on the size of the SNF expression is exponential in the size of the given expression. Each step of constructing an SNF expression takes polynomial time in the length of its constituent SNF expressions. Therefore, any rational expression can be converted to an equivalent sumfree normal form in exponential time. \square

Remark 2.11. *Rewriting a rational expression in SNF may result in an exponential blow-up, both in the number of summands and the size of each summand.*

For instance, consider the expression $E = (a + b)^n$ for some $n > 0$. It can be shown that any equivalent expression of E in SNF will have at least 2^n summands. Since $L(E)$ contains exactly 2^n distinct words, any equivalent expression for E must not contain a Kleene star. In that case, if an expression in SNF had fewer than 2^n terms, it would necessarily omit at least one word from $(a + b)^n$, contradicting its equivalence to E . Now, consider the expression $E' = (E\#)^*$. Any equivalent SNF expression for E' will have at least one summand of exponential size, and the expression $E \cdot \$ \cdot E'$ in SNF will have exponentially many summands of exponential size.

Words and their Combinatorics

Contents

| | |
|--|----|
| 3.1 Combinatorial Tools on Words | 23 |
| 3.2 Metric on Words | 35 |
| 3.3 Notes | 39 |

3.1 Combinatorial Tools on Words

We recall some standard notions from combinatorics on words and associated facts.

Cyclic shift and Quotients. A *cyclic shift* of a word w is a word w' such that $w = uv$ and $w' = vu$ for some words $u, v \in A^*$. If u and v are words such that u is a *prefix* of v , the *left quotient* of v by u , denoted by $u^{-1}v$, is the word x such that $v = ux$. Similarly, the *right quotient* of $v = xu$ by u , denoted as vu^{-1} , is the word x . The word u^ω denotes the unique infinite word $u \cdot u \cdot \dots$ (ω times). Let $u[i]$ denote the i th letter of u , and let $u[i \dots j]$ denote the *factor* of u from index i to j where $1 \leq i \leq j \leq |u|$.

Prefix Delay and Suffix Delay. Recall that the *delay* between words u and v , denoted by $\text{delay}(u, v)$, is the pair (u', v') such that $u = \ell u'$ and $v = \ell v'$ where ℓ is the longest common prefix of u and v . If u and v are words such that one of them is a prefix of

another, we define the *prefix delay* between u and v as

$$[u, v]_L = \begin{cases} u^{-1}v & \text{if } u \text{ is a prefix of } v \\ v^{-1}u & \text{if } v \text{ is a prefix of } u \end{cases}$$

Similarly, the *suffix delay* of two words u and v such that one of them is a *suffix* of another, denoted by $[u, v]_R$, is vu^{-1} if u is a suffix of v and uv^{-1} if v is a suffix of u . For example, $[abaa, ab]_L = aa = [ab, abaa]_L$ and $[abaa, aa]_R = ab = [aa, abaa]_R$.

Primitive Words and Roots. A word u is said to be a *power* of a word v if $u \in v^*$. Two words $u, v \in A^*$ satisfy a *nontrivial relation* $\alpha = \beta$ for $\alpha, \beta \in \{u, v\}^+$ if α and β are not identical as words over $\{u, v\}$, but $\alpha = \beta$ in A^* .

Proposition 3.1 (Choffrut and Karhumäki (1997))

Two words u and v satisfy a *nontrivial relation* iff u and v are *powers* of the same word.

Proof. If $u \in t^*$ and $v \in t^*$ for some word t , then it is obvious that u and v satisfy any nontrivial relation. For the other direction, we prove by induction on $|u| + |v|$. If $|u| = |v| = 1$, then clearly the implication holds.

Let $\alpha = \beta$ be the nontrivial relation satisfied by $u, v \in A^*$. Since α and β are not identical over $\{u, v\}$, by removing the common prefixes of α and β (in $\{u, v\}^*$), we assume $\alpha = u\alpha'$ and $\beta = v\beta'$ with $\alpha', \beta' \in \{u, v\}^*$. WLOG assume that $|u| > |v|$. Thus, there exists a word w such that

$$u = vw. \tag{3.1}$$

If $w = \varepsilon$, we are done. So let $w \neq \varepsilon$. By replacing each occurrence of u by vw in α and β , we get words over $\{v, w\}$, and let it be α_1 and β_1 respectively. Then, $\alpha_1 = \beta_1$ in A^* , and α_1 and β_1 are nonidentical over $\{v, w\}$ since α_1 starts with vw and β_1 starts with vv . Since $|v| + |w| < |u| + |v|$, v and w are powers of the same word by induction hypothesis. By Equation (3.1), the same holds for u and v . \square

A word u is *primitive* if it cannot be expressed as a power of any strictly smaller word. For example, aba is primitive but $abab$ is not. A word ρ is called a *primitive root* of a word u if $u = \rho^n$ for $n \geq 1$ and ρ is a primitive word.

Proposition 3.2

Every word has a unique **primitive root**.

Proof. Let u be a nonempty word. Assume for contradiction that u has two distinct primitive roots, say ρ_1 and ρ_2 such that $\rho_1 \neq \rho_2$ and $u = \rho_1^m = \rho_2^n$ for $m, n \geq 1$. Thus, ρ_1 and ρ_2 satisfy a nontrivial relation and hence are powers of the same word by **Proposition 3.1**. Thus, there exists a nonempty word t such that $\rho_1 \in t^*$ and $\rho_2 \in t^*$. Since both ρ_1 and ρ_2 are primitive words, we deduce $\rho_1 = t = \rho_2$, and hence contradicts that $\rho_1 \neq \rho_2$. \square

By ρ_u we denote the unique primitive root of the word u .

Proposition 3.3

The **primitive root** of a word can be computed in time linear in the length of the word.

Proof. Assume that u is a nonempty word. We claim that ρ is the primitive root of a word u iff ρ is the smallest nonempty prefix of u such that $uu = \rho u u'$ for some word u' , and $|\rho|$ divides $|u|$. The direction (\rightarrow) is trivial since $uu \in \rho \rho^*$. For the other direction, assume that $uu = \rho u u'$ for some word u' such that $|\rho|$ divides $|u|$. We deduce $u \in \rho^*$. Since ρ is the smallest prefix with the said property, ρ is the primitive root of u .

Therefore, to compute the primitive root of u , find the first nontrivial occurrence of u in uu , and let the offset be the prefix of uu before this occurrence. This can be done in linear time using the classical KMP pattern matching algorithm ([Knuth et al., 1977](#)). If the length of the offset divides the length of u , the offset is the primitive root of the word u . \square

The first theorem of Lyndon-Schützenberger relates primitivity and commutativity.

Theorem 3.4 (Lyndon and Schützenberger (1962))

Two words u and v commute, i.e., $uv = vu$, iff they have the same **primitive root**.

Proof. Follows from **Proposition 3.1** and **Proposition 3.2**. \square

We lift the notion of primitive root to a pair and a set of pairs as follows.

Definition 3.5 (Primitive Root of a Set of Pairs)

The *primitive root* of a pair (u, v) is the pair (ρ_u, ρ_v) , and the primitive root of a set of pairs P , denoted by $R(P)$, is the set of all primitive roots of each pair in P , i.e., $R(P) = \{(\rho_u, \rho_v) \mid (u, v) \in P\}$.

For example, $\{(ab, ba), (bab, abb)\}$ is the primitive root of $\{(abab, baba), (bab, abb)\}$.

Conjugate Words. A pair of words (u, v) is *conjugate* if v can be obtained from u by cyclic shifts, i.e., there exist words x, y such that $u = xy$ and $v = yx$. For example, $(aaab, aaba)$ is a conjugate pair with $x = a$ and $y = aab$. We use notation $u \sim v$ to denote that the pair of words u and v are conjugate. Conjugacy is an equivalence relation on the set of words, satisfying *reflexivity* ($u \sim u$), *symmetry* (if $u \sim v$, then $v \sim u$), and *transitivity* (if $u \sim v$ and $v \sim w$, then $u \sim w$).

The second theorem of Lyndon and Schützenberger gives a complete characterisation of the conjugacy of words.

Theorem 3.6 (Lyndon and Schützenberger (1962))

A pair of nonempty words (u, v) is *conjugate* iff there exists a word z such that $uz = zv$. Moreover, $z \in (xy)^*x$ where x and y are such that $u = xy$ and $v = yx$.

Proof. If (u, v) is conjugate, there exist words x, y such that $u = xy$ and $v = yx$. For all $z = (xy)^n x$ with $n \geq 0$, $uz = xy(xy)^n x = (xy)^n xyx = zv$.

Assume that $uz = zv$. By induction on n , for all $n \geq 1$

$$u^n z = u^{n-1} u z \stackrel{\text{I.H.}}{=} u^{n-1} v z = z v^{n-1} v = z v^n.$$

Now choose n such that $n|u| \geq |z| > (n-1)|u|$. Since $u^n z = z v^n$, we deduce that $z = u^{n-1} x$ and $u^n = zy$ for some words x and y . Also, $u = xy$ since $u^n = zy = u^{n-1} xy$.

$$\begin{aligned} v^n &= yz && \text{(Since } u^n z = z v^n \text{ and } u^n = zy) \\ &= y(xy)^{n-1} x && \text{(Since } z = u^{n-1} x \text{ and } u = xy) \\ &= (yx)^n. \end{aligned}$$

Since $|u| = |v|$, $v = yx$. Thus $u = xy$, $v = yx$ and $z \in (xy)^*x$. □

By symmetry of conjugacy, there also exists a word z' such that $z'u = vz'$ where $z' \in (yx)^*y$.

Proposition 3.7

Deciding if a pair of words is **conjugate** can be done in linear time.

Proof. We claim that two words u and v are conjugate if and only if $|u| = |v|$ and there is an occurrence of v in uu . If u and v are conjugates, then $u = xy$ and $v = yx$ for some nonempty words x, y , implying $|u| = |v|$ and $uu = xyxy = xvy$, and hence v occurs in uu . Conversely, if v occurs in uu and $|u| = |v|$, then either $u = v$ or there exist words x, y, p and q such that $v = xy$ and $u = px = yq$. Since $|u| = |v|$, we get $|p| = |y|$. This implies $p = y$ since $u = px = yq$. Therefore, $u = yx$. Hence u and v are conjugate words.

Therefore, deciding whether a pair of words (u, v) is conjugate can be done by checking if $|u| = |v|$ and verifying whether v occurs in uu . This can be done in linear time using the KMP pattern matching algorithm (Knuth et al., 1977). \square

A conjugate pair of words and its primitive root are connected.

Proposition 3.8

If u is **primitive** and (u, v) is **conjugate**, then v is **primitive**.

Proof. Assume that v is not primitive, i.e., there exists a word s such that $v = s^k$ for some $k > 1$. Clearly, any cyclic shift of v is a k th power of a cyclic shift of s . Thus, since $u \sim v$, we get $u = t^k$ where $s \sim t$. Since $k > 1$, this contradicts that u is primitive. \square

Proposition 3.9

If a pair (u, v) is **conjugate**, then its **primitive root** (ρ_u, ρ_v) is also **conjugate**. Moreover, $(u, v) = (\rho_u, \rho_v)^n$ for some $n \geq 1$.

Proof. Follows directly from proof of Proposition 3.8 and uniqueness of primitive root of a word (Proposition 3.2). \square

Fine and Wilf Theorems. Below is a fundamental result on words by Fine and Wilf (1965).

Theorem 3.10 (Fine and Wilf Theorem)

Two nonempty words u and v have the same **primitive root** if and only if the words u^ω and v^ω have a common **prefix** of length $|u| + |v| - \gcd(|u|, |v|)$.

Proof. We reduce the general case to the case where $\gcd(|u|, |v|) = 1$. If $\gcd(|u|, |v|) = d$ for $d \neq 1$, then $|u| = dm$ and $|v| = dn$ with $\gcd(m, n) = 1$. Consider u and v as elements of $(A^d)^*$, i.e., over the alphabet A^d . In the larger alphabet, $\gcd(|u|, |v|) = 1$.

So we assume $|u| = m$, $|v| = n$ and $\gcd(m, n) = 1$. Assume that u^ω and v^ω have a common prefix x of length $m + n - 1$. We show that u and v are powers of a single letter. WLOG assume that $m < n$. It suffices to show that the first $n - 1$ letters of x are equal. By assumption, we have the following

$$x[i] = x[i + m] \quad 1 \leq i \leq n - 1 \quad (3.2)$$

$$x[j] = x[j + n] \quad 1 \leq j \leq m - 1 \quad (3.3)$$

Let $i, j \in [n - 1]$ and $j \equiv i + m \pmod{n}$. Then either $j = i + m$ or $j = i + m - n$. In the former case, $x[i] = x[j]$ by Equation (3.2), and in the latter case, $x[j] = x[j + n]$ by Equation (3.3) since $j = i + m - n \leq m - 1$. Thus,

$$x[i] = x[i + m] = x[j + n] = x[j].$$

Hence, $x[i] = x[j]$ whenever $i, j \in [n - 1]$ and $j - i \equiv m \pmod{n}$. But since m and n are relatively prime, any element of the set $[n - 1]$ is equal to modulo n to a multiple of m . This shows that the first $n - 1$ letters of x are equal. Thus, u and v are powers of the same letter and hence have the same primitive root. \square

The above theorem can be adapted to yield conjugate primitive roots.

Theorem 3.11 (Conjugate Fine and Wilf Theorem)

For any two words u and v , if u^ω and v^ω have a common **factor** of length at least $|u| + |v| - \gcd(|u|, |v|)$, then the **primitive roots** of u and v are **conjugates**.

Proof. Let x be the common factor u^ω and v^ω such that $|x| \geq |u| + |v| - \gcd(|u|, |v|)$. There exist words u_1 and v_1 such that $u \sim u_1$, $v \sim v_1$, and x is a common prefix of u_1^ω and v_1^ω . Since $|u| = |u_1|$ and $|v| = |v_1|$, we get $|x| \geq |u_1| + |v_1| - \gcd(|u_1|, |v_1|)$. Using

Theorem 3.10, u_1 and v_1 have the same primitive root, say ρ , i.e., for some $n, m \geq 1$

$$u_1 = \rho^n \text{ and } v_1 = \rho^m. \quad (3.4)$$

Using arguments from the proof of the [Proposition 3.8](#), we get $u = \rho_1^n$ and $v = \rho_2^m$, where $\rho_1 \sim \rho \sim \rho_2$ using [Equation \(3.4\)](#). By [Proposition 3.8](#), ρ_1 and ρ_2 are also primitive. Therefore, the primitive roots of u and v are conjugates. \square

The bound $|u| + |v| - \gcd(|u|, |v|)$ is known as the *Fine and Wilf index* of u and v .

Cuts and Uniqueness of Cuts of Primitive Pairs. A *cut* of a conjugate pair (u, v) is a pair of words (x, y) such that $u = xy$ and $v = yx$. Alternatively, we say that u has a cut at position $|x|$, or equivalently, v has a cut at position $|y|$. If either x or y is the empty word, then we say the cut is *empty*. Otherwise, the cut is *nonempty*. For example, the pair $(aabb, bbaa)$ has a cut (aa, bb) . There can be several cuts for a conjugate pair. For instance, the pair $(abab, baba)$ has cuts (a, bab) and (aba, b) . Empty cuts are possible only for a pair of identical words.

If u and v are conjugates and one of them is primitive, by [Proposition 3.8](#), the other is also primitive. A pair (u, v) is *primitive* if both u and v are primitive words. For such pairs, their cuts are also special.

Proposition 3.12 (Uniqueness of Cuts of Primitive Pairs)

Let (u, v) be a *conjugate primitive* pair. If $u \neq v$, then (u, v) has a unique *cut* (x, y) . Otherwise, the only two possible *cuts* of (u, v) are (u, ε) and (ε, v) .

Proof. By definition, if pair (u, v) is conjugate, then there exists a cut (x, y) such that $u = xy$ and $v = yx$. Assume u and v are distinct, and hence x and y have to be nonempty. It suffices to show that x and y are unique if u and v are primitive.

For the sake of contradiction, assume that (x, y) is not unique, i.e., there exists a different cut (x', y') for (u, v) , i.e., $u = x'y'$, $v = y'x'$ and $x' \neq x, y' \neq y$. WLOG, assume that $|x| > |x'|$. Therefore there exists a nonempty word p such that $x = x'p$ and $y' = py$. Substituting for x in v , we get $v = yx = yx'p$ and substituting for y' in v , we obtain $v = y'x' = pyx'$. Therefore yx' and p commutes, i.e., $yx'p = pyx'$. By [Theorem 3.4](#), they have the same primitive root. Since p and yx' are nonempty

words, $v = pyx'$ is a power of some smaller word. Hence v is not primitive and it is a contradiction. Therefore, (u, v) has a unique cut.

In the case where $u = v$ and u being primitive, two possible cuts are (u, ε) and (ε, v) , i.e., the empty cuts. Imagine there is a nonempty cut (x, y) . Since $u = v$, we get $xy = yx$. Using [Theorem 3.4](#), u and v are powers of a smaller word and hence not primitive. Thus the only possible cuts are the empty cuts when u is primitive and $u = v$. \square

It is known that a primitive word cannot be equal to any of its nontrivial cyclic shifts, see for instance [Smyth \(2002\)](#); [Bai et al. \(2016\)](#). We give a reformulation of this fact below in a fashion that is suitable for some of the proofs in [Chapter 5](#).

Lemma 3.13 (Cut Lemma)

Assume (u, v) is a [conjugate primitive](#) pair.

- I. If (u, v) is a *distinct* pair with the unique [cut](#) (x, y) , then the following equalities *cannot* hold for any nonempty words x', x'', y', y'' such that $x = x'x''$ and $y = y'y''$.
 - (a) $xy = x''yx'$.
 - (b) $xy = y''xy'$.
 - (c) $yx = y''xy'$.
 - (d) $yx = x''yx'$.
 - (e) $xy = yx$.
- II. In the special case when $u = v$, there are two [empty cuts](#) (u, ε) and (ε, u) . In both cases, the equality $u = u''u'$ *cannot* hold for any nonempty words u', u'' such that $u = u'u''$.

Proof. Consider the case when (u, v) is a distinct pair with the unique nonempty cut (x, y) . It suffices to show that if any of the equalities hold, there exists a different nonempty cut of the primitive pair (u, v) contradicting [Proposition 3.12](#).

1. In the case of I. (a), the other nonempty cut is (x'', yx') since $(xy, yx) = (x''yx', yx'x'')$.

2. In the case of I. (b), the other nonempty cut is $(y''x, y')$ since $(xy, yx) = (y''xy', y'y''x)$.
3. When I. (c) is true, we obtain a different nonempty cut (xy', y'') because $(xy, yx) = (xy'y'', y''xy')$.
4. If I. (d) holds, the other nonempty cut is $(x', x''y)$ since $(xy, yx) = (x'x''y, x''yx')$.
5. If I. (e) holds, the other nonempty cut is (y, x) since $(xy, yx) = (yx, xy)$ and $x \neq y$ (since $u \neq v$).

Consider the special case when $u = v$. If the equality $u = u''u'$ holds, then we obtain $u = u'u'' = u''u'$. Therefore, u' and u'' commutes. Since u' and u'' are nonempty words, u is a power of some smaller word using Theorem 3.4. Hence u is not primitive and it is a contradiction. \square

A consequence of the above lemma is that the cut of the primitive root decides the cuts of its powers.

Proposition 3.14

Let (u, v) be a *distinct conjugate primitive* pair with the unique **cut** (x, y) . Any **cut** of the pair (u^n, v^n) for $n \geq 1$ is of the form (x', y') where $x' \in (xy)^*x$ and $y' \in (yx)^*y$.

Proof. Let $(u', v') = (u^n, v^n)$ for some $n \geq 1$. The lemma is trivially true for $n = 1$ by the uniqueness of cut of primitive pairs by Proposition 3.12.

Consider the case when $n \geq 2$. Substituting for $u = xy$ and $v = yx$ in u' and v' ,

$$\begin{aligned} u' &= \overbrace{u \cdots u}^{n \text{ times}} = xy \cdots xy \\ v' &= v \cdots v = yx \cdots yx \end{aligned}$$

We show that cut in u' will always be at the end of some x and all other cases lead to one of the Cases I. (a) to I. (e) of Cut Lemma.

Case 1: When the cut is at the end of y . I.e., there exists a cut (p, q) for (u', v') such that $p \in (xy)^+$. Then

$$u' = \overbrace{xy \cdots xy}^p \overbrace{xy \cdots xy}^q \quad (3.5)$$

$$v' = yx \cdots yxyx \cdots yx = qp = \overbrace{xy \cdots xy}^q \overbrace{xy \cdots xy}^p \quad (3.6)$$

Equating the suffixes of v' of length $|xy|$ in both side of the Equation (3.6), we deduce $xy = yx$, i.e., $u = v$. It satisfies Case I. (e) of Cut Lemma. Hence a contradiction.

Case 2: When the cut is strictly within some x or y . We will make a further case analysis: when there is an xy present before the cut, and when there is an xy present after the cut (since $n \geq 2$).

Suppose the cut in u' is in the i^{th} xy for $i > 1$, i.e., there is an xy present before the cut.

1. When the cut is within x , i.e., there exists a cut (p, q) of (u', v') such that $p \in (xy)^+x'$ where x' is a nonempty proper prefix of x and $x = x'x''$ for some word x'' . Now,

$$u' = \cdots \overbrace{x'x''yx'}^p \overbrace{x''y \cdots}^q \quad (3.7)$$

$$v' = yx'x'' \cdots yx'x'' = qp = \overbrace{x''y \cdots}^q \overbrace{x'x''yx'}^p \quad (3.8)$$

As before, equating the suffixes of v' of length $|xy|$ on both sides of Equation (3.8), we obtain

$$yx = yx'x'' = x''yx'$$

Here x' and x'' satisfies Case I. (d) of Cut Lemma. Hence a contradiction.

2. When the cut is within y , i.e., there exists a cut (p, q) of (u', v') such that $p \in (xy)^+xy'$ where y' is a nonempty prefix of y and $y = y'y''$ for some word y'' . Then,

$$u' = \cdots \overbrace{xy'y''xy'}^p \overbrace{y'' \cdots}^q \quad (3.9)$$

$$v' = y'y''x \cdots y'y''x = qp = \overbrace{y'' \cdots}^q \overbrace{xy'y''xy'}^p \quad (3.10)$$

On both sides of the Equation (3.10), equating the suffixes of v' of length $|xy|$, we get

$$yx = y'y''x = y''xy'$$

that is Case I. (c) of Cut Lemma. Hence a contradiction.

The case when there is an xy after the cut is symmetric and leads to Cases I. (a) and I. (b) of Cut Lemma.

Since we have eliminated all of the other cases, the only possible cuts of the pair (u', v') are of the form $((xy)^*x, (yx)^*y)$. \square

The below lemma gives the relationship between the cuts of two conjugate primitive pairs of equal length, particularly when there is a specific combination of these pairs which is conjugate.

Lemma 3.15 (Equal Length Lemma)

Let $(u_1, v_1), (u_2, v_2)$ be two conjugate primitive pairs of equal length (i.e., $|u_1| = |u_2|$) and let (x_1, y_1) and (x_2, y_2) be their cuts respectively. Any pair $(u_1, v_1)^{\ell_1}(u_2, v_2)^{\ell_2}$ where $\ell_1 > 2, \ell_2 > \ell_1 + 2$, is conjugate only if either $x_1 = x_2$ or $y_1 = y_2$.

Proof. Let $(u, v) = (u_1, v_1)^{\ell_1}(u_2, v_2)^{\ell_2}$ such that $\ell_1 > 2$ and $\ell_2 > \ell_1 + 2$.

$$\begin{aligned} u &= \overbrace{u_1 \cdots u_1}^{\ell_1 \text{ times}} \overbrace{u_2 u_2 \cdots u_2 u_2}^{\ell_2 \text{ times}} \\ v &= v_1 \cdots v_1 v_2 v_2 \cdots v_2 v_2 \end{aligned}$$

If (u, v) is conjugate, then they have a cut say (p, q) . We have two cases: when the cut in u is within $u_1^{\ell_1}u_2$ or it is after $u_1^{\ell_1}u_2$. In both cases, we show that either $x_1 = x_2$ or $y_1 = y_2$, or both.

Case 1: When the cut in u is within $u_1^{\ell_1}u_2$. In this case, the cut in v is within the suffix $v_2^{\ell_2+1}$ since the $|u_1| = |u_2| = |v_1|$ and $\ell_2 > \ell_1 + 2$. Substituting (u_1, v_1) and (u_2, v_2) with (x_1y_1, y_1x_1) and (x_2y_2, y_2x_2) ,

$$\begin{aligned} u &= x_1y_1 \cdots x_1y_1 x_2y_2 \cdots x_2y_2 = pq \\ v &= y_1x_1 \cdots y_1x_1 \cdots y_2x_2 \underbrace{y_2x_2}_{\text{cut region}} \cdots = qp \end{aligned}$$

Since $\ell_2 > \ell_1 + 2$, there exist at least one y_2x_2 before the cut in v . We compare the suffixes of q in both u and v . Since q ends with x_2y_2 in u , the cut in v should be at the end of a y_2 by Cases I. (a), I. (b), I. (e) and II. of Cut Lemma. Hence p can be of the form x_2 or $(x_2y_2)^+x_2$ depending upon if the cut in v is within the last y_2x_2 or not.

$$\begin{aligned} u &= x_1y_1 \cdots x_1y_1x_2y_2 \cdots x_2y_2 = pq \\ v &= \underbrace{y_1x_1 \cdots y_1x_1 \cdots y_2x_2y_2}_q \underbrace{x_2 \cdots}_p \end{aligned}$$

Suppose $p \in (x_2y_2)^+x_2$, then equating the prefixes of p in u and v of length $|x_2y_2| = |x_1y_1|$ (Since $|u_1| = |u_2|$), we obtain $x_2y_2 = x_1y_1$. Substituting this in u ,

$$\begin{aligned} u &= x_1y_1 \cdots x_1y_1x_1y_1 \cdots x_1y_1 = pq \\ v &= \underbrace{y_1x_1 \cdots y_1x_1 \cdots y_2x_2y_2}_q \underbrace{x_2y_2 \cdots x_2}_p \end{aligned}$$

Now we compare the prefixes of q in u and v . Since q starts with y_1x_1 in v , from Cases I. (c), I. (d), I. (e) and II. of Cut Lemma, the cut in u should be at the end of x_1 . Therefore, $p \in (x_1y_1)^+x_1$ in u . Also, $p \in (x_2y_2)^+x_2$ in v . Since $|x_1y_1| = |x_2y_2|$ and $|x_1|, |x_2| < |x_1y_1|$, we can deduce $p = (x_1y_1)^i x_1 = (x_2y_2)^i x_2$ for some i . Hence, $x_1 = x_2$. Therefore, it implies $y_1 = y_2$ since $x_1y_1 = x_2y_2$ and hence, (u_1, v_1) and (u_2, v_2) are identical.

Suppose $p = x_2$ in v . Here, p in u is within the first x_1y_1 since $|x_1y_1| = |x_2y_2|$. Moreover, $p = x_1$ since the only possible cut in u will be at the end of x_1 by Cases I. (c), I. (d), I. (e) and II. of Cut Lemma (comparing the prefixes of q in u and v). Hence $x_1 = x_2$.

Case 2: Cut in u is after $u_1^{\ell_1}u_2$. This case is symmetric. For the sake of completeness we prove it. Substituting (u_1, v_1) and (u_2, v_2) with (x_1y_1, y_1x_1) and (x_2y_2, y_2x_2) ,

$$\begin{aligned} u &= x_1y_1 \cdots x_1y_1 \cdots x_2y_2 \overbrace{x_2y_2}^{\text{cut region}} \cdots = pq \\ v &= y_1x_1 \cdots y_1x_1y_2x_2 \cdots y_2x_2 = qp \end{aligned}$$

Note that there is at least one x_2y_2 before the cut. We compare the suffixes of p in u and v . Since p ends with y_2x_2 in v , the cut in u should be at the end of x_2 by Cases I. (c),

I. (d), I. (e) and II. of Cut Lemma.

$$\begin{aligned} u &= \overbrace{x_1 y_1 \cdots x_1 y_1 \cdots x_2 y_2 x_2 \cdots y_2}^p \overbrace{\cdots y_2}^q \\ v &= y_1 x_1 \cdots y_1 x_1 y_2 x_2 \cdots y_2 x_2 = qp \end{aligned}$$

Hence q is of the form y_2 or $(y_2 x_2)^+ y_2$ depending upon if the cut in u is within the last $x_2 y_2$ or not.

If $q \in (y_2 x_2)^+ y_2$, then comparing the prefixes of q of length $|y_2 x_2| = |y_1 x_1|$ (Since $|v_1| = |v_2|$) in u and v , we obtain $y_2 x_2 = y_1 x_1$. Substituting this in v ,

$$\begin{aligned} u &= \overbrace{x_1 y_1 \cdots x_1 y_1 \cdots x_2 y_2 x_2 \cdots y_2}^p \overbrace{\cdots y_2}^q \\ v &= y_1 x_1 \cdots y_1 x_1 y_1 x_1 \cdots y_1 x_1 = qp \end{aligned}$$

We compare the prefixes of p in u and v . Since p starts with $x_1 y_1$ in u , the cut in v should be at the end of y_1 using Cases I. (a), I. (b), I. (e) and II. of Cut Lemma. Therefore, $q \in (y_1 x_1)^+ y_1$ in v and $q \in (y_2 x_2)^+ y_2$ in u . Hence, as before we can deduce that $y_1 = y_2$. It also implies $x_1 = x_2$ since $y_1 x_1 = y_2 x_2$ and thus (u_1, v_1) and (u_2, v_2) are identical..

If $q = y_2$. The cut in v is within the first $y_1 x_1$. In fact, $q = y_1$ since the only possible cut in u will be at the end of y_1 by Cases I. (a), I. (b), I. (e) and II. of Cut Lemma (comparing the prefixes of p in u and v). Hence $y_1 = y_2$. \square

3.2 Metric on Words

For meaningfully comparing two words (or sequences, vectors, functions, etc.), it is often necessary to have a measure that quantifies their (dis)similarity. It usually consists of associating a nonnegative number with two words that indicate how different they are from each other. This usually defines the distance between words. A metric on a set is used to measure the distance between any two elements of the set.

Definition 3.16 (Metric on Words)

A *metric on words* over the alphabet A is a function $d : A^* \times A^* \rightarrow \mathbb{R} \cup \{\infty\}$ such that for any words u, v and w in A^* ,

1. $d(u, v) = 0 \iff u = v$ (*separation*),
2. $d(u, v) = d(v, u)$ (*symmetry*),
3. $d(u, v) \leq d(u, w) + d(w, v)$ (*triangle inequality*).

A metric is *integer-valued* if it has range $\mathbb{N} \cup \{\infty\}$. A trivial metric on words is the *discrete metric* — distance between words u and v , denoted by $d_\infty(u, v)$, is 0 if $u = v$ and ∞ otherwise. Another straightforward distance on words is the absolute difference of their lengths (denoted as d_{len}), i.e., $d_{len}(u, v) = \text{abs}(|u| - |v|)$. This is a *pseudo-metric* since the distance between two distinct words can be zero, i.e., does not satisfy the separation property of a metric.

An important class of metrics in the context of word transducers is *edit distances*. Loosely speaking, *edits* are operations that transform words, such as *inserting* a letter, *deleting* a letter, letter-to-letter *substitutions*, *adjacent transpositions* (swapping adjacent letters), *cyclic shifts* etc.

For a fixed set of edit operations C , the *edit distance* w.r.t. C between words u and v , is the minimum number of edits in C required to transform u to v if it is possible, and ∞ otherwise. Table 1.1 summarizes common edit distances along with their corresponding allowed operations. Edit distances are studied in coding (Levenshtein, 1966; Ullman, 1966), parsing (Aho and Peterson, 1972), speech recognition (Okuda et al., 1976; Ackroyd, 1980), molecular biology (Eppstein et al., 1990; Karp, 1993) etc. For a detailed overview of the history and applications of edit distances, see (Kruskal, 1983). We discuss each of these distances given in Table 1.1 in detail below.

Hamming Distance (d_h). It is an edit distance that permits only letter-to-letter substitutions. It is often used to quantify the number of positions in which two words of the same length differ. For example, $d_h(aba, aab) = 2$ while $d_h(aba, ab) = \infty$. The hamming distance was originally introduced by Hamming (1950) in the theory of error detecting and error correcting codes where the distance measures the error introduced by a noise

over a channel when a message is sent between its source and destination. The hamming distance of two words of length n can be computed in $\mathcal{O}(n)$ with a trivial algorithm.

Transposition distance (d_t). This metric allows only the swapping of adjacent letters, making the distance between words of different lengths infinite. For instance, $d_t(abc, cab) = 2$ (swap b and c , then a and c), while $d_t(aba, aa) = \infty$. Transposition distance is useful in scenarios where swapping adjacent characters is a natural operation, such as in keyboard typing errors or genomic sequence analysis. This metric is an adaptation of Kendall tau distance (Kendall, 1938), a metric on permutations. This distance is studied for sequences by Cicirello (2020) and showed that the distance between two words of length n can be computed in $\mathcal{O}(n \log n)$.

Levenshtein distance (d_l). It is a well-known metric named after Levenshtein (1966). It is an edit distance that allows operations: insertions, deletions and substitutions. For example, $d_l(acaba, babab) = 2$ (insert b at the beginning, delete a at the end, and substitute c with b). Since insertions and deletions are permitted, the words being compared need not be of the same length. Levenshtein distance is essential in applications such as text similarity analysis, information retrieval, and spell-checking (where it helps suggest corrections based on proximity to dictionary words). Levenshtein distance between two words of length m and n can be computed in $\mathcal{O}(mn)$ using dynamic programming (Wagner and Fischer, 1974).

Longest Common subsequence (d_{lcs}). It is an edit distance that allows only insertions and deletions. For instance, $d_{lcs}(abab, ba) = 2$. It is widely used in text compression algorithms to detect the longest repeated subsequences and in version control systems (e.g., Git, diff tools) to track differences between two versions of a file. One of the earliest references to this distance was by Levenshtein (1966) and Wagner and Fischer (1974), where the longest common subsequence is a special case of Levenshtein distance. The classical dynamic programming approach computes the longest common subsequence distance between two words of lengths m and n in $\mathcal{O}(mn)$. However, later research led to faster algorithms with improvements up to logarithmic factors (Hirschberg, 1977; Bringmann and Künnemann, 2018).

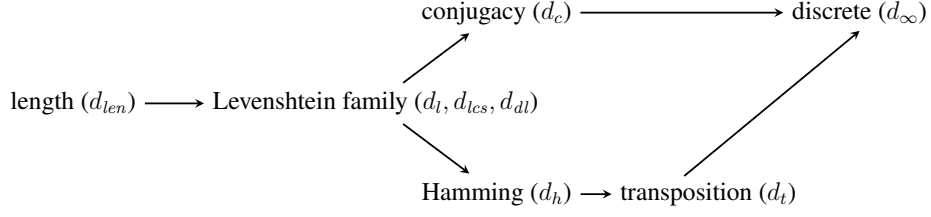


Fig. 3.1. The boundedness preorder of edit distances.

Damerau-Levenshtein distance (d_{dl}). It is an extension of Levenshtein distance that also allows the swapping of adjacent letters. This makes it particularly useful for detecting and correcting typographical errors. It was first introduced by [Damerau \(1964\)](#) and hence named after him. An example would be $d_{dl}(acb, cbc) = 2$ (swap c and b , and then substitute a with c). Given two words of length m and n , their Damerau-Levenshtein distance can be computed in $\mathcal{O}(mn)$.

Conjugacy (d_c). The conjugacy distance, also known as cyclic distance, is an edit distance that permits only [cyclic shifts](#). Two words have a finite conjugacy distance iff one is a cyclic shift of the other, and hence they must be of identical length. For example, $d_c(abab, baba) = 1$. In contrast, abc and cba are not cyclic shifts of one another, resulting $d_c(abc, cba) = \infty$. It has applications in various fields, particularly where rotation-invariant structures are important. The conjugacy distance between two words of length n can be computed in $\mathcal{O}(n)$ using the algorithm by [Knuth et al. \(1977\)](#) (adapting [Proposition 3.7](#)).

Since many of these edit operations are obtained by combinations of the others, we can relate these metrics. The notation $d_1 \leq d_2$ is an abbreviation for $d_1(u, v) \leq d_2(u, v)$ for all words u, v . We can also relate the metrics up to boundedness (See [Colcombet \(2013\)](#) for a detailed introduction). Let $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ be a *correction* function. Usual examples are increments (e.g. $x \mapsto x + 2$), scaling (e.g. $x \mapsto 2 \cdot x$) etc. We extend α to the domain $\mathbb{N} \cup \{\infty\}$ by letting $\alpha(\infty) = \infty$. We write $d_1 \lesssim d_2$ to mean that there is some α such that $d_1 \leq \alpha \circ d_2$. Clearly, if $d_1 \leq d_2$ then $d_1 \lesssim d_2$. If $d_1 \lesssim d_2$ and $d_2 \lesssim d_1$, we write $d_1 \approx d_2$ (this is known as the cost equivalence or the boundedness equivalence). If two functions f and g are cost-equivalent then f and g are bounded over precisely the same family of subsets (See [Proposition 1](#) of [Colcombet \(2013\)](#)).

Lemma 3.17

The **metrics** defined in Table 1.1 are related as follows:

1. $d_{len} \leq d \leq d_{\infty}$, for each **edit distance metric** $d \in \{d_l, d_h, d_t, d_c, d_{lcs}, d_{dl}\}$.
2. $d_l \approx d_{lcs} \approx d_{dl}$.
3. $d_l \leq d_h \lesssim d_t$, $d_h \not\lesssim d_l$ and $d_t \not\lesssim d_h$.
4. $d_l \lesssim d_c$ and $d_c \not\lesssim d_l$.
5. d_c and d_t as well as d_c and d_h are incomparable, i.e., $d_h \not\lesssim d_c$, $d_c \not\lesssim d_h$ and $d_t \not\lesssim d_c$, $d_t \not\lesssim d_c$.

Proof. 1. From the definition of the metrics.

2. Clearly $d_l \leq d_{lcs}$. Since one substitution can be achieved by an insertion and a deletion, $d_{lcs} \leq 2 \cdot d_l$. Hence $d_l \approx d_{lcs}$. Similarly $d_{dl} \leq d_l$ and since a transposition is equivalent to two substitutions $d_l \leq 2 \cdot d_{dl}$. Therefore $d_l \approx d_{dl}$.
3. It is obvious that $d_l \leq d_h$. Since a transposition is equivalent to two substitutions $d_h \leq 2 \cdot d_t$. Hence, $d_l \leq d_h \lesssim d_t$. However $d_h \not\lesssim d_l$ and $d_t \not\lesssim d_h$; for example, $d_l(aa, a) = 1$ while $d_h(aa, a) = \infty$, and $d_h(aa, ab) = 1$ but $d_t(aa, ab) = \infty$.
4. Since a cyclic shift is achieved by an insertion and a deletion, $d_l \leq 2 \cdot d_c$. But $d_c \not\lesssim d_l$, for instance $d_l(aa, ab) = 1$ but $d_c(aa, ab) = \infty$.
5. Consider the family of words $\{((ab)^k, (ba)^k) \mid k \geq 0\}$, $d_h, d_t \rightarrow \infty$, but d_c is 1. Conversely, for the family of words $\{(ba^k b, aba^{k-1} b) \mid k \geq 1\}$, $d_t = 1$, $d_h = 2$, but d_c is ∞ . \square

The above relations between metrics listed in Table 1.1 are depicted in Figure 3.1.

3.3 Notes

The proofs of Theorems 3.6 and 3.11, as well as Propositions 3.1, 3.2, and 3.8, are adapted from Choffrut and Karhumäki (1997). The proof of Theorem 3.10 is taken from Lothaire (1997).

Comparing Word Transducers

Contents

| | | |
|-----|--|----|
| 4.1 | Introduction | 41 |
| 4.2 | Metric over Transducers | 42 |
| 4.3 | Diameter of a Relation | 48 |
| 4.4 | Index of a Relation in a Composition Closure of a Relation | 50 |
| 4.5 | Conclusion | 56 |
| 4.6 | Notes | 56 |

4.1 Introduction

In the literature, the concept of distance between two words is lifted naturally to distance between a word and a set of words, or between two sets of words, and so on. There is a long line of research of this kind: computing the edit distance between two languages — usually defined as the smallest distance between any two pairs from the respective sets. It could be between a word and a [regular language](#) ([Wagner, 1974](#); [Allauzen and Mohri, 2008](#)), two regular languages ([Mohri, 2003](#)), a regular language and itself ([Konstantinidis,](#)

¹The contributions presented in this chapter are published in the proceedings of ICALP 2024 under the title “[Edit distance of finite state transducers](#)”. This is a joint work with Dr. C. Aiswarya and Dr. Amaldev Manuel.

2007), or a regular language and a context-free language (Han et al., 2012). In all these settings there are efficient algorithms for computing the edit distances.

In this chapter, we define the notion of distance between functions (or relations) and explore related concepts, including the diameter of a relation and the index of a relation in the composition closure of another. We show the connection between these notions in the context of rational functions and rational relations and conclude with our decidability results.

4.2 Metric over Transducers

We lift a given metric on words to the functions/relations defined by transducers. By applying metric on output words, we first define a metric over word-to-word functions and later on word-to-word relations. The metric on transducers is inherited from the metric on the functions or relations they define.

Definition 4.1 (Metric on Functions)

Let d be a metric on words over the alphabet B . Given two partial functions $f, g : A^* \rightarrow B^*$, we define their distance as follows

$$d(f, g) = \begin{cases} \sup \{ d(f(w), g(w)) \mid w \in \text{dom}(f) \} & \text{if } \text{dom}(f) = \text{dom}(g) \\ \infty & \text{otherwise} \end{cases}$$

Proposition 4.2

d is a metric on functions.

Proof. Clearly, since d is a metric on words, for functions f and g , $d(f, g) = 0$ if and only if $f = g$, and $d(f, g) = d(g, f)$. Let $f, g, h : A^* \rightarrow B^*$ be three functions. It remains to show that $d(f, g) \leq d(f, h) + d(h, g)$.

Assume the domains of f and g are different. Then, either $\text{dom}(f) \neq \text{dom}(h)$ or $\text{dom}(h) \neq \text{dom}(g)$. In both cases, $d(f, g)$ and $d(f, h) + d(h, g)$ are ∞ . Therefore, assume that the domains of f, g and h are the same, call it L . Since for each word w in L , $d(f(w), g(w)) \leq d(f(w), h(w)) + d(h(w), g(w))$ by virtue of d being a metric, it

follows that

$$\begin{aligned}
 d(f, g) &= \sup \{ d(f(w), g(w)) \mid w \in L \} \\
 &\leq \sup \{ d(f(w), h(w)) + d(h(w), g(w)) \mid w \in L \} \\
 &\leq \sup \{ d(f(w), h(w)) \mid w \in L \} + \sup \{ d(h(w), g(w)) \mid w \in L \} \\
 &= d(f, h) + d(h, g) .
 \end{aligned}
 \quad \square$$

The *adjacent functions* introduced by Reutenauer and Schutzenberger (1991) is related to the notion of distance between functions w.r.t. prefix distance. The *prefix distance* between two words u and v is defined as $d_p(u, v) = |u'| + |v'|$ where $(u', v') = \text{delay}(u, v)$. Two functions f and g are *adjacent* if

$$\sup \{ d_p(f(w), g(w)) \mid w \in \text{dom}(f) \cap \text{dom}(g) \} < \infty.$$

Equivalently, if f and g have identical domains, they are adjacent when their prefix distance is finite. The adjacency of *rational functions* are studied for their characterisation and is shown to be decidable (Reutenauer and Schutzenberger, 1991).

Analogous to metrics on word-to-word functions, we can define metrics on word-to-word relations. Unlike functions, a relation maps an input word to a set of output words. This necessitates a measure of distance between two sets of words (or languages). The distance between two languages L and L' can be defined in a standard way, by resorting to the Hausdorff distance, denoted by $H_d(L, L')$. Informally, it is defined to be the least upper bound of all the distances from a point in one set to the closest point in the other set. More precisely, the *directed* distance from L to L' is defined as

$$\vec{d}(L, L') = \sup_{w \in L} \inf_{w' \in L'} d(w, w') \quad (4.1)$$

The *Hausdorff* distance between L and L' is defined as

$$H_d(L, L') = \max \{ \vec{d}(L, L'), \vec{d}(L', L) \} \quad (4.2)$$

Definition 4.3 (Metric on Relations)

Given a *metric* d on words, and two *relations* $R, S \subseteq A^* \times B^*$, the distance between R and S is defined as follows.

$$d(R, S) = \begin{cases} \sup \{ H_d(R(w), S(w)) \mid w \in \text{dom}(R) \} & \text{if } \text{dom}(R) = \text{dom}(S) \\ \infty & \text{otherwise} \end{cases}$$

4. COMPARING WORD TRANSDUCERS

Therefore, $d(R, S) < \infty$ iff $\text{dom}(R) = \text{dom}(S)$ and there exists $k \in \mathbb{N}$ such that for all word u in the domain and any output $v_1 \in R(u)$, there exists some output $v_2 \in S(u)$ satisfying $d(v_1, v_2) \leq k$, and vice-versa.

Proposition 4.4

d is a **metric** on **relations**.

Proof. As the Hausdorff distance H_d is a metric on languages, it follows that d is also a metric on relations, by an argument similar to [Proposition 4.2](#). \square

Since d is metric on relations, by [separation](#) property, two relations are equivalent if and only if their distance is exactly 0. Recalling that checking the [equivalence](#) of two [rational relations](#) is undecidable ([Fischer and Rosenberg, 1968](#)), it is immediate to conclude that the distance between rational relations is not computable since the former problem simply reduces to computing the distance and validating if it is 0. So, it is natural to restrict our attention to the distance between rational functions.

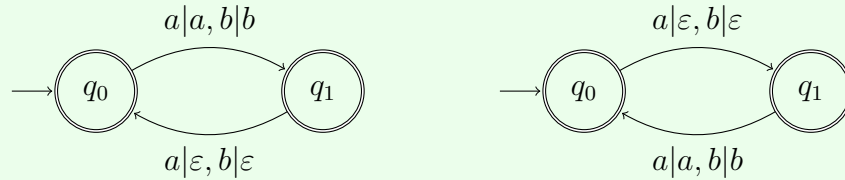
Definition 4.5 (Distance between Transducers)

The distance between two transducers \mathcal{T} and \mathcal{S} , denoted by $d(\mathcal{T}, \mathcal{S})$, is defined to be the distance between the rational relations (or functions) defined by them.

For [functional transducers](#), the value $d(\mathcal{T}, \mathcal{S})$ is an upper bound on how dissimilar the outputs of transducers \mathcal{T} and \mathcal{S} can be on any input. Following are some examples.

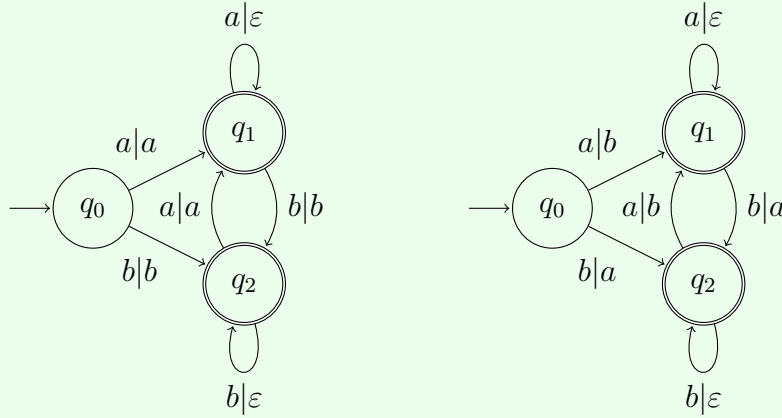
Example 4.6

The transducers \mathcal{T}_1 (on the left) and \mathcal{T}_2 (on the right) output the letters at the odd and even positions respectively. For any input word u , $\text{abs}(|\mathcal{T}_1(u)| - |\mathcal{T}_2(u)|) \leq 1$. Hence $d_{\text{len}}(\mathcal{T}_1, \mathcal{T}_2) = 1$. For input word $(ab)^n$ where $n \geq 1$, the outputs produced by \mathcal{T}_1 and \mathcal{T}_2 are a^n and b^n respectively. Since n substitutions are required to convert a^n to b^n , the [Levenshtein](#) distance $d_l(a^n, b^n) = n$. Therefore $d_l(\mathcal{T}_1, \mathcal{T}_2) = \infty$.



Example 4.7

The transducer \mathcal{T}_4 (on the left) replaces each block of a 's by a single a and each block of b 's by a single b . Similarly, \mathcal{T}_5 (on the right) substitutes a block of a 's by a single b and a block of b 's by a single a . The output words produced by the transducers on any input word is an alternate sequence of a 's and b 's. If \mathcal{T}_4 outputs aba , then \mathcal{T}_5 produces its complement, i.e., bab . Thus, the Levenshtein distance between the output words on any input is at most 2, while the **Hamming** distance is exactly the length of the output words. Hence, $d_l(\mathcal{T}_4, \mathcal{T}_5) = 2$, but $d_h(\mathcal{T}_4, \mathcal{T}_5) = \infty$.



We ask some computational and boundedness questions about the distance between two given transducers w.r.t. a metric d , namely **distance**, **closeness**, and **k -closeness** problems as follows.

| Problem | Input | Question |
|------------------------|--|---|
| Distance problem | transducers \mathcal{T}, \mathcal{S} | $d(\mathcal{T}, \mathcal{S})$? |
| Closeness problem | transducers \mathcal{T}, \mathcal{S} | Is $d(\mathcal{T}, \mathcal{S}) < \infty$? |
| k -Closeness problem | integer k , transducers \mathcal{T}, \mathcal{S} | Is $d(\mathcal{T}, \mathcal{S}) \leq k$? |

Table 4.1. Problems regarding distance between two transducers w.r.t. the metric d .

We say two transducers \mathcal{T} and \mathcal{S} are **close** (resp. **k -close**, for $k \in \mathbb{N}$) w.r.t. d if $d(\mathcal{T}, \mathcal{S}) < \infty$ (resp. $d(\mathcal{T}, \mathcal{S}) \leq k$). Closeness with respect to the **discrete metric** d_∞ is precisely the **equivalence** problem. In the case of edit distances, closeness means that any output of \mathcal{T} can be converted to an output of \mathcal{S} by doing a bounded number of edits.

This is particularly relevant for comparing transducers with text-based output, such as spell checkers.

Remark 4.8. From [Definition 4.1](#), it is easy to verify that [Lemma 3.17](#) holds for transducers as well. If $d_1 \lesssim d_2$, then it is easy to see that if transducers \mathcal{T} and \mathcal{S} are not *close* w.r.t. d_1 , then they are not *close* w.r.t. d_2 either.

Closeness and k -closeness correspond to a boundedness problem and an upper bound problem on the distance respectively. For *integer-valued* metrics, the distance problem reduces to closeness and k -closeness as follows.

Proposition 4.9

Let d be an *integer-valued metric*. The *distance* problem w.r.t. d is computable if and only if *k -closeness* and *closeness* problems w.r.t. d are decidable.

Proof. Clearly, if we can compute the distance w.r.t. d then we can decide k -closeness as well as closeness. For the other direction, given two transducers, we first check if they are close and if it is we perform an exponential search — check if they are k -close for $k = 2^0, 2^1, 2^2, \dots$ till it fails and subsequently perform a binary search on the interval $[2^n, 2^{n+1}]$, $n \in \mathbb{N}$ that contains the distance. \square

Since the distance between transducers, or the rational relations they define, is generally uncomputable, we look at the computation of distance between functional transducers. Given two such transducers \mathcal{T} and \mathcal{S} we can define a *distance function* which maps each word w to the distance between their outputs on w as follows.

Definition 4.10 (Distance function)

The *distance function* $f_{\mathcal{T}, \mathcal{S}}^d : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ of *functional transducers* \mathcal{T} and \mathcal{S} is

$$f_{\mathcal{T}, \mathcal{S}}^d(w) = \begin{cases} d(\mathcal{T}(w), \mathcal{S}(w)) & \text{if } w \in \text{dom}(\mathcal{T}) \cap \text{dom}(\mathcal{S}) \\ \infty & \text{otherwise} \end{cases}$$

The transducers \mathcal{T} and \mathcal{S} are close w.r.t. a metric d if their *domains* are the same and their distance function $f_{\mathcal{T}, \mathcal{S}}^d$ is *limited* (i.e., $< \infty$ on its domain). Similarly k -closeness w.r.t. d of \mathcal{T} and \mathcal{S} reduces to k -limitedness of $f_{\mathcal{T}, \mathcal{S}}^d$ (i.e., $\leq k$ on its domain). Limitedness

problems are well-studied in the context of weighted automata (Leung and Podolskiy, 2004; Colcombet, 2021). Therefore, when the distance function $f_{\mathcal{T},\mathcal{S}}^d$ is computable by a $(\min, +)$ -automaton, the distance between \mathcal{T} and \mathcal{S} is computable due to Proposition 4.9.

However, there are distance functions that are not computable by weighted automata. Let $A = \{a, b\}$. Consider the functional transducers $\mathcal{T}_1, \mathcal{T}_2 : A^* \rightarrow A^*$ with the domain a^*b^* defining the functions $a^pb^q \mapsto a^p$, $a^pb^q \mapsto a^q$ respectively (\mathcal{T}_1 outputs the a 's and erases the b 's, \mathcal{T}_2 erases a 's and renames the b 's as a 's). It is easily checked that their distance function w.r.t. the Levenshtein family ($d \in \{d_l, d_{lcs}, d_{dl}\}$) is $f_{\mathcal{T}_1, \mathcal{T}_2}^d : a^pb^q \mapsto \text{abs}(p - q)$.

If $f : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ is a function computed by weighted automata ($(\min, +)$ or $(\max, +)$ or B -automata (Colcombet, 2009)), then the language $L_{f \leq k} = \{w \in A^* \mid f(w) \leq k\}$ is regular for each $k \leq \mathbb{N}$. Hence the function $f_{\mathcal{T}_1, \mathcal{T}_2}^d$ is not realised by any of them (consider the language $L_{f_{\mathcal{T}_1, \mathcal{T}_2}^d \leq 0} = \{a^n b^n \mid n \geq 0\}$). In fact, it can be shown that the function $f_{\mathcal{T}_1, \mathcal{T}_2}^d$ is not computed even up to boundedness (Colcombet et al., 2016).

To decide the closeness or k -closeness of functional transducers \mathcal{T}_1 and \mathcal{T}_2 , we first check whether they have the same domain, which reduces to testing the equivalence of their underlying automata. If $\text{dom}(\mathcal{T}_1) = \text{dom}(\mathcal{T}_2)$, we construct an output product transducer of \mathcal{T}_1 and \mathcal{T}_2 that defines the relation $\{(\mathcal{T}_1(w), \mathcal{T}_2(w)) \mid w \in \text{dom}(\mathcal{T}_1)\}$ consisting of all pairs of output words produced by \mathcal{T}_1 and \mathcal{T}_2 on any input. The interesting case for deciding closeness lies in analysing the nonempty loops in this output product transducer. If no such loops exist, the relation contains only finitely many output pairs, and their distance can be computed effectively. Notably, we show that any input-output word pair generated along these loops must be conjugate.

In Chapter 6, using the results of conjugacy from Chapter 5, we show that both closeness and k -closeness problems are decidable for rational functions w.r.t. all the integer-valued metrics given in Table 1.1 (see Theorem 6.2). Hence, we get the following result by virtue of Proposition 4.9.

Theorem 4.11

The distance between rational functions with respect to the metrics given in Table 1.1 are computable.

4.3 Diameter of a Relation

A closely related notion to distance between functions is the diameter of a relation. The term “diameter” originates from graph theory where it refers to the length of the shortest path between the most distanced nodes in a graph.

Definition 4.12 (Diameter of a Relation)

The *diameter* of a relation R with respect to a metric d , denoted by $dia_d(R)$, is the supremum of the distance of the related words in R .

$$dia_d(R) = \sup \{ d(u, v) \mid (u, v) \in R \}$$

Example 4.13

Consider the relation $R = \{((ab)^n, (ba)^n) \mid n \geq 1\}$. For all $n \geq 1$, the *Hamming* distance is $d_h((ab)^n, (ba)^n) = 2n$ (substitute a with b and vice-versa), while *Levenshtein* distance is $d_l((ab)^n, (ba)^n) = 2$ (delete and insert a in the beginning and end respectively). Thus, $dia_{d_l}(R) = 2$, but $dia_{d_h}(R) = \infty$.

Similar to the questions asked in Table 4.1, we can ask the questions given in Table 4.2 about the diameter of a relation w.r.t. a metric d , namely *diameter*, *bounded diameter*, *k-bounded diameter* problems as follows.

| Problem | Input | Question |
|-------------------------------|----------------------------|--------------------------|
| Diameter problem | relation R | $dia_d(R)$? |
| Bounded diameter problem | relation R | Is $dia_d(R) < \infty$? |
| k -Bounded diameter problem | integer k , relation R | Is $dia_d(R) \leq k$? |

Table 4.2. Problems regarding diameter of a relation w.r.t. the metric d .

A relation has *bounded* (resp. *k-bounded*) diameter w.r.t. a metric d if the diameter of the relation w.r.t. d is finite (resp. $\leq k$). Frougny and Sakarovitch (1991) studied *rational relations* with *bounded delay*, which are exactly the rational relations with bounded diameter when the distance over words is measured by their length difference (d_{len}). In

fact, they showed the computability of the diameter of a rational relation w.r.t. d_{len} , and we extend the results to [edit distances](#).

Towards this, we show that the diameter problem mutually reduces to the [distance](#) problem of rational functions. Thus, their closeness and boundedness problems are also interreducible. The correspondence between distance and diameter problem for rational relations follows from the theorem below. A [morphism](#) $\phi : A^* \rightarrow B^*$ is a function satisfying $\phi(\varepsilon) = \varepsilon$ and $\phi(uv) = \phi(u)\phi(v)$ for all $u, v \in A^*$. It can be seen as a function [defined](#) by a one-state transducer with no output at final states, i.e., $o(q) = \varepsilon$ for all final states q .

Theorem 4.14 (Nivat (1968))

Let A and B be [alphabets](#). The following conditions are equivalent.

1. R is a [rational relation](#) over $A^* \times B^*$.
2. There exists an [alphabet](#) C , two [morphisms](#) $\phi : C^* \rightarrow A^*$ and $\psi : C^* \rightarrow B^*$ and a [regular language](#) $L \subseteq C^*$ such that $R = \{(\phi(w), \psi(w)) \mid w \in L\}$.

Diameter to Distance. Given a rational relation R , we can create two [functional transducers](#) \mathcal{T}_1 and \mathcal{T}_2 such that $dia_d(R) = d(\mathcal{T}_1, \mathcal{T}_2)$ for any metric d . The domain for these transducers corresponds to the set L in [Theorem 4.14](#). For each transition in \mathcal{T}_1 and \mathcal{T}_2 that involves an input alphabet symbol σ , we set the outputs to be $\phi(\sigma)$ and $\psi(\sigma)$ in [Theorem 4.14](#), respectively. Since ϕ and ψ are morphisms, \mathcal{T}_1 and \mathcal{T}_2 recognise the functions $\{(w, \phi(w)) \mid w \in L\}$ and $\{(w, \psi(w)) \mid w \in L\}$ respectively. Since the domain of these transducers is identical, the distance between \mathcal{T}_1 and \mathcal{T}_2 with respect to any metric d is $d(\mathcal{T}_1, \mathcal{T}_2) = \sup \{d(\phi(w), \psi(w)) \mid w \in L\}$, that is equivalent to the diameter of R w.r.t. metric d .

Distance to Diameter. Given two transducers \mathcal{T}_1 and \mathcal{T}_2 , if their domains are not equal, their distance is infinite. Otherwise, when $dom(\mathcal{T}_1) = dom(\mathcal{T}_2)$, we define the relation

$$R = \{(\mathcal{T}_1(w), \mathcal{T}_2(w)) \mid w \in dom(\mathcal{T}_1)\}.$$

By the definitions of diameter and distance, $d(\mathcal{T}_1, \mathcal{T}_2) = dia_d(R)$ for any metric d . The relation R is rational given by the [output product](#) transducer of \mathcal{T}_1 and \mathcal{T}_2 obtained by ignoring the input word in $\mathcal{T}_1 \times \mathcal{T}_2$ ([cartesian product](#) of \mathcal{T}_1 and \mathcal{T}_2).

Therefore, the problems regarding the distance between rational functions are interreducible to the problems about the diameter of a rational relation. Since the distance between rational functions is computable (see [Theorem 4.11](#)) as well as [closeness](#) and [k-closeness](#) are decidable (see [Theorem 6.2](#)) for all metrics listed in [Table 1.1](#), we get the following result.

Theorem 4.15

The [diameter](#), [bounded diameter](#) and [k-bounded diameter](#) problem stated in [Table 4.2](#) are decidable for a [rational relation](#) w.r.t. [metrics](#) given in [Table 1.1](#).

Rational relations that are bounded with respect to the [discrete metric](#) are simply those containing only identical pairs, and deciding this reduces to the problem of checking the [equivalence](#) of two rational functions. Recall that the latter problem is decidable ([Blattner and Head, 1977](#); [Gurari and Ibarra, 1983](#)), and hence deciding whether a rational relation is a subset of the [identity relation](#) is decidable.

4.4 Index of a Relation in a Composition Closure of a Relation

Another related notion to the distance between functions and the diameter of a relation is that of the index of a relation in the composition closure of another. Towards defining the problem, we first define the composition closure of a relation.

Definition 4.16 (Composition Closure of a Relation)

Let S be a [relation](#) over $A^* \times A^*$. Let $S^{(n)}$ denote the [composition](#) of S with itself $n \geq 0$ times ($S^{(0)}$ is taken to be the [identity relation](#)), and let $S^{\leq(n)}$ denotes the [composition](#) of S with itself at most n times, i.e.,

$$S^{\leq(n)} = S^{(0)} \cup S^{(1)} \cup \dots \cup S^{(n)}.$$

The [composition closure](#) of S , denoted as $S^{(*)}$, is defined as $S^{(*)} = \bigcup_{i \geq 0} S^{(i)}$.

Notice that we use parenthesis around the superscript to indicate that the base operation is composition, and not concatenation.

Definition 4.17 (Index of a Rational Relation in a Composition Closure)

Let S be a relation over $A^* \times A^*$. An *index* of a relation R in the composition closure of S , denoted as $\text{Index}(R, S)$, is the smallest integer k such that R is contained in $S^{\leq(k)}$, i.e., $R \subseteq S^{\leq(k)}$.

Example 4.18

Consider a relation S over $\{a, b\}^* \times \{a, b\}^*$ that deletes the first a if exists on any input. Fix an integer $k > 0$ and let R be the relation that deletes the first k a 's from the input if exists. In fact, both R and S are functions. The index of R in $S^{(*)}$ is k since for any input word $u \in \{a, b\}^*$, $R(u) \in S^{\leq(k)}(u)$.

Consider another relation R' that deletes all a 's from the input. For any $k > 0$, the pair $(a^{k+1}, \varepsilon) \in R'$ but $(a^{k+1}, \varepsilon) \notin S^{\leq(k)}$. Thus, the index of R' in $S^{(*)}$ is ∞ .

As seen in the case of the distance and diameter problem, we can ask some questions about the index of a relation in the composition closure of a relation, namely, *index*, *bounded index*, and *k-bounded index* problems as follows.

| Problem | Input | Question |
|-----------------------------------|-------------------------------|------------------------------------|
| Index problem | relation R, S | $\text{Index}(R, S)$? |
| Bounded (or finite) index problem | relation R, S | Is $\text{Index}(R, S) < \infty$? |
| k -Bounded index problem | integer k , relation R, S | Is $\text{Index}(R, S) \leq k$? |

Table 4.3. Problems regarding the index of a relation in the composition closure of another.

We say a relation R has *bounded* (resp. *k-bounded*) index in the composition closure of a relation S if the index of R in $S^{(*)}$ is finite (resp. $\leq k$). The following lemma shows that checking the boundedness of the index problem for an arbitrary rational relation is difficult.

Lemma 4.19

It is undecidable to check if a rational relation has a bounded index in the composition closure of an arbitrary rational relation.

Proof. The proof is by reducing a version of the membership problem of the Turing machine to the bounded index problem of a rational relation.

A Turing machine is a mathematical model of computation consisting of 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_{in}, q_{accept}, q_{reject})$ where Q is a finite set of states; $q_{in}, q_{accept}, q_{reject} \in Q$ is the start state, accept state and reject state respectively; Σ, Γ are the finite input and tape alphabet respectively; and $\delta : Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function — given a state and the current tape symbol (pointed by the tape head) the transition writes a new symbol, changes state and moves the tape-head one cell either to the left or to the right. A configuration of a Turing machine is a snapshot of the current state of the machine. It can be expressed as a word consisting of three components: the tape contents to the left of the tape head, the current state, and the tape contents to the right of the tape head (including the current cell). For example, configuration $x_1x_2 \cdots x_nqy_1y_2 \cdots y_m$, where $x_i, y_i \in \Gamma$, indicates that tape content is $x_1 \cdots x_ny_1 \cdots y_m$ followed by infinite blanks, the tape head is pointing at y_1 and the current state is q . On an input word w , the initial configuration is $q_{in}w$, and a configuration of the form $xq_{accept}y$ and $xq_{reject}y$, $x, y \in \Gamma^*$, are called accepting and rejecting respectively. Let \vdash represent a relation between two configurations C_1, C_2 such that $C_1 \vdash C_2$ if C_2 is a possible next valid configuration. For instance, for a transition $\delta(q, a) = (q', b, R)$ and current configuration $xqay$, where $x, y \in \Gamma^*, a \in \Gamma$, a next valid configuration would be $xbq'y$. Hence $xqay \vdash xbq'y$. It is easy to verify that \vdash is a rational relation. A computation on a Turing machine is a sequence $C_0 \vdash C_1 \cdots \vdash C_m$ for some $m \geq 0$ with C_0 being the initial configuration. This is denoted by $C_0 \vdash_m C_m$. The computation is accepting (*resp.* rejecting) if C_m is an accepting (*resp.* rejecting) configuration.

Reduction: First we recall a version of the membership problem for Turing machines.

Input: a Turing machine M and an input word x .

Output: Yes, if M accepts x and erases the tape contents; No, otherwise.

Given input M and x , let q_{in} and q_{accept} be the initial and accepting state of M respectively. We can define a singleton relation $R = \{(q_{in}x, q_{accept})\}$. Let S be the \vdash relation over configurations of M . Both R and S are rational relations.

R has a bounded index in $S^{(*)}$ iff there exists some $k \geq 0$ such that $q_{in}x \vdash_k q_{accept}$. This is equivalent to saying that R has a bounded index in $S^{(*)}$ iff M accepts x and erases the tape contents. Since the membership problem for the Turing machine is undecidable (Kozen, 1997), we can conclude that it is undecidable to check if a rational

relation has a bounded index in the composition closure of a rational relation. \square

However, we show that the index problem is decidable w.r.t. a large class of rational relations defined below.

Definition 4.20 (Metrizable Rational Relation)

Let S be a **rational relation** over $A^* \times A^*$. Let $d_S : A^* \times A^* \rightarrow \mathbb{N} \cup \{\infty\}$ be the distance between two vertices in the graph of S , i.e., for any two words u and v , $d_S(u, v)$ is the smallest i such that $v \in S^{(i)}(u)$, and ∞ otherwise.

We say S is a **d -metrizable** rational relation for a **metric** d if $d_S \approx d$.

Proposition 4.21

Let R be a **rational relation** and S be a **d -metrizable** rational relation for an **integer-valued metric** d for which $d_{len} \lesssim d$. If **bounded diameter** w.r.t. d is decidable for a **rational relation**, then $\text{Index}(R, S)$ is computable as well as the **bounded index** and **k -bounded index** of R in **composition closure** of S are decidable.

Proof. Similar to the **distance** problem, the index problem is computable iff the bounded index and k -bounded index problem are decidable. For a rational relation R and d -metrizable rational relation S , we show that $\text{Index}(R, S) < \infty$ iff $\text{dia}_d(R) < \infty$ as follows.

$$\begin{aligned}
 \text{dia}_d(R) < \infty &\iff \exists k \in \mathbb{N} \text{ s.t. } \forall (u, v) \in R, d(u, v) \leq k \\
 &\iff \exists k' \in \mathbb{N} \text{ s.t. } \forall (u, v) \in R, d_S(u, v) \leq k' \quad (\text{Since } d_S \approx d) \\
 &\iff \forall (u, v) \in R, v \in S^{\leq(k')}(u) \quad (\text{Definition 4.20 of } d_S) \\
 &\iff \text{Index}(R, S) < \infty.
 \end{aligned}$$

Thus, if the boundedness of diameter w.r.t. d is decidable for a rational relation, then we can decide if $\text{Index}(R, S) < \infty$. If so, then it suffices to decide if $\text{Index}(R, S) \leq k$ for $k = 0, 1, \dots$ and output the smallest k as the index of R in the composition closure of S .

Recall that a relation has a **bounded delay** if the relation has a bounded diameter w.r.t. d_{len} . Since $\text{dia}_d(R) < \infty$ and $d_{len} \lesssim d$, the rational relation R has a bounded delay.

Similarly, S also has a bounded delay since for all $(u, v) \in S$,

$$d_S(u, v) = 1 \quad (\text{By Definition 4.20 of } d_S) \quad (4.3)$$

$$\Rightarrow \exists k \in \mathbb{N} \text{ s.t. } d(u, v) \leq k \quad (\text{Since } d_S \approx d) \quad (4.4)$$

$$\Rightarrow \exists k' \in \mathbb{N} \text{ s.t. } d_{len}(u, v) \leq k' \quad (\text{Since } d_{len} \lesssim d) \quad (4.5)$$

Since S has bounded delay, for any $k \in \mathbb{N}$, $S^{\leq(k)}$ is also a rational relation with bounded delay. It is shown by Frougny and Sakarovitch (1991) (Corollary 2) that rational relations with bounded delay are closed under set difference. Also, checking whether a rational relation is empty is decidable (Berstel, 1979). For any $k \in \mathbb{N}$, deciding $\text{Index}(R, S) \leq k$ reduces to checking if $R \subseteq S^{\leq(k)}$ (or equivalently, $R \setminus S^{\leq(k)} = \emptyset$), and hence decidable. \square

Since the boundedness of diameter w.r.t. a metric d listed in Table 1.1 is decidable (see Theorem 4.15), and $d_{len} \lesssim d$ (see Lemma 3.17), we get the following result by applying Proposition 4.21.

Theorem 4.22

The [index](#), [bounded index](#) and [k-bounded index](#) problem stated in Table 4.3 are decidable for a [rational relation](#) in a [composition closure](#) of a [d-metrizable](#) rational relation for a [metric](#) d given in Table 1.1.

A close and (almost) dual notion to metrizable rational relation is that of a metric that defines a rational relation.

Definition 4.23 (Rationalizable Distance)

A distance d on words [rationalizable](#) if the relation $S_d = \{(u, v) \mid d(u, v) = 1\}$, called the [distance relation](#) of d , is rational.

Example 4.24

Consider the [Hamming](#) distance d_h . We can construct a rational relation $S_h = \{(u, v) \mid u \text{ and } v \text{ differ only in exactly one position}\}$. For example, let $A = \{a, b\}$ and $S_h(aba) = \{bba, aaa, abb\}$. For this, construct a [transducer](#) that nondeterministically chooses a position and replaces the input letter with other letters in the alphabet.

In fact, we have the following result about the rationalizability of edit distances referred in Table 1.1.

Proposition 4.25

Every edit distance d given in Table 1.1 is rationalizable.

Proof. Let d be an edit distance. Let C be the set of permissible edits in d . We can construct a rational relation S such that $S = \{(u, v) \mid d(u, v) = 1\}$. For $d \in \{d_l, d_h, d_t, d_{lcs}, d_{dl}\}$, we can construct a transducer that nondeterministically chooses a position and an edit from set C and applies that edit to the chosen position. For example, the rational relation S_{d_l} over $A^* \times A^*$ for the Levenshtein distance is as follows. For all, $u, v \in A^*$, $(u, v) \in S_{d_l}$ if and only if there exists $x, y \in A^*$, $a, b \in A$ with $a \neq b$, such that either one of the following is true.

1. $u = xy, v = xay$ (insertion).
2. $u = xay, v = xy$ (deletion).
3. $u = xay, v = xby$ (substitution).

For conjugacy distance d_c , the distance relation S_{d_c} consists of all pairs (u, v) such that either $u = ax$ and $v = xa$, or $u = xa$ and $v = ax$ for $x \in A^*$, $a \in A$. For this, we can construct a transducer that nondeterministically remembers (or guesses) the first (or last) letter of the input word and outputs it at the end (or beginning) along with the rest of the input word. \square

Therefore, the distance relation of all the metrics listed in Table 1.1 are rational, and hence a metrizable rational relation. Therefore, we get the decidability for the problems in Table 4.3 for rational relation in a composition closure of the rationalizable distance d given in Table 4.3. In the context of d -metrizable relations, the diameter and index of a relation are equivalent up to boundedness, while they are exactly the same in the context of rationalizable distances, as stated in the following proposition.

Proposition 4.26

The **diameter** of a **rational relation** R w.r.t. a **rationalizable** distance d is equal to the **index** of the **rational relation** R in the **composition closure** of the **distance relation** of d .

Proof. Assume that the diameter of a relation R w.r.t. a distance d is ∞ . We claim that the index of R in $S_d^{(*)}$ is also ∞ where S_d is the distance relation of d . Suppose not, i.e., let $k < \infty$ be the index of R in $S_d^{(*)}$. Thus, $\forall (u, v) \in R, v \in S_d^{\leq(k)}(u)$. Since S_d is the distance relation of d , $\forall (u, v) \in R, d(u, v) \leq k$. However, this contradicts the fact that $\text{dia}_d(R) = \infty$. Hence, the index of R in $S_d^{(*)}$ is infinite. Similarly, we can prove the other direction. Now, suppose the diameter of R w.r.t. d is finite, i.e., for some $k \in \mathbb{N}$,

$$\begin{aligned} \text{dia}_d(R) = k &\iff \forall (u, v) \in R, d(u, v) \leq k \text{ and } \exists (u, v) \in R, d(u, v) = k \\ &\iff \forall (u, v) \in R, v \in S_d^{\leq(k)}(u) \text{ and } \exists (u, v) \in R, v \in S_d^{(k)}(u) \\ &\iff \text{Index}(R, S_d) = k. \end{aligned}$$

□

4.5 Conclusion

We study the problems stated in Tables 4.1, 4.2 and 4.3 and show that they are decidable for the **metrics** in Table 1.1. The **index** and **bounded index** problems stated in Table 4.3 are undecidable in general for **rational relations** (see Lemma 4.19), but they are decidable in the composition closure of **d -metrizable** rational relations. This decidability follows from a correspondence with the **diameter** problems, which in turn, are shown to be decidable via their interreductions to the **distance** problems. Recall that, w.r.t. a metric d , distance problem is computable iff both **closeness** and **k -closeness** are decidable (see Proposition 4.9). It remains to show the decidability of closeness and k -closeness from Table 4.1. They are addressed in Chapter 6, where their decidability is shown using results from Chapter 5.

4.6 Notes

A problem that is similar in spirit to the **distance** problem is the *robustness problem*. We say a **transducer** \mathcal{T} is *robust* w.r.t. a distance d if there is a nontrivial relation R

between the distance between two input words and distance between their corresponding outputs on \mathcal{T} . For instance, R could be *Lipschitz continuity* — there is some $k > 0$ such that for all $u, v \in \text{dom}(\mathcal{T})$, $d(\mathcal{T}(u), \mathcal{T}(v)) \leq k \cdot d(u, v)$, or *locally Lipschitz continuity* — there exists $b, k > 0$ such that for all $u, v \in \text{dom}(\mathcal{T})$ if $d(u, v) < b$ then $d(\mathcal{T}(u), \mathcal{T}(v)) \leq k \cdot d(u, v)$, etc. Sometimes, weaker notions of distance are considered (for instance by dropping the [triangle inequality](#)), and respective distances are called *cost* or *similarity* functions. The work by [Samanta et al. \(2013\)](#) solves the locally Lipschitz continuity problem for [sequential](#) and [unambiguous](#) transducers using reversal bounded counter automata. The problem is shown to be undecidable for Lipschitz continuity even for sequential transducers and the decidability is shown for the class that has a bound on the delay between input and output words ([Henzinger et al., 2014](#)). The robustness problems study the relationship between the distance between two input words and their corresponding outputs of a transducer, while we study the distance between the outputs of two transducers over any input. In fact, this allows us to compare two transducers based on their output words produced.

A problem related to the [diameter](#) of a [rational relation](#) is *reflexivity* of rational relations. A relation $R \subseteq A^* \times A^*$ is k -reflexive, for some integer $k \in \mathbb{N}$, if every element u in the domain of R is at a distance at most k from some element of the range v , with $(u, v) \in R$, and vice versa. A relation is almost reflexive if $k < \infty$. The bounded diameter of a relation R asks if there exists an integer $k \in \mathbb{N}$ such that every element u in the domain is at most k distance from *every* element v in its range with $(u, v) \in R$ and vice-versa. Reflexivity has shown to be undecidable for rational relations ([Johnson, 1986](#)), and even for [sequential](#) functions ([Choffrut and Pighizzini, 2002](#)) with respect to classical distances including [Hamming](#) and [Levenshtein](#) distance.

In 1966, Brzozowski raised the question of *finite power property* on regular languages — it takes a regular language L as input and asks whether there exists some positive integer n such that $(L + \varepsilon)^n = L^*$. It was solved in 1979 by [Hashiguchi \(1979\)](#) and [Simon \(1978\)](#), independently. We studied the [bounded index](#) (or finite index) property of a rational relation in the iterative [composition](#) of another relation. Notice that the finite index property is different from the finite power property in two respects. One, it is over relations and not languages, and secondly and more importantly, the iteration is obtained by relation composition and not concatenation.

CHAPTER 5

Deciding Conjugacy of a Rational Relation

Contents

| | | |
|-------|--|-----|
| 5.1 | Introduction | 60 |
| 5.2 | Conjugacy of a Rational Relation | 62 |
| 5.2.1 | Conjugacy of a Sumfree Expression | 63 |
| 5.3 | Common Witness Theorems | 66 |
| 5.3.1 | Common Witness and its Characterisations | 66 |
| 5.3.2 | Common Witness Theorem for Kleene Closure | 72 |
| 5.3.3 | Common Witness Theorem for Monoid Closure | 73 |
| 5.4 | Existence of Common Witness for Kleene Closure | 78 |
| 5.4.1 | For a Finite Set of Pairs | 79 |
| 5.4.2 | For an Infinite Set of Pairs | 84 |
| 5.5 | Existence of Common Witness for Monoid Closure | 85 |
| 5.5.1 | Common Witness of a Singleton Redux | 85 |
| 5.5.2 | Common Witness of a Sumfree Set | 95 |
| 5.6 | Computing Witness of a Sumfree Expression | 106 |

¹An extended abstract of the contributions presented in this chapter is published in the proceedings of DLT 2024 under the title “[Deciding conjugacy of a rational relation](#)”. This is a joint work with Dr. C. Aiswarya and Dr. Amaldev Manuel.

| | |
|--------------------------|-----|
| 5.7 Conclusion | 111 |
| 5.8 Notes | 111 |

5.1 Introduction

Conjugacy of two elements u and v in a group² can be defined as any of the following equivalent cases:

1. $uz = zv$ for some z ,
2. $u = xy$ and $v = yx$ for some x, y .

The conjugacy problem asks if a given pair of elements in a finitely presented group (typically infinite) is conjugate. It along with the word and isomorphism problems constitutes the classical triad of decision problems on groups identified by Dehn in 1912 (Peifer, 2015). Dehn’s prescient choice turned out to be instrumental not only in mathematics, but also in the theory of semigroups/monoids and automata in computer science. It turns out that the above conditions are equivalent for free monoids (i.e., when u, v, z, x, y are taken to be words over some finite alphabet). This is the well-known second theorem of Lyndon-Schützenberger (See Theorem 3.6). But unlike in the case of groups where condition (1) is taken to be the definition of conjugacy, in the case of monoids condition (2) is taken as the definition of conjugacy.

Conjugacy problem is solvable in polynomial time over free monoids and free groups. We consider a generalisation of the problem to a finitely-presented possibly infinite set of pairs of words. In this chapter, we address the decidability of the following fundamental question:

1. Given a *rational relation* R , are all the pairs of words in R conjugates?

A set of pairs of words (or relation) is said to be *conjugate* if every pair in the set is conjugate. This is also known as *conjugacy of a relation*. Checking the conjugacy of a rational relation is equivalent to checking if all the input-output pairs of a relation defined by a *nondeterministic finite state transducer* are conjugates. Checking a number of properties of word transducers, for instance *sequentiality* (can the given transducer

²A group is a monoid where each element also has an inverse.

be [determinised](#)?) or finite sequentiality (is the given transducer equivalent to a disjoint union of [sequential](#) transducers?), bounded edit-distance (is the [edit distance](#) between the respective outputs of the given transducers finite?) etc. amounts to checking conjugacy of the rational relations defined by the strongly connected components of the transducer and certain specific properties of the underlying acyclic graph of strongly connected components. Loosely speaking, conjugacy of the relations defined by the strongly connected components imply that the loops of the transducer are pumpable. Likewise, given two [functional](#) transducers with identical domain, checking if the output pairs of these transducers are conjugates on all input, reduces to checking conjugacy of the rational relation of output pairs defined by their [output product](#) transducer. A converse result also holds: checking conjugacy of rational relation is equivalent to checking conjugacy of output pairs of two functional transducer with identical domain by virtue of Nivat's theorem (See [Theorem 4.14](#)).

We provide a definitive answer to Question 1 by introducing the concept of a common witness of a relation. A *witness* of a conjugate pair (u, v) is a word z such that either $uz = zv$ (*inner witness*) or $zu = vz$ (*outer witness*). Succinctly, a word z is a *common inner (resp. outer) witness* of a relation, if for every pair (u, v) in the relation, z is an inner witness (*resp.* outer witness) of (u, v) . We show that a rational relation is conjugate if and only if each of its sumfree rational components has a common witness, i.e., either a common inner witness or a common outer witness. This characterisation of conjugacy is a main contribution of the chapter. It is in fact a generalisation of Lyndon-Schützenberger theorem characterising conjugacy of two words.

Subsequently, when dealing with a rational relation R , there are two interesting questions regarding the common witness:

2. *Is there a common witness for the relation R ?*
3. *Given a word z , is it a common witness of R ?*

Question 3 proves to be comparatively more tractable, as it can be reduced to verifying whether the rational relation $R' = \{(uz, zv) \mid (u, v) \in R\}$ (or, $R' = \{(zu, vz) \mid (u, v) \in R\}$) is a subset of [identity relation](#). In fact, the decidability of the [twinning property](#) of a transducer is connected to Question 3. It is further elaborated in [Section 5.8](#).

Question 2, on the other hand, is more difficult *a priori* as we do not have a bound on the size of a possible common witness. We provide a decision procedure for Question 2.

This is another contribution of the chapter. Our characterisation of conjugacy via common witness, together with this procedure, yields an algorithm for deciding conjugacy.

5.2 Conjugacy of a Rational Relation

We assume that the **rational relation** is given as a **rational expression** over the **monoid** $A^* \times B^*$. Furthermore, if there is a pair in the relation containing a letter in the symmetric difference of A and B , then the pair as well as the relation is not conjugate. Since this can be easily checked, the nontrivial part of the problem is when the alphabets are identical, i.e., when $A = B$. Thus, we assume that the given rational expression is over $A^* \times A^*$ for a fixed finite alphabet A . A rational expression E is **conjugate** if the relation it defines is conjugate, i.e., for all $(u, v) \in L(E)$, u and v are **conjugate**.

Example 5.1

The rational expression $E_1 = (\varepsilon, a)(ab, ba)^*(a, \varepsilon)$ denotes the relation $\{((ab)^n a, a(ba)^n) \mid n \geq 0\}$. The expression $E_2 = ((a, aa) + (b, \varepsilon))^*$ represents $\{(u, v) \mid v \text{ is obtained from } u \text{ by duplicating } a\text{'s and discarding } b\text{'s}\}$. The expression E_1 is conjugate, but E_2 is not.

Our main result is summarised by the following theorem.

Theorem 5.2

Conjugacy of rational relations is decidable.

The union operation of rational relations preserves conjugacy, i.e., if R_1 and R_2 are conjugate, then $R_1 \cup R_2$ is also conjugate. However, conjugacy is not preserved under the product and Kleene closure.

Example 5.3

Consider $E_1 = (ab, ba)$, $E_2 = (ca, ac)$ and $E_3 = (ac, ca)$, all of which are conjugates. $E_1 \cdot E_3$ is conjugate but $E_1 \cdot E_2$ is not. Also $(E_1 + E_3)^*$ is conjugate, but $(E_1 + E_2)^*$ is not.

Since every rational expression can be written as a sum of **sumfree expressions** (those that does not use the union) (See [Lemma 2.9](#)), for proving [Theorem 5.2](#) we only need to decide the conjugacy of a rational relation given by a sumfree expression.

In the rest of this section, we give an overview of determining the conjugacy of sumfree expressions.

5.2.1 Conjugacy of a Sumfree Expression

Recall that the family \mathcal{F} of sumfree expressions over a monoid $\mathbf{M} = (M, \cdot, 1)$ is defined inductively as $\mathcal{F} = \bigcup_{i \geq 0} \mathcal{F}_i$, where $\mathcal{F}_0 = M \cup \{\emptyset\}$ and $\mathcal{F}_{i+1} = \mathcal{MK}\mathcal{F}_i$ for each $i \geq 0$.

We now proceed to solve the conjugacy of sumfree expressions. We use pairs of lowercase Greek letters (α, β) with suitable modifications to denote pairs of words over $A^* \times A^*$. Clearly \emptyset and $(\varepsilon, \varepsilon)$ are conjugates. For an expression of the form (α, β) , it is straightforward to check conjugacy (see [Proposition 3.7](#)). Thus, the conjugacy problem is decidable for the class of expressions \mathcal{F}_0 .

To show the decidability of the conjugacy problem for the whole family \mathcal{F} , it suffices to show that if the problem is decidable for \mathcal{F}_i , $i \geq 0$, then it is also decidable for $\mathcal{K}\mathcal{F}_i$ and $\mathcal{F}_{i+1} = \mathcal{MK}\mathcal{F}_i$. By induction on i the decidability extends to the whole family \mathcal{F} .

Assume that conjugacy is decidable for \mathcal{F}_i . Let E be an expression in \mathcal{F}_i and hence $E^* \in \mathcal{K}\mathcal{F}_i$. Since $L(E) \subseteq L(E^*)$,

Proposition 5.4

If the expression E^* is conjugate, then E is conjugate.

Because conjugacy is decidable for \mathcal{F}_i , we can check whether E is conjugate. Therefore, to show the decidability of conjugacy for $\mathcal{K}\mathcal{F}_i$, it suffices to show the decidability of the following question.

Question 1 (Conjugacy of Kleene Closures). *Given a conjugate sumfree expression E , is E^* conjugate?*

Next, assume that conjugacy is decidable for $\mathcal{K}\mathcal{F}_i$. For $k > 0$, let

$$E = (\alpha_0, \beta_0)E_1^*(\alpha_1, \beta_1) \cdots (\alpha_{k-1}, \beta_{k-1})E_k^*(\alpha_k, \beta_k) \quad (5.1)$$

5. DECIDING CONJUGACY OF A RATIONAL RELATION

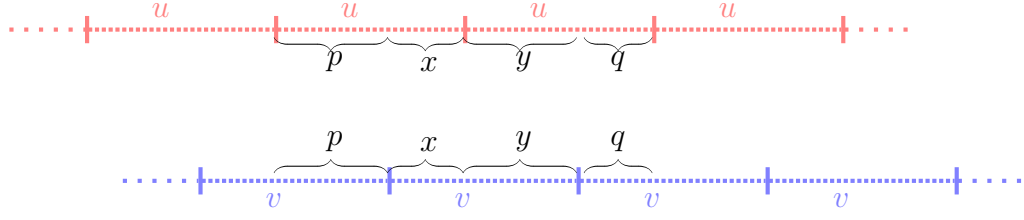


Fig. 5.1. v as infix of uu .

be an expression in \mathcal{MKF}_i where E_1^*, \dots, E_k^* are from \mathcal{KF}_i . Analogous to the case of Kleene closures, E is conjugate only if E_1^*, \dots, E_k^* are conjugate, as shown below.

Lemma 5.5

If the expression $E = (\alpha_0, \beta_0)F^*(\alpha_1, \beta_1)$ is conjugate, then F^* is conjugate.

Proof. If F^* is an empty set, then it is conjugate. Otherwise, assume that (u, v) is a nonempty pair that belongs to F^* . Therefore, (u^ℓ, v^ℓ) for each $\ell \geq 0$ also belongs to F^* . We can safely assume that $|u| = |v|$, otherwise each iteration will increase the difference in length between u^ℓ and v^ℓ , leading to nonconjugacy of E .

Let $k = |\alpha_0| + |\beta_0| + |\alpha_1| + |\beta_1|$. Consider the pair $(\alpha_0, \beta_0)(u^\ell, v^\ell)(\alpha_1, \beta_1)$ where $\ell = 2^k$ (some value much larger than k). Since ℓ is much larger than k and $(\alpha_0 u^\ell \alpha_1, \beta_0 v^\ell \beta_1)$ is conjugate, there exist large factors of u^ℓ and v^ℓ that match as shown in Figure 5.1. Since $|u| = |v|$, we can infer that u is a factor of vv , and v is a factor of uu .

Since v is an factor of uu , the following holds as shown in Figure 5.1. There exist words x, y, p , and q such that $v = xy$ and $u = px = yq$. Since $|u| = |v|$, length of p and length of y are the same, that implies $p = y$ (since $u = px = yq$). Therefore, $u = yx$. Hence u and v are conjugate words. Since the pair (u, v) was arbitrary, F^* is conjugate. \square

We can generalize the above lemma to the general form of sumfree expressions.

Corollary 5.6

If the expression $E = (\alpha_0, \beta_0)E_1^*(\alpha_1, \beta_1) \cdots (\alpha_{k-1}, \beta_{k-1})E_k^*(\alpha_k, \beta_k)$, for $k > 0$, is conjugate, then each of $E_1^*, E_2^*, \dots, E_k^*$ is conjugate.

Proof. If E is conjugate, then for each $i \in \{1, \dots, k\}$,

$$(\alpha_0 \cdots \alpha_{i-1}, \beta_0 \cdots \beta_{i-1})E_i^*(\alpha_i \cdots \alpha_k, \beta_i \cdots \beta_k) \subseteq E$$

is conjugate. Therefore, from [Lemma 5.5](#) we get that each of E_1^*, \dots, E_k^* is conjugate. \square

Since the conjugacy of \mathcal{KF}_i is decidable, we can check whether E_1^*, \dots, E_k^* are conjugate expressions. Thus, to show the decidability of \mathcal{MKF}_i , it suffices to show the decidability of the following question.

Question 2 (Conjugacy of Monoid Closures). *Given conjugate sumfree expressions E_1^*, \dots, E_k^* , is the expression $E = (\alpha_0, \beta_0)E_1^*(\alpha_1, \beta_1) \cdots E_k^*(\alpha_k, \beta_k)$ conjugate?*

We show that [Question 1](#) and [Question 2](#) can be effectively answered. The idea is to use the notion of common witness that we mentioned in the beginning (further elaborated in [Definition 5.11](#)). In [Section 5.3](#), we present two common witness theorems that address the above questions:

1. Let G be an arbitrary set of conjugate pairs. The set G^* is conjugate if and only if G has a common witness ([Theorem 5.17](#)).
2. Let G_1^*, \dots, G_k^* , $k > 0$, be arbitrary sets of conjugate pairs. The set

$$(\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1) \cdots (\alpha_{k-1}, \beta_{k-1})G_k^*(\alpha_k, \beta_k),$$

called a *sumfree set*, is conjugate if and only if it has a common witness ([Theorem 5.26](#)).

Remark 5.7. *Note that the assumption of conjugacy of the sets G, G_1^*, \dots, G_k^* is not necessary. However, if they are not conjugate then the corresponding sets will neither have a common witness nor be conjugate, and the statements will be vacuously true (Since [Proposition 5.4](#) and [Corollary 5.6](#) also hold for arbitrary sets).*

[Item 2](#) is a generalisation of [Item 1](#), and its proof relies on [Item 1](#). Both theorems are generalisations of the Lyndon-Schützenberger theorem. We present in detail the above common witness theorems for addressing [Question 1](#) and [Question 2](#), along with the proofs of the easier directions in [Section 5.3](#). However, the difficult directions require a detailed case analysis, and we complete the proof of common witness theorems for

Kleene closure and monoid closure in Section 5.4 and Section 5.5 respectively. When G, G_1^*, \dots, G_k^* are rational sumfree expressions of pairs, the above theorems are *effective*, that is a common witness, if exists, is computable in polynomial time in the length of the expression, which is outlined in Section 5.6. Thus, we get a decision procedure proving our main theorem (Theorem 5.2).

5.3 Common Witness Theorems

In this section, it is shown that an infinite set of pairs that is generated by a **sumfree set** is **conjugate** if and only if there is a word witnessing its conjugacy.

5.3.1 Common Witness and its Characterisations

We have already seen from the second theorem of Lyndon-Schützenberger (Theorem 3.6) that if two words u and v are conjugates, then there exist words z, z' such that $uz = zv$ and $z'u = vz'$. This leads to the following notion.

Definition 5.8 (Inner and Outer Witness)

Given a conjugate pair (u, v) , the word z is an *inner witness* of (u, v) if $uz = zv$. Similarly, z is an *outer witness* of (u, v) if $zu = vz$. A pair of words has a *witness* if it has either an inner witness or an outer witness.

Given a conjugate pair (u, v) , the set of all inner witnesses of (u, v) is

$$\{z \mid uz = zv\} = \cup_{\{(x,y) \mid u=xy, v=yx\}} (xy)^* x.$$

Similarly, the set of all outer witnesses of (u, v) is

$$\{z \mid zu = vz\} = \cup_{\{(x,y) \mid u=xy, v=yx\}} (yx)^* y.$$

An inner witness of a pair (u, v) is an outer witness of the pair (v, u) .

Example 5.9

The pair (aba, baa) has inner witnesses $(aba)^*a$ and outer witnesses $(baa)^*ba$.

The following proposition shows that a pair and its primitive root share the same set of witnesses.

Proposition 5.10

Let (u, v) be a pair such that $(u, v) \in (\rho_u, \rho_v)^*$. The following are equivalent for a word z .

1. z is an **inner** (resp. **outer**) witness of (u, v) .
2. z is an **inner** (resp. **outer**) witness of (ρ_u, ρ_v) .

Proof. We prove $(1) \iff (2)$ for inner witness. The outer witness case is symmetric.

$(2) \Rightarrow (1)$: Assume that z is an inner witness of (ρ_u, ρ_v) . By induction on n , we prove that z is also an inner witness of $(\rho_u, \rho_v)^n$ for all $n \geq 1$, i.e., $\rho_u^n z = z \rho_v^n$. It is true when $n = 1$. For all $n > 1$,

$$\begin{aligned}
 \rho_u^n z &= \rho_u^{n-1} \rho_u z \\
 &= \rho_u^{n-1} z \rho_v && \text{(Since } \rho_u z = z \rho_v \text{)} \\
 &= z \rho_v^{n-1} \rho_v && \text{(Inductive Hypothesis)} \\
 &= z \rho_v^n
 \end{aligned}$$

Since $(u, v) \in (\rho_u, \rho_v)^*$, z is also an inner witness for (u, v) .

$(1) \Rightarrow (2)$: By [Theorem 3.6](#), (u, v) is conjugate, and hence its primitive root (ρ_u, ρ_v) is conjugate as well by [Proposition 3.9](#). Consider the case when $u \neq v$. It follows that $\rho_u \neq \rho_v$. According to [Proposition 3.12](#), the pair (ρ_u, ρ_v) has a unique **cut**, denoted as (x, y) . From [Proposition 3.14](#), all cuts of (u, v) are of the form $((xy)^*x, (yx)^*y)$. From [Theorem 3.6](#), an inner witness of (u, v) belongs to

$$((xy)^*x(yx)^*y)^*(xy)^*x = (xy)^*x$$

and hence is an inner witness of (ρ_u, ρ_v) .

In the case where $u = v$, it follows that $\rho_u = \rho_v$. Consequently, any witness z for the pair (u, u) belongs to the set u^* that is a subset of ρ_u^* . Thus, z is also a witness for the pair (ρ_u, ρ_v) , since ρ_u^* consists of witnesses of (ρ_u, ρ_v) . \square

5. DECIDING CONJUGACY OF A RATIONAL RELATION

We generalise the notion of a witness of a pair to a set of pairs.

Definition 5.11 (Common Witness)

A word is a *common inner witness* of a set of pairs P if it is an *inner witness* of each pair in P . Similarly, a word is a *common outer witness* of P if it is an *outer witness* of each pair in P .

A set of pairs has a *common witness* if it has either a common inner witness or a common outer witness. A *rational expression* E has a common witness if the set $L(E)$ has a common witness.

The structure of a common witness of a set of pairs is obtained from Theorem 3.6.

Proposition 5.12

Let P be a set of pairs of words. The following are equivalent.

1. z is a *common inner witness* of P .
2. There exists a *cut* (x, y) of each pair $(u, v) \in P$ such that $z \in \bigcap_{(u,v) \in P} (xy)^*x$.
3. $z \in \bigcap_{(u,v) \in P} \bigcup_{\{(x,y) \mid u=xy, v=yx\}} (xy)^*x$

The statement for *common outer witness* is analogous.

Proof. Follows from Definition 5.11 and Theorem 3.6. □

Example 5.13

Consider the set $P = \{(ab, ba), (abab, baba)\}$. The pair (ab, ba) has a unique cut (a, b) , and the pair $(abab, baba)$ has two cuts: (a, bab) and (aba, b) . The word a is a common inner witness of P since a belongs to both $(ab)^*a$ and $(abab)^*a$ (using the first cut). Similarly, aba is also a common inner witness of P since aba belongs to both $(ab)^*a$ and $(abab)^*aba$ (using the second cut). Notice that aba is not in the intersection of $(ab)^*a$ and $(abab)^*a$.

Proposition 5.10 connecting witness of a conjugate pair and its root can be lifted to a set of conjugate pairs and its root, i.e., the common witnesses of a set of conjugate pairs G and its primitive root $R(G)$ are the same as follows.

Proposition 5.14

Let G be a set of pairs of words such that $G \subseteq R(G)^*$. The following are equivalent for a word z .

1. z is a **common inner** (resp. **common outer**) witness of G .
2. z is a **common inner** (resp. **common outer**) witness of $R(G)$.

Proof. We prove for common inner witness. The common outer witness case is symmetric.

z is a common inner witness of G

$$\iff z \text{ is an inner witness of } (u, v) \text{ for each } (u, v) \in G$$

$$\iff z \text{ is an inner witness of } (\rho_u, \rho_v) \text{ for each } (u, v) \in G \quad (\text{By Proposition 5.10})$$

$$\iff z \text{ is a common inner witness of } R(G). \quad \square$$

When a set is not conjugate, clearly it has no common witness. However, even when a set is conjugate, it may have both common inner and outer witnesses, or only common inner witness, or only common outer witness, or neither of them as shown below.

Example 5.15

Consider the set $P = \{(ab, ba), (ac, ca)\}$. The pair (ab, ba) has inner witnesses $(ab)^*a$ and outer witnesses $(ba)^*b$. Similarly, the pair (ac, ca) has inner witnesses $(ac)^*a$ and outer witnesses $(ca)^*c$. According to Proposition 5.12, the set P has a unique common inner witness $a = (ab)^*a \cap (ac)^*a$, but it does not have any common outer witness since $(ba)^*b \cap (ca)^*c = \emptyset$.

The set $\{(ab, ba), (abab, baba)\}$ has both common inner witnesses

$$(ab)^*a = (ab)^*a \cap ((abab)^*aba \cup (abab)^*a)$$

and common outer witnesses

$$(ba)^*b = (ba)^*b \cap ((baba)^*b \cup (baba)^*bab).$$

However, the set $\{(ab, ba), (ba, ab)\}$ has no common witnesses since $(ab)^*a \cap (ba)^*b = \emptyset$.

Next we analyse the number of common witnesses a set of pairs can have.

Proposition 5.16

Let G be a set of pairs of words with $G \subseteq R(G)^*$. The following are equivalent.

1. G has more than one common witness.
2. G has infinitely many common witnesses.
3. G has infinitely many common inner witnesses.
4. G has infinitely many common outer witnesses.
5. All the pairs in G have the same primitive root.

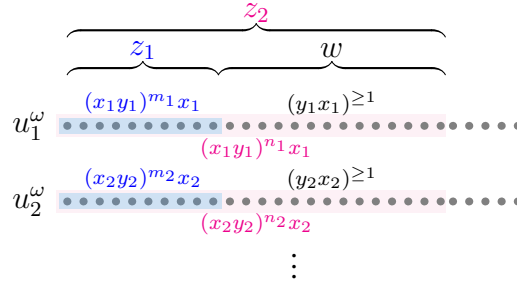
Proof. We prove $(1) \Rightarrow (2) \Rightarrow (1)$ and $(1) \Rightarrow (5) \Rightarrow (3), (4) \Rightarrow (1)$.

$(2) \Rightarrow (1)$ and $(3), (4) \Rightarrow (1)$ is obvious. We first show $(5) \Rightarrow (3), (4)$. Since all the pairs in G have the same primitive root, $R(G)$ is singleton. Thus, $R(G)$ has infinitely many common inner as well as common outer witnesses by Theorem 3.6. Since G and $R(G)$ have the same witnesses (Proposition 5.14), G also has infinitely many common inner and common outer witnesses.

Now it remains to show that $(1) \Rightarrow (2)$ and $(1) \Rightarrow (5)$. If the set G has two common witnesses, namely z_1 and z_2 , then according to Proposition 5.14, z_1 and z_2 are also common witnesses of $R(G)$. Let $R(G) = \{(u_i, v_i) \mid i \in I\}$ where I is an Index set.

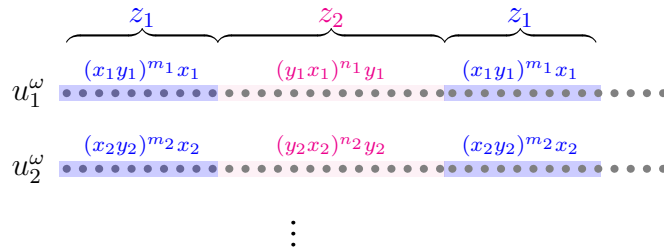
Suppose z_1 and z_2 are two common inner witnesses of $R(G)$. We choose cuts (x_i, y_i) for each $(u_i, v_i) \in R(G)$ such that both z_1 and z_2 belongs to $\bigcap_{i \in I} (x_i y_i)^* x_i$. Since $R(G)$ is a set of primitive pairs, (x_i, y_i) is the unique cut of (u_i, v_i) when $u_i \neq v_i$ by Proposition 3.12. When $u_i = v_i$, it has only two empty cuts, namely (ε, u_i) and (u_i, ε) (if $u_i = v_i$). The inner witnesses obtained using cut (ε, u_i) is a superset of inner witnesses obtained using (u_i, ε) . Hence, we can choose cut $(x_i, y_i) = (\varepsilon, u_i)$ for pair (u_i, v_i) when $u_i = v_i$. Therefore, as stated in Proposition 5.12, both z_1 and z_2 belongs to $\bigcap_{i \in I} (x_i y_i)^* x_i$.

Without loss of generality, assume that $|z_1| < |z_2|$. As depicted in Figure 5.2, a common factor $w \in \bigcap_{i \in I} (y_i x_i)^{\geq 1}$ exists for each u_i^ω that can be repeated one after another in u_i^ω to get longer and longer common inner witnesses. By symmetry, when z_1 and z_2 are both common outer witnesses of $R(G)$, we get infinitely many common outer witnesses.


 Fig. 5.2. When there are at least two common inner witnesses z_1, z_2 .

Now, WLOG assume that z_1 is a common *inner* witness and z_2 is a common *outer* witness of $R(G)$, where $z_1 \neq z_2$. Here also we choose cuts (x_i, y_i) for each $(u_i, v_i) \in R(G)$ such that z_1 belongs to $\bigcap_{i \in I} (x_i y_i)^* x_i$ and z_2 belongs to $\bigcap_{i \in I} (y_i x_i)^* y_i$. For each distinct primitive pair in $R(G)$, there exists a unique cut. However, for identical primitive pairs, we fix a cut based on the values of z_1 and z_2 . We consider two cases: either $z_1, z_2 \neq \varepsilon$, or exactly one of z_1 and z_2 is equal to ε . In the case where $z_1, z_2 \neq \varepsilon$, for primitive pairs (u_i, v_i) such that $u_i = v_i$, we can choose either of the two empty cuts as (x_i, y_i) , resulting in $z_1 \in (x_i y_i)^* x_i$ and $z_2 \in (y_i x_i)^* y_i$. In the second case, if $z_1 = \varepsilon$, we select the cut $(x_i, y_i) = (\varepsilon, u_i)$. This choice ensures that $z_1 \in (x_i y_i)^* x_i$ and $z_2 \in (y_i x_i)^* y_i$ (since $z_2 \neq \varepsilon$). Similarly, if $z_2 = \varepsilon$, we choose the cut $(x_i, y_i) = (u_i, \varepsilon)$. Consequently, we can conclude that z_1 belongs to $\bigcap_{i \in I} (x_i y_i)^* x_i$ and z_2 belongs to $\bigcap_{i \in I} (y_i x_i)^* y_i$.

As shown in Figure 5.3, concatenating $z_1 \cdot z_2 \cdot z_1$ in u_i^ω , we get one more common inner witness z_3 for $R(G)$. By the above argument, $R(G)$ has infinitely many common witnesses. Since witnesses of $R(G)$ are also witnesses of G (Proposition 5.14), it implies that G itself has infinitely many common witnesses. This completes the proof of $(1) \Rightarrow (2)$.


 Fig. 5.3. When there is 1 common inner witness z_1 and 1 common outer witness z_2 .

In both the cases, we get that $(x_1y_1)^\omega = (x_2y_2)^\omega = \dots$ and $(y_1x_1)^\omega = (y_2x_2)^\omega = \dots$. Hence, from Fine and Wilf Theorem ([Theorem 3.10](#)), all u_i 's have the same primitive root. Similarly, all v_i 's has the same primitive root. This proves $(1) \Rightarrow (5)$. \square

Therefore, a set of pairs can have no common witness, a unique common witness, or infinitely many common witnesses.

5.3.2 Common Witness Theorem for Kleene Closure

Lyndon-Schützenberger theorem characterises conjugacy of a pair of words. We generalise the notion in [Theorem 3.6](#) to an infinite set of pairs closed under concatenation. The question we ask is: “Given an arbitrary set of pairs G , is G^* conjugate?”

A problem much related to the above question is the *Conjugate Post Correspondence problem* by [Finkel et al. \(2023\)](#): given a finite set of pairs G , does there exist of a pair $(u, v) \in G^*$ such that u and v are conjugate? This problem is shown to be undecidable by reduction to the word problem of a special type of semi-Thue systems. We show that the universal version of this problem — checking if all the pairs in G^* are conjugate — is decidable.

If G^* has a common witness, then each pair in G^* has a witness and is conjugate. Hence G^* is conjugate. We prove the converse, namely, if G^* is conjugate, then it has a common witness. The below theorem characterises conjugacy of a freely generated set of pairs of words.

Theorem 5.17 (Common Witness Theorem for Kleene Closure)

Let G be an arbitrary conjugate set of pairs of words. The following are equivalent.

1. G^* is conjugate.
2. G^* has a common witness z .
3. G has a common witness z .
4. $R(G)$ has a common witness z .

Proof. We prove $(4) \Rightarrow (3) \Rightarrow (2) \Rightarrow (1) \Rightarrow (4)$. The only nontrivial direction is $(1) \Rightarrow (4)$ that is proved in [Section 5.4](#).

(4) \Rightarrow (3) Since G is conjugate, $G \subseteq R(G)^*$ by [Proposition 3.9](#). Thus, (4) \Rightarrow (3) follows from [Proposition 5.14](#).

(3) \Rightarrow (2) Suppose there exists a common inner witness z of the set G . Hence $\forall (u, v) \in G$, $uz = zv$. Let (u', v') be any arbitrary element from G^* , i.e., $(u', v') = (u_1 \cdots u_n, v_1 \cdots v_n)$ for some $n \geq 1$ and $(u_i, v_i) \in G$ for $1 \leq i \leq n$. By induction on n , we equate $u'z = zv'$ as follows. Thus, z is a common inner witness of G^* . The proof for common outer witness is symmetric.

$$\begin{aligned} u'z &= u_1 \cdots u_{n-1}u_nz \\ &= u_1 \cdots u_{n-1}zv_n && \text{(Since } u_nz = zv_n\text{)} \\ &= zv_1 \cdots v_{n-1}v_n && \text{(Inductive Hypothesis)} \\ &= zv'. \end{aligned}$$

(2) \Rightarrow (1) Follows from [Theorem 3.6](#). □

Remark 5.18. Note that whenever G^* is conjugate even though G and $R(G)$ have the same common witnesses, G^* need not be equal to $R(G)^*$. Indeed, $G^* \subseteq R(G)^*$.

Corollary 5.19

Let E be a [rational expression](#) of pairs. E^* is [conjugate](#) if and only if E^* as well as E has a [common witness](#).

Below is an illustration of the common witness theorem for a set of pairs that is not rational.

Example 5.20

Let $G = \{(ab^p, b^pa) \mid p \text{ is a prime number}\}$. The word $a \in \bigcap_{p \in \mathbb{N}, p \text{ is a prime}} (ab^p)^*a$ is a common inner witness of the set G . It is also easy to verify that G^* is conjugate and a is a common inner witness of G^* .

5.3.3 Common Witness Theorem for Monoid Closure

Next we prove the common witness theorem for monoid closures, i.e., [sumfree sets](#) of the form

$$M = (\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1)G_2^* \cdots (\alpha_{k-1}, \beta_{k-1})G_k^*(\alpha_k, \beta_k), k > 0.$$

where $G_1^*, G_2^*, \dots, G_k^*$ are arbitrary sets of conjugate pairs. We show that such a set is conjugate if and only if it has common witness. Note that this does not generalise to arbitrary sets of pairs, in particular, rational sets using sum.

Example 5.21

$(ab, ba)^* + (ba, ab)^*$ is an infinite conjugate set with *no* common witness.

Definition 5.22 (Redux, Singleton Redux)

Let M be the **sumfree set**

$$(\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1)G_2^* \cdots (\alpha_{k-1}, \beta_{k-1})G_k^*(\alpha_k, \beta_k), k > 0.$$

The **redux** of M is the pair $(\alpha_0 \cdots \alpha_k, \beta_0 \cdots \beta_k)$ obtained by substituting each G_i^* by the empty pair $(\varepsilon, \varepsilon)$. A **singleton redux** of M is a set obtained by substituting all but one of the G_i^* 's by the empty pair $(\varepsilon, \varepsilon)$. They are of the form $(\alpha_0 \cdots \alpha_{i-1}, \beta_0 \cdots \beta_{i-1})G_i^*(\alpha_i \cdots \alpha_k, \beta_i \cdots \beta_k)$ where $1 \leq i \leq k$.

Example 5.23

Consider the set $M = (a, a)(baa, aba)^*(b, a)(aab, baa)^*(a, b)$. The redux of M is (aba, aab) , and its singleton reduxes are $(a, a)(baa, aba)^*(ba, ab)$ and $(ab, aa)(aab, baa)^*(a, b)$.

If a sumfree set has a common witness, it is conjugate. We prove the converse, i.e., if a sumfree set is conjugate, then it has a common witness and that is in the intersection of the common witnesses of the singleton reduxes of the set.

Following is the common witness theorem for a sumfree set with only one Kleene star, i.e., sets of the form $(\alpha_0, \beta_0)G^*(\alpha_1, \beta_1)$. In short it states that such a set is conjugate if and only if it has a common witness that is determined by the common witnesses of $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$.

Theorem 5.24

Let $M = (\alpha_0, \beta_0)G^*(\alpha_1, \beta_1)$ be a **sumfree set** with nonempty **redux** $(\alpha_0\alpha_1, \beta_0\beta_1)$. The following are equivalent.

1. M is **conjugate**.
2. There exist a **common witness** of $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$.
3. M has a **common witness**.

Furthermore,

- (a) If the set $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ has a unique **common inner witness** z' , then M has a unique **common witness**

$$z = [\alpha_0 z', \beta_0]_R = [\alpha_1, z' \beta_1]_L.$$

Moreover, if $|\alpha_0 z'| \geq |\beta_0|$ or equivalently $|\alpha_1| \leq |z' \beta_1|$, then z is a **common inner witness**, otherwise it is a **common outer witness**.

- (b) If the set $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ has a unique **common outer witness** z' , then M has a unique **common witness**

$$z = [\alpha_0, \beta_0 z']_R = [z' \alpha_1, \beta_1]_L.$$

Moreover, if $|z' \alpha_1| \geq |\beta_1|$ or equivalently $|\alpha_0| \leq |\beta_0 z'|$, then z is a **common outer witness**, otherwise it is a **common inner witness**.

- (c) If $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ has infinitely many **common witnesses**, then M is a subset of powers of the **primitive root** of its **redux**. Thus, M has infinitely many **common witnesses**.

The proof of the above theorem as well as the the proof of the general case of sumfree sets below are given in [Section 5.5](#).

Example 5.25

Let $M = (\alpha_0, \beta_0)G^*(\alpha_1, \beta_1)$ be a sumfree set with one Kleene star where

$$\begin{pmatrix} \alpha_0 \\ \beta_0 \end{pmatrix} = \begin{pmatrix} ab \\ b \end{pmatrix}, G = \left\{ \begin{pmatrix} bab \\ abb \end{pmatrix} \right\}, \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} b \\ ab \end{pmatrix}.$$

The redux of M is $(\alpha_0\alpha_1, \beta_0\beta_1) = (abb, bab)$. The set

$$G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\} = \{(bab, abb)\} \cup \{(bab, abb)\} = \{(bab, abb)\}$$

and, hence it has infinitely many common witnesses. By [Theorem 5.24 \(c\)](#), M is a subset of powers of the primitive root of the redux, i.e., $M = (abb, bab)^+$. Therefore, M has infinitely many witnesses same as those of (abb, bab) .

A singleton redux of a sumfree set is nothing but a sumfree set with only one Kleene star. Given any sumfree set M , if M is conjugate, each of its singleton reduxes is conjugate. From [Theorem 5.24](#), a singleton redux of M has a common witness. Further, we prove that M has a common witness that is the common witness of each of its singleton reduxes. The below theorem characterises the conjugacy of a general sumfree set.

Theorem 5.26 (Common Witness Theorem for Monoid Closure)

Let M be a [sumfree set](#). The following are equivalent.

1. M is [conjugate](#).
2. Each [singleton redux](#) of M has a [common witness](#) z .
3. M has a [common witness](#) z .

See [Example 5.28](#). As a consequence, we have the following corollary.

Corollary 5.27

A sumfree expression is [conjugate](#) if and only if the expression has a [common witness](#).

Example 5.28

Let $M = (\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1)G_2^*(\alpha_2, \beta_2)$ be a sumfree set with two Kleene star where

$$\begin{pmatrix} \alpha_0 \\ \beta_0 \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}, G_1 = \left\{ \begin{pmatrix} ac \\ ca \end{pmatrix} \right\}, \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} ab \\ b \end{pmatrix}, G_2 = \left\{ \begin{pmatrix} bab \\ bab \end{pmatrix} \right\}, \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \varepsilon \\ b \end{pmatrix}.$$

The redux of M is $(\alpha_0\alpha_1\alpha_2, \beta_0\beta_1\beta_2) = (bab, abb)$. The set M has two singleton reduxes,

$$M_1 = \begin{pmatrix} \alpha_0 \\ \beta_0 \end{pmatrix} G_1^* \begin{pmatrix} \alpha_1\alpha_2 \\ \beta_1\beta_2 \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix} \begin{pmatrix} ac \\ ca \end{pmatrix}^* \begin{pmatrix} ab \\ bb \end{pmatrix}$$

and,

$$M_2 = \begin{pmatrix} \alpha_0\alpha_1 \\ \beta_0\beta_1 \end{pmatrix} G_2^* \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} bab \\ ab \end{pmatrix} \begin{pmatrix} bab \\ bab \end{pmatrix}^* \begin{pmatrix} \varepsilon \\ b \end{pmatrix}.$$

The set

$$G_1 \cup \{(\alpha_1\alpha_2\alpha_0, \beta_1\beta_2\beta_0)\} = \{(ac, ca)\} \cup \{(abb, bba)\} = \{(ac, ca), (abb, bba)\}$$

has a unique common inner witness $z_1 = a = (ac)^*a \cap (abb)^*a$ and no common outer witness since $(ca)^*c \cap (bba)^*bb = \emptyset$. By [Theorem 5.24 \(a\)](#), the unique common inner witness of the singleton redux M_1 of M is $[\alpha_0z_1, \beta_0]_R = [ba, a]_R = b$. The set

$$G_2 \cup \{(\alpha_2\alpha_0\alpha_1, \beta_2\beta_0\beta_1)\} = \{(bab, bab)\}$$

has infinitely many common witnesses. Thus the singleton redux M_2 is a subset of powers of the primitive root of the redux using [Theorem 5.24 \(c\)](#), i.e., $M_2 = (bab, abb)^+$. Thus M_2 have infinitely many common inner witnesses $(bab)^*b$ and common outer witnesses $(abb)^*ab$.

By [Theorem 5.26](#), M has a unique common inner witness $b \cap (bab)^*b = b$, that equals to the intersection of the common inner witness of its singleton reduxes M_1 and M_2 .

5.4 Existence of Common Witness for Kleene Closure

In this section, we prove the direction $(1) \Rightarrow (4)$ of [Theorem 5.17](#) recalled in the following lemma.

Lemma 5.29

For a set of pairs G , if G^* is [conjugate](#) then $R(G)$ has a [common witness](#).

We prove the lemma when G is finite by case analysis and then extend it for a countably infinite set of pairs using a compactness argument. Towards proving this, we need the following auxiliary result proven using [Lemma 3.15](#). A corollary of the proposition below is that whenever G is finite and G^* is conjugate with $R(G)$ consisting of only pairs of identical length, then $R(G)$ has a common witness.

Proposition 5.30

Let $G = \{(u_i, v_i) \mid i \in [k]\}$ be a set of $k \geq 2$ [conjugate](#) pairs such that all pairs in $R(G)$ are of equal length. Let (x_i, y_i) be a [cut](#) of the [primitive](#) pair $(\rho_{u_i}, \rho_{v_i}) \in R(G)$. If $(u_1, v_1)^*(u_2, v_2)^* \cdots (u_k, v_k)^*$ is [conjugate](#) then either $x_1 = x_2 = \cdots = x_k$ or $y_1 = y_2 = \cdots = y_k$.

Proof. Proof is by induction on the number of pairs in $R(G)$.

1. *Base Case:* When $R(G)$ has only 2 pairs, i.e., let $R(G) = \{(\rho_{u_1}, \rho_{v_1}), (\rho_{u_2}, \rho_{v_2})\}$. There exist ℓ_1, ℓ_2 such that $\ell_2 > \ell_1 + 2, \ell_1 > 2$ and $(\rho_{u_1}, \rho_{v_1})^{\ell_1} (\rho_{u_2}, \rho_{v_2})^{\ell_2} \in (u_1, v_1)^*(u_2, v_2)^*$ and hence it is conjugate. From [Lemma 3.15](#) we get either $x_1 = x_2$ or $y_1 = y_2$.
2. *Inductive Case:* Let us assume that the statement is true for k equal length pairs in $R(G)$, i.e., assume that $R(G) = \{(\rho_{u_1}, \rho_{v_1}), \dots, (\rho_{u_k}, \rho_{v_k})\}$. By induction hypothesis, we get that if $(u_1, v_1)^* \cdots (u_k, v_k)^*$ is conjugate then $x_1 = \cdots = x_k$ or $y_1 = \cdots = y_k$. WLOG, assume $x_1 = \cdots = x_k$. We aim to prove for $k + 1$ pairs. Let $G' \supseteq G$ be such that $R(G') = R(G) \cup \{(\rho_{u_{k+1}}, \rho_{v_{k+1}})\}$ where $(\rho_{u_{k+1}}, \rho_{v_{k+1}})$ is a conjugate primitive root of $(u_{k+1}, v_{k+1}) \in G'$ and have identical length to that of pairs in $R(G)$. Assume that $(u_1, v_1)^* \cdots (u_k, v_k)^* (u_{k+1}, v_{k+1})^*$ is conjugate. Let (x_{k+1}, y_{k+1}) be a cut of $(\rho_{u_{k+1}}, \rho_{v_{k+1}})$. There exists the set

of pairs $\{(\rho_{u_i}, \rho_{v_i})^{\ell_i}(\rho_{u_{k+1}}, \rho_{v_{k+1}})^{\ell_{k+1}} \mid \ell_{k+1} > \ell_i + 2, \ell_i > 2, 1 \leq i \leq k\} \subset (u_1, v_1)^* \cdots (u_k, v_k)^*(u_{k+1}, v_{k+1})^*$ that is conjugate. Thus, the pairs (ρ_{u_i}, ρ_{v_i}) and $(\rho_{u_{k+1}}, \rho_{v_{k+1}})$ satisfy either $x_i = x_{k+1}$ or $y_i = y_{k+1}$ by Lemma 3.15. There are two cases:

- (a) Suppose there exist an i such that $x_i = x_{k+1}$. Since $i \in [k]$ and $x_1 = \cdots = x_k$, we conclude $x_1 = \cdots = x_k = x_{k+1}$ as required.
- (b) Otherwise $y_i = y_{k+1}$ for all i . Then it follows that $y_1 = \cdots = y_k = y_{k+1}$. \square

5.4.1 For a Finite Set of Pairs

The common witness theorem for a finite set is a straightforward corollary of the below lemma.

Lemma 5.31

Let $G = \{(u_i, v_i) \mid i \in [k]\}$ be a finite set of $k \in \mathbb{N}^+$ pairs. If the set $(u_1, v_1)^* \cdots (u_k, v_k)^*$ is conjugate then $R(G)$ as well G has a common witness.

Proof. When $k = 1$, G has only one pair (u, v) and by assumption it is conjugate. By Theorem 3.6, (u, v) has a witness. From Proposition 5.10, we obtain that the witnesses of $R(G) = \{(\rho_u, \rho_v)\}$ are same as that of (u, v) .

Next we assume that $k > 1$. Let \approx be the equivalence relation on G whereby $(u, v) \approx (u', v')$ if $\rho_u \sim \rho_{u'}$, i.e., the primitive roots of the pairs are conjugates. Assume that \approx has d equivalence classes. Clearly $1 \leq d \leq k$. We do a cases analysis on whether $d = 1$ or otherwise.

If \approx has only one equivalence class, then the primitive roots of all the pairs in G are conjugates. Consequently, their lengths are identical. Since $(u_1, v_1)^* \cdots (u_k, v_k)^*$ is conjugate and all the pairs in $R(G)$ have identical lengths, by Proposition 5.30, $R(G)$ has a common witness.

Now we assume that $d > 1$. Choose d pairs $(u_1, v_1), (u_2, v_2), \dots, (u_d, v_d)$ from each equivalence class. We construct a pair $(u, v) \in (u_1, v_1)^*(u_2, v_2)^* \cdots (u_d, v_d)^* \subseteq (u_1, v_1)^* \cdots (u_k, v_k)^*$ and show that (u, v) is conjugate only if $R(G)$ has a common witness.

5. DECIDING CONJUGACY OF A RATIONAL RELATION

Let m be the least common multiple of $|u_1|, \dots, |u_d|$. Let $\ell_{ij} = |u_i| + |u_j| - \gcd(|u_i|, |u_j|) > 0$ for $1 \leq i, j \leq d$ and $i \neq j$. Let $\ell = \max \{\ell_{ij} \mid 1 \leq i, j \leq d, i \neq j\}$. Let N be a multiple of m that is $> 2\ell$.

Let $(u, v) = (u_1, v_1)^{j_1} (u_2, v_2)^{j_2} \dots (u_d, v_d)^{j_d}$ such that $j_1, \dots, j_d > 2$ and $|u_i^{j_i}| = N$ for each $1 \leq i \leq d$.

$$\begin{aligned} u &= \overbrace{u_1 \dots u_1}^N \overbrace{u_2 \dots u_2}^N \dots \overbrace{u_d \dots u_d}^N \\ v &= v_1 \dots v_1 v_2 \dots v_2 \dots v_d \dots v_d \end{aligned}$$

Since (u, v) is conjugate, it has a **cut**, say (p, q) . Substituting each pair in (u, v) with their **primitive roots**, we get

$$\begin{aligned} u &= \rho_{u_1} \dots \rho_{u_1} \rho_{u_2} \dots \rho_{u_2} \dots \rho_{u_d} \dots \rho_{u_d} = pq \\ v &= \rho_{v_1} \dots \rho_{v_1} \rho_{v_2} \dots \rho_{v_2} \dots \rho_{v_d} \dots \rho_{v_d} = qp \end{aligned}$$

Let (x_i, y_i) be a cut of (ρ_{u_i}, ρ_{v_i}) for $1 \leq i \leq d$. Let B_1, B_2, \dots, B_d represent the blocks in u , and let B'_1, B'_2, \dots, B'_d represent the blocks in v .

The cut in u can be either within the first block B_1 , or the last block B_d , or anywhere between the first and the last blocks in u . We do a case analysis on all the possible cuts of (u, v) and show that there exists a common witness of $R(G)$ in each of the cases.

Case 1: When the cut in u is in the first block B_1 . We make a further case analysis depending upon if the cut is within the first half or the second half of the first block.

Suppose the cut in u is within the first half of the block, i.e., p is of length at most $N/2$. In this case, since the length of each block are equal, the cut in v is within the second half of the last block B'_d , i.e., p is a suffix of v of length at most $N/2$.

$$\begin{aligned} u &= \overbrace{\rho_{u_1} \dots \rho_{u_1}}^p \overbrace{\rho_{u_1} \rho_{u_2} \dots \rho_{u_2} \dots \rho_{u_d} \dots \rho_{u_d}}^q \\ v &= \underbrace{\rho_{v_1} \dots \rho_{v_1} \rho_{v_2} \dots \rho_{v_2} \dots \rho_{v_d} \dots \rho_{v_d}}_q \underbrace{\rho_{v_d} \dots \rho_{v_d}}_p \end{aligned}$$

We obtain $q = p^{-1}(B_1 B_2 \dots B_d) = (B'_1 B'_2 \dots B'_d) p^{-1}$.

Claim 5.32. *The following holds for each $1 \leq i \leq d$.*

1. B_i is of the form pq_i and B'_i is of the form $q_i p$, and

2. p is of the form $(x_i y_i)^{m_i} x_i$ and q_i is of the form $(y_i x_i)^{m_i} y_i$ for some $m_i \geq 0$.

Proof. Proof is by induction on i .

1. *Base Case:* when $i = 1$. We compare the **prefixes** of q in u and v . Since $|p| \leq N/2$, the prefix of q in u must begin within the first block B_1 . Also, there must be at least one occurrence of the **factor** $\rho_{u_1} = x_1 y_1$ following the cut. Since q in v starts with $\rho_{v_1} = y_1 x_1$, the cut should be at the end of x_1 by Cases I. (c), I. (d), I. (e) and II. of **Cut Lemma**. Hence $p = (x_1 y_1)^{m_1} x_1$ for some integer $m_1 \geq 0$. Consequently, the prefix of q in the block B_1 , denoted as q_1 , is of the form $y_1 (x_1 y_1)^{n_1}$, for some $n_1 > 0$. After matching q_1 in v , we observe that a factor equal to p appears in the **suffix** of the block B'_1 , as shown below.

$$\begin{aligned} u &= \overbrace{(x_1 y_1)^{m_1} x_1}^p \overbrace{y_1 (x_1 y_1)^{n_1}}^{q_1} \overbrace{\rho_{u_2} \cdots \rho_{u_2} \cdots \rho_{u_d} \cdots \rho_{u_d}}^{q_1^{-1} q} \\ v &= \underbrace{y_1 (x_1 y_1)^{n_1}}_{q_1} \underbrace{(x_1 y_1)^{m_1} x_1}_{=p} \rho_{v_2} \cdots \rho_{v_2} \cdots \rho_{v_d} \cdots \rho_{v_d} = qp \end{aligned}$$

2. *Inductive Case:* Assume the claim is true for first i blocks where $1 \leq i < k$.

$$\begin{aligned} u &= \overbrace{(x_1 y_1)^{m_1} x_1}^p \overbrace{y_1 (x_1 y_1)^{n_1}}^{q_1} \cdots \overbrace{(x_i y_i)^{m_i} x_i}^p \overbrace{y_i (x_i y_i)^{n_i}}^{q_i} \overbrace{\rho_{u_{i+1}} \cdots \rho_{u_d}}^{q'' = (q_1 p \cdots q_{i-1} p q_i)^{-1} q} = pq \\ v &= \underbrace{y_1 (x_1 y_1)^{n_1}}_{q_1} \underbrace{(x_1 y_1)^{m_1} x_1}_p \cdots \underbrace{y_i (x_i y_i)^{n_i}}_{q_i} \underbrace{(x_i y_i)^{m_i} x_i}_p \rho_{v_{i+1}} \cdots \rho_{v_d} = qp \end{aligned}$$

Let q'' denote the remaining suffix of q in u after the i -th block B_i . We obtain $q'' = (q_1 p \cdots q_{i-1} p q_i)^{-1} q$. By comparing the prefixes of q'' in u and v , we get that the suffix of the block B'_i in v , that is equal to p , matches within the first half of the block B_{i+1} in u since $|p| < N/2$. Moreover, the matching should end at x_{i+1} by Cases I. (c), I. (d), I. (e) and II. of **Cut Lemma**. Hence $p = (x_{i+1} y_{i+1})^{m_{i+1}} x_{i+1}$ for some integer $m_{i+1} \geq 0$. Consequently, the remaining suffix of the block B_{i+1} , denoted by q_{i+1} , is of the form $(y_{i+1} x_{i+1})^{n_{i+1}} y_{i+1}$ for some integer $n_{i+1} > 0$. After matching q_{i+1} in B'_{i+1} , we observe that a factor equal to p appears in the suffix of the block B'_{i+1} . Hence, $B_{i+1} = p q_{i+1}$ and $B'_{i+1} = q_{i+1} p$.

□ *Claim 5.32*

From the above claim, it follows that $q = q_1 p q_2 p \cdots p q_d$ and

$$p = (x_1 y_1)^{m_1} x_1 = (x_2 y_2)^{m_2} x_2 = \cdots = (x_d y_d)^{m_d} x_d$$

for $m_1, \dots, m_d \geq 0$.

From [Claim 5.33](#) (see below), the above equation holds for any two pairs between the equivalence classes. By [Proposition 5.12](#), we obtain p is a common inner witness of $R(G)$.

Next, we assume the cut in u is in the second half of B_1 . We compare the prefixes of q in u and v and deduce that there exists a common factor of length at least $N/2 > \ell > \ell_{12}$ between block B'_1 and block B_2 . From [Theorem 3.11](#), ρ_{v_1} is conjugate to ρ_{u_2} , that is in turn is conjugate to ρ_{v_2} . From [transitivity](#) of conjugacy, ρ_{v_1} is conjugate to ρ_{v_2} , which contradicts the fact that (u_1, v_1) and (u_2, v_2) belong to different equivalence classes. Hence, a cut in the second half of B_1 is not possible.

Case 2: When the cut in u is in the last block B_d . We make a further case analysis depending upon if the cut in u is in the first half or second half of the last block B_d . The proof is symmetric to that of the previous case. In particular, when the cut is in the second half of the last block,

$$q = (y_1 x_1)^{m_1} y_1 = (y_2 x_2)^{m_2} y_2 = \cdots = (y_d x_d)^{m_d} y_d.$$

By [Claim 5.33](#) and [Proposition 5.12](#), q is a common outer witness of $R(G)$.

Case 3: When the cut in u is within the block B_j for $1 < j < d$. WLOG assume that the cut in u is within the first half of the block B_j . In this case, the cut in v will be within the second half of the block B_{d-j+1} .

$$\begin{aligned} u &= \overbrace{\rho_{u_1} \cdots \rho_{u_1} \cdots \rho_{u_j}}^p \cdots \overbrace{\rho_{u_j} \cdots \cdots \rho_{u_d} \cdots \rho_{u_d}}^q \\ v &= \underbrace{\rho_{v_1} \cdots \rho_{v_1} \cdots \rho_{v_{d-j+1}}}_{q} \cdots \underbrace{\rho_{v_{d-j+1}} \cdots \rho_{u_d} \cdots \rho_{u_d}}_p \end{aligned}$$

By matching q in u and v , we get ρ_{v_1} and ρ_{u_j} shares a common factor of length at least $N/2 > \ell > \ell_{1j}$ and hence they are conjugates to each other by [Theorem 3.11](#). Since ρ_{u_j} is conjugate to ρ_{v_j} , by transitivity of conjugacy we obtain ρ_{v_1} and ρ_{v_j} are conjugates.

This contradicts the fact that (u_1, v_1) and (u_j, v_j) belongs to different equivalence classes. Hence cut in B_j where $1 < j < d$ is not possible.

Claim 5.33. *Let (u_i, v_i) and (u_j, v_j) be two pairs in different equivalence class where $i, j \in [d]$. Let (x_i, y_i) and (x_j, v_j) be a cut of the primitive root of (u_i, v_i) and (u_j, v_j) respectively. The following holds.*

1. *If $(x_i y_i)^{m_i} x_i = (x_j y_j)^{m_j} x_j$, then for any pair $(u_{j'}, v_{j'})$ that belongs to the same equivalence class as (u_j, v_j) with $(x_{j'}, y_{j'})$ being the cut of its primitive root, $(x_i y_i)^{m_i} x_i = (x_{j'} y_{j'})^{m_j} x_{j'}$.*
2. *If $(y_i x_i)^{m_i} y_i = (y_j x_j)^{m_j} y_j$, then for any pair $(u_{j'}, v_{j'})$ that belongs to the same equivalence class as (u_j, v_j) with $(x_{j'}, y_{j'})$ being the cut of its primitive root, $(y_i x_i)^{m_i} y_i = (y_{j'} x_{j'})^{m_j} y_{j'}$.*

Proof. We prove Item 1. The proof of Item 2 is symmetric.

For contradiction, assume that $(x_i y_i)^{m_i} x_i = (x_j y_j)^{m_j} x_j \neq (x_{j'} y_{j'})^{m_j} x_{j'}$. Construct a pair (\bar{u}, \bar{v}) similar to (u, v) by keeping only the pairs (u_i, v_i) , (u_j, v_j) and $(u_{j'}, v_{j'})$ such that the total length of each pairs are equal to N . WLOG, assume, $i < j < j'$

$$\bar{u} = \overbrace{u_i \cdots u_i}^N \overbrace{u_j \cdots u_j}^N \overbrace{u_{j'} \cdots u_{j'}}^N$$

$$\bar{v} = v_i \cdots v_i v_j \cdots v_j v_{j'} \cdots v_{j'}$$

Since $(\bar{u}, \bar{v}) \subseteq (u_1, v_1)^* \cdots (u_k, v_k)^*$, it is conjugate with a cut. After doing a case analysis on the cut, we get three possible cases.

1. There exist integers $n_i, n_j, n_{j'} \geq 0$ such that $(x_i y_i)^{n_i} x_i = (x_j y_j)^{n_j} x_j = (x_{j'} y_{j'})^{n_{j'}} x_{j'}$.

Both $n_i = m_i$ and $n_j = m_j$. Otherwise if either $n_i \neq m_i$ or $n_j \neq m_j$, then the set of pairs $\{(u_i, v_i), (u_j, v_j)\}$ has more than one common witness. By [Proposition 5.16](#), they have infinitely many common witnesses, and their primitive roots are the same. This contradicts that (u_i, v_i) and (u_j, v_j) are in different equivalence class. Since (u_j, v_j) and $(u_{j'}, v_{j'})$ belongs to the same equivalence class, $|x_j y_j| = |x_{j'} y_{j'}|$. Hence, we get $n_{j'} = n_j = m_j$. Thus, $(x_i y_i)^{m_i} x_i = (x_j y_j)^{m_j} x_j = (x_{j'} y_{j'})^{m_j} x_{j'}$.

2. There exist integers $n_i, n_j, n_{j'} \geq 0$ such that $(y_i x_i)^{n_i} y_i = (y_j x_j)^{n_j} y_j = (y_{j'} x_{j'})^{n_{j'}} y_{j'}$.

Since $(x_i y_i)^{m_i} x_i = (x_j y_j)^{m_j} x_j$, we get there is more than one common witness for the set of pairs $\{(u_i, v_i), (u_j, v_j)\}$ and from [Proposition 5.16](#), we get they have infinitely many common witnesses. Thus the primitive roots are the same and contradict that (u_i, v_i) and (u_j, v_j) are in different equivalence classes.

3. The block of u_i 's, u_j 's and $u_{j'}$'s share a common factor of length at least $N/2 \geq \ell$ and via [Theorem 3.11](#), we get their primitive roots are conjugates to each other. This contradicts that (u_i, v_i) and (u_j, v_j) are in different equivalence class.

□*Claim 5.33*

Hence, for a finite set of pairs G , $(u_1, v_1)^* \cdots (u_k, v_k)^*$ is conjugate only if $R(G)$ has a common witness. By [Proposition 5.14](#), we also conclude that G has a common witness. □

Hence, we proved [Lemma 5.29](#) for the finite case.

5.4.2 For an Infinite Set of Pairs

We now extend [Lemma 5.29](#) from a finite set to an infinite set of pairs.

Lemma 5.34 (Compactness Theorem)

Let G be an infinite set of pairs. If every finite subset of G has a **common witness**, then G has a **common witness**.

Proof. From [Proposition 5.16](#), if a set has a witness, it has exactly one common witness or infinitely many common witnesses. Given that every finite subset of G has a common witness, there are two possible cases: a finite subset of G with a unique witness exists, or every finite subset of G has infinitely many witnesses.

1. Assume that there exists a finite subset G_f of G with exactly one common witness, say z . We claim that z is a common witness of G as well. By assumption, the finite set $G_f \cup \{(u, v)\}$ has a common witness, for any pair $(u, v) \in G$. Moreover, the witness for this set must be z ; otherwise, it contradicts the uniqueness of the witness of G_f . This implies that z is a witness for any pair in G . Hence z is a common witness of G .

2. Next we assume that every finite subset of G has infinitely many common witnesses. Take any pair (u_i, v_i) and (u_j, v_j) from G . The set $\{(u_i, v_i), (u_j, v_j)\}$ is a finite set with infinitely many witnesses by assumption. Therefore, from [Proposition 5.16](#), both (u_i, v_i) and (u_j, v_j) have the same primitive root. Since primitive roots are unique, the primitive root of every pair in G is the same. From [Proposition 5.10](#), the witnesses of the primitive root is same as that of the witnesses of each pair in G . Hence, G has a common witness. \square

The proof of [Lemma 5.29](#) is a straightforward corollary of the [Compactness Theorem](#). If G^* is conjugate, then the closure of every finite subset of G is also conjugate. Using [Lemma 5.31](#), every finite subset of G has a common witness. Using [Compactness Theorem](#), G has a common witness. From [Proposition 5.14](#), we conclude that $R(G)$ has a common witness.

This concludes the proof of Common Witness Theorem ([Theorem 5.17](#)).

5.5 Existence of Common Witness for Monoid Closure

In this section, we prove the equivalence between conjugacy and the presence of a [common witness](#) in [sumfree sets](#). We begin by proving [Theorem 5.24](#) for sumfree sets that contain only one Kleene star. Subsequently, we establish [Theorem 5.26](#) that extends the result to general sumfree sets.

5.5.1 Common Witness of a Singleton Redux

We prove [Theorem 5.24](#) by showing $(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (1)$. In this subsection, we assume that the sumfree set has a nonempty [redux](#) as stated in [Theorem 5.24](#). It is trivial that $(3) \Rightarrow (1)$.

Now we proceed to prove $(1) \Rightarrow (2)$, namely, if a sumfree set $M = (\alpha_0, \beta_0)G^*(\alpha_1, \beta_1)$ is conjugate, then there exists a common witness of $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$. We first prove this direction when G is just a singleton set and later generalise it to any arbitrary set of pairs G .

Proposition 5.35

If a set $M = (\alpha_0, \beta_0)(u, v)^*(\alpha_1, \beta_1)$ is **conjugate** then the set $\{(u, v), (\alpha_1\alpha_0, \beta_1\beta_0)\}$ has a **common witness**.

Proof. Since M is conjugate, (u, v) is also conjugate by [Lemma 5.5](#). Consider the pair $(u', v') = (\alpha_0, \beta_0)(u, v)^{4n}(\alpha_1, \beta_1)$ where $n|u| \geq 2 \cdot (\text{length of the redux of } M)$. Let (x, y) denote a **cut** of the primitive root of the conjugate pair (u, v) .

We now examine the possible cuts, say (p, q) , of (u', v') and show that in each case, a common witness of $\{(u, v), (\alpha_1\alpha_0, \beta_1\beta_0)\}$ exists.

Case 1: When the cut in u' is within α_0 . Then $p = \alpha'_0$ is a **prefix** of α_0 and $\alpha_0 = \alpha'_0\alpha''_0$ for some word α''_0 . Substituting (u, v) with powers of (xy, yx) ,

$$\begin{aligned} u' &= \overbrace{\alpha'_0}^p \overbrace{\alpha''_0 xy \cdots xy \alpha_1}^q \\ v' &= \beta_0 yx \cdots yx \beta_1 \end{aligned}$$

Comparing prefixes of q in u' and v' , we obtain three possible cases for β_0 .

- (a) β_0 is a proper prefix of α''_0 : After matching β_0 with the prefix of q in u' , we find that the remaining **suffix** of α''_0 matches with the prefix of the block $yx \cdots yx$ in v' . Since the total length of the block $yx \cdots yx$ is greater than $4|\alpha_0|$, it follows that α''_0 must end within the first half of the block. Furthermore, using Cases [I. \(a\)](#), [I. \(b\)](#), [I. \(e\)](#) and [II. of Cut Lemma](#), it should end after a y since there exists an xy after α''_0 in u' . Thus, we can express α''_0 as

$$\alpha''_0 = \beta_0(yx)^m y \tag{5.2}$$

for some integer $m \geq 0$. Continuing to match q in u' and v' , we obtain

$$u' = \alpha'_0 \alpha''_0 \overbrace{xy \cdots x}^= (yx)^m y \alpha_1 = pq \tag{5.3}$$

$$v' = \underbrace{\beta_0(yx)^m y}_{\alpha''_0} \underbrace{xy \cdots x}_{=} \beta_1 = qp = \alpha''_0 \overbrace{xy \cdots x}^= (yx)^m y \alpha_1 \alpha'_0 \tag{5.4}$$

By equating the sets for v' on both sides of [Equation \(5.4\)](#), we get

$$\beta_1 = (yx)^m y \alpha_1 \alpha'_0. \tag{5.5}$$

Concatenating Equation (5.2) and Equation (5.5), we obtain

$$(yx)^m y \alpha_1 \alpha_0 = \beta_1 \beta_0 (yx)^m y .$$

From Theorem 3.6, we get $(yx)^m y$ is a outer witness for $(\alpha_1 \alpha_0, \beta_1 \beta_0)$, and it is also a outer witness of (u, v) using Proposition 5.10. Therefore, $(yx)^m y$ is a common outer witness of $\{(u, v), (\alpha_1 \alpha_0, \beta_1 \beta_0)\}$.

(b) The case when $\beta_0 = \alpha_0''$:

$$u' = \alpha_0' \alpha_0'' \overbrace{xy \cdots xy}^{\quad \quad \quad} \alpha_1 = pq \quad (5.6)$$

$$v' = \underbrace{\beta_0}_{\alpha_0''} \underbrace{yx \cdots yx}_{\quad \quad \quad} \beta_1 = qp = \alpha_0'' \overbrace{xy \cdots xy}^{\quad \quad \quad} \alpha_1 \alpha_0' \quad (5.7)$$

Equating v' on both sides of the Equation (5.7), we get that $xy = yx$ and

$$\beta_1 = \alpha_1 \alpha_0' . \quad (5.8)$$

Appending the equation $\beta_0 = \alpha_0''$ to Equation (5.8), we get $\alpha_1 \alpha_0 = \beta_1 \beta_0$. Hence $(\alpha_1 \alpha_0, \beta_1 \beta_0)$ is conjugate with ε as a witness. Since $xy = yx$, we can also deduce that (u, v) is an identical pair with ε as a witness. Therefore, ε is a common witness of $\{(u, v), (\alpha_1 \alpha_0, \beta_1 \beta_0)\}$.

(c) α_0'' is a proper prefix of β_0 : Since the total length of block $xy \cdots xy$ is at least $4|\beta_0|$, it follows that β_0 must end within the first half of the block of xy . Moreover, it should end after an x by Cases I. (c), I. (d), I. (e) and II. of Cut Lemma since there is at least one yx after β_0 in v' . Therefore,

$$\beta_0 = \alpha_0'' (xy)^m x \quad (5.9)$$

for some integer $m \geq 0$. Continuing with the analysis, we have:

$$u' = \alpha_0' \overbrace{\alpha_0'' (xy)^m x}^{\beta_0} \overbrace{yx \cdots yx}^{\quad \quad \quad} \alpha_1 = pq \quad (5.10)$$

$$v' = \beta_0 \underbrace{yx \cdots yx}_{\quad \quad \quad} (xy)^m x \beta_1 = qp = \overbrace{\alpha_0'' (xy)^m x}^{\beta_0} \overbrace{yx \cdots yx}^{\quad \quad \quad} \alpha_1 \alpha_0' \quad (5.11)$$

By equating the sets for v' on both sides of Equation (5.11), we get

$$(xy)^m x \beta_1 = \alpha_1 \alpha'_0. \quad (5.12)$$

Concatenating Equation (5.9) and Equation (5.12), we obtain

$$\alpha_1 \alpha_0 (xy)^m x = (xy)^m x \beta_1 \beta_0.$$

From Theorem 3.6, we get $(xy)^m x$ is an inner witness of $(\alpha_1 \alpha_0, \beta_1 \beta_0)$, and it is also an inner witness for (u, v) using Proposition 5.10. Thus, $(xy)^m x$ is a common inner witness of $\{(u, v), (\alpha_1 \alpha_0, \beta_1 \beta_0)\}$.

Case 2: When the cut in u' is within the block of $(u, v) \cdots (u, v)$. Then p ends within the block of (u, v) 's. There are two cases based on whether the cut in u' is within the first half or the second half of the block of $(u, v) \cdots (u, v)$.

(a) When p ends within the first half of the block of (u, v) 's:

$$\begin{aligned} u' &= \alpha_0 \overbrace{xy \cdots xy}^{\text{cut region}} \overbrace{xy \cdots xy}^{\geq 2n \text{ times}} \alpha_1 = pq \\ v' &= \beta_0 yx \cdots yxyx \cdots yx \beta_1 = qp \end{aligned}$$

We compare the prefixes of q in u' and v' . Since the length of the remaining half of the block of xy 's is still greater than $2n|u| > 2|\beta_0|$, it follows that β_0 in v' matches within the block of xy 's in u' and there is at least one xy occurring after it. Moreover, it ends after an x by Cases I. (c), I. (d), I. (e) and II. of Cut Lemma, as there is at least one yx in v' after β_0 . Therefore,

$$p\beta_0 = \alpha_0(xy)^m x \quad (5.13)$$

for some integer $m \geq 0$.

$$u' = \overbrace{\alpha_0(xy)^m x}^{p\beta_0} \overbrace{yx \cdots y}^{\quad} \alpha_1 = pq \quad (5.14)$$

$$v' = \beta_0 \underbrace{yx \cdots y}_{\quad} (xy)^m x \beta_1 = qp = \beta_0 \overbrace{yx \cdots y}^{\quad} \alpha_1 p \quad (5.15)$$

By equating the sets for v' on both sides of Equation (5.15), we get

$$\begin{aligned} (xy)^m x \beta_1 = \alpha_1 p &\Rightarrow (xy)^m x \beta_1 \beta_0 = \alpha_1 p \beta_0 && \text{(Appending } \beta_0) \\ &\Rightarrow (xy)^m x \beta_1 \beta_0 = \alpha_1 \alpha_0 (xy)^m x && \text{(Substituting Equation (5.13))} \end{aligned}$$

Therefore we obtain,

$$\alpha_1 \alpha_0 (xy)^m x = (xy)^m x \beta_1 \beta_0 .$$

From Theorem 3.6, $(xy)^m x$ is an inner witness of $(\alpha_1 \alpha_0, \beta_1 \beta_0)$, and it is also an inner witness for (u, v) using Proposition 5.10. Therefore, $(xy)^m x$ is a common inner witness of $\{(u, v), (\alpha_1 \alpha_0, \beta_1 \beta_0)\}$.

(b) When p ends within the second half of the block of (u, v) 's:

$$\begin{aligned} u' &= \alpha_0 \overbrace{xy \cdots xy}^{\geq 2n \text{ times}} \overbrace{xy \cdots xy}^{\text{cut region}} \alpha_1 = pq \\ v' &= \beta_0 yx \cdots yxyx \cdots yx \beta_1 = qp \end{aligned}$$

We compare the suffixes of p in u' and v' . Since the suffix of p within the block xy is still greater than $2n|u| > 2|\beta_1|$, it follows that the suffix β_1 in v' matches within the block of xy 's in u' and there is at least one xy occurring before it. Moreover, it starts with a y by Cases I. (c), I. (d), I. (e) and II. of Cut Lemma since there is at least one yx before β_1 . Hence,

$$\beta_1 q = (yx)^m y \alpha_1 \tag{5.16}$$

for some integer $m \geq 0$.

$$u' = \alpha_0 \overbrace{xy \cdots x}^{\beta_1 q} (yx)^m y \alpha_1 = pq \tag{5.17}$$

$$v' = \beta_0 (yx)^m y \underbrace{xy \cdots x}_{\beta_1} \beta_1 = qp = q \alpha_0 \overbrace{xy \cdots x}^{\beta_1} \beta_1 \tag{5.18}$$

By equating v' on both sides of the Equation (5.18), we get

$$\begin{aligned} \beta_0 (yx)^m y = q \alpha_0 &\Rightarrow \beta_1 \beta_0 (yx)^m y = \beta_1 q \alpha_0 && \text{(Concatenating } \beta_1 \text{ on left side)} \\ &\Rightarrow \beta_1 \beta_0 (yx)^m y = (yx)^m y \alpha_1 \alpha_0 && \text{(Substituting Equation (5.16))} \end{aligned}$$

Therefore, we obtain

$$(yx)^m y \alpha_1 \alpha_0 = \beta_1 \beta_0 (yx)^m y .$$

From [Theorem 3.6](#), we get $(yx)^m y$ is an outer witness of $(\alpha_1 \alpha_0, \beta_1 \beta_0)$, and it is also an outer witness for (u, v) using [Proposition 5.10](#). Therefore, $(yx)^m y$ is a common outer witness of $\{(u, v), (\alpha_1 \alpha_0, \beta_1 \beta_0)\}$.

Case 3: When the cut in u' is within α_1 . Then $q = \alpha_1''$ is a suffix of α_1 and $\alpha_1 = \alpha_1' \alpha_1''$ for some word α_1' .

$$\begin{aligned} u' &= \overbrace{\alpha_0 x y \cdots x y \alpha_1'}^p \overbrace{\alpha_1''}^q \\ v' &= \beta_0 y x \cdots y x \beta_1 \end{aligned}$$

This case is symmetric to *Case 1*, where the cut in u' is within α_0 .

Thus, the pair $(u', v') \in M$ is conjugate only if there exists a common witness for the set $\{(u, v), (\alpha_1 \alpha_0, \beta_1 \beta_0)\}$. \square

Now we extend the above proposition to an arbitrary set G .

Lemma 5.36

If a set $M = (\alpha_0, \beta_0) G^* (\alpha_1, \beta_1)$ is [conjugate](#) then one of the following is true:

1. G has infinitely many [common witnesses](#): In this case, each pair in G shares the same [primitive root](#) that has a [common witness](#) with $(\alpha_1 \alpha_0, \beta_1 \beta_0)$.
2. G has a unique [common inner](#) (*resp.* [common outer](#)) witness z : In this case, z is the unique [common inner](#) (*resp.* [common outer](#)) witness of $G \cup \{(\alpha_1 \alpha_0, \beta_1 \beta_0)\}$. Moreover, there exist a pair $(u, v) \in G$ such that z is the unique [common inner](#) (*resp.* [common outer](#)) witness of set $\{(u, v), (\alpha_1 \alpha_0, \beta_1 \beta_0)\}$.

Proof. Since M is conjugate, G^* is conjugate by [Lemma 5.5](#). Further, G^* is conjugate iff G has a common witness by [Theorem 5.17](#). The set G has two possibilities: it either has a unique common witness or infinitely many common witnesses using [Proposition 5.16](#).

Assume that G has infinitely many common witnesses. From [Proposition 5.16](#), each pair in G have the same primitive root, say (ρ, ρ') . The common witnesses of G are the same as that of the witnesses of (ρ, ρ') using [Proposition 5.14](#). Since M is conjugate, $(\alpha_0, \beta_0)(\rho^n, \rho'^n)^*(\alpha_1, \beta_1)$ is conjugate for some $n \geq 1$. From [Proposition 5.35](#), there exists a common witness of $\{(\rho^n, \rho'^n), (\alpha_1\alpha_0, \beta_1\beta_0)\}$. Furthermore, the witness of (ρ, ρ') is the same as the witness of (ρ^n, ρ'^n) by [Proposition 5.10](#). Therefore, we can conclude that there exists a common witness of $\{(\rho, \rho'), (\alpha_1\alpha_0, \beta_1\beta_0)\}$, and thus of $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$.

Next, we assume that G has a unique common witness. WLOG, assume that G has a unique common inner witness z . It suffices to show that z is an inner witness of $(\alpha_1\alpha_0, \beta_1\beta_0)$. Every pair in G^* has a common witness with $(\alpha_1\alpha_0, \beta_1\beta_0)$ using [Proposition 5.35](#). If there exists a pair (u, v) in G^* such that $\{(u, v), (\alpha_1\alpha_0, \beta_1\beta_0)\}$ has infinitely many common witnesses, then both (u, v) and $(\alpha_1\alpha_0, \beta_1\beta_0)$ have the same primitive root by [Proposition 5.16](#). Since z is an inner witness of (u, v) , z is an inner witness of its primitive root, and thus also an inner witness of $(\alpha_1\alpha_0, \beta_1\beta_0)$ using [Proposition 5.10](#). Suppose instead that every pair in G^* has a unique common witness with $(\alpha_1\alpha_0, \beta_1\beta_0)$. Since G has a unique common inner witness z , there exists two distinct pairs (u, v) and (u', v') of length greater than z such that z is the unique common inner witness for the set $\{(u, v), (u', v')\}$. These pairs can be expressed as $(u, v) = (zr, rz)$ and $(u', v') = (zr', r'z)$, where $r \neq r'$. Consider the following three pairs formed using (u, v) and (u', v') :

$$(u_1, v_1) = (u'uu, v'vv) = (zr'zr, r'zr'z),$$

$$(u_2, v_2) = (uu'u, vv'v) = (zr'zr', r'zr'z),$$

$$(u_3, v_3) = (uuu', vv'v') = (zr'zr'z, r'zr'z).$$

For $1 \leq i, j \leq 3$ with $i \neq j$, the following properties hold:

1. $|(u_i, v_i)| = |(u_j, v_j)|$.
2. $|\rho_{u_i}, \rho_{v_i}| = |\rho_{u_j}, \rho_{v_j}|$; Since u_i and u_j (resp. v_i and v_j) are conjugates, their primitive roots are conjugates, and hence are of equal length, i.e., $|\rho_{u_i}| = |\rho_{u_j}|$ (resp. $|\rho_{v_i}| = |\rho_{v_j}|$).

3. $(u_i, v_i) \neq (u_j, v_j)$; Otherwise, if $u_i = u_j$ (or $v_i = v_j$), it would follow that $r'zr = r'zr'$ (or, $r'zr'zr = r'zr'zr'$), implying that r' is a outer witness of $(zr, rz) = (u, v)$. Since r' is also an outer witness of $(u', v') = (zr', r'z)$, it contradicts the assumption that z is a unique common witness of the set $\{(u, v), (u', v')\}$. Hence, $u_i \neq u_j$ and $v_i \neq v_j$.
4. $(\rho_{u_i}, \rho_{v_i}) \neq (\rho_{u_j}, \rho_{v_j})$; Since $u_i \neq u_j$ (resp. $v_i \neq v_j$) and $|\rho_{u_i}| = |\rho_{u_j}|$ (resp. $|\rho_{v_i}| = |\rho_{v_j}|$), we get that $\rho_{u_i} \neq \rho_{u_j}$ (resp. $\rho_{v_i} \neq \rho_{v_j}$).

Let (x_i, y_i) be the cut of (ρ_{u_i}, ρ_{v_i}) for $i \in [3]$. The set $\{(u_i, v_i) \mid i \in [3]\}$ has a unique common witness; otherwise their primitive roots will be same by [Proposition 5.16](#) which is a contradiction. Since z is an inner witness of each (u_i, v_i) , z is the unique common inner witness of $\{(u_i, v_i) \mid i \in [3]\}$, and thus of $\{(\rho_{u_i}, \rho_{v_i}) \mid i \in [3]\}$ by [Proposition 5.14](#). Since $(u_1, v_1)^*(u_2, v_2)^*(u_3, v_3)^* \subset G^*$ is conjugate and $|\rho_{u_1}, \rho_{v_1}| = |\rho_{u_2}, \rho_{v_2}| = |\rho_{u_3}, \rho_{v_3}|$, we get that $z = x_1 = x_2 = x_3$ by [Proposition 5.30](#).

By assumption, each pair (u_i, v_i) has a unique common witness with $(\alpha_1\alpha_0, \beta_1\beta_0)$. Thus, at least two of these three pairs, (u_i, v_i) and (u_j, v_j) for $i, j \in [3]$ and $i \neq j$, such that either both have unique common inner witnesses with $(\alpha_1\alpha_0, \beta_1\beta_0)$, or both have unique common outer witnesses with $(\alpha_1\alpha_0, \beta_1\beta_0)$. Assume that the set $\{(u_i, v_i), (\alpha_1\alpha_0, \beta_1\beta_0)\}$ and $\{(u_j, v_j), (\alpha_1\alpha_0, \beta_1\beta_0)\}$ both has a unique common inner witness, say z_i and z_j respectively. Let (α, β) be the primitive root of $(\alpha_1\alpha_0, \beta_1\beta_0)$. There exist integers $m_i, m_j, n_i, n_j \in \mathbb{N}$ such that

$$\begin{aligned} z_i &= (x_i y_i)^{m_i} x_i = (\alpha \beta)^{n_i} \alpha \\ z_j &= (x_j y_j)^{m_j} x_j = (\alpha \beta)^{n_j} \alpha \end{aligned}$$

If both $m_i, m_j \geq 1$, then $x_i y_i$ and $x_j y_j$ are both equal length prefixes of $(\alpha \beta)^* \alpha$, and hence $x_i y_i = x_j y_j$, which contradicts that $\rho_{u_i} \neq \rho_{u_j}$. Hence, we get that either $m_i = 0$ or $m_j = 0$. WLOG, assume that $m_i = 0$, i.e., $x_i = (\alpha \beta)^{n_i} \alpha$ and since $z = x_i = x_j$, we get $z \in (\alpha \beta)^* \alpha$, and hence is an inner witness of $(\alpha_1\alpha_0, \beta_1\beta_0)$. Infact, z is the unique common inner witness of (u_i, v_i) and $(\alpha_1\alpha_0, \beta_1\beta_0)$. The case where the set $\{(u_i, v_i), (\alpha_1\alpha_0, \beta_1\beta_0)\}$ and $\{(u_j, v_j), (\alpha_1\alpha_0, \beta_1\beta_0)\}$ both has a unique common outer witness is invalid and leads to a contradiction, since we get y_i and y_j are equal length prefixes of $(\beta \alpha)^* \beta$, and hence $y_i = y_j$. This contradicts $\rho_{u_i} \neq \rho_{u_j}$ since $x_i = x_j$. \square

It remains to prove (2) \Rightarrow (3) in [Theorem 5.24](#). It is a direct corollary of the lemma below.

Lemma 5.37

Let $M = (\alpha_0, \beta_0)G^*(\alpha_1, \beta_1)$ be a sumfree set with nonempty [redux](#). If there exists a [common witness](#) z' for $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$, then one of the following cases is true:

- (a) If z' is a unique [common inner witness](#), then M has a unique [common witness](#) $z = [\alpha_0 z', \beta_0]_R = [\alpha_1, z' \beta_1]_L$. Moreover, if $|\alpha_0 z'| \geq |\beta_0|$ or equivalently $|\alpha_1| \leq |z' \beta_1|$, then z is a [common inner witness](#), otherwise it is a [common outer witness](#).
- (b) If z' is a unique [common outer witness](#), then M has a unique [common witness](#) $z = [\alpha_0, \beta_0 z']_R = [z' \alpha_1, \beta_1]_L$. Moreover, if $|z' \alpha_1| \geq |\beta_1|$ or equivalently $|\alpha_0| \leq |\beta_0 z'|$, then z is a [common outer witness](#), otherwise it is a [common inner witness](#).
- (c) If $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ have infinitely many [common witnesses](#), then M is a subset of powers of the [primitive root](#) of its [redux](#). Thus, M has infinitely many [common witnesses](#).

Proof. There are two cases to consider depending upon if $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ has a common inner or outer witness.

Case (a): When z' is a common inner witness of $G \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$. The following equations hold:

$$\alpha_1 \alpha_0 z' = z' \beta_1 \beta_0 \quad (5.19)$$

$$u z' = z' v \text{ for any pair } (u, v) \in G^* \quad (5.20)$$

We claim that $z = [\alpha_0 z', \beta_0]_R = [\alpha_1, z' \beta_1]_L$ is a common witness for M . There are two cases depending upon whether β_0 is a suffix of $\alpha_0 z'$ or vice-versa in [Equation \(5.19\)](#).

- (a) When β_0 is a suffix of $\alpha_0 z'$ or equivalently, when α_1 is a prefix of $z' \beta_1$. We get $z = \alpha_0 z' \beta_0^{-1} = \alpha_1^{-1} z' \beta_1$, or equivalently $\alpha_0 z' = z \beta_0$ and $\alpha_1 z = z' \beta_1$. We show that z is a common inner witness for M .

For any $(u, v) \in G^*$,

$$\begin{aligned}
 \alpha_0 u \alpha_1 z &= \alpha_0 u z' \beta_1 && \text{(Substituting } \alpha_1 z = z' \beta_1) \\
 &= \alpha_0 z' v \beta_1 && (z' \text{ is an inner witness of } (u, v)) \\
 &= z \beta_0 v \beta_1. && \text{(Substituting } \alpha_0 z' = z \beta_0)
 \end{aligned}$$

(b) When $\alpha_0 z'$ is a suffix of β_0 or equivalently $z' \beta_1$ is a prefix of α_1 . We get $z = \beta_0 (\alpha_0 z')^{-1} = (z' \beta_1)^{-1} \alpha_1$, or equivalently $\alpha_1 = z' \beta_1 z$ and $z \alpha_0 z' = \beta_0$. We show that z is a common outer witness for M .

For any $(u, v) \in G^*$,

$$\begin{aligned}
 z \alpha_0 u \alpha_1 &= z \alpha_0 u z' \beta_1 z && \text{(Substituting } \alpha_1 = z' \beta_1 z) \\
 &= z \alpha_0 z' v \beta_1 z && (z' \text{ is an inner witness of } (u, v)) \\
 &= \beta_0 v \beta_1 z. && \text{(Substituting } z \alpha_0 z' = \beta_0)
 \end{aligned}$$

Case (b): When z' is a common outer witness of G and $(\alpha_1 \alpha_0, \beta_1 \beta_0)$. Therefore, the following equations hold:

$$z' \alpha_1 \alpha_0 = \beta_1 \beta_0 z' \tag{5.21}$$

$$z' u = v z' \text{ for any pair } (u, v) \in G^* \tag{5.22}$$

We claim that $z = [\alpha_0, \beta_0 z']_R = [z' \alpha_1, \beta_1]_L$ is a witness for M . There are two cases depending upon if α_0 is a suffix of $\beta_0 z'$ or vice-versa in [Equation \(5.21\)](#).

(a) When α_0 is a suffix of $\beta_0 z'$ or equivalently, β_1 is a prefix of $z' \alpha_1$. We get $z = \beta_0 z' \alpha_0^{-1} = \beta_1^{-1} z' \alpha_1$, or equivalently $z \alpha_0 = \beta_0 z'$ and $z' \alpha_1 = \beta_1 z$. We show that z is a common outer witness for M .

For any $(u, v) \in G^*$,

$$\begin{aligned}
 z \alpha_0 u \alpha_1 &= \beta_0 z' u \alpha_1 && \text{(Substituting } z \alpha_0 = \beta_0 z') \\
 &= \beta_0 v z' \alpha_1 && (z' \text{ is an outer witness of } (u, v)) \\
 &= \beta_0 v \beta_1 z && \text{(Substituting } z' \alpha_1 = \beta_1 z)
 \end{aligned}$$

- (b) If $\beta_0 z'$ is a suffix of α_0 or equivalently, $z' \alpha_1$ is a prefix of β_1 . Therefore, $z = \alpha_0 (\beta_0 z')^{-1} = (z' \alpha_1)^{-1} \beta_1$, or equivalently $\alpha_0 = z \beta_0 z'$ and $z' \alpha_1 z = \beta_1$. We show that z is a common inner witness for M .

For any $(u, v) \in G^*$,

$$\begin{aligned} \alpha_0 u \alpha_1 z &= z \beta_0 z' u \alpha_1 z && \text{(Substituting } \alpha_0 = z \beta_0 z') \\ &= z \beta_0 v z' \alpha_1 z && (z' \text{ is an outer witness of } (u, v)) \\ &= z \beta_0 v \beta_1 && \text{(Substituting } z' \alpha_1 z = \beta_1) \end{aligned}$$

Therefore, if there exists a common witness z' for G and $(\alpha_1 \alpha_0, \beta_1 \beta_0)$, then there also exists a common witness z for M . Furthermore, since the redux remains constant in the word equation that relates the common witness of M and those of $G \cup \{(\alpha_1 \alpha_0, \beta_1 \beta_0)\}$, each distinct common witness of M corresponds to a distinct common witness of $G \cup \{(\alpha_1 \alpha_0, \beta_1 \beta_0)\}$, and vice-versa. Consequently, if z' is the unique common witness for $G \cup \{(\alpha_1 \alpha_0, \beta_1 \beta_0)\}$ then z is the unique common witness of M , and conversely.

Suppose the set $G \cup \{(\alpha_1 \alpha_0, \beta_1 \beta_0)\}$ has infinitely many witnesses. According to [Proposition 5.16](#), all the pairs in that set have the same primitive root, let us say (ρ, ρ') . Therefore, $G^*(\alpha_1, \beta_1)(\alpha_0, \beta_0)$ is a subset of powers of (ρ, ρ') and is conjugate. Since M is a cyclic shift of $G^*(\alpha_1, \beta_1)(\alpha_0, \beta_0)$ and is also conjugate, it is also a subset of powers of a primitive root, let us say (ρ_m, ρ'_m) , that is a cyclic shift of (ρ, ρ') . Moreover, α_1 (*resp.* β_1) is an inner (*resp.* outer) witness of (ρ, ρ_m) (*resp.* (ρ', ρ'_m)). We observe that (ρ_m, ρ'_m) is the primitive root of the redux of M . Hence, M is a subset of powers of the primitive root of its redux. \square

5.5.2 Common Witness of a Sumfree Set

We prove $(1) \Rightarrow (2), (3) \Rightarrow (1)$ and $(2) \iff (3)$ in [Theorem 5.26](#). $(3) \Rightarrow (1)$ is straightforward. We show $(2) \iff (3)$ first in the following lemma.

Lemma 5.38

Given a **sumfree set** $M = (\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1)G_2^* \cdots G_k^*(\alpha_k, \beta_k)$. The following are equivalent.

1. z is a **common inner** (*resp.* **common outer**) witness of M .
2. z is a **common inner** (*resp.* **common outer**) witness of each of its **singleton redux**.

Proof. Let $M = (\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1)G_2^* \cdots G_k^*(\alpha_k, \beta_k)$ be a sumfree set. For $i \in \{1, \dots, k\}$, let M_i denote the singleton redux of M keeping only the Kleene star G_i^* , i.e.,

$$M_i = (\alpha_0 \cdots \alpha_{i-1}, \beta_0 \cdots \beta_{i-1})G_i^*(\alpha_i \cdots \alpha_k, \beta_i \cdots \beta_k).$$

The proof of (1) \Rightarrow (2) is trivial, i.e., if z is a common witness of M , then z is also a common witness of $M_i \subseteq M$.

We prove (2) \Rightarrow (1). WLOG, assume that z is a common *inner* witness of each M_i 's. If the redux is an empty pair $(\varepsilon, \varepsilon)$, then $M = G_1^*G_2^* \cdots G_k^*$. It is easy to prove that z is a common inner witness of M by showing that for any arbitrary pair $(u_i, v_i) \in G_i^*$, $u_1u_2 \cdots u_kz = zv_1v_2 \cdots v_k$. Proof by induction on k .

$$\begin{aligned} u_1 \cdots u_{k-1}u_kz &= u_1 \cdots u_{k-1}zv_k && \text{(Since } u_kz = zv_k\text{)} \\ &= zv_1 \cdots v_{k-1}v_k. && \text{(By Induction Hypothesis)} \end{aligned}$$

Now, assume that the redux is nonempty. Since each M_i has a common witness, there exist a common witness, say z_i , for $G_i \cup \{(\alpha_i \cdots \alpha_k \alpha_0 \cdots \alpha_{i-1}, \beta_i \cdots \beta_k \beta_0 \cdots \beta_{i-1})\}$ by [Theorem 5.24](#). There are three possible cases for z_i .

- (a) z_i is a unique common inner witness. Therefore, for any pair $(u_i, v_i) \in G_i^*$,

$$u_iz_i = z_iv_i \tag{5.23}$$

$$\alpha_i \cdots \alpha_k \alpha_0 \cdots \alpha_{i-1}z_i = z_i\beta_i \cdots \beta_k \beta_0 \cdots \beta_{i-1} \tag{5.24}$$

$$z = \alpha_0 \cdots \alpha_{i-1}z_i(\beta_0 \cdots \beta_{i-1})^{-1} = (\alpha_i \cdots \alpha_k)^{-1}z_i\beta_i \cdots \beta_k \text{ (By Lemma 5.37 (a))} \tag{5.25}$$

(b) z_i is a unique common outer witness. Therefore, for any pair $(u_i, v_i) \in G_i^*$,

$$z_i u_i = v_i z_i \quad (5.26)$$

$$z_i \alpha_i \cdots \alpha_k \alpha_0 \cdots \alpha_{i-1} = \beta_i \cdots \beta_k \beta_0 \cdots \beta_{i-1} z_i \quad (5.27)$$

$$z = \alpha_0 \cdots \alpha_{i-1} (\beta_0 \cdots \beta_{i-1} z_i)^{-1} = (z_i \alpha_i \cdots \alpha_k)^{-1} \beta_i \cdots \beta_k \text{ (By Lemma 5.37 (b))} \quad (5.28)$$

(c) When $G_i \cup \{(\alpha_i \cdots \alpha_k \alpha_0 \cdots \alpha_{i-1}, \beta_i \cdots \beta_k \beta_0 \cdots \beta_{i-1})\}$ has infinitely many witnesses, by Proposition 5.16, it is a set of powers of the same primitive root say (ρ_i, ρ'_i) . Therefore z_i belongs to witnesses of (ρ_i, ρ'_i) . From Lemma 5.37 (c), the set M_i reduces to a set of powers of the primitive root of the redux $(\alpha_0 \cdots \alpha_k, \beta_0 \cdots \beta_k)$, say (ρ, ρ') . Note that $\alpha_i \cdots \alpha_k$ (*resp.* $\alpha_0 \cdots \alpha_{i-1}$) is an inner (*resp.* outer) witness of (ρ_i, ρ) . Similarly $\beta_i \cdots \beta_k$ (*resp.* $\beta_0 \cdots \beta_{i-1}$) is an inner (*resp.* outer) witness of (ρ'_i, ρ') .

We show that z is a common inner witness of M , i.e., for any arbitrary pair $(u_i, v_i) \in G_i^*$ (possibly empty), we prove $\alpha_0 u_1 \alpha_1 u_2 \alpha_2 \cdots \alpha_{k-1} u_k \alpha_k z = z \beta_0 v_1 \beta_1 v_2 \beta_2 \cdots \beta_{k-1} v_k \beta_k$. The proof is by induction on the number of singleton reduces, $0 \leq i \leq k$.

Base Case: When $i = 0$, it is vacuously true since z is a witness of the redux.

Inductive Case: Assume for induction that it is true for the first $i - 1$ singleton reduces, i.e.,

$$\alpha_0 u_1 \alpha_1 u_2 \alpha_2 \cdots u_{i-1} \alpha_{i-1} \cdots \alpha_k z = z \beta_0 v_1 \beta_1 v_2 \beta_2 \cdots v_{i-1} \beta_{i-1} \cdots \beta_k .$$

We prove it for the first i singleton reduces, i.e., we show

$$\alpha_0 u_1 \alpha_1 u_2 \alpha_2 \cdots u_{i-1} \alpha_{i-1} u_i \alpha_i \cdots \alpha_k z = z \beta_0 v_1 \beta_1 v_2 \beta_2 \cdots v_{i-1} \beta_{i-1} v_i \beta_i \cdots \beta_k .$$

Following are the three possible cases for the common witness z_i of the set $G_i \cup \{(\alpha_i \cdots \alpha_k \alpha_0 \cdots \alpha_{i-1}, \beta_i \cdots \beta_k \beta_0 \cdots \beta_{i-1})\}$.

1. When z_i is a unique common inner witness. From Equation (5.25), $z = (\alpha_i \cdots \alpha_k)^{-1} z_i \beta_i \cdots \beta_k$.

$$\begin{aligned}
 & \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} u_i \alpha_i \alpha_{i+1} \cdots \alpha_k z \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} u_i z_i \beta_i \cdots \beta_k && \text{(Subs. } z) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} z_i v_i \beta_i \cdots \beta_k && (u_i z_i = z_i v_i) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} \alpha_i \cdots \alpha_k z (\beta_i \cdots \beta_k)^{-1} v_i \beta_i \cdots \beta_k && \text{(Subs. } z_i) \\
 &= z \beta_0 v_1 \beta_1 \cdots v_{i-1} \beta_{i-1} \beta_i \cdots \beta_k (\beta_i \cdots \beta_k)^{-1} v_i \beta_i \cdots \beta_k && \text{(I.H.)} \\
 &= z \beta_0 v_1 \beta_1 \cdots v_{i-1} \beta_{i-1} v_i \beta_i \cdots \beta_k. && \text{(Simplifying)}
 \end{aligned}$$

2. When z_i is a unique outer witness. By Equation (5.28), $z = (z_i \alpha_i \cdots \alpha_k)^{-1} \beta_i \cdots \beta_k$.

$$\begin{aligned}
 & \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} u_i \alpha_i \alpha_{i+1} \cdots \alpha_k z \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} u_i \alpha_i \alpha_{i+1} \cdots \alpha_k (z_i \alpha_i \cdots \alpha_k)^{-1} \beta_i \cdots \beta_k && \text{(Subs. } z) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} u_i z_i^{-1} \beta_i \cdots \beta_k && \text{(Simplifying)} \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} z_i^{-1} v_i \beta_i \cdots \beta_k && (u_i = z_i^{-1} v_i z_i) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} \alpha_i \cdots \alpha_k z (\beta_i \cdots \beta_k)^{-1} v_i \beta_i \cdots \beta_k && \text{(Subs. } z_i^{-1}) \\
 &= z \beta_0 v_1 \beta_1 \cdots v_{i-1} \beta_{i-1} \beta_i \cdots \beta_k (\beta_i \cdots \beta_k)^{-1} v_i \beta_i \cdots \beta_k && \text{(I.H.)} \\
 &= z \beta_0 v_1 \beta_1 \cdots v_{i-1} \beta_{i-1} v_i \beta_i \cdots \beta_k. && \text{(Simplifying)}
 \end{aligned}$$

3. Consider the case where $G_i \cup \{(\alpha_i \cdots \alpha_k \alpha_0 \cdots \alpha_{i-1}, \beta_i \cdots \beta_k \beta_0 \cdots \beta_{i-1})\}$ have infinitely many witnesses, i.e., when z_i is a witness of the primitive root (ρ_i, ρ'_i) of $G_i \cup \{(\alpha_i \cdots \alpha_k \alpha_0 \cdots \alpha_{i-1}, \beta_i \cdots \beta_k \beta_0 \cdots \beta_{i-1})\}$. Here (u_i, v_i) is some m^{th} power of (ρ_i, ρ'_i) . Since z is a witness of a singleton redux, it is also a witness of the redux $(\alpha_0 \cdots \alpha_k, \beta_0 \cdots \beta_k)$, and hence a witness of its primitive root (ρ, ρ') . We also know that $\alpha_i \cdots \alpha_k$ is an inner witness of (ρ_i, ρ) and $\beta_i \cdots \beta_k$ is an inner

witness of (ρ'_i, ρ') .

$$\begin{aligned}
 & \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} u_i \alpha_i \alpha_{i+1} \cdots \alpha_k z \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} (\rho_i)^m \alpha_i \cdots \alpha_k z && \text{(Subs. } u_i) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} \alpha_i \cdots \alpha_k (\rho)^m z && (\alpha_i \cdots \alpha_k \text{ is i.w. of } (\rho_i, \rho)) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} \alpha_i \cdots \alpha_k z (\rho')^m && (z \text{ is a witness of } (\rho, \rho')) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} \cdots \alpha_k z (\beta_i \cdots \beta_k)^{-1} (\rho'_i)^m \beta_i \cdots \beta_k && (\beta_i \cdots \beta_k \text{ is i.w. of } (\rho'_i, \rho')) \\
 &= \alpha_0 u_1 \alpha_1 \cdots u_{i-1} \alpha_{i-1} \cdots \alpha_k z (\beta_i \cdots \beta_k)^{-1} v_i \beta_i \cdots \beta_k && \text{(Subs. } v_i = (\rho'_i)^m) \\
 &= z \beta_0 v_1 \beta_1 \cdots v_{i-1} \beta_{i-1} \beta_i \cdots \beta_k (\beta_i \cdots \beta_k)^{-1} v_i \beta_i \cdots \beta_k && \text{(I.H.)} \\
 &= z \beta_0 v_1 \beta_1 \cdots v_{i-1} \beta_{i-1} v_i \beta_i \cdots \beta_k. && \text{(Simplifying)}
 \end{aligned}$$

Thus z is a common inner witness of M . □

We prove (1) \Rightarrow (2) in [Theorem 5.26](#) in the case when a sumfree set contains only two Kleene stars. Later we extend it to an arbitrary number of Kleene stars.

Lemma 5.39

Let $M = (\alpha_0, \beta_0) G_1^*(\alpha_1, \beta_1) G_2^*(\alpha_2, \beta_2)$. If M is **conjugate** then there exists a **common witness** z such that z is a **common witness** for each of its **singleton reduxes**.

Proof. Consider the singleton redux of M denoted as M_1 and M_2 . We have $M_1 = (\alpha_0, \beta_0) G_1^*(\alpha_1 \alpha_2, \beta_1 \beta_2)$ and $M_2 = (\alpha_0 \alpha_1, \beta_0 \beta_1) G_2^*(\alpha_2, \beta_2)$. Since M is conjugate, it follows that M_1 and M_2 are also conjugate. According to [Theorem 5.24](#), both M_1 and M_2 has a common witness.

If both M_1 and M_2 have infinitely many common witnesses, we can conclude, based on the third item in [Lemma 5.37](#), that both M_1 and M_2 are subsets of powers of the primitive root of the redux of M . Thus, any witness for the primitive root of the redux is also a witness for both M_1 and M_2 using [Proposition 5.10](#). Therefore, it holds true that when both M_1 and M_2 have infinitely many common witnesses.

Let us consider the scenario where exactly one of M_1 and M_2 has a unique common witness. Without loss of generality, let us assume that M_1 has a unique common witness, while M_2 has infinitely many common witnesses. According to [Lemma 5.37](#), we can conclude that M_2 is a subset of powers of the primitive root of the redux. Consequently,

the witnesses of M_2 are the same as the witnesses of the primitive root of the redux. Since the unique witness, say z , for M_1 is also a witness for the redux, we can apply [Proposition 5.10](#) to conclude that z is also a witness for the primitive root of the redux. Therefore, z is also a witness for M_2 .

If both M_1 and M_2 have a unique common witness. From [Lemma 5.37](#), G_1 has a unique common witness z_1 with $(\alpha_1\alpha_2\alpha_0, \beta_1\beta_2\beta_0)$. Moreover, From [Lemma 5.36](#), there exist a pair $(u_1, v_1) \in G$ such that it has a unique common witness with $(\alpha_1\alpha_2\alpha_0, \beta_1\beta_2\beta_0)$ equal to z_1 . Similarly, there exists a pair $(u_2, v_2) \in G_2$ that has a unique common witness with $(\alpha_2\alpha_0\alpha_1, \beta_2\beta_0\beta_1)$, that is same as the unique witness of G_2 say z_2 .

Consider the set $M' = (\alpha_0, \beta_0)(u_1, v_1)^*(\alpha_1, \beta_1)(u_2, v_2)^*(\alpha_2, \beta_2)$, which is a subset of the sumfree set M and therefore M' is conjugate.

We construct a pair in M' and do a case analysis on its cuts. Our objective is to show that a nontrivial relation exists between z_1, z_2 , and the redux for all possible cuts. By equating this relation, we can establish that the unique witnesses of M_1 and M_2 are identical.

Let m be the least common multiple of $|u_1|$ and $|u_2|$, and let C be the length of the redux. We choose n as the smallest number such that $nm \geq C$. Let (x_1, y_1) and (x_2, y_2) be a cut of the primitive root of (u_1, v_1) and (u_2, v_2) respectively.

Consider a pair $(u', v') \in M'$ as follows.

$$\begin{aligned} u' &= \alpha_0 \overbrace{u_1 \cdots u_1}^{2nm} \alpha_1 \overbrace{u_2 \cdots u_2}^{8nm} \alpha_2 \\ v' &= \beta_0 v_1 \cdots v_1 \beta_1 v_2 \cdots v_2 \beta_2 \end{aligned}$$

Since M' is conjugate, (u', v') is conjugate with some cut denoted as (p, q) . We do a case analysis based on the cuts possible and show that M_1 and M_2 share the same unique witness. There are two cases to consider: when the cut in u' occurs at most within the initial $2nm$ length of the block of $x_2y_2 \cdots x_2y_2$, or after it.

Substituting (u_i, v_i) with powers of (x_iy_i, y_ix_i) we get,

$$\begin{aligned} u' &= \alpha_0 \overbrace{x_1y_1 \cdots x_1y_1}^{2nm} \alpha_1 \overbrace{x_2y_2 \cdots x_2y_2}^{2nm} \overbrace{x_2y_2 \cdots x_2y_2}^{6nm} \alpha_2 \\ v' &= \beta_0 y_1x_1 \cdots y_1x_1 \beta_1 y_2x_2 \cdots y_2x_2 y_2x_2 \cdots y_2x_2 \beta_2 \end{aligned}$$

Case 1: When the cut p in u' ends at most within the first $2nm$ length of block $x_2y_2 \cdots x_2y_2$.

$$\begin{aligned} u' &= \overbrace{\alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 x_2 y_2 \cdots x_2 y_2}^{\text{cut region}} \overbrace{x_2 y_2 \cdots x_2 y_2}^{\geq 6nm} \alpha_2 \\ v' &= \beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 y_2 x_2 \cdots y_2 x_2 y_2 x_2 \cdots y_2 x_2 \beta_2 \end{aligned}$$

In this case, the total length of p is less than $5nm$. As the total length of the block consisting of $y_2 x_2$ is at least $8nm$, the cut in v' is at most within the suffix of the block $y_2 x_2 \cdots y_2 x_2$. We compare the suffixes of q in u' and v' . Since the length of the remaining block of $y_2 x_2$ before the cut is still greater than $3nm$, we conclude that α_2 in u' matches at most within the $y_2 x_2$'s in v' .

$$v' = \beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 y_2 x_2 \cdots \underbrace{\cdots y_2 x_2 \beta_2}_{=\alpha_2 p}.$$

There are three possible cases for $\alpha_2 p$.

- (a) $\alpha_2 p$ is a proper suffix of β_2 . We continue comparing the suffixes of q , and deduce that β_2 starts within the block of $x_2 y_2$'s, and there is at least one occurrence of $x_2 y_2$ before it. Moreover, by Cases I. (c), I. (d), I. (e) and II. of [Cut Lemma](#), we determine that β_2 starts from y_2 since there is at least one $y_2 x_2$ preceding β_2 in v' . Therefore, we can express β_2 as $(y_2 x_2)^{m_2} y_2 \alpha_2 p$, where m_2 is an integer greater than or equal to 0. Let $w_2 = (y_2 x_2)^{m_2} y_2$. Continuing the matching of q in u' and v' ,

$$\begin{aligned} u' &= \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 \overbrace{x_2 \cdots x_2}^{\quad} w_2 \alpha_2 = pq \\ v' &= \beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 w_2 \underbrace{x_2 \cdots x_2}_{=} \underbrace{\beta_2}_{w_2 \alpha_2 p} = qp \end{aligned}$$

On matching further, we deduce $p = \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 (\beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 w_2)^{-1}$.

By substituting it in the equation $w_2 \alpha_2 p = \beta_2$, we obtain

$$w_2 \alpha_2 \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 = \beta_2 \beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 w_2.$$

We deduce w_2 is an outer witness for $(\alpha_2 \alpha_0, \beta_2 \beta_0)(x_1 y_1, y_1 x_1) \cdots (x_1 y_1, y_1 x_1)(\alpha_1, \beta_1)$ using [Theorem 3.6](#). Furthermore, w_2 is the unique outer witness for this set, as

(u_1, v_1) and $(\alpha_1\alpha_2\alpha_0, \beta_1\beta_2\beta_0)$ share a unique common witness z_1 . By applying [Lemma 5.37](#), we can equate w_2 accordingly. There are two cases to consider based on whether z_1 is a unique common inner or outer witness.

a) When z_1 is a unique inner witness of (u_1, v_1) and $(\alpha_1\alpha_2\alpha_0, \beta_1\beta_2\beta_0)$.

$$w_2 = \beta_2\beta_0(\alpha_2\alpha_0z_1)^{-1} = (z_1\beta_1)^{-1}\alpha_1. \quad (5.29)$$

We obtain $w_2\alpha_2\alpha_0\alpha_1 = \beta_2\beta_0\beta_1w_2$ by solving [Equation \(5.29\)](#). Since $w_2 \in (y_2x_2)^*y_2$, w_2 is a common outer witness of (x_2, y_2) and $(\alpha_2\alpha_0\alpha_1, \beta_2\beta_0\beta_1)$. Since z_2 is the unique common witness of (u_2, v_2) and $(\alpha_2\alpha_0\alpha_1, \beta_2\beta_0\beta_1)$, we get $w_2 = z_2$, which implies that z_2 is the unique common outer witness.

From [Theorem 5.24](#), the unique witness of M_1 is $[\alpha_0z_1, \beta_0]_R$ and the unique witness of M_2 is $[\beta_0\beta_1z_2, \alpha_0\alpha_1]_R$. We show that they are equal as follows.

$$\begin{aligned} [\beta_0\beta_1z_2, \alpha_0\alpha_1]_R &= [\beta_0\beta_1z_2, \alpha_0z_1\beta_1z_2]_R \quad (\text{Using Equation (5.29)}) \\ &= [\beta_0, \alpha_0z_1]_R \quad ([uw, vw]_R = [u, v]_R \text{ for all } w, u, v) \\ &= [\alpha_0z_1, \beta_0]_R. \quad ([u, v]_R = [v, u]_R \text{ for all } u, v) \end{aligned}$$

Thus the witness of M_1 and M_2 are the same.

b) When z_1 is a unique outer witness of (u_1, v_1) and $(\alpha_1\alpha_2\alpha_0, \beta_1\beta_2\beta_0)$.

$$w_2 = \beta_2\beta_0z_1(\alpha_2\alpha_0)^{-1} = \beta_1^{-1}z_1\alpha_1. \quad (5.30)$$

We get $w_2\alpha_2\alpha_0\alpha_1 = \beta_2\beta_0\beta_1w_2$ by solving [Equation \(5.30\)](#). Since $w_2 \in (y_2x_2)^*y_2$, w_2 is a common outer witness of (x_2, y_2) and $(\alpha_2\alpha_0\alpha_1, \beta_2\beta_0\beta_1)$. Since z_2 is the unique common witness of (u_2, v_2) and $(\alpha_2\alpha_0\alpha_1, \beta_2\beta_0\beta_1)$, we get $w_2 = z_2$, which implies z_2 is the unique common outer witness.

From [Theorem 5.24](#), the unique witness of M_1 is $[\beta_1\beta_2, z_1\alpha_1\alpha_2]_L$ and the unique witness of M_2 is $[\beta_2, z_2\alpha_2]_L$. We show that they are equal as follows.

$$\begin{aligned} [\beta_1\beta_2, z_1\alpha_1\alpha_2]_L &= [\beta_1\beta_2, \beta_1z_2\alpha_2]_L \quad (\text{Using Equation (5.30)}) \\ &= [\beta_2, z_2\alpha_2]_L. \quad ([wu, wv]_L = [u, v]_L \text{ for all } w, u, v) \end{aligned}$$

Thus the witness of M_1 and M_2 are the same.

- (b) $\alpha_2 p = \beta_2$. In this case by further matchings in q we get $x_2 y_2 = y_2 x_2$. Thus $u_2 = v_2$ and hence (u_2, v_2) is an identical cycle with ε as witness. It is the same as the above case where $w_2 = \varepsilon$.
- (c) When β_2 is proper suffix of $\alpha_2 p$. β_1 is proper suffix of $\alpha_1 p$. We continue by comparing the suffixes of q in u' and v' . Since $|\alpha_2 p| \leq C + 5nm \leq 6nm$ and the total length of the block of $y_2 x_2$'s is at least $8nm$, $\alpha_2 p$ starts within the $y_2 x_2$'s, and there is at least one occurrence of $x_2 y_2$ before that. Furthermore, it starts from x_2 by using Cases I. (a), I. (b), I. (e) and II. of [Cut Lemma](#), as there exists at least one $x_2 y_2$ before α_2 in u' . Thus, we can write $\alpha_2 p = (x_2 y_2)^{m_2} x_2 \beta_2$, where $m_2 \geq 0$. Let $w_2 = (x_2 y_2)^{m_2} x_2$.

$$\begin{aligned} u' &= \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 w_2 \overbrace{y_2 \cdots y_2}^{\alpha_2 p} \alpha_2 \\ v' &= \beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 \underbrace{y_2 \cdots y_2}_{\alpha_2 p} \underbrace{w_2 \beta_2}_{\alpha_2 p} \end{aligned}$$

On matching further, we deduce $p = \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 w_2 (\beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1)^{-1}$. By substituting it in the equation $\alpha_2 p = (x_2 y_2)^{m_2} x_2 \beta_2$, we obtain

$$\alpha_2 \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 w_2 = w_2 \beta_2 \beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 w_2 .$$

We deduce w_2 is an inner witness for $(\alpha_2 \alpha_0, \beta_2 \beta_0)(x_1 y_1, y_1 x_1) \cdots (x_1 y_1, y_1 x_1)(\alpha_1, \beta_1)$ using [Theorem 3.6](#). Furthermore, w_2 is the unique inner witness for this set, as (u_1, v_1) and $(\alpha_1 \alpha_2 \alpha_0, \beta_1 \beta_2 \beta_0)$ share a unique common witness z_1 . By applying [Lemma 5.37](#), we can equate w_2 accordingly. There are two cases to consider based on whether z_1 is a unique common inner or outer witness.

- a) When z_1 is a unique inner witness of (u_1, v_1) and $(\alpha_1 \alpha_2 \alpha_0, \beta_1 \beta_2 \beta_0)$.

$$w_2 = \alpha_2 \alpha_0 z_1 (\beta_2 \beta_0)^{-1} = (\alpha_1)^{-1} z_1 \beta_1 . \quad (5.31)$$

We deduce $\alpha_2 \alpha_0 \alpha_1 w_2 = w_2 \beta_2 \beta_0 \beta_1$ by solving [Equation \(5.31\)](#). Since $w_2 \in (x_2 y_2)^* x_2$, w_2 is a common inner witness of (x_2, y_2) and $(\alpha_2 \alpha_0 \alpha_1, \beta_2 \beta_0 \beta_1)$. Since z_2 is the unique common witness of (u_2, v_2) and $(\alpha_2 \alpha_0 \alpha_1, \beta_2 \beta_0 \beta_1)$, we get $w_2 = z_2$, which implies z_2 is the unique common inner witness.

From [Theorem 5.24](#), the unique witness of M_1 is $[\alpha_0 z_1, \beta_0]_R$ and the unique witness of M_2 is $[\alpha_0 \alpha_1 z_2, \beta_0 \beta_1]_R$. We show that they are equal as follows.

$$\begin{aligned} [\alpha_0 \alpha_1 z_2, \beta_0 \beta_1]_R &= [\alpha_0 z_1 \beta_1, \beta_0 \beta_1]_R \quad (\text{Using Equation (5.31)}) \\ &= [\alpha_0 z_1, \beta_0]_R. \quad ([uw, vw]_R = [u, v]_R \text{ for all } w, u, v) \end{aligned}$$

Thus the witness of M_1 and M_2 are the same.

b) When z_1 is a unique outer witness of (u_1, v_1) and $(\alpha_1 \alpha_2 \alpha_0, \beta_1 \beta_2 \beta_0)$.

$$w_2 = \alpha_2 \alpha_0 (\beta_2 \beta_0 z_1)^{-1} = (z_1 \alpha_1)^{-1} \beta_1. \quad (5.32)$$

We obtain $\alpha_2 \alpha_0 \alpha_1 w_2 = w_2 \beta_2 \beta_0 \beta_1$ by solving [Equation \(5.32\)](#). Since $w_2 \in (x_2 y_2)^* x_2$, w_2 is a common inner witness of (x_2, y_2) and $(\alpha_2 \alpha_0 \alpha_1, \beta_2 \beta_0 \beta_1)$. Since z_2 is the unique common witness of (u_2, v_2) and $(\alpha_2 \alpha_0 \alpha_1, \beta_2 \beta_0 \beta_1)$, we get $w_2 = z_2$, which implies z_2 is the unique common inner witness.

From [Theorem 5.24](#), the unique witness of M_1 is $[\beta_1 \beta_2, z_1 \alpha_1 \alpha_2]_L$ and the unique witness of M_2 is $[\alpha_2, z_2 \beta_2]_L$. We show that they are equal as follows.

$$\begin{aligned} [\beta_1 \beta_2, z_1 \alpha_1 \alpha_2]_L &= [z_1 \alpha_1 z_2 \beta_2, z_1 \alpha_1 \alpha_2]_L \quad (\text{Using Equation (5.32)}) \\ &= [z_2 \beta_2, \alpha_2]_L \quad ([wu, wv]_L = [u, v]_L \text{ for all } w, u, v) \\ &= [\alpha_2, z_2 \beta_2]_L. \quad ([u, v]_L = [v, u]_L \text{ for all } u, v) \end{aligned}$$

Thus the witness of M_1 and M_2 are the same.

Case 2: When the cut in u' ends after the first $2nm$ length of the block $x_2 y_2 \cdots x_2 y_2$.

$$\begin{aligned} u' &= \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 \overbrace{x_2 y_2 \cdots x_2 y_2}^{\geq 2nm} \overbrace{x_2 y_2 \cdots x_2 y_2}^{\text{cut region}} \\ v' &= \beta_0 y_1 x_1 \cdots y_1 x_1 \beta_1 y_2 x_2 \cdots y_2 x_2 y_2 x_2 \cdots y_2 x_2 \beta_2 \end{aligned}$$

We compare the suffixes of p in u' and v' . In v' , β_2 starts matching within the block of $x_2 y_2$'s in u' since the length of β_2 is at most C , which is less than or equal to nm , and the length of the block of $x_2 y_2$'s before the cut is at least $2nm$.

$$u' = \alpha_0 x_1 y_1 \cdots x_1 y_1 \alpha_1 x_2 y_2 \cdots \overbrace{\cdots x_2 y_2}^{\beta_2 q} \alpha_2$$

There are three possible cases for $\beta_2 q$, which are symmetric to the three cases for $\alpha_2 p$ discussed earlier:

(a) $\beta_2 q$ is a proper suffix of α_2 .

(b) $\beta_2 q = \alpha_2$.

(c) α_2 is a proper suffix of $\beta_2 q$. □

Theorem 5.40

If a **sumfree set** M is **conjugate**, then the set of all **singleton reduxes** of M has a **common witness**.

Proof. Let M consist of k singleton reduxes where $k \geq 1$. If the redux of M is empty, then each singleton redux is of the form G_i^* , $1 \leq i \leq k$ and $M = G_1^* G_2^* \cdots G_k^*$. If all G_i^* has infinitely many witnesses, each G_i^* is the power of a primitive root, say (ρ_i, ρ'_i) with the same witnesses as G_i^* . There exists integers $n_i > 0$, such that $P = (\rho_1, \rho'_1)^{n_1} \cdots (\rho_k, \rho'_k)^{n_k} \subset M$ and is conjugate. From [Lemma 5.31](#) and [Proposition 5.14](#), there exist a common witness for the set $\{(\rho_i, \rho'_i) \mid 1 \leq i \leq k\}$. Hence, a common witness exists for the set of all G_i^* 's. The case when some G_i^* has a unique witness is similar. For each G_i^* that has a unique common witness, there exists a finite subset of G_i , say $\{u_{i_1}, \dots, u_{i_m}\}$, $m > 1$, that shares the unique witness. For those G_i^* , we replace the primitive roots with $(u_{i_1}, v_{i_1})^* \cdots (u_{i_m}, v_{i_m})^*$ in P . From [Lemma 5.31](#), we get a common witness exists for $\bigcup_{1 \leq i \leq k} G_i^*$.

Now assume that the redux of M is nonempty. If all singleton redux has infinitely many common witnesses, then each singleton redux is a subset of powers of the primitive root of the redux by [Theorem 5.24\(c\)](#). Hence, any witness for the primitive root of the redux is a witness for any singleton redux of M . Therefore, a common witness exists for the set of singleton reduxes of M .

Suppose there are ℓ singleton reduxes with unique witnesses $z_{n_1}, z_{n_2}, \dots, z_{n_\ell}$, where $1 \leq \ell \leq k$. We claim that $z_{n_1} = z_{n_2} = \cdots = z_{n_\ell}$. For any two positions $i, j \in \{n_1, \dots, n_\ell\}$, since the subset of M obtained by keeping the Kleene star at positions i and j is conjugate, according to [Lemma 5.39](#), $z_i = z_j$.

Thus, all the unique witnesses of the ℓ singleton reduxes are the same, and let it be z . Since z is also a witness for the redux, as stated in [Proposition 5.10](#), it is a witness for the primitive root of the redux. Therefore, z is also a witness for all singleton reduxes with infinitely many witnesses, as they are sets of powers of the primitive root of the redux. Hence, z is a common witness of each singleton reduxes of M . □

5.6 Computing Witness of a Sumfree Expression

In this section, we give a decision procedure to compute the **common witness** of a **sumfree expression**, if it exists. A sumfree expression can have no common witness, a unique common witness, or infinitely many common witnesses. Thus, the set of common witnesses (abbreviated as the **witness set**) is either empty, singleton, or infinite. Whenever there are infinitely many common witnesses for an expression, the witnesses are the same as those of its **primitive root** (Proposition 5.16). In that case, we compute the primitive root as their finite representation.

The following proposition shows that there is a bound to the size of the unique common witness of two **conjugate** pairs if it exists, which aids in computing the common witness of two pairs in linear time.

Proposition 5.41

If the set of two **conjugate primitive** pairs (u_1, v_1) and (u_2, v_2) have a unique **common witness** z , then $|z| \leq 2 \cdot \max(|u_1|, |u_2|)$.

Proof. Let z be a common inner witness. Therefore, $z = (x_1y_1)^{n_1}x_1 = (x_2y_2)^{n_2}x_2$ for some $n_1, n_2 \geq 0$ where (x_i, y_i) is a **cut** of the primitive pair (u_i, v_i) for $i \in \{1, 2\}$. We claim either n_1 or n_2 is less than 2. Suppose not, i.e., $n_1 \geq 2$ and $n_2 \geq 2$. Thus u_1^ω and u_2^ω share a common prefix of length at least $|u_1| + |u_2|$. From Theorem 3.10, they have the same primitive root. It implies that $x_1y_1 = x_2y_2$ since u_1 and u_2 are primitive words. Since $(x_1y_1)^{n_1}x_1 = (x_2y_2)^{n_2}x_2$, $x_1y_1 = x_2y_2$ and $|x_1|, |x_2| < |x_1y_1|$, we obtain $n_1 = n_2$, and hence, $x_1 = x_2$. This implies $y_1 = y_2$ and hence, $y_1x_1 = y_2x_2$. Both (u_1, v_1) and (u_2, v_2) are the same word; thus, they have infinitely many common witnesses, that is a contradiction. Hence $|z| \leq 2 \cdot \max\{|u_1|, |u_2|\}$.

The case for common outer witness is symmetric. □

The above proposition holds true for any two conjugate pairs (not necessarily primitive) by Proposition 5.14.

Proposition 5.42

The **witness set** of two **conjugate** pairs can be computed in linear time.

Proof. Since the common witness for a set and its primitive root are the same (Proposition 5.14), we compute the witness set for the primitive roots of the given two conjugate pairs. We can compute the primitive roots in linear time by Proposition 3.3. Let $G = \{(u_1, v_1), (u_2, v_2)\}$ be the set of primitive roots of the given conjugate pairs and let (x_i, y_i) be the cut of (u_i, v_i) for $i \in \{1, 2\}$. The cut of the primitive pair (u_i, v_i) can be computed in linear time by finding the first nontrivial occurrence of v_i in $u_i u_i$.

One of the following possibilities holds true for G : it has no common witness, a unique common witness, or infinitely many common witnesses. The following algorithm outlines the computation of the witness set of G :

1. Check if the primitive pairs are identical, i.e., verify if $u_1 = u_2$ and $v_1 = v_2$. If yes, then G has infinitely many common witnesses by Proposition 5.16. The witness is finitely represented by the primitive pair (u_1, v_1) . This step takes linear time w.r.t. the length of the primitive pairs.
2. If the pairs are not identical, then check if G has a unique common witness using Proposition 5.41 as follows: WLOG assume that $|u_1| > |u_2|$. According to Proposition 5.41, if a unique common witness exists for G , its length is at most $2 \cdot \max(|u_1|, |u_2|) = 2 \cdot |u_1|$. Thus, it suffices to check whether $(x_1 y_1)^\omega$ and $(x_2 y_2)^\omega$ share equal prefixes of length $|x_1|$ or $|x_1 y_1 x_1|$, that also end in x_2 . If it is satisfied, then G has a unique common witness. This step can be performed in linear time w.r.t. the length of the primitive pairs.
3. If none of the above holds, then G has no common witness.

The overall complexity of the algorithm is linear w.r.t the length of the primitive roots of the given pairs. \square

The witness set of a sumfree expression is equal to the intersection of witness sets of each of its singleton reduces. So first, we show how to compute the witness set of a sumfree expression with only one Kleene star, in effect the witness set of a singleton redux. Using this procedure, we show how to compute the witness set of a general sumfree expression.

Lemma 5.43

Let $M = (\alpha_0, \beta_0)E^*(\alpha_1, \beta_1)$ be a **sumfree expression**. Given the **witness set** of E , we can compute the **witness set** of M in $\mathcal{O}(m + n)$ time where m is the **length** of the expression M , and n is the size of the **common witness** of E .

Proof. If **redux** of M is empty, then the witness set of M is same as the witness set of E by [Theorem 5.17](#). Now assume that M has a nonempty **redux**. From [Theorem 5.24](#), M has a common witness iff $E \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ has a common witness. The common witness of M is computed from the common witness of E and $(\alpha_1\alpha_0, \beta_1\beta_0)$.

The idea is to check if a common witness exists for $E \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ using [Lemma 5.36](#). If it exists, using that we compute the common witness for M . There are two possibilities for common witnesses of E :

1. E has a unique common inner (*resp.* outer) witness z . By [Item 2](#) of [Lemma 5.36](#), it suffices to check if z is a common inner (*resp.* outer) witness of $(\alpha_1\alpha_0, \beta_1\beta_0)$. This can be checked in $\mathcal{O}(m + n)$ time using [Theorem 3.6](#). If so, z is the common witness of $E \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$. Now compute the common witness of M using [Theorem 5.24\(a\)](#) (*resp.* [Theorem 5.24\(b\)](#)). This can be computed in $\mathcal{O}(m + n)$ time. Otherwise, $E \cup \{(\alpha_1\alpha_0, \beta_1\beta_0)\}$ has no common witness and hence, M has no common witness by [Theorem 5.24](#).
2. E has infinitely many common witnesses. In this case, the witnesses of E are the same as that of its primitive root, say (ρ, ρ') . From [Item 1](#) of [Lemma 5.36](#), it suffices to check if (ρ, ρ') and $(\alpha_1\alpha_0, \beta_1\beta_0)$ has a common witness. For this, first check if the primitive root of $(\alpha_1\alpha_0, \beta_1\beta_0)$ is equal to (ρ, ρ') . This step takes time $\mathcal{O}(m + n)$. We have two cases:
 - (a) If $(\alpha_1\alpha_0, \beta_1\beta_0)$ have same primitive root as that of E , then E and $(\alpha_1\alpha_0, \beta_1\beta_0)$ have infinitely many common witnesses by [Proposition 5.16](#). In this case, M is a set of powers of the primitive root of the **redux** by [Theorem 5.24\(c\)](#). Thus M has infinitely many witnesses. Compute the primitive root of its **redux** using [Proposition 3.3](#). This step takes $\mathcal{O}(m)$ time.
 - (b) Otherwise, compute the unique common witness of (ρ, ρ') and $(\alpha_1\alpha_0, \beta_1\beta_0)$ if it exists using [Proposition 5.42](#). If so, we are back to *Case 1*; otherwise the

set M has no common witness. This step takes $\mathcal{O}(m + n)$ time. \square

Using the above algorithm, we compute the common witness of a general sumfree expression as follows.

Lemma 5.44

Let M be a **sumfree expression**. Given the **witness set** of each Kleene star in M , we can compute the **witness set** of M in time $\mathcal{O}(m \cdot (m + n))$ where m is the **length** of the expression and n is the maximum size among the given **common witnesses**.

Proof. From Lemma 5.38, the witness set of M is the intersection of the witness sets of its singleton reduxes. Check if each singleton redux of M has a common witness and compute it using Lemma 5.43. This step takes $\mathcal{O}(m \cdot (m + n))$. If there is a singleton redux with no common witness, then M has no common witness by Theorem 5.26.

Assume that M has a nonempty redux. The algorithm is as follows. Firstly, check if the redux of M is conjugate using Proposition 3.7 (in time $\mathcal{O}(m)$). If yes, then compute the primitive root of the redux, say (ρ_m, ρ'_m) , using Proposition 3.3 (in time $\mathcal{O}(m)$). Otherwise, M has no common witness. There are two cases depending on whether all singleton redux has infinitely many witnesses.

- (a) If all the singleton reduxes have infinitely many witnesses, then M is a set of powers of the primitive root of the redux (ρ_m, ρ'_m) by Theorem 5.24(c). Thus, M has infinitely many common witnesses.
- (b) If there exists a singleton redux with a unique common witness, say z , then for all other singleton reduxes of M with a unique witness z' , check if $z = z'$ (for all other singleton reduxes z is already a witness by virtue of being a witness of the redux of M). If so, z is the unique common witness of M ; otherwise M has no common witness.

This takes $\mathcal{O}(mn + m)$ time. Consider the case where the redux of M is empty. Let M be of the form $E_1^* E_2^* \cdots E_k^*$. We iterate over $i \in \{1, \dots, k - 1\}$.

- (a) If both E_i and E_{i+1} has infinitely many witnesses, then the witness set of E_i^* and E_{i+1}^* is finitely represented by their primitive roots. Check if the primitive

roots are the same. If yes, continue the procedure for $i = i + 1$. Otherwise, check and compute the common witness, say z , of the primitive roots if exists using [Proposition 5.42](#) (in time $\mathcal{O}(n)$). If not, then M has no common witness. Otherwise, check if z is a common witness of all singleton reduxes E_{i+2}^*, \dots, E_k^* (This can be done in $\mathcal{O}(n)$ for each singleton redux — If a singleton redux has a unique witness then check if it is equal to z . Otherwise, if it has infinitely many witness, then it is represented by a primitive root. It suffices to check if z is a witness of that primitive root).

- (b) If either one of E_i and E_{i+1} has a unique witness, say z , then check if z is a common witness of all singleton reduxes E_{i+2}^*, \dots, E_k^* .

This step takes $\mathcal{O}(mn + n)$ time. Overall, it takes $\mathcal{O}(m \cdot (m + n))$ time. \square

Computation of the Witness Set: Given a sumfree expression M , we compute its witness set bottom-up. We start from the innermost Kleene star. It is a pair of words (u, v) . First, we check if (u, v) is conjugate using [Proposition 3.7](#). If yes, then there are infinitely many common witnesses for $(u, v)^*$, namely the witnesses of its primitive root, otherwise M has no witness. This step can be done in a time linear in the length of (u, v) . Now we recursively use [Lemma 5.44](#) to compute the common witness of the expression under the Kleene star in each level. If there is no common witness for any level of Kleene star expression, then M is not conjugate.

To find out the complexity of the decision procedure, it suffices to estimate the maximum length of a witness involved in the computation.

Length of the Witness of a Sumfree Expression: We claim that if a sumfree expression M is conjugate, then there exists a witness of length linear in size of M . If M has infinitely many witnesses, from [Proposition 5.16](#), M is a set of powers of a primitive root. Therefore, there exists a witness of length that is less than that of the length of the primitive root. Next, suppose M has a unique common witness. In that case, there exists a subexpression E_i^* such that E_i^* has a unique common witness, and all Kleene star appearing in E_i has infinitely many witnesses. Thus, all of them have a common witness of length at most $|E_i|$. Therefore, there is a singleton redux M_i of E_i^* that has a unique witness z_i . The size of z_i is linear in M_i and the size of the witnesses of subexpressions

of E_i . Both are upper bounded by size of M . Furthermore, the common witnesses for all subsequent levels is unique (if it exists) and its length is bounded by $|M|$.

Complexity of the Algorithm: Since the size of the common witness of M is linear in $|M|$, by Lemma 5.44, the overall complexity of computing a common witness of a sumfree expression is $\mathcal{O}(h \cdot m^2)$ where h is the [star height](#) of M and m is the [length](#) of the expression.

5.7 Conclusion

It is shown that the conjugacy problem of a rational relation is decidable. The current decision procedure proceeds through the analysis of rational expressions. In its essence, it is analogous to the boundedness checking of distance automata using factorisation trees ([Colcombet, 2021](#)), though explicit use of factorisation trees are avoided using sumfree rational expressions instead. An obvious question is the existence of an automata-theoretic proof. Factorisation forests remain the primary tool to settle boundedness questions on automata and by that standard the proof approach taken in this chapter is natural and quite possibly the most intuitive.

Computing a witness of a given sumfree expression, if one exists, can be done in polynomial time. However, converting a rational expression into a sum of sumfree expressions may result in an exponential blowup (see [Remark 2.11](#) and [Lemma 2.9](#)). Thus, the algorithm presented in the chapter is of exponential time. It remains to find the precise complexity of this problem.

5.8 Notes

The [twinning property](#) of a transducer, used in characterising the [sequentiality](#) of [rational function](#), is related to conjugacy. The original definition of twinning has the following equivalent form. Let \mathcal{T} be a [trim transducer](#) with input alphabet A and output alphabet B , and let \mathcal{T}^2 denote the [cartesian product](#) of \mathcal{T} with itself. A transducer \mathcal{T} is *twinning* if for all initial state (p_0, q_0) in \mathcal{T}^2 , for all states (p, q) that are reachable in \mathcal{T}^2 , for all

words $u, v \in A^*$ and $u_1, u_2, v_1, v_2 \in B^*$, if

$$(p_0, q_0) \xrightarrow{u|(u_1, u_2)}_{\mathcal{T}^2} (p, q) \xrightarrow{v|(v_1, v_2)}_{\mathcal{T}^2} (p, q)$$

then either $v_1 = v_2 = \varepsilon$, or there exists a finite word z such that either $u_2 = u_1 z$ and $v_1 z = z v_2$ (equivalently z is an **inner witness** of (v_1, v_2)), or $u_1 = u_2 z$ and $z v_1 = v_2 z$ (equivalently z is an **outer witness** of (v_1, v_2)).

For deciding twinning, since the input words in \mathcal{T}^2 are inconsequential, compute rational relations over pairs of output words along the loops rooted at reachable states in \mathcal{T}^2 . Checking the twinning property reduces to computing a number of instances of the following problem: decide if a given rational relation is conjugate with a given witness.

Generalisation of the twinning property called *weak twinning* is used to characterise *multi-sequential* (also called *plurisubsequential* or *finitely sequential*) functions (Choffrut and Schützenberger, 1986) and relations (Jecker and Filiot, 2018). This property also can be stated as conjugacy of suitably defined rational relations with respect to certain predetermined witnesses. The results in this chapter holds the potential to define more general properties of transducers where the witnesses are not known a priori, for instance determinising approximately (detailed in Chapter 7).

A generalisation of Lyndon-Schützenberger’s theorem (Theorem 3.6) to infinite sets, though with no comparison to ours, is considered by Cassaigne et al. (2001) and Karhumäki (2001), where solutions to the language equation $XZ = ZY$, where X, Y, Z are sets of words, are given for special cases. The general solution is still open.

Computing Distance of Rational Functions

Contents

| | | |
|-------|---|-----|
| 6.1 | Introduction | 113 |
| 6.2 | Deciding Closeness for Edit Distances | 115 |
| 6.2.1 | Closeness w.r.t. Levenshtein Family and Conjugacy | 117 |
| 6.2.2 | Closeness w.r.t. Hamming and Transposition | 121 |
| 6.3 | Deciding k -closeness for Edit distances | 130 |
| 6.3.1 | k -closeness for Hamming and Transposition | 131 |
| 6.4 | Conclusion | 133 |

6.1 Introduction

We study the problems stated in Table 4.1 regarding the distance between [rational functions](#) (or, equivalently those definable by [functional transducers](#)) and prove that they are decidable for the [metrics](#) in Table 1.1. Recall that [distance](#) problem w.r.t. a

¹The contributions presented in this chapter are published in the proceedings of ICALP 2024 under the title “[Edit distance of finite state transducers](#)”. This is a joint work with Dr. C. Aiswarya and Dr. Amaldev Manuel.

integer-valued metric d is computable if and only if both closeness and k -closeness are decidable (See Proposition 4.9). Therefore, it suffices to show that both closeness and k -closeness are decidable for all the metrics in Table 4.1.

Given two functional transducers, the first step in deciding closeness as well as k -closeness is to verify if the domains of the transducers are the same. This reduces to checking the equivalence of the underlying automata of the transducers. The underlying automata for functional transducers are nondeterministic in general and checking their equivalence is PSPACE-complete (Stockmeyer and Meyer, 1973). However, if the given transducers are unambiguous (resp. sequential), then the underlying automata are unambiguous (resp. deterministic), and checking their equivalence is in polynomial time (Stearns and Hunt III, 1985). Thus, from now on we assume that the domains of the transducers given as input to the closeness or k -closeness problem are identical.

For functional transducers with identical domains, the problems in Table 4.1 can be reduced to that for sequential transducers by considering the cartesian product of the functional transducers as follows. We use representation 1 for transducers in this chapter.

Proposition 6.1

Let d be a metric on words. For each pair of functional transducers \mathcal{T}_1 and \mathcal{T}_2 with identical domain, there exists a deterministic finite state automaton \mathcal{A} and output functions λ'_1, o'_1 and λ'_2, o'_2 such that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$ where the sequential transducer $\mathcal{T}'_i = (\mathcal{A}, \lambda'_i, o'_i), i \in \{1, 2\}$. Furthermore, the size of the automaton \mathcal{A} is polynomial in the size of \mathcal{T}_1 and \mathcal{T}_2 .

Proof. Assume we are given two functional transducers \mathcal{T}_1 and \mathcal{T}_2 with the input alphabet A and the output alphabet B . The idea is to construct a deterministic automaton whose language is the set of pairs of accepting runs of \mathcal{T}_1 and \mathcal{T}_2 on words in their domain.

For convenience, we can assume that \mathcal{T}_1 and \mathcal{T}_2 have exactly one initial state each, denoted by i_1 and i_2 . Let Δ_i, Q_i and $F_i, i \in \{1, 2\}$, be the transitions, the set of states, and the set of final states of \mathcal{T}_i .

The input alphabet of \mathcal{A} is going to be $A' = \Delta_1 \times \Delta_2$. The deterministic automaton \mathcal{A} has the set of states $Q_1 \times Q_2$, initial state (i_1, i_2) , set of final states $F_1 \times F_2$, and the set of transitions (denoted as Δ) of the form $((p, q), (\delta, \chi), (p', q'))$ such that $\delta \in \Delta_1, \chi \in \Delta_2$ are of the form (p, a, p') and (q, a, q') for some letter $a \in A$. It is easy to verify that the

language accepted by \mathcal{A} , denoted as $L' \subseteq (\Delta_1 \times \Delta_2)^*$, is the language of all sequences of pairs of the form $\rho = (\delta_1, \chi_1) \cdots (\delta_k, \chi_k)$, $k \geq 1$ and for each $i \in [k]$, $\delta_i \in \Delta_1$, $\chi_i \in \Delta_2$, and there exists a word $w = a_1 \cdots a_k$ such that

1. δ_i and χ_i are transitions on the same letter a_i ,
2. $\delta_1 \cdots \delta_k$ (denoted as $\pi_1(\rho)$) is an **accepting** run of \mathcal{T}_1 on w ,
3. $\chi_1 \cdots \chi_k$ (denoted as $\pi_2(\rho)$) is an accepting run of \mathcal{T}_2 on w .

Let $\lambda_i : \Delta_i \rightarrow B^*$, $o_i : F_i \rightarrow B^*$, $i \in \{1, 2\}$, be the output functions of \mathcal{T}_i . We define $\lambda'_1 : \Delta \rightarrow B^*$ and $\lambda'_2 : \Delta \rightarrow B^*$ as follows: for $\psi = ((p, q), (\delta, \chi)(p', q')) \in \Delta$, we let $\lambda'_1(\psi) = \lambda_1(\delta)$ and $\lambda'_2(\psi) = \lambda_2(\chi)$. Similarly, we define $o'_1, o'_2 : F_1 \times F_2 \rightarrow B^*$ as $o'_1(p, q) = o_1(p)$ and $o'_2(p, q) = o_2(q)$, for each $(p, q) \in F_1 \times F_2$.

We take \mathcal{T}'_1 and \mathcal{T}'_2 to be the automaton \mathcal{A} with the output functions λ'_1, o'_1 and λ'_2, o'_2 respectively. Clearly, they are of polynomial size w.r.t. \mathcal{T}_1 and \mathcal{T}_2 .

It remains to show that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$. It is easy to show that there is a correspondence between L and L' , namely for each $w \in L$, there exists a $\rho \in L'$ such that $\pi_1(\rho)$ and $\pi_2(\rho)$ are accepting runs of \mathcal{T}_1 and \mathcal{T}_2 on w respectively such that $\mathcal{T}_1(w) = \mathcal{T}'_1(\rho)$ and $\mathcal{T}_2(w) = \mathcal{T}'_2(\rho)$. Similarly, for each $\rho \in L'$, there exists a $w \in L$ such that $\mathcal{T}_1(w) = \mathcal{T}'_1(\rho)$ and $\mathcal{T}_2(w) = \mathcal{T}'_2(\rho)$. Thus, $\{d(\mathcal{T}_1(w), \mathcal{T}_2(w)) \mid w \in L\} = \{d(\mathcal{T}_1(\rho), \mathcal{T}_2(\rho)) \mid \rho \in L'\}$. Hence, we conclude that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$. \square

Henceforth, we solve closeness and k -closeness problems for sequential transducers. The decidability of closeness and k -closeness for sequential transducers are shown in Section 6.2 and Section 6.3 respectively w.r.t. metrics listed in Table 1.1. By virtue of Proposition 6.1, we get the following result.

Theorem 6.2

The **closeness** and **k -closeness** problems for **functional transducers** with respect to all the **metrics** given in Table 1.1 are decidable.

6.2 Deciding Closeness for Edit Distances

Recall that the **distance** between two **rational functions** is mutually reducible to the **diameter** of **rational relation** (See Section 4.3). Hence, the **closeness** problem is interreducible

to the **bounded diameter** problem. Specifically, given two transducers \mathcal{T}_1 and \mathcal{T}_2 with identical domain, for any **metric** d ,

$$d(\mathcal{T}_1, \mathcal{T}_2) < \infty \iff \text{dia}_d(R) < \infty \quad (6.1)$$

where $R = \{(\mathcal{T}_1(w), \mathcal{T}_2(w)) \mid w \in \text{dom}(\mathcal{T}_1)\}$ is a rational relation **defined** by the **output product** transducer of \mathcal{T}_1 and \mathcal{T}_2 obtained by ignoring the input word in $\mathcal{T}_1 \times \mathcal{T}_2$ (**cartesian product** of \mathcal{T}_1 and \mathcal{T}_2).

For **Hamming** and **transposition** distances, we show the decidability of the closeness of transducers, while for the **Levenshtein family** of distances — **Levenshtein**, **Damerau-Levenshtein** and **longest common subsequence**, and **conjugacy** distance, we show the decidability of boundedness of diameter of a rational relation.

Conjugacy of words plays an important role in deciding boundedness of diameter.

Lemma 6.3

Let x, y, x', y', u, v be words over A^* and $c, k \in \mathbb{N}$. For any **metric** d in Table 1.1, if $d(xu^\ell y, x'v^{c\ell}y') \leq k$ for all $\ell \geq 0$, then $|u| = |v^c|$ and the **primitive roots** of u and v are **conjugate**.

Proof. Since $d(xu^\ell y, x'v^{c\ell}y') \leq k$, we get $|u| = |v^c|$. Otherwise, as ℓ increases the length difference of $xu^\ell y$ and $x'v^{c\ell}y'$ increases, and hence their distance will not be bounded.

Since $|u| = |v^c|$, either both u and v are nonempty, or $u = v = \varepsilon$. In the latter case, u and v are conjugate. Assume that u and v are nonempty words. Take $\ell = 2^{|u|+|v|}$. Since $d(xu^\ell y, x'v^{c\ell}y') \leq k$ there exist large **factors** of u 's and v 's that match. In fact, u 's and v 's overlap at least of length $|u| + |v|$. By Fine and Wilf's theorem (See **Theorem 3.11**), the primitive roots of u and v are conjugate. \square

Proposition 6.4

Let R be a **rational relation** defined by a **trim transducer** \mathcal{T} . If $\text{dia}_d(R) < \infty$ for a **metric** d given in Table 1.1, then every pair of input-output words generated by loops in \mathcal{T} are **conjugate**.

Proof. Let $\text{dia}_d(R) \leq k$ for some $k \in \mathbb{N}$. Let (u, v) be an input-output pair produced by a loop in \mathcal{T} rooted at some state q . Hence (u^ℓ, v^ℓ) for each $\ell \geq 0$ is also a pair in a loop

rooted at q . Since \mathcal{T} is trim, there exist an initial state q_0 and a final state q_f in \mathcal{T} such that

$$q_0 \xrightarrow{x|x'}_{\mathcal{T}} q \xrightarrow{u|v}_{\mathcal{T}} q \xrightarrow{y|y'}_{\mathcal{T}} q_f$$

Since $\text{dia}_d(R) \leq k$, $d(xu^\ell y, x'v^\ell y') \leq k$ for $\ell \geq 0$. Using [Lemma 6.3](#), we get $|u| = |v|$ and ρ_u and ρ_v , the primitive roots of u and v respectively, are conjugates. Thus, $|\rho_u| = |\rho_v|$, and hence (u, v) is a power of (ρ_u, ρ_v) since $|u| = |v|$. Therefore, u and v are conjugate since the powers of a conjugate pair are also conjugate ([Proposition 5.10](#)). Since the pair (u, v) was arbitrary, any pair generated by a loop in \mathcal{T} is conjugate. \square

In the context of the closeness of two transducers \mathcal{T}_1 and \mathcal{T}_2 , the above proposition implies that all output word pairs along the loop in $\mathcal{T}_1 \times \mathcal{T}_2$ must be conjugate. We show that this is sufficient for the Levenshtein distance. For conjugacy distance, \mathcal{T}_1 and \mathcal{T}_2 are [close](#) iff all output pairs in $\mathcal{T}_1 \times \mathcal{T}_2$ are conjugate. In the case of Hamming distance, which only includes substitutions, it suffices to check if the output pairs along the loops in $\mathcal{T}_1 \times \mathcal{T}_2$ after some shifted delay is identical. This holds true for transposition distance, but additionally, we also need to check if the words are permutations of each other.

6.2.1 Closeness w.r.t. Levenshtein Family and Conjugacy

The Levenshtein family of distances — Levenshtein, Damerau-Levenshtein, and longest common subsequence are all equivalent with up to boundedness by [Lemma 3.17](#) and [Remark 4.8](#), i.e., for a rational relation R , $\text{dia}_{d_l}(R) < \infty \iff \text{dia}_{d_{lcs}}(R) < \infty \iff \text{dia}_{d_{dl}}(R) < \infty$. Henceforth, it suffices to decide diameter boundedness for Levenshtein distance.

In this section, we prove the decidability of boundness for Levenshtein and conjugacy distance. We assume the rational relation is given as a [rational expression](#) over pairs of words. The diameter problem of a rational relation w.r.t. a metric d extends naturally to rational expressions by letting

$$\text{dia}_d(E) = \sup \{d(u, v) \mid (u, v) \in L(E)\}.$$

Clearly, $\text{dia}_d(R) = \text{dia}_d(E)$ for a rational expression E defining R . Thus, the diameter and boundedness problems of a rational relation reduces to the corresponding problems for a rational expression over pairs.

Recall that a rational expression is **sumfree** if it does not use sum (i.e., $+$), and every rational expression is equivalent to a sum of sumfree expressions (see [Lemma 2.9](#)). The proposition below implies that to show a bounded diameter for a sum of sumfree expressions, it suffices to show a bounded diameter for each of its constituent sumfree expressions.

Proposition 6.5

Let $E = E_1 + \dots + E_\ell, \ell \geq 1$ be a **rational expression** of pairs. For all word metrics d , $dia_d(E) = \max(dia_d(E_1), \dots, dia_d(E_\ell))$.

Proof. It suffices to show that if $E = E_1 + E_2$, then $dia_d(E) = \max(dia_d(E_1), dia_d(E_2))$. Since $L(E_i) \subseteq L(E), i \in \{1, 2\}$, we can deduce that $dia_d(E_i) \leq dia_d(E)$ and hence $\max(dia_d(E_1), dia_d(E_2)) \leq dia_d(E)$. It remains to show that $dia_d(E) \leq \max(dia_d(E_1), dia_d(E_2))$. We have two cases.

1. If $dia_d(E) = k$, for $k \in \mathbb{N}$, then for each pair in E the distance is at most k , and there is a pair $(u, v) \in L(E)$ with distance k . Then, for each pair in E_1 as well as E_2 the distance is at most k , and the expression that contains the pair (u, v) has diameter k .
2. If $dia_d(E) = \infty$, then either there is a pair (u, v) with distance ∞ , in which case one of $dia_d(E_1), dia_d(E_2)$ is ∞ . Otherwise, there is an infinite subset of pairs $L \subseteq L(E)$ such that for each $k \in \mathbb{N}$ there is a pair with the distance at least k . Since L is infinite, one of $L \cap L(E_1), L \cap L(E_2)$ is infinite, and the corresponding expression has diameter ∞ . \square

Recall that an expression is conjugate if every pair generated by the expression is conjugate. In the context of conjugacy distance, the bounded diameter of a rational expression necessarily implies that the expression is conjugate. Otherwise, if there exists a pair $(u, v) \in L(E)$ such that u is not conjugate to v , then $d_c(u, v) = \infty$, thus $dia_{d_c}(E) = \infty$. In fact, this is also a sufficient condition. The proof relies on our results from [Chapter 5](#) that studies the conjugacy of rational expression over pairs of words. It crucially uses the notion of **common witness** of a set of pairs. Recall that an expression E has a common witness if the set $L(E)$ has a common witness. The existence of a common witness bounds the diameter of an expression w.r.t. conjugacy distance as follows.

Proposition 6.6

If a **rational expression** over pairs E has a **common witness** z , then $dia_{dc}(E) \leq |z|$.

Proof. Since E has a common witness, either $\forall (u, v) \in L(E), uz = zv$, or $\forall (u, v) \in L(E), zu = vz$. WLOG, assume that $\forall (u, v) \in L(E), uz = zv$. Now, for any pair $(u, v) \in L(E)$:

1. If $|u| > |z|$, then z is a prefix of u and suffix of v and hence $(u, v) = (zp, pz)$ for some word $p \in A^*$. Therefore $d_c(u, v) \leq |z|$ since v can be obtained by $|z|$ left cyclic shifts of u .
2. Otherwise, when $|u| \leq |z|$, the number of cyclic shifts required to transform u to v (note that u and v are conjugate since they have a witness) is less than $|u| \leq |z|$. \square

The following lemma characterises bounded diameter w.r.t. conjugacy distance.

Lemma 6.7

A **rational expression** over pairs of words has **bounded diameter** w.r.t. **conjugacy** distance iff the expression is **conjugate**. Furthermore, it is decidable to check if a **rational expression** has a **bounded diameter** w.r.t. **conjugacy** distance.

Proof. One direction is trivial. Assume E to be an arbitrary rational expression of pairs and is conjugate. Let $E = E_1 + E_2 + \dots + E_k$, for $k \in \mathbb{N}^+$, where E_1, E_2, \dots, E_k are sumfree expressions. Since E is conjugate, each of its sumfree constituent E_i for $i \in [k]$ is also conjugate. Using **Corollary 5.27**, each E_i has a common witness, say z_i . From **Proposition 6.6**, $dia_{dc}(E_i) \leq |z_i|$. By **Proposition 6.5**, $dia_{dc}(E) = \max\{|z_i| \mid i \in [k]\}$, and hence it is bounded w.r.t. conjugacy distance.

Therefore, to decide if E has a bounded diameter w.r.t. conjugacy distance, it suffices to check if E is conjugate. It is shown to be decidable in **Chapter 5** (See **Theorem 5.2**). \square

Now consider the case of Levenshtein distances. From **Proposition 6.4**, if an expression has bounded diameter w.r.t. Levenshtein distances, it is necessary that every pair generated by a Kleene star in the expression needs to be conjugate. Using common witness, we show that it is also a sufficient condition.

Proposition 6.8

If a rational expression of pairs E has a common witness z , then $\text{dia}_{d_l}(E) \leq 2|z|$.

Proof. Since E has a common witness, $\text{dia}_{d_c}(E) \leq |z|$ by Proposition 6.6. Thus, $\forall(u, v) \in L(E), d_c(u, v) \leq |z|$. From Lemma 3.17, $d_l \lesssim d_c$. In fact, $d_l(u, v) \leq 2 \cdot d_c(u, v)$ for all words u and v , since a cyclic shift can be achieved by an insertion and a deletion. Therefore, $\forall(u, v) \in L(E), d_l(u, v) \leq 2|z|$, and hence $\text{dia}_{d_l}(E) \leq 2|z|$. \square

Lemma 6.9

A rational relation R has a bounded diameter w.r.t. Levenshtein distance iff all Kleene stars in any expression generating R are conjugate. Further, it is decidable to check if a rational expression has a bounded diameter w.r.t. Levenshtein distance.

Proof. Let E be a rational expression generating R . From Proposition 6.4, if R has bounded diameter w.r.t. Levenshtein distance, then each Kleene star in E generates a conjugate pair of words. For the other direction, WLOG assume that the rational expression E is equivalent to a sum of sumfree expressions using Lemma 2.9. From Proposition 6.5, to show bounded diameter for a sum of sumfree expressions, it suffices to show bounded diameter for each of its constituent sumfree expressions. The general form of a sumfree expression is

$$E' = (\alpha_0, \beta_0)E_1^*(\alpha_1, \beta_1) \cdots (\alpha_{k-1}, \beta_{k-1})E_k^*(\alpha_k, \beta_k)$$

where $k \in \mathbb{N}$, for $0 \leq j \leq k$, (α_j, β_j) is a (possibly empty) pair of words, and for each $i \in [k]$, E_i is a sumfree expression. By assumption, each E_i^* for $i \in [k]$ is conjugate. By Corollary 5.19, E_i^* has a common witness, say z_i . From Proposition 6.8, $\text{dia}_{d_l}(E_i^*) \leq 2|z_i|$. Further, the diameter of the sumfree expression,

$$\text{dia}_{d_l}(E') \leq \sum_{j \in \{0 \dots k\}} d_l(\alpha_j, \beta_j) + \sum_{i \in \{1 \dots k\}} \text{dia}_{d_l}(E_i^*) = \sum_{j \in \{0 \dots k\}} d_l(\alpha_j, \beta_j) + 2 \sum_{i \in \{1 \dots k\}} |z_i|,$$

and hence bounded.

Therefore, checking if a rational expression has bounded diameter w.r.t. Levenshtein distance reduces to checking the existence of a common witness of each Kleene star in its sumfree constituents, and thus decidable. \square

For a sumfree rational expression, a common witness, if exists, can be computed in polynomial time (See Section 5.6), and thus closeness w.r.t. Levenshtein and conjugacy distances are decidable in polynomial time. However, converting a rational expression to a sum of sumfree rational expressions can cause an exponential blow-up both in the number of summands and the size of each summand (see Section 2.3). Hence, we have the following result.

Theorem 6.10

Checking if a rational relation, given as a rational expression, has a bounded diameter w.r.t. Levenshtein family of distances and conjugacy distance is decidable in exponential time.

6.2.2 Closeness w.r.t. Hamming and Transposition

Towards deciding closeness for Hamming and transposition, we need the following characterisation of closeness w.r.t. the length metric d_{len} . The closeness w.r.t. length metric reduces to checking if a rational relation has a bounded delay (See Equation (6.1)), and it is decidable (Frougny and Sakarovitch, 1991).

Proposition 6.11

Consider the sequential transducers $\mathcal{T}_1, \mathcal{T}_2$ defined by the deterministic finite state automaton \mathcal{A} and output functions λ_1, o_1 and λ_2, o_2 respectively. The following are equivalent.

1. $d_{len}(\mathcal{T}_1, \mathcal{T}_2)$ is finite.
2. There is $k \in \mathbb{N}$ such that on any $w \in L(\mathcal{A})$, the difference in lengths of the outputs of $\mathcal{T}_1, \mathcal{T}_2$ on any prefix of w is bounded by k .

Proof. (2) \Rightarrow (1) is straightforward. For (1) \Rightarrow (2), WLOG assume that automaton \mathcal{A} is trimmed, i.e., all states are accessible (reachable from the initial state) and coaccessible (from each state there is a path to some final state). Let ℓ be the maximum difference in the output lengths on any single transition, and n be the number of states of \mathcal{A} . We claim that, if the difference in output lengths of the transducers on any input is bounded, then $k = n\ell$ validates our proposition. Note that the difference in the output lengths on

any partial input u is at most $|u|\ell$ and we show that this does not exceed k . Suppose $|u|\ell > k = n\ell$, then $|u| > n$, and hence there is a state that is repeated on u such that the repeated part has a nonzero difference in the output lengths. That is, $u = u_1u_2u_3$, with $u_2 \neq \varepsilon$ such that the states reached after u_1 and that reached after u_1u_2 are the same, say q , and $|\lambda_1(\rho)| - |\lambda_2(\rho)| \neq 0$, where ρ is the unique partial run from q on u_2 . If $w = uv$ for some $w \in L(\mathcal{A})$ then $u_1u_2^iu_3v \in L(\mathcal{A})$ for all i , and the difference in the output lengths become unbounded with increasing i . This contradicts that $d_{len}(\mathcal{T}_1, \mathcal{T}_2)$ is finite. Therefore, the length difference in the outputs of $\mathcal{T}_1, \mathcal{T}_2$ on any prefix of w is bounded by $k = n\ell$. \square

By [Proposition 6.1](#), it suffices to consider two sequential transducers with a common underlying deterministic finite state automaton \mathcal{A} . Let $\mathcal{T}_1 = (\mathcal{A}, \lambda_1, o_1)$ and $\mathcal{T}_2 = (\mathcal{A}, \lambda_2, o_2)$ be two sequential transducers. WLOG, we make the following assumptions.

1. (Property \star) Automaton \mathcal{A} is trimmed, i.e., all states are accessible (reachable from the initial state) and coaccessible (has a path to a final state).
2. (Property \ddagger) \mathcal{T}_1 and \mathcal{T}_2 produce output words of identical length; otherwise the Hamming as well as transposition distance will be ∞ . We can check this property: rename all the output letters in \mathcal{T}_1 and \mathcal{T}_2 to a and check their [equivalence](#).
3. The difference in the length (or *delay*) of the partial outputs of \mathcal{T}_1 and \mathcal{T}_2 is at most $k \in \mathbb{N}$ (By [Proposition 6.11](#)).

Let Q and $F \subseteq Q$ be the set of states and final states of \mathcal{A} respectively, and let $q_0 \in Q$ be the initial state. For states $p, q \in Q$, Let $M_{p,q}$ be the set of pairs (u, v) such that there is a run ρ from p to q and $u = \lambda_1(\rho)$ and $v = \lambda_2(\rho)$. Extending this notation, for a state $q_f \in F$, let M'_{q,q_f} be the set of pairs (u, v) such that $u = u' \cdot o_1(q_f)$, $v = v' \cdot o_2(q_f)$ and $(u', v') \in M_{q,q_f}$.

Let q be a state of the automaton. If (α, β) and (α', β') are two pairs in $M_{q_0,q}$, then $|\alpha| - |\beta| = |\alpha'| - |\beta'|$, or else one of the pairs in $\{(\alpha\alpha'', \beta\beta''), (\alpha'\alpha'', \beta'\beta'')\}$ will have different lengths, where (α'', β'') is some pair in M_{q,q_f} , for some $q_f \in F$, guaranteed by Property (\ddagger) . Therefore, with each state q , we can associate the delay of a run reaching it, called the [delay at](#) q , denoted by ∂_q , as $|\alpha| - |\beta|$. Clearly $\partial_q \leq k$. By a symmetric argument, if (α, β) and (α', β') are two pairs in M_{q,q_f} , where q_f is some final state, then $|\alpha| - |\beta| = |\alpha'| - |\beta'| = -\partial_q$. This also implies that for all $(u, v) \in M_{q,q}$, $|u| = |v|$.

For each state q , either $M_{q,q} = \{(\varepsilon, \varepsilon)\}$, or $M_{q,q}$ is infinite. Let q be a state for which $M_{q,q}$ is infinite. For a delay $\partial \in \mathbb{Z}$, a pair $(u, v) \in M_{q,q}$ where $n = |u| > \partial$, we define the *interior* of the pair (u, v) as

$$\text{interior}_{\partial}(u, v) = \begin{cases} (u[1 \dots n - \partial], v[\partial + 1 \dots n]) & \text{if } \partial \geq 0 \\ (u[\partial + 1 \dots n], v[1 \dots n - \partial]) & \text{if } \partial < 0 \end{cases}$$

For example, $\text{interior}_1(abc, def) = (ab, ef)$ and $\text{interior}_{-1}(abc, def) = (bc, de)$. We also define the *Left-Border* and *Right-Border* of the pair (u, v) as

$$\text{lborder}_{\partial}(u, v) = \begin{cases} v[1 \dots \partial] & \text{if } \partial \geq 0 \\ u[1 \dots \partial] & \text{if } \partial < 0 \end{cases} \quad \text{rborder}_{\partial}(u, v) = \begin{cases} u[n - \partial + 1 \dots n] & \text{if } \partial \geq 0 \\ v[n - \partial + 1 \dots n] & \text{if } \partial < 0 \end{cases}$$

Claim 6.12. *Hamming distance between \mathcal{T}_1 and \mathcal{T}_2 is unbounded iff there exists a state $q \in Q$ and $(u, v) \in M_{q,q}$ such that $|u| = |v| > \partial_q$, and $u' \neq v'$ where $(u', v') = \text{interior}_{\partial_q}(u, v)$.*

Proof. The Figure 6.1 depicts the situation described by (2).

(\leftarrow): Assume there exists a state $q \in Q$ and $(u, v) \in M_{q,q}$ such that $|u| = |v| > \partial_q$, and $u' \neq v'$ where $(u', v') = \text{interior}_{\partial_q}(u, v)$. Let $(\alpha_0, \beta_0) \in M_{q_0,q}$ and $(\alpha_1, \beta_1) \in M'_{q,q_f}$. Consider the pair $(u_i, v_i) = (\alpha_0 u^i \alpha_1, \beta_0 v^i \beta_1)$, $i \geq 1$ (shown in Figure 6.2). Since $u' \neq v'$, we can deduce that $d_h(u_i, v_i) \geq i$. Hence $d_h(\mathcal{T}_1, \mathcal{T}_2) = \infty$.

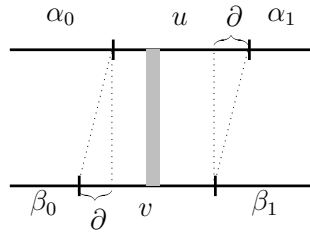


Fig. 6.1. An edit in the interior of u and v .

(\rightarrow): Assume $d_h(\mathcal{T}_1, \mathcal{T}_2) = \infty$. Assume \mathcal{A} has n states and the maximum length of an output produced on any transition or at the end-of-input is ℓ . Choose a *run* ρ of \mathcal{A} such that the distance between the outputs produced on $\rho = \delta_1 \cdots \delta_m$, $m > 0$ is at least $((k+2)n+1)\ell$. We can associate each edit in $\lambda_1(\rho)$ with the transition δ_i such that the edit happens in $\lambda_1(\delta_i)$ for $i \in [m]$. Since there are $((k+2)n+1)\ell$ edits, there are at least

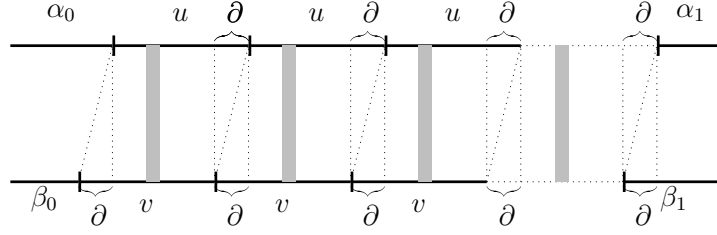


Fig. 6.2. Words that require arbitrarily large number of edits.

$(k + 2)n + 1$ transitions in ρ whose output words are edited. Associate each transition with its source state. By pigeonhole principle, there is a state q such that $\rho = \rho_1 \cdot \rho_2 \cdot \rho_3$ where

1. ρ_1 is a run from the initial state q_0 to q ,
2. ρ_2 is a run from q to itself,
3. ρ_3 is a run from q to a final state q_f , and
4. there are at least $(k + 1)$ edits in the factor $\lambda_1(\rho_2)$.

Let $u = \lambda_1(\rho_2)$ and $v = \lambda_2(\rho_2)$. Clearly $|u| = |v|$ and $|u| \geq (k + 1)$. Since the edits in u are at least $k + 1$, there is a position on which the pair $\text{interior}_{\partial_q}(u, v)$ differ.

□*Claim 6.12*

Lemma 6.13

The closeness problem of two functional transducers with identical domain w.r.t. Hamming distance is decidable in polynomial time.

Proof. By Proposition 6.1, it suffices to consider two sequential transducers $\mathcal{T}_1 = (\mathcal{A}, \lambda_1, o_1)$ and $\mathcal{T}_2 = (\mathcal{A}, \lambda_2, o_2)$ with a common underlying deterministic finite state automaton \mathcal{A} . Let Q and $F \subseteq Q$ be the set of states and final states of \mathcal{A} respectively, and let $q_0 \in Q$ be the initial state. Assume that \mathcal{A} is trim.

We begin by verifying that Property (\ddagger) holds, i.e., \mathcal{T}_1 and \mathcal{T}_2 produce output words of the same length. Recall that this is done by checking their equivalence after renaming all output letters to a , and it takes polynomial time. This step also ensures that the delay at each state is unique. Now the unique delay at each state can be computed in polynomial

time. The delay at the initial state is set to be zero. We then process each transition of \mathcal{A} using a breadth-first search and compute the delay of the target state.

Now, it suffices to decide the characterisation in [Claim 6.12](#), i.e., for each state q for which $M_{q,q}$ is infinite, we need to check if the $\text{interior}(q) = \{\text{interior}_{\partial_q}(u, v) \mid (u, v) \in M_{q,q}\}$ consist of only identical pairs of words.

We construct a *forward* and *backward* gadget for each state that unfolds the loop up to its delay in both directions trimming the *lborder* and *rborder* respectively. For a state q with the delay ∂_q , if $\partial_q > 0$, then the forward gadget of q will produce pairs $\{(u[1 \dots \partial_q], \varepsilon) \mid (u, v) \in M_{q,q}, |u| > \partial_q\}$, and the backward gadget of q will produce pairs $\{(\varepsilon, v[(n - \partial_q) \dots n]) \mid (u, v) \in M_{q,q}, n = |v| > \partial_q\}$. The construction is symmetric for $\partial_q < 0$.

Instead of constructing these gadgets for each state, we combine the construction for each strongly connected component (SCC). In polynomial time, we can find all the strongly connected components (SCCs) of the automaton \mathcal{A} . For an SCC containing a set of states $S \subseteq Q$, let ∂ be the maximum absolute value of delay among all the states in S . We then construct a forward and backward gadget (or automaton) \mathcal{F}_S and \mathcal{B}_S respectively with states from $S \times \{-\partial, -\partial + 1, \dots, 0, 1, 2, \dots, \partial\}$ with output functions $\lambda_{f_1}, \lambda_{f_2}$ for \mathcal{F}_S and $\lambda_{b_1}, \lambda_{b_2}$ for \mathcal{B}_S .

Initially, \mathcal{F}_S contains states of the form (q, ∂_q) and \mathcal{B}_S contains states of the form $(q, -\partial_q)$ for $q \in S$. For a transition $\delta = (p, a, q)$ in \mathcal{A} where $a \in A$ and $p, q \in S$ with $n_1 = |\lambda_1(\delta)|$ and $n_2 = |\lambda_2(\delta)|$, we define transitions as follows.

- In forward gadget, for each state $(p, i) \in \mathcal{F}_S$, we add a transition $\delta_f = ((p, i), a, (q, j))$ such that

$$\begin{aligned}
 & 1. \text{ if } i > 0, \text{ then } \lambda_{f_1}(\delta_f) = \lambda_1(\delta), \lambda_{f_2}(\delta_f) = \begin{cases} \varepsilon & \text{if } i \geq n_2 \\ \lambda_2(\delta)[i + 1 \dots n_2] & \text{otherwise} \end{cases} \\
 & \quad \text{and assign } j = i - n_2 \text{ if } i \geq n_2; \text{ else } j = 0. \\
 & 2. \text{ if } i < 0, \text{ then } \lambda_{f_2}(\delta_f) = \lambda_2(\delta), \lambda_{f_1}(\delta_f) = \begin{cases} \varepsilon & \text{if } i + n_1 \leq 0 \\ \lambda_1(\delta)[i + 1 \dots n_1] & \text{otherwise} \end{cases} \\
 & \quad \text{and assign } j = i + n_1 \text{ if } i + n_1 \leq 0; \text{ else } j = 0.
 \end{aligned}$$

- In backward gadget, for each state $(q, i) \in \mathcal{B}_S$, we add a transition $\delta_b = ((p, j), a, (q, i))$ such that

1. if $i > 0$, then $\lambda_{b_1}(\delta_b) = \lambda_1(\delta)$, $\lambda_{b_2}(\delta_b) = \begin{cases} \varepsilon & \text{if } i \geq n_2 \\ \lambda_2(\delta)[1 \dots n_2 - i] & \text{otherwise} \end{cases}$
and assign $j = i - n_2$ if $i \geq n_2$; else $j = 0$.
2. if $i < 0$, then $\lambda_{b_2}(\delta_b) = \lambda_2(\delta)$, $\lambda_{b_1}(\delta_b) = \begin{cases} \varepsilon & \text{if } i + n_1 \leq 0 \\ \lambda_1(\delta)[1 \dots n_1 - i] & \text{otherwise} \end{cases}$
and assign $j = i + n_1$ if $i + n_1 \leq 0$; else $j = 0$.

We construct such a forward and backward gadget for each SCC. For each state q in an SCC with a set of states S , we construct an automaton \mathcal{A}_{S_q} by concatenating \mathcal{F}_S , \mathcal{A} restricted to states in S , and \mathcal{B}_S by adding following additional transitions.

- For each state $(p, 0) \in \mathcal{F}_S$, $p \in \mathcal{A}$, add $((p, 0), \varepsilon, p)$ with ε output labels.
- For each state $p \in \mathcal{A}$, $(p, 0) \in \mathcal{B}_S$, add $(p, \varepsilon, (p, 0))$ with ε output labels.

Let $\lambda'_1 = \lambda_{f_1} \cup \lambda_1 \cup \lambda_{b_1}$ and $\lambda'_2 = \lambda_{f_2} \cup \lambda_2 \cup \lambda_{b_2}$ be the output labels in \mathcal{A}_{S_q} , and o'_1, o'_2 maps each state to ε in \mathcal{A}_{S_q} . Let the initial state of \mathcal{A}_{S_q} being (q, ∂_q) and final state being $(q, -\partial_q)$. It is easy to see that the automaton \mathcal{A}_{S_q} is unambiguous and is of polynomial size w.r.t. \mathcal{A} . To verify [Claim 6.12](#), it suffices to check if the unambiguous transducers $(\mathcal{A}_{S_q}, \lambda'_1, o'_1)$ and $(\mathcal{A}_{S_q}, \lambda'_2, o'_2)$ are equivalent for each state q in a SCC with a set of states S . \square

Next we show closeness w.r.t. transposition distance. We write $u \equiv v$ to denote that words u and v are permutations of each other. The *alphabetic vector* of a word over the alphabet A , denoted by \vec{u} , is the sequence $(|w|_{a_i})_{a_i \in A}$ for some fixed ordering of A . It is easy to observe that two words are permutations of each other if their alphabetic vectors are the same.

Claim 6.14. *Transposition distance between \mathcal{T}_1 and \mathcal{T}_2 is unbounded if and only if one of the following holds*

1. There is a pair $(u, v) \in M'_{q_0, q_f}$, $q_f \in F$ such that $u \not\equiv v$.
2. There exists a state $q \in Q$ and $(u, v) \in M_{q, q}$ such that $|u| = |v| > \partial_q$, and $u' \neq v'$ where $(u', v') = \text{interior}_{\partial_q}(u, v)$.

3. There exists a state $q \in Q$ such that $M_{q,q}$ is infinite, and for each pair $(u, v) \in M_{q,q}$ of length at least $|\partial_q|$, $\text{interior}_{\partial}(u, v)$ is identical. Further, there are pairs $(u, v) \in M_{q,q}$ and $(\alpha, \beta) \in M_{q_0,q}$ (resp. M'_{q,q_f}) such that: If $\partial_q \geq 0$, then $\alpha \not\equiv \beta \cdot \text{lborder}(u, v)$ (resp. $\text{rborder}(u, v) \cdot \alpha \not\equiv \beta$), and if $\partial_q < 0$, then $\alpha \cdot \text{lborder}(u, v) \not\equiv \beta$ (resp. $\alpha \not\equiv \text{rborder}(u, v) \cdot \beta$).

Proof. (\leftarrow): It is obvious that if Item 1 is true, then the transposition distance between \mathcal{T}_1 and \mathcal{T}_2 is unbounded. Therefore we assume that the output pairs of the transducers are permutations of each other. For Item 2, the proof is the same as in Claim 6.12. Next we consider Item 3. The cases are symmetric. Assume that there exist a pair $(u, v) \in M_{q,q}$, $(\alpha, \beta) \in M_{q_0,q}$, and WLOG $\partial_q \geq 0$ such that $\alpha \not\equiv \beta \cdot \text{lborder}(u, v)$. Let (α', β') be some pair in M'_{q,q_f} . Consider the pair $(u_i, v_i) = (\alpha u^i \alpha', \beta v^i \beta')$, $i \geq 1$.

Let $(x, x) = \text{interior}_{\partial_q}(u, v)$, $z_1 = \text{lborder}(u, v)$, $z_2 = \text{rborder}(u, v)$. By assumption $\alpha \not\equiv \beta z_1$, and hence $z_2 \alpha' \not\equiv \beta'$. Since interior of (u, v) is (x, x) , we can deduce that $\alpha z_2 \alpha' \equiv \beta z_1 \beta'$. Therefore $\vec{\alpha} - \beta z_1 = z_2 \vec{\alpha}' - \vec{\beta}'$. This means that the transpositions have to cancel out the differences in the vectors at each end of the word. We can prove by induction that it requires at least $|x|$ transpositions to mitigate a difference of 1, while keeping the alphabetic vector of the middle portion the same. Hence we deduce that $d_t(u_i, v_i) \geq i$.

(\rightarrow): If $d_t(\mathcal{T}_1, \mathcal{T}_2) = \infty$, either there is a pair of outputs (u, v) such that $d_t(u, v) = \infty$ (This is Item 1), or all the output pairs are permutations of each other and there is an infinite set of pairs $S = \{(u_i, v_i) \mid i > 0\}$ such that $d_t(u_i, v_i) \geq i$.

In the latter case, we show that either Item 2 or Item 3 holds. We say the set S is *error-bounded* if there is an $r > 0$ such that u_i and v_i differ in at most r positions. Clearly, there are sets with bounded errors on which d_t is infinite. We do case analysis.

If there is an infinite set of pairs $S = \{(u_i, v_i) \mid i > 0\}$ such that $d_t(u_i, v_i) \geq i$ that is *not* error-bounded, we proceed as in the proof of Claim 6.12 and obtain Item 2 by pigeonhole principle.

If the set of all output pairs is error-bounded, then clearly for states q such that $M_{q,q}$ is infinite, the interior of all the sufficiently large pairs in $M_{q,q}$ are identical. Moreover, since the output pairs are permutations of each other there is a state q such that $|M_{q,q}| = \infty$ and there is a partial run from q_0 to q (or a partial run from q to q_f) whose output words are not permutations of each other. \square Claim 6.14

Lemma 6.15

The **closeness** problem of two **functional transducers** with identical **domain** w.r.t. **transposition** distance is decidable in polynomial time.

Proof. By [Proposition 6.1](#), it suffices to consider two sequential transducers $\mathcal{T}_1 = (\mathcal{A}, \lambda_1, o_1)$ and $\mathcal{T}_2 = (\mathcal{A}, \lambda_2, o_2)$ with a common underlying deterministic finite state automaton \mathcal{A} .

The Condition 2 in [Claim 6.14](#) is the same as that of the characterisation in [Claim 6.12](#), and thus can be checked in polynomial time by [Lemma 6.13](#). If it does not satisfy Condition 2, then the transducers are not close. Otherwise, we get that for any state q in \mathcal{A} , *interior*(q) produces only identical pairs of outputs. In fact, we can ignore the *interior* of the loops, and by only keeping *lborder* and *rborder* of any loops, we construct two transducers \mathcal{T}'_1 and \mathcal{T}'_2 whose underlying automaton is a polynomially sized acyclic automaton from \mathcal{A} such that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$ for $d \in \{d_h, d_t\}$. Towards this, we construct a forward and backward gadget similar to the one constructed in [Lemma 6.13](#). However, in this case, instead of trimming the *lborder* (*resp.* *rborder*) in the forward (*resp.* backward) gadget, we keep only *lborder* (*resp.* *rborder*) and make the other component ε , i.e., for a state q with the delay ∂_q , if $\partial_q > 0$, then the forward gadget of q will produce pairs $\{(\varepsilon, v[1 \dots \partial_q]) \mid (u, v) \in M_{q,q}, |v| \geq \partial_q\}$, and the backward gadget of q will produce pairs $\{(u[(n - \partial_q) \dots n], \varepsilon) \mid (u, v) \in M_{q,q}, n = |u| > \partial_q\}$. Symmetrically for $\partial_q < 0$. We construct such a forward and backward gadget for each SCC in \mathcal{A} same as in proof of [Lemma 6.13](#), with states from $S \times \{-\partial, -\partial + 1, \dots, 0, 1, 2, \dots, \partial\}$ where S is the set of states in the SCC and ∂ be the maximum absolute value of delay among all the states in S . Let $\lambda_{f_1}, \lambda_{f_2}$ and $\lambda_{b_1}, \lambda_{b_2}$ denote the output function of transitions within the forward and backward gadgets respectively.

We construct two transducers $\mathcal{T}'_i = (\mathcal{A}', \lambda'_i, o'_i)$ for $i \in [2]$ where \mathcal{A}' is an acyclic automaton obtained from \mathcal{A} as follows: construct a directed acyclic graph of strongly connected components from \mathcal{A} . Then, replace each SCC with its combined forward and backward gadget by merging the states of the form $(p, 0)$ in both of these gadgets where p is a state in the SCC. The output functions λ'_i for $i \in [2]$ consists of output functions λ_{f_i} and λ_{b_i} within the gadgets. Further, for any transition $\delta = (p, a, q)$ between two adjacent SCC, we add a transition from $\delta' = ((p, -\partial_p), a, (q, \partial_q))$ connecting the combined gadgets of the two different SCCs with $\lambda'_i(\delta') = \lambda_i(\delta)$. This combined gadget

is acyclic. The initial state of \mathcal{A}' is set to be (q_0, ∂_{q_0}) where q_0 is the initial state of \mathcal{A} , and final states of \mathcal{A}' are set of all states $(q_f, -\partial_{q_f})$ with $o'_i(q_f, -\partial_{q_f}) = o_i(q_f)$ such that q_f is a final state in \mathcal{A} . Therefore, we get a polynomially sized acyclic automaton \mathcal{A}' from \mathcal{A} by removing the *interior* of any state in \mathcal{A} . Since characterisation in [Claim 6.12](#) and Condition 2 in [Claim 6.14](#) is satisfied for \mathcal{T}_1 and \mathcal{T}_2 , we get $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$ for $d \in \{d_h, d_t\}$.

Now, for checking Conditions 1 and 3 in [Claim 6.14](#), we need to compute a vector (or sequence) of length $|A|$ for some fixed ordering of A where A is the alphabet of \mathcal{A} such that each position in the vector tells the difference in the count of a letter encountered till now. It can either be a positive value or a negative value. For example, for $A = \{a, b\}$, $(2 \ -3)$ means there are 2 a 's more (or less) in the output of the first (or second) transducer and there are 3 b 's less (or more) in the output of the first (or second) transducer seen till now. We compute such a vector for each state in \mathcal{A}' . Assume that \mathcal{A}' is trimmed. Initially, we set a zero vector (i.e., $(0 \ 0)$) for the initial state of \mathcal{A}' . For each transition in \mathcal{A}' , we compute the vector of the target state from the computed vector of the origin state. For verifying Condition 3, we need to check if the vector associated with the states of the form $(p, 0)$ in \mathcal{A}' is a zero vector. This ensures that the prefixes of the output words produced before entering the interior of the loops have the same alphabetic vectors. Similarly, for verifying Condition 2, check if the vector associated with the final states in \mathcal{A}' after accommodating the difference in the outputs produced by the final state, is a zero vector. For each state in \mathcal{A}' , we can associate a unique vector. Otherwise, if there are two different vectors associated, then since \mathcal{A}' is trimmed, there exists an accepting path visiting this state that reaches the final state with different vectors, and hence consequently violating Condition 1 in [Claim 6.14](#). Therefore, we can compute the unique vector associated with each state in \mathcal{A}' and verify Conditions 1 and 3 in [Claim 6.14](#). \square

Remark 6.16. *Note that if two transducers \mathcal{T}_1 and \mathcal{T}_2 are close w.r.t. Hamming or transposition distance, there exist two transducers \mathcal{T}'_1 and \mathcal{T}'_2 with a common underlying acyclic automaton such that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$ for $d \in \{d_h, d_t\}$.*

Since checking equivalence of domain of two unambiguous transducers is in polynomial time ([Stearns and Hunt III, 1985](#)), the closeness of unambiguous transducers w.r.t. hamming and transposition distance is decidable in polynomial time via [Lemma 6.13](#) and [Lemma 6.15](#) respectively.

6.3 Deciding k -closeness for Edit distances

Our decision procedure for checking k -closeness for transducers involves constructing an automaton that nondeterministically performs at most k edit operations on the output of the first transducer to match with the output of the second transducer and accepts the input word if the matching is successful. We need to check whether there are input instances which are not accepted in this automaton. We extract a finite state automaton of size exponential in k that achieves this. This is a generic approach independent of the particular edit operations. However, for Hamming and transposition distances, we have a direct polynomial time procedure for deciding k -closeness, detailed in Section 6.3.1.

Theorem 6.17

The k -closeness problem for functional transducers with identical domains is decidable w.r.t. all the metrics listed in Table 1.1.

Proof. By virtue of Proposition 6.1, it suffices to consider two sequential transducers $\mathcal{T}_1 = (\mathcal{A}, \lambda_1, o_1)$ and $\mathcal{T}_2 = (\mathcal{A}, \lambda_2, o_2)$ with a common underlying deterministic finite state automaton \mathcal{A} . First, check if the transducers are close w.r.t. length metric. By Proposition 6.11, there is a maximum delay $\partial \in \mathbb{N}$ between any partial outputs of \mathcal{T}_1 and \mathcal{T}_2 on any input.

For all metrics $d \in \{d_l, d_h, d_t, d_{lcs}, d_{dl}\}$, fix a set of edits C supported in the metric. We construct an automaton $\mathcal{A}_{C,k}$ that reads an input word $w \in L(\mathcal{A})$ and accepts it if we can perform at most k edits in C to $\mathcal{T}_1(w)$ and obtain $\mathcal{T}_2(w)$. For this the states of automaton $\mathcal{A}_{C,k}$ are those of \mathcal{A} augmented with a budget b and unmatched leftover word of the form (u, ε) or (ε, u) with $|u| \leq \max(\partial, k)$. Initially, the budget is k , and the word is $(\varepsilon, \varepsilon)$. While performing a transition δ of \mathcal{A} , from a state annotated with b and (ε, u) , the automaton $\mathcal{A}_{C,k}$ nondeterministically performs $b' \leq b$ edits from C so as to partially match $(\lambda_1(\delta), u\lambda_2(\delta))$. It then moves on to the target state with the reduced budget $b - b'$ and the updated leftover word. The automaton accept w only if it reaches a final state q_f with budget $b \geq 0$, leftover word (u, v) such that $d(u o_1(q_f), v o_2(q_f)) \leq b$. The distance between \mathcal{T}_1 and \mathcal{T}_2 w.r.t. C is at most k iff $L(\mathcal{A}_{C,k}) = L(\mathcal{A})$.

For conjugacy distance, we construct a new automaton \mathcal{A}_k that remembers a prefix of length at most k of the output of the first (or second) transducer and matches the shifted output with the remaining output of the transducer. For this, the states of automaton \mathcal{A}_k

are those of \mathcal{A} augmented with a word of the form (u, v) with $\text{abs}(|u| - |v|) \leq \partial$, and either $|u| \leq k$ or $|v| \leq k$. Here, u (*resp.* v) is a suitable candidate for the prefix of the output of the first (*resp.* second) transducer to be matched with the suffix of the output of the second (*resp.* first) transducer. Initially the word is $(\varepsilon, \varepsilon)$. While performing a transition δ of \mathcal{A} , from a state annotated with (u, v) , the automaton \mathcal{A}_k either choose to perform step 1 (stores the output of first and second transducer of at most k length) or until nondeterministically chooses to perform step 2 where it starts matching the output of first and second transducer and finally check if the stored output matches with the remaining output.

1. Go to the target state with updated word as $(u', v') = (u\lambda_1(\delta), v\lambda_2(\delta))$ only if $\text{abs}(|u'| - |v'|) \leq \partial$, and $|u'| \leq k$ or $|v'| \leq k$.
2. There are two symmetric cases. Start to partially match the output of the first (*resp.* second) transducer $u\lambda_1(\delta)$ (*resp.* $v\lambda_2(\delta)$) with that of the upcoming output of the second (*resp.* first) transducer, i.e., $\lambda_2(\delta)$ (*resp.* $\lambda_1(\delta)$) only if $|v| \leq k$ (*resp.* $|u| \leq k$). It then moves on to the target state with updated word (ε, v) (*resp.* (u, ε)) and also the leftover word obtained after matching $u\lambda_1(\delta)$ and $\lambda_2(\delta)$ (*resp.* $v\lambda_2(\delta)$ and $\lambda_1(\delta)$). For the upcoming transitions on \mathcal{A} , the matching continues until the output of the second (*resp.* first) transducer is exhausted. While matching, the state may need to additionally store the leftover words of the form (w, ε) or (ε, w) . After that, it checks if the stored word v (*resp.* u) matches the first (*resp.* second) transducer's remaining output.

The transducers are k -close w.r.t. conjugacy distance iff $L(\mathcal{A}_k) = L(\mathcal{A})$.

Note that the size of $\mathcal{A}_{C,k}$ and \mathcal{A}_k is exponential in k (as it has to keep track of the unprocessed words). □

6.3.1 k -closeness for Hamming and Transposition

For [Hamming](#) and [transposition](#) distance, using the characterisations used for deciding their [closeness](#) (see [Claim 6.12](#) and [Claim 6.14](#)), we give a co-NP procedure for deciding k -closeness and co-NP \cap NP procedure for computing the distance.

Theorem 6.18

The k -closeness problem w.r.t. [Hamming](#) and [transposition](#) distance are decidable for [unambiguous transducers](#) in co-NP time.

Proof. Given two unambiguous transducers \mathcal{T}_1 and \mathcal{T}_2 , check in polynomial time that if they are close w.r.t. Hamming (or Transposition) distance. If not, then the distance is ∞ and they are not k -close. Assume they are close. By [Remark 6.16](#), there exist two transducers \mathcal{T}'_1 and \mathcal{T}'_2 with a common underlying acyclic automaton such that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$ for $d \in \{d_h, d_t\}$. Now, it suffices to check if all paths from an initial to a final state in the acyclic automaton require only at most k edits (substitutions in the case of Hamming distance, while adjacent transpositions in the case of Transposition distance) to convert the output of the first transducer to that of second. The complement of this problem is asking if there exists a path that requires at least $k + 1$ edits to convert one output to another. This is in NP since given a certificate, we can polynomially compute the number of edits required to convert one output to another (for hamming - $\mathcal{O}(n)$ and for transposition - $\mathcal{O}(n \log n)$) ([Cicirello, 2020](#)) where n is the length of the output words), and hence verify polynomially that whether it requires $k + 1$ edits. Since the complement of this problem is in NP, the problem is in co-NP. \square

The k -distance problem between two transducers w.r.t. a metric d , for a given $k \geq 0$, asks whether the distance between the transducers w.r.t. d is exactly k . As a consequence of the above result, we obtain the following.

Theorem 6.19

The k -distance problem of two [unambiguous transducers](#) w.r.t. [Hamming](#) as well as [transposition](#) distance is the intersection of an NP language and a co-NP language.

Proof. We begin by checking whether the given transducers \mathcal{T}_1 and \mathcal{T}_2 are *close* with respect to the Hamming or transposition distance, which can be done in polynomial time. If they are not close, their distance is ∞ and the answer is No.

If the transducers are close, there exist two transducers \mathcal{T}'_1 and \mathcal{T}'_2 with a common underlying acyclic automaton \mathcal{A}' such that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$ for $d \in \{d_h, d_t\}$ (See [Remark 6.16](#)). Moreover, the distance is bounded by the product of the number of transitions in \mathcal{A}' and the maximum output length on any single transition. The distance

between \mathcal{T}'_1 and \mathcal{T}'_2 is exactly k if all accepting paths in \mathcal{A}' requires at most k edits and there is at least one accepting path which requires at least k edits. The former is a co-NP language (as shown in [Theorem 6.18](#)), while the latter is in NP. Thus, deciding whether the distance equals k w.r.t. Hamming and transposition is the intersection of a language in NP and a language in co-NP.

□

6.4 Conclusion

It is shown that [distance](#) between two [rational functions](#) w.r.t. common [edit distances](#) is computable by deciding [closeness](#) and [k-closeness](#). We leave open the question of finding the precise computational complexity of the problems in [Tables 4.1](#). The current decision procedure for closeness w.r.t. [conjugacy](#) and [Levenshtein family](#) of distances proceeds through the analysis of the [bounded diameter](#) of [rational expressions](#). One could directly work on automata, but it is not enough to check for the conjugacy of simple cycles, as there can be complex strongly connected components.

Approximate Problems on Transducers

Contents

| | | |
|-------|--|-----|
| 7.1 | Introduction | 135 |
| 7.2 | Twinning Properties | 137 |
| 7.3 | Approximate Determinisation of Rational Functions | 146 |
| 7.3.1 | Approximate Determinisation for Levenshtein Family | 149 |
| 7.4 | Approximate Problems for Rational Relations | 155 |
| 7.4.1 | Approximate Functionality | 156 |
| 7.4.2 | Approximate Determinisation | 157 |
| 7.4.3 | Approximate Uniformisation | 159 |
| 7.5 | Conclusion | 160 |

7.1 Introduction

In this chapter, we consider approximate versions of two fundamental problems in **transducers**, namely **determinisation** and **functionality**. Recall that **sequential functions**, those **defined** by input-deterministic transducers, form a strict subclass of **rational**

¹The contributions presented in this chapter are published in the proceedings of ICALP 2025 under the title “**Approximate Problems for Finite Transducers**”. This is a joint work with Prof. Emmanuel Filiot, Dr. Ismaël Jecker and Dr. Khushraj Madnani.

functions. It turns out that nondeterminism is needed for finite transducers to capture rational functions. A canonical example is the function f_{last} in [Example 2.4](#) which moves the last symbol upfront. We have already seen that the function f_{last} is not sequential, in other words, there is no [sequential transducer](#) recognising f_{last} . The determinisation problem asks: given an arbitrary finite state transducer, does it define a sequential function? or, can the transducer be (input-) determinised? The determinisation problem has been, for instance, extensively considered for weighted automata ([Mohri, 1997](#)), and a long-standing open problem is whether this problem is decidable for $(\mathbb{N}, max, +)$ -automata ([Kirsten and Lombardy, 2009](#)).

Even though the function f_{last} is not exactly sequential, it turns out that f_{last} is *almost* sequential, in the sense that it is *close* to some sequential function, for instance the [identity function](#) id . Two functions or relations are considered *close* if the distance between them is finite. The functions f_{last} and id are close w.r.t. [Levenshtein](#) distance, in the sense that $d_l(f_{last}, id)$ is finite, but they are not close w.r.t. [Hamming](#) distance. This raises a natural and fundamental problem, called *approximate determinisation* problem (for a [metric](#) d): given a finite state transducer recognising a function f , does there exists a sequential function s such that $d(f, s)$ is finite? The approximate determinisation problem has been extensively studied for weighted automata ([Buchsbaum et al., 2001](#); [Aminof et al., 2013](#)) and quantitative automata ([Boker and Henzinger, 2012, 2014](#)), but, nothing was known for transducers to the best of our knowledge.

In this chapter, our main result is the decidability of approximate determinisation of finite state transducers for the [Levenshtein family](#) of distances. For exact determinisation, determinisable finite state transducers are characterized by the so-called [twinning property](#) ([Choffrut, 1977](#); [Béal and Carton, 2002](#); [Béal et al., 2003](#)), a structural pattern with the requirement that the [delay](#) between any two outputs on the same input must not increase when taking synchronized cycles of the transducer. As noticed in [Chapter 6](#), bounded (Levenshtein) [edit distance](#) is closely related to the notion of the [word conjugacy](#). In this chapter, we consider an *approximate version of the twinning property* (abbreviated as ATP), with no constraints on the delay, but instead requires that the output words produced on the synchronised loops are conjugate. It turns out that ATP is not sufficient to characterise approximately determinisable transducers, and an extra property is needed. The *strongly connected twinning property* (abbreviated as STP), is a pattern that requires to hold twinning property within strongly connected components of the finite state transducer.

We show that a transducer \mathcal{T} is approximately determinisable (for Levenshtein family of distances) iff both ATP and STP hold and, if they do, we show how to approximately determinise \mathcal{T} . We also prove that both ATP and STP are decidable properties of a transducer.

Later, approximate versions of problems for [rational relations](#) are considered in this chapter. The *approximate functionality problem* asks whether a given rational relation R is close to a rational function, in the sense that $d(R, f)$ is finite for some rational function f . We prove that the approximate functionality problem is decidable for all the metrics given in [Table 1.1](#). We generalise the approximate determinisation problem to rational relations as well, which amounts to deciding whether a given rational relation R is close to a sequential function f such that $d(R, f)$ is finite. In fact, the characterisation for approximate determinisation of rational functions also holds for rational relations as well.

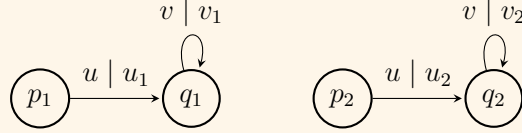
Finally, we consider the *approximate (sequential) uniformisation problem*. In its exact version, this problem asks whether for a given rational relation R , there exists a sequential function f with the same domain, and whose graph is included in R . This problem is closely related to a synthesis problem, but is unfortunately undecidable ([Carayol and Löding, 2011](#); [Filiot et al., 2016](#)). We consider its approximate variant, where instead of requiring that the graph of f is included in R , we require that it is close to some function whose graph is included in R . However, despite this relaxation, we show that the problem is still undecidable.

7.2 Twinning Properties

The class of [sequential functions](#) has been characterised by [transducers](#) satisfying the so-called [twinning property](#). In this section, we recall this property and introduce two of its variants, namely *approximate twinning property* and *strongly connected twinning property*. We also show that these properties on transducers are decidable as well as independent of the representation of the transducers. All these properties are expressed as particular conditions on twinning patterns defined as follows. In this chapter, we use [representation 2](#) for transducers.

Definition 7.1 (Twinning Pattern)

A *twinning pattern* for a transducer $\mathcal{T} = (Q, I, \Delta, F, o)$ over input alphabet A and output alphabet B is a tuple $(p_1, q_1, p_2, q_2, u, v, u_1, v_1, u_2, v_2) \in Q^4 \times (A^*)^2 \times (B^*)^4$ such that the following runs (graphically depicted) exist:



Recall that the *delay* between two words u and v is the pair (u', v') such that $u = \ell u'$ and $v = \ell v'$ where ℓ is the longest common *prefix* of u and v .

Definition 7.2 (Twinning Property (TP))

Let \mathcal{T} be a *trim transducer*. We say that \mathcal{T} satisfies *twinning property* if for each *twinning pattern* $(p_1, q_1, p_2, q_2, u, v, u_1, v_1, u_2, v_2)$ such that p_1, p_2 are initial, $\text{delay}(u_1, u_2) = \text{delay}(u_1 v_1, u_2 v_2)$ holds.

It is well-known that a function *recognised* by a trim transducer \mathcal{T} is sequential iff \mathcal{T} satisfies the twinning property (Choffrut, 1977; Béal et al., 2003). We now define its approximate variant.

Definition 7.3 (Approximate Twinning Property (ATP))

A *trim transducer* \mathcal{T} satisfies *approximate twinning property* if for each *twinning pattern* $(p_1, q_1, p_2, q_2, u, v, u_1, v_1, u_2, v_2)$ where p_1, p_2 are initial, the words v_1 and v_2 are *conjugate*.

Definition 7.4 (Strongly Connected Twinning Property (STP))

A *trim transducer* \mathcal{T} satisfies *strongly connected twinning property* if for each *twinning pattern* $(p_1, q_1, p_2, q_2, u, v, u_1, v_1, u_2, v_2)$ such that $p_1 = p_2$ (not necessarily initial), p_1, q_1, q_2 are in the same strongly connected component of \mathcal{T} , then $\text{delay}(u_1, u_2) = \text{delay}(u_1 v_1, u_2 v_2)$ holds.

Note that a transducer satisfying ATP or STP does not necessarily satisfy TP. For example, the transducer shown in Example 2.4, which recognises the function f_{last} ,

satisfies both ATP and STP but fails to satisfy TP. However, the converse holds as shown below.

Proposition 7.5

Each trim transducer satisfying TP also satisfies ATP and STP.

Proof. Let \mathcal{T} be a trim transducer which satisfies TP. We first show that \mathcal{T} satisfies ATP. Consider a twinning pattern $(p_1, q_1, p_2, q_2, u, v, u_1, v_1, u_2, v_2)$ of \mathcal{T} such that p_1, p_2 are initial states. Then, $\text{delay}(u_1, u_2) = \text{delay}(u_1 v_1, u_2 v_2) = \text{delay}(u_1 v_1^i, v_1 v_2^i)$ for all $i \geq 0$, which implies that $|v_1| = |v_2|$. Therefore, for all $n \geq 0$, there exists i such that v_1^i and v_2^i have a common factor of length at least n . By Fine and Wilf's theorem (see Theorem 3.11), it implies that v_1 and v_2 have conjugate primitive roots, and since $|v_1| = |v_2|$, we get that v_1 and v_2 are conjugate using Proposition 5.10.

Now, we show that \mathcal{T} satisfies STP. Let $(p, q_1, p, q_2, u, v, u_1, v_1, u_2, v_2)$ be some twinning pattern where p, q_1, q_2 are in the same SCC. Since \mathcal{T} is trim, p is accessible from the initial state q_0 , by a run $q_0 \xrightarrow{\alpha|\beta}_{\mathcal{T}} p$. Note that $(q_0, q_1, q_0, q_2, \alpha u, v, \beta u_1, v_1, \beta u_2, v_2)$ is also a twinning pattern. Hence by the twinning property,

$$\text{delay}(\beta u_1, \beta u_2) = \text{delay}(\beta u_1 v_1, \beta u_2 v_2) \quad (7.1)$$

Since $\text{delay}(\beta u_1, \beta u_2) = \text{delay}(u_1, u_2)$ and $\text{delay}(\beta u_1 v_1, \beta u_2 v_2) = \text{delay}(u_1 v_1, u_2 v_2)$, by substituting it in Equation (7.1), we get $\text{delay}(u_1, u_2) = \text{delay}(u_1 v_1, u_2 v_2)$, and hence satisfies STP. Note that we didn't use the assumption that p, q_1, q_2 are in the same SCC. \square

The following lemma states that ATP and STP are preserved between transducers up to finite edit distance, and so in particular between equivalent transducers. This shows that these properties do not depend on the representation of the transductions (or relations), not even on the representation of close transductions.

Lemma 7.6

Let \mathcal{T} and \mathcal{S} be two **trim transducers** such that $\text{dom}(\mathcal{T}) = \text{dom}(\mathcal{S})$ and there exists a metric d in **Table 1.1** and a constant $C \in \mathbb{N}$ for which

$$\forall u \in \text{dom}(\mathcal{T}), v \in \mathcal{T}(u), v' \in \mathcal{S}(u), d(v, v') \leq C.$$

Then for $P \in \{\text{ATP}, \text{STP}\}$, \mathcal{S} satisfies P iff \mathcal{T} satisfies P .

Proof. Let us suppose that \mathcal{S} satisfies $P \in \{\text{ATP}, \text{STP}\}$, and show that so does \mathcal{T} . The converse is symmetric. We first show this for **ATP**, and later for **STP**.

For ATP. Let $(p_1, p_2, q_1, q_2, u, v, u_1, u_2, v_1, v_2)$ be an instance of the twinning pattern for \mathcal{T} with p_1 and q_1 initial. Since \mathcal{T} is trim, there exist words w, w', w_1, w_2 and two accepting states p_f, q_f such that:

$$\begin{array}{ccccccc} p_1 & \xrightarrow{u|u_1} \mathcal{T} & p_2 & \xrightarrow{v|v_1} \mathcal{T} & p_2 & \xrightarrow{w|w_1} \mathcal{T} & p_f \\ q_1 & \xrightarrow{u|u_2} \mathcal{T} & q_2 & \xrightarrow{v|v_2} \mathcal{T} & q_2 & \xrightarrow{w'|w_2} \mathcal{T} & q_f \end{array}$$

Since $\text{dom}(\mathcal{T}) = \text{dom}(\mathcal{S})$, for all $i \geq 1$, $uv^i w, uv^i w' \in \text{dom}(\mathcal{T}) = \text{dom}(\mathcal{S})$. Iterating the loop of \mathcal{T} on v sufficiently many times causes \mathcal{S} to also loop on some **power** of v . Formally, there exist $\ell, m, n \in \mathbb{N}$, initial states p'_1, q'_1 and states p'_2, q'_2, p'_f, q'_f of \mathcal{S} , and words $u'_1, u'_2, v'_1, v'_2, w'_1, w'_2$ such that:

$$\begin{array}{ccccccc} p_1 & \xrightarrow{uv^\ell|u_1v_1^\ell} \mathcal{T} & p_2 & \xrightarrow{v^m|v_1^m} \mathcal{T} & p_2 & \xrightarrow{v^n w|v_1^n w_1} \mathcal{T} & p_f \\ q_1 & \xrightarrow{uv^\ell|u_2v_2^\ell} \mathcal{T} & q_2 & \xrightarrow{v^m|v_2^m} \mathcal{T} & q_2 & \xrightarrow{v^n w'|v_2^n w_2} \mathcal{T} & q_f \\ p'_1 & \xrightarrow{uv^\ell|u'_1} \mathcal{S} & p'_2 & \xrightarrow{v^m|v'_1} \mathcal{S} & p'_2 & \xrightarrow{v^n w|w'_1} \mathcal{S} & p'_f \\ q'_1 & \xrightarrow{uv^\ell|u'_2} \mathcal{S} & q'_2 & \xrightarrow{v^m|v'_2} \mathcal{S} & q'_2 & \xrightarrow{v^n w'|w'_2} \mathcal{S} & q'_f \end{array}$$

Since $d(\mathcal{T}(x), \mathcal{S}(x)) \leq C$ for all $x \in \text{dom}(\mathcal{T})$. Then in particular for all $k \geq 0$:

$$d(u_1 v_1^{\ell+km+n} w_1, u'_1 v_1'^k w'_1) \leq C \text{ and } d(u_2 v_2^{\ell+km+n} w_2, u'_2 v_2'^k w'_2) \leq C.$$

From the above inequalities, using **Lemma 6.3**, we get that ρ_{v_i} and $\rho_{v'_i}$, the primitive roots of v_i and v'_i respectively, are conjugate and $|v_i^m| = |v'_i|$ for $i \in [2]$. Since \mathcal{S} satisfies

the ATP, we also have that v'_1 and v'_2 are conjugate, and hence their primitive roots are conjugate and $|v'_1| = |v'_2|$. By [transitivity](#) of the conjugacy relation, we get that ρ_{v_1} and ρ_{v_2} are conjugate. Further, $|v_1| = |v_2|$ since $|\rho_{v_1}| = |\rho_{v_2}|$ and $|v_1^m| = |v'_1| = |v'_2| = |v_2^m|$. Therefore, v_1 and v_2 are conjugate, and thus the ATP holds for \mathcal{T} as well.

For STP. We suppose that \mathcal{S} satisfies the STP, and show that \mathcal{T} satisfies it too. Let us pick a twinning pattern $(p, q_1, p, q_2, u, v, u_1, v_1, u_2, v_2)$ in \mathcal{T} such that p, q_1, q_2 are in the same strongly connected component, as in the definition of the STP ([Definition 7.4](#)):

$$\begin{aligned} \rho_1 : & q_I \xrightarrow{x|x_0}_{\mathcal{T}} p \xrightarrow{u|u_1}_{\mathcal{T}} q_1 \xrightarrow{v|v_1}_{\mathcal{T}} q_1 \xrightarrow{w^+|w_1}_{\mathcal{T}} p \xrightarrow{z|z_0}_{\mathcal{T}} q_F, \\ \rho_2 : & q_I \xrightarrow{x|x_0}_{\mathcal{T}} p \xrightarrow{u|u_2}_{\mathcal{T}} q_2 \xrightarrow{v|v_2}_{\mathcal{T}} q_2 \xrightarrow{w^-|w_2}_{\mathcal{T}} p \xrightarrow{z|z_0}_{\mathcal{T}} q_F. \end{aligned}$$

Here the runs with inputs x and z are witnesses of the fact that \mathcal{T} is trim and the runs with inputs w^+ and w^- are witnesses of the fact that p, q_1, q_2 are in the same strongly connected component. Similar to the proof of ATP, we use these runs to build an instance of STP in \mathcal{S} . To that end, let N denote the number of states of \mathcal{S} . We study the run of \mathcal{S} on the word :

$$y = xy_1y_2 \dots y_{2N}z, \text{ where } y_i = (uv^{N^2}w^+)^i uv^{N^2}w^- \text{ for every } 1 \leq i \leq 2N.$$

Remark that $y \in \text{dom}(\mathcal{T})$, as a run over y can be obtained by pumping and combining the subruns of ρ_1 and ρ_2 . Therefore, as $\text{dom}(\mathcal{T}) = \text{dom}(\mathcal{S})$, there is also a run of \mathcal{S} on y , and since \mathcal{S} has N states this run will visit some state r three times between the y_i . Formally, there exist $1 \leq i < j < k \leq 2N$ such that, if we let $\bar{x} = xy_1y_2 \dots y_{i-1}$ and $\bar{z} = y_ky_{k+1} \dots y_{2N}z$, we get:

$$r_I \xrightarrow{\bar{x}|\bar{x}_0}_{\mathcal{S}} r \xrightarrow{y_i \dots y_{j-1}|y'}_{\mathcal{S}} r \xrightarrow{y_j \dots y_{k-1}|y''}_{\mathcal{S}} r \xrightarrow{\bar{z}|\bar{z}_0}_{\mathcal{S}} r_F.$$

Now remark that the input words labeling both loops on r start with the [prefix](#) $(uv^{N^2}w^+)^i uv^{N^2}$, which is followed by w^- in the first loop and w^+ in the second. Since \mathcal{S} has N states, the last block of v^{N^2} occurring in this common prefix will visit a synchronised subloop in both loops. Formally, there is a decomposition $\bar{u}\bar{u}'\bar{v}\bar{w}^+$ of

$y_i \dots y_{j-1}$ and a decomposition $\bar{u}\bar{u}'\bar{v}\bar{w}^-$ of $y_j \dots y_{k-1}$ satisfying:

$$\begin{aligned} \bar{u} &= (uv^{N^2}w^+)^i, & \bar{u}' &= uv^a, & \bar{v} &= v^b, & \bar{w}^+ &= v^c w^- y_{i+1} \dots y_{j-1}, \\ \bar{w}^- &= v^c w^+ (uv^{N^2}w^+)^{j-i-1} uv^{N^2} w^- y_{j+1} \dots y_{k-1}, \\ \bar{\rho}_1 : & s_I \xrightarrow{\bar{x}|\bar{x}_0}_S r \xrightarrow{\bar{u}|\bar{u}_1}_S r_1 \xrightarrow{\bar{u}'|\bar{u}'_1}_S s_1 \xrightarrow{\bar{v}|\bar{v}_1}_S s_1 \xrightarrow{\bar{w}^+|\bar{w}_1}_S r \xrightarrow{\bar{z}|\bar{z}_0}_S s_F, \\ \bar{\rho}_2 : & s_I \xrightarrow{\bar{x}|\bar{x}_0}_S r \xrightarrow{\bar{u}|\bar{u}_2}_S r_2 \xrightarrow{\bar{u}'|\bar{u}'_2}_S s_2 \xrightarrow{\bar{v}|\bar{v}_2}_S s_2 \xrightarrow{\bar{w}^-|\bar{w}_2}_S r \xrightarrow{\bar{z}|\bar{z}_0}_S s_F. \end{aligned}$$

Since \mathcal{S} satisfies the STP by hypothesis, $\text{delay}(\bar{u}_1\bar{u}'_1, \bar{u}_2\bar{u}'_2) = \text{delay}(\bar{u}_1\bar{u}'_1\bar{v}_1, \bar{u}_2\bar{u}'_2\bar{v}_2)$. We now combine this with the fact that the distance between the outputs of \mathcal{T} and \mathcal{S} is bounded by C to show that $\text{delay}(u_1, u_2) = \text{delay}(u_1v_1, u_2v_2)$, which concludes the proof of the lemma. First, let us pump the subruns of ρ_1 and ρ_2 to match the inputs of $\bar{\rho}_1$ and $\bar{\rho}_2$:

$$\begin{aligned} \rho'_1 : & q_I \xrightarrow{\bar{x}|\bar{x}'_0}_T p \xrightarrow{\bar{u}|\bar{u}_0}_T p \xrightarrow{\bar{u}'|\bar{u}_1v_1^a}_T q_1 \xrightarrow{\bar{v}|\bar{v}_1^b}_T q_1 \xrightarrow{\bar{w}^+|\bar{w}'_1}_T p \xrightarrow{\bar{z}|\bar{z}'_0}_T q_F, \\ \rho'_2 : & q_I \xrightarrow{\bar{x}|\bar{x}'_0}_T p \xrightarrow{\bar{u}|\bar{u}_0}_T p \xrightarrow{\bar{u}'|\bar{u}_2v_2^a}_T q_2 \xrightarrow{\bar{v}|\bar{v}_2^b}_T q_2 \xrightarrow{\bar{w}^-|\bar{w}'_2}_T p \xrightarrow{\bar{z}|\bar{z}'_0}_T q_F. \end{aligned}$$

We introduce a standard property of the delay, proved for instance by [Filiot et al. \(2020\)](#):

Claim 7.7. *For every words s_1, s_2, t_1, t_2 we have that $\text{delay}(s_1, s_2) = \text{delay}(s_1t_1, s_2t_2)$ if and only if either $t_1 = t_2 = \varepsilon$, or $|t_1| = |t_2|$ and there is no mismatch between $s_1t_1^m$ and $s_2t_2^m$ for all $m \in \mathbb{N}$.*

Therefore, in order to show that $\text{delay}(u_1, u_2) = \text{delay}(u_1v_1, u_2v_2)$, we now suppose that $v_1 \neq \varepsilon$ or $v_2 \neq \varepsilon$, and we show that $|v_1| = |v_2|$ and there is no mismatch between $u_1v_1^m$ and $u_2v_2^m$ for all $m \in \mathbb{N}$.

Same output length. We first show that $|v_1| = |v_2|$. Observe that for every $\lambda \in \mathbb{N}$ we have :

$$\begin{aligned} \mathcal{S}(\bar{x}\bar{u}\bar{u}v^{a+\lambda \cdot b}\bar{w}^+\bar{z}) &= \bar{x}_0\bar{u}_1\bar{u}'_1\bar{v}_1^\lambda\bar{w}_1\bar{z}_0, & \mathcal{S}(\bar{x}\bar{u}\bar{u}v^{a+\lambda \cdot b}\bar{w}^-\bar{z}) &= \bar{x}_0\bar{u}_2\bar{u}'_2\bar{v}_2^\lambda\bar{w}_2\bar{z}_0, \\ \mathcal{T}(\bar{x}\bar{u}\bar{u}v^{a+\lambda \cdot b}\bar{w}^+\bar{z}) &= x'_0u_0u_1v_1^{a+\lambda \cdot b}w'_1z'_0, & \mathcal{T}(\bar{x}\bar{u}\bar{u}v^{a+\lambda \cdot b}\bar{w}^-\bar{z}) &= x'_0u_0u_2v_2^{a+\lambda \cdot b}w'_2z'_0. \end{aligned}$$

By the supposition in the statement of the Lemma, $d(\bar{x}_0\bar{u}_1\bar{u}'_1\bar{v}_1^\lambda\bar{w}_1\bar{z}_0, x'_0u_0u_1v_1^{a+\lambda \cdot b}w'_1z'_0) \leq C$ and $d(\bar{x}_0\bar{u}_2\bar{u}'_2\bar{v}_2^\lambda\bar{w}_2\bar{z}_0, x'_0u_0u_2v_2^{a+\lambda \cdot b}w'_2z'_0) \leq C$. This implies that $|v_1|^b = \bar{v}_1$ and $|v_2|^b = \bar{v}_2$, as otherwise choosing λ large enough yields outputs with a length difference

too large to be fixed with at most C number of edits. Hence, it suffices to show $|\bar{v}_1| = |\bar{v}_2|$. This is implied by the fact $\text{delay}(\bar{u}_1\bar{u}'_1, \bar{u}_2\bar{u}'_2) = \text{delay}(\bar{u}_1\bar{u}'_1\bar{v}_1, \bar{u}_2\bar{u}'_2\bar{v}_2)$ along with the [Claim 7.7](#).

No mismatch. As we have shown that $|v_1| = |v_2|$, let us now pick $m \in \mathbb{N}$ and show that there is no mismatch between $u_1v_1^m$ and $u_2v_2^m$ to conclude the proof via [Claim 7.7](#). Remark that the words $u_1v_1^m$ and $u_2v_2^m$ appear as subwords of the outputs of the runs ρ'_1 and ρ'_2 . We now derive three key equations on output words ([Equation \(7.2\)](#), [Equation \(7.3\)](#) and [Equation \(7.4\)](#)) by pumping and assembling parts of ρ'_1 , ρ'_2 , $\bar{\rho}_1$ and $\bar{\rho}_2$. We then combine these equations to get the desired result. We begin by introducing some notation. Let $M \in \mathbb{N}$ satisfying

$$M > \frac{m \cdot |v_1| + C + \max(|\bar{x}_0|, |x'_0|)}{b \cdot |v_1|}.$$

Moreover, we consider the following output words :

$$\begin{aligned} \alpha_1 &= u_0u_1v_1^{a+Mb}, & \alpha_2 &= u_0u_2v_2^{a+Mb}, & \beta_1 &= w'_1, & \beta_2 &= w'_2, \\ \bar{\alpha}_1 &= \bar{u}_1\bar{u}'_1\bar{v}_1^M, & \bar{\alpha}_2 &= \bar{u}_2\bar{u}'_2\bar{v}_2^M, & \bar{\beta}_1 &= \bar{w}_1, & \bar{\beta}_2 &= \bar{w}_2. \end{aligned}$$

First, remark that for every $\lambda \in \mathbb{N}$ we have :

$$\begin{aligned} \mathcal{S}(\bar{x}(\bar{u}\bar{u}'\bar{v}^M\bar{w}^+)^{\lambda}\bar{z}) &= \bar{x}_0(\bar{\alpha}_1\bar{\beta}_1)^{\lambda}\bar{z}_0, & \mathcal{S}(\bar{x}(\bar{u}\bar{u}'\bar{v}^M\bar{w}^-)^{\lambda}\bar{z}) &= \bar{x}_0(\bar{\alpha}_2\bar{\beta}_2)^{\lambda}\bar{z}_0, \\ \mathcal{T}(\bar{x}(\bar{u}\bar{u}'\bar{v}^M\bar{w}^+)^{\lambda}\bar{z}) &= x'_0(\alpha_1\beta_1)^{\lambda}z'_0, & \mathcal{T}(\bar{x}(\bar{u}\bar{u}'\bar{v}^M\bar{w}^-)^{\lambda}\bar{z}) &= x'_0(\alpha_2\beta_2)^{\lambda}z'_0. \end{aligned}$$

Since the edit distance between the outputs of \mathcal{T} and \mathcal{S} is bounded by C , this yields :

$$|\alpha_1\beta_1| = |\bar{\alpha}_1\bar{\beta}_1| \text{ and } |\alpha_2\beta_2| = |\bar{\alpha}_2\bar{\beta}_2|. \quad (7.2)$$

Second, for every $\lambda \in \mathbb{N}$ we have :

$$\begin{aligned} \mathcal{S}(\bar{x}(\bar{u}\bar{u}'\bar{v}^M\bar{w}^+\bar{u}\bar{u}'\bar{v}^M\bar{w}^+\bar{u}\bar{u}'\bar{v}^M\bar{w}^-)^{\lambda}\bar{z}) &= \bar{x}_0(\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2\bar{\beta}_2)^{\lambda}\bar{z}_0, \\ \mathcal{T}(\bar{x}(\bar{u}\bar{u}'\bar{v}^M\bar{w}^+\bar{u}\bar{u}'\bar{v}^M\bar{w}^+\bar{u}\bar{u}'\bar{v}^M\bar{w}^-)^{\lambda}\bar{z}) &= x'_0(\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2)^{\lambda}z'_0. \end{aligned}$$

If we set $\lambda = C$, the hypothesis in the statement of the Lemma yields :

$$d(x'_0(\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2)^C z'_0, \bar{x}_0(\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2\bar{\beta}_2)^C \bar{z}_0) < C.$$

Observe that at least one copy of the subword $\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2$ is preserved (i.e., no internal deletion or insertion) while editing $x'_0(\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2)^\lambda z'_0$ into $\bar{x}_0(\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2\bar{\beta}_2)^\lambda \bar{z}_0$ via $C - 1$ edits. As this copy of $\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2$ can be shifted by at most $C - 1$ letters, and as it is originally shifted by $\max(|x'_0|, |\bar{x}_0|)$ letters with respect to its matching copy of $\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2\bar{\beta}_2$ in $\bar{x}_0(\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2\bar{\beta}_2)^C \bar{z}_0$, there exists $\delta \in \mathbb{N}$ satisfying $|\delta| < C + \max(|\bar{x}_0|, |x'_0|)$ such that :

$$\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2[i] = \bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2\bar{\beta}_2[i + \delta] \quad (7.3)$$

$$\text{for all } \max(0, \delta) < i \leq \min(|\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2|, |\alpha_1\beta_1\alpha_1\beta_1\alpha_2\beta_2| - \delta).$$

Finally, as $\text{delay}(\bar{u}_1\bar{u}'_1, \bar{u}_2\bar{u}'_2) = \text{delay}(\bar{u}_1\bar{u}'_1\bar{v}_1, \bar{u}_2\bar{u}'_2\bar{v}_2)$, [Claim 7.7](#) yields that

$$\bar{\alpha}_1[i] = \bar{\alpha}_2[i] \text{ for every } 1 \leq i \leq \min(|\bar{\alpha}_1|, |\bar{\alpha}_2|). \quad (7.4)$$

We now combine all equations. For every $\max(0, \delta) < i \leq \min(|\alpha_1|, |\alpha_2| - \delta)$ we get :

$$\begin{aligned} \alpha_1[i] &= \bar{\alpha}_1[i + \delta] && \text{by (7.3),} \\ &= \bar{\alpha}_2[i + \delta] && \text{by (7.4),} \\ &= (\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2)[|\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1| + i + \delta] \\ &= (\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_2)[|\alpha_1\beta_1\alpha_1\beta_1| + i + \delta] && \text{by (7.2),} \\ &= (\alpha_1\beta_1\alpha_1\beta_1\alpha_2)[|\alpha_1\beta_1\alpha_1\beta_1| + i] && \text{by (7.3),} \\ &= \alpha_2[i]. \end{aligned}$$

If $\delta < 0$ we still need to address some values of i . For every $1 \leq i \leq -\delta$ we get :

$$\begin{aligned}
 \alpha_1[i] &= (\alpha_1\beta_1\alpha_1)[|\alpha_1\beta_1| + i] \\
 &= (\alpha_1\beta_1\alpha_1)[|\bar{\alpha}_1\bar{\beta}_1| + i] && \text{by (7.2),} \\
 &= (\bar{\alpha}_1\bar{\beta}_1)[|\bar{\alpha}_1\bar{\beta}_1| + i + \delta] && \text{by (7.3) since } i + \delta < 1, \\
 &= (\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1)[|\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1| + i + \delta] \\
 &= (\bar{\alpha}_1\bar{\beta}_1\bar{\alpha}_1\bar{\beta}_1)[|\alpha_1\beta_1\alpha_1\beta_1| + i + \delta] && \text{by (7.2),} \\
 &= (\alpha_1\beta_1\alpha_1\beta_1\alpha_2)[|\alpha_1\beta_1\alpha_1\beta_1| + i] && \text{by (7.3) since } i \geq 1, \\
 &= \alpha_2[i].
 \end{aligned}$$

These last two series of equations ensure that α_1 and α_2 have a large common prefix :

$$\alpha_1[i] = \alpha_2[i] \text{ for every } 1 \leq i \leq \min(|\alpha_1|, |\alpha_2| - \delta). \quad (7.5)$$

Then we can conclude by noticing that, thanks to the choice of

$$M > \frac{m \cdot |v_1| + C + \max(|\bar{x}_0|, |x'_0|)}{b \cdot |v_1|} > \frac{m}{b} + \frac{|\delta|}{b \cdot |v_1|},$$

the words $\alpha_1 = u_0u_1v_1^{a+Mb}$ and $\alpha_2 = u_0u_2v_2^{a+Mb}$ can be written as $u_0u_1v_1^mv_1^\mu$, respectively $u_0u_2v_2^mv_2^\mu$, for some $\mu \in \mathbb{N}$ such that $|v_1^\mu| = |v_2^\mu| > |\delta|$. Therefore, as desired, Equation (7.5) implies that there is no mismatch between $u_1v_1^m$ and $u_2v_2^m$. \square

Note that the above lemma does not necessarily hold for TP. For instance, although $d_l(f_{last}, id) < \infty$ and the transducer defining the **identity function** satisfies TP, the transducer defining f_{last} does not.

We now show that checking each of the variants of the twinning property is decidable.

Lemma 7.8

It is decidable whether a **transducer** satisfies **ATP** and **STP**.

Proof. Let \mathcal{T} be a transducer recognising relation R .

Deciding ATP. Let \mathcal{T}^2 be the [cartesian product](#) of \mathcal{T} by itself. Verifying whether \mathcal{T} satisfies ATP reduces to checking that every loop in \mathcal{T}^2 produces a pair of conjugate output words. For each state $(p, q) \in \mathcal{T}^2$, we define a rational relation $R_{(p,q)}$ consisting of pairs of output words produced by the loops in \mathcal{T}^2 rooted at (p, q) . The transducer defining $R_{(p,q)}$ is obtained from \mathcal{T}^2 by disregarding the inputs and setting both the initial and final state to be (p, q) . Checking whether every pair of words in a given rational relation is conjugate is decidable (See [Theorem 5.2](#)), which entails decidability of ATP.

Deciding STP. The twinning property is decidable in polynomial time ([Béal et al., 2003](#)). To decide STP, we first decompose the transducer into SCCs (in polynomial time). Then, for each SCC and each state p in \mathcal{T} , the SCC is seen as a transducer with initial state p on which we check the twinning property. \square

7.3 Approximate Determinisation of Rational Functions

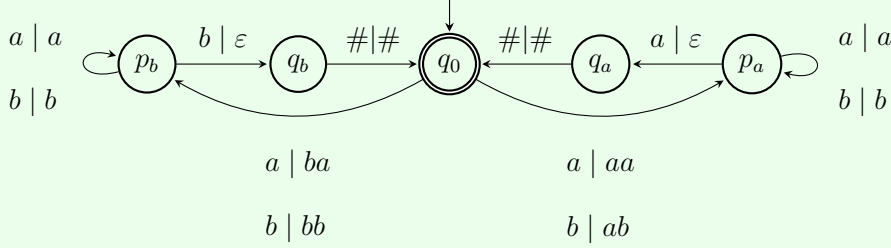
In this section, we define *approximately determinisable functions* w.r.t. a [metric](#) d and give decidable properties on transducers to characterise them for the [Levenshtein family](#) of distances.

Definition 7.9 (Approximate Determinisation)

A [rational function](#) $f : A^* \rightarrow B^*$ is *approximately determinisable* w.r.t. a [metric](#) d if there exists a [sequential function](#) $g : A^* \rightarrow B^*$ such that $d(f, g) < \infty$. In this case, we also say that g approximately determinises f w.r.t. d .

Example 7.10

The function f_{last} is approximately determinisable w.r.t. [Levenshtein](#) distance (See [Figure 7.1](#)). The function f_{last}^* (depicted below) that maps $u_1\# \cdots u_n\#$, for $n \geq 1$, to $f_{last}(u_1)\# \cdots f_{last}(u_n)\#$ for some separator $\#$ is *not* approximately determinisable w.r.t Levenshtein distance.



The [approximate determinisation](#) problem asks whether a rational function given as a functional transducer is approximately determinisable. We prove the following.

Theorem 7.11

The [approximate determinisation](#) problem for [rational functions](#) w.r.t. [Levenshtein family](#) of distances (d_l, d_{lcs}, d_{dl}) is decidable.

To prove the theorem, we show that [ATP](#) and [STP](#) characterise rational functions that can be approximately determinised w.r.t Levenshtein family ([Lemma 7.18](#)). [Theorem 7.11](#) then follows directly, as these three properties are decidable ([Lemma 7.8](#)).

We now outline the proof strategy. The full proof for the Levenshtein family is presented in [Section 7.3.1](#). We show with [Proposition 7.14](#) that ATP and STP are necessary conditions for approximate determinisation with respect to the Levenshtein family, a consequence of [Lemma 7.6](#). The main challenge lies in proving that these properties are sufficient. To prove it, we first show that ATP alone suffices for certain subclasses of [functional transducers](#), namely multi-sequential functions and series-sequential functions.

Definition 7.12 (Multi-sequential Functions)

A [multi-sequential](#) function is a rational function [recognised](#) by finite disjoint unions of [sequential transducers](#). For instance, f_{last} is multi-sequential, as $f_{last} = f_a \cup f_b$ where $f_a : ua \rightarrow au$ and $f_b : ub \rightarrow bu$ are [sequential functions](#).

Multi-sequential functions have been studied by Choffrut and Schützenberger (1986) where checking if a rational function is multi-sequential is shown to be decidable. The symmetric counterpart of multi-sequential is the class of *series-sequential functions* defined below.

Definition 7.13 (Series-sequential Functions)

A rational function is *series-sequential* if it is recognised by a finite disjoint union of sequential transducers $\mathcal{D}_1, \dots, \mathcal{D}_k$ where additionally there is a single transition from a (not necessarily final) state of \mathcal{D}_i to the initial state of the next transducer \mathcal{D}_{i+1} for every $1 \leq i < k$. The initial state of the entire construction coincides with that of \mathcal{D}_1 . Intuitively, this structure corresponds to the concatenation of the sequential transducers $\mathcal{D}_1, \dots, \mathcal{D}_k$, and we will denote it by $\mathcal{D}_1 \cdots \mathcal{D}_k$, and called a *series-sequential transducer*.

We show that ATP alone suffices for the approximate determinisation of multi-sequential (Lemma 7.16) and unambiguous series-sequential (Lemma 7.17) functions. However, for rational functions in general, ATP does not suffice. For example, the transducer for f_{last}^* depicted in Example 7.10 satisfies ATP but is not approximately determinisable. To extend this result to all rational functions, we incorporate STP. Given a functional transducer \mathcal{T} satisfying STP, we transform each strongly connected component of \mathcal{T} into a sequential transducer, effectively decomposing \mathcal{T} into a finite union of concatenations of sequential transducers. We then leverage our results for series-sequential and multi-sequential functions to approximate this structure with a sequential function (Lemma 7.18).

Figure 7.1 illustrates the main construction technique used in these proofs: Starting with a transducer \mathcal{T} that we aim to approximate, we construct a sequential transducer \mathcal{D} as follows. We apply the powerset construction to \mathcal{T} , introducing a distinguished state (marked by a \bullet in the figure) in each subset. The output is determined by the distinguished state's production. If the distinguished state reaches a point where it has no continuation, we simply transition to another distinguished state. We show that ATP, combined with a carefully chosen priority scheme for selecting distinguished states, ensures *bounded distance* for Levenshtein family.

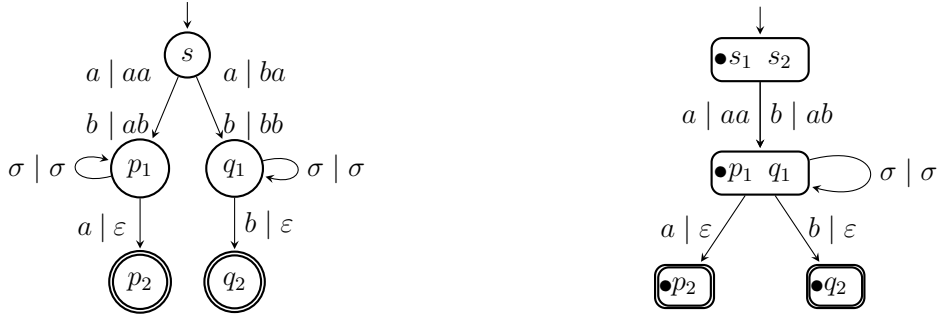


Fig. 7.1. A nondeterministic transducer \mathcal{T} (left), along with a sequential approximations \mathcal{D} (right) w.r.t. the Levenshtein family of distance.

7.3.1 Approximate Determinisation for Levenshtein Family

We give a decidable characterisation of approximately determinisable rational functions w.r.t. Levenshtein family of distances — Levenshtein (d_l), Longest common subsequence (d_{lcs}) and Damerau-Levenshtein (d_{dl}). They are all equivalent up to boundedness, i.e., for any two rational functions f, g , $d_{lcs}(f, g) < \infty \iff d_l(f, g) < \infty \iff d_{dl}(f, g) < \infty$. We show that a rational function is approximately determinisable w.r.t. Levenshtein family if and only if the transducer that defines the function satisfies both ATP and STP. One direction is a consequence of Lemma 7.6 as follows.

Proposition 7.14

If a rational function given by a trim transducer \mathcal{T} is approximately determinisable w.r.t. a metric $d \in \{d_l, d_{lcs}, d_{dl}\}$ then \mathcal{T} satisfies both ATP and STP.

Proof. Given that the rational function given by \mathcal{T} is approximately determinisable, i.e., there exists a sequential function given by a deterministic transducer \mathcal{D} such that $d(\mathcal{T}, \mathcal{D}) < \infty$. Since \mathcal{D} is a sequential transducer, it satisfies the twinning property, and consequently, \mathcal{D} also satisfies ATP and STP by Proposition 7.5. Since $d(\mathcal{T}, \mathcal{D}) < \infty$, we can conclude that \mathcal{T} also satisfies ATP as well as STP by applying Lemma 7.6. \square

Towards proving the other direction, we prove the following lemma, which provides a bound on the distance between the output words produced by distinct runs of a transducer on the same prefix of an input word using Lemma 6.9.

Lemma 7.15

Let \mathcal{T} be a **trim transducer** satisfying the **ATP**. Then, there exists a constant $N_{\mathcal{T}} \in \mathbb{N}$ such that for any two output words $v, v' \in B^*$ produced via two distinct **runs** of \mathcal{T} from an initial state on the same **prefix** of an input word, $d(v, v') \leq N_{\mathcal{T}}$ for $d \in \{d_l, d_{lcs}, d_{dl}\}$.

Proof. Let \mathcal{T}^2 be the **cartesian product** of \mathcal{T} by itself. By designating all states of \mathcal{T}^2 as final, and disregarding the input word, we obtain the **output product** transducer that defines the relation R_o , consisting of all pairs of output words produced by distinct runs of \mathcal{T} on the same prefix of an input word. Since \mathcal{T} satisfies **ATP**, every pair of output words produced by loops in \mathcal{T}^2 on any input are **conjugate**. Consequently, since R_o is obtained by ignoring the input word from \mathcal{T}^2 , it satisfies the condition in **Lemma 6.9**. Hence, the diameter of R_o w.r.t. Levenshtein family of distances is bounded.

Let $\text{dia}_d(R_o) \leq k$. By the definition of **diameter**, it follows that $d(u, v) \leq k$ for all $(u, v) \in R_o$. As a result, the distance between any two output words produced by distinct runs of \mathcal{T} on the same word is at most k . Setting $N_{\mathcal{T}} = k$ completes the proof. \square

For subclasses of rational functions, namely multi-sequential and series-sequential functions, we show that **ATP** is a sufficient condition for approximate determinisation.

Lemma 7.16

A **multi-sequential function** given by a **trim transducer** \mathcal{T} is **approximately determinisable** w.r.t. a **metric** $d \in \{d_l, d_{lcs}, d_{dl}\}$ iff \mathcal{T} satisfies the **ATP**.

Proof. (\rightarrow) is direct by **Proposition 7.14**.

(\leftarrow) Since the transducer \mathcal{T} is multi-sequential, it is equivalent to some finite union of sequential transducers $\mathcal{U} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_k$ for some $k \in \mathbb{N}$ where $\mathcal{D}_i = (Q_i, s_i, \delta_i, F_i, o_i)$ is a sequential trim transducer for $i \in [k]$. By **Lemma 7.6** and since \mathcal{T} satisfies **ATP**, the transducer \mathcal{U} also satisfies **ATP**. We construct a sequential transducer \mathcal{D} that approximately determinises \mathcal{T} , which intuitively is simply the cartesian product of $\mathcal{D}_1, \dots, \mathcal{D}_k$ which on each transition produces the output of the smallest index transducer \mathcal{D}_i for which that transition is defined. Let $\mathcal{D} = (Q, s, \delta, F, o)$ where

1. The set of states $Q = Q'_1 \times Q'_2 \times \dots \times Q'_k$ is the cartesian product of the state set Q'_i for

$i \in [k]$ such that $Q'_i = Q_i \cup \{d\}$ where d represents a dead state. The initial state is $s = (s_1, s_2, \dots, s_k)$, and the set of final states F is $\{(p_1, p_2, \dots, p_k) \mid \exists i p_i \in F_i\}$.

2. The function $\delta : Q \times A \rightarrow Q \times B^*$ is defined as: $\delta((p_1, p_2, \dots, p_k), a) = ((q_1, q_2, \dots, q_k), x)$ where for each $i \in [k]$, either $\delta_i(p_i, a) = (q_i, x_i)$ or, if $p_i = d$ or $\delta_i(p_i, a)$ is undefined, then $q_i = d$. The output x is set to x_j , where $j \in [k]$ is the smallest index for which the transition $\delta_j(p_j, a)$ is defined.
3. The output function $o : F \rightarrow B^*$ is defined as $o((p_1, p_2, \dots, p_k)) = o_i(p_i)$ where $i \in [k]$ is the smallest index such that $p_i \in F_i$.

We show that $d(\mathcal{D}, \mathcal{T}) < \infty$ w.r.t. Levenshtein family. From the construction, it is clear that \mathcal{D} and \mathcal{T} have the same **domain**. Consider an input word $u \in \text{dom}(\mathcal{T})$. Let $i \in [k]$ be the smallest index such that $u \in \text{dom}(\mathcal{D}_i)$, with output word, say v . The transducer \mathcal{D} on input u produces output, say v' , by concatenating output on each transition over the input word produced by the smallest index transducer.

Thus, as shown in Figure 7.2, v' can be decomposed into $v_{i_1} v_{i_2} \dots v_{i_n}$ where $i_1 < i_2 < \dots < i_n = i$ and for each $i_j \in [i]$, v_{i_j} is the output produced by \mathcal{D}_{i_j} along the partial run of u . For each v_{i_j} , let v'_{i_j} denote the prefix of the output produced by \mathcal{D}_{i_j} upto v_{i_j} . The output produced by \mathcal{T} on u via \mathcal{D}_i is $v = v'_i v_i = v'_{i_n} v_{i_n}$.

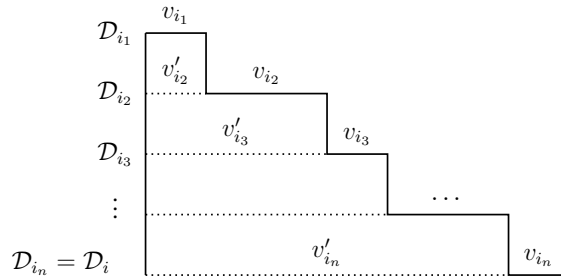


Fig. 7.2. Decomposition of an output produced by \mathcal{D}

Observe that $v'_{i_j} v_{i_j}$ and $v'_{i_{j+1}}$ (for $i_1 < i_j < i_n$) is the output produced by \mathcal{D}_{i_j} and $\mathcal{D}_{i_{j+1}}$ on the same prefix of input u . By Lemma 7.15, we obtain $d(v'_{i_j} v_{i_j}, v'_{i_{j+1}}) \leq N_T$. Similarly, since v_{i_1}, v'_{i_2} are the outputs of \mathcal{D}_{i_1} and \mathcal{D}_{i_2} on the same input prefix, we get

$$d(v_{i_1}, v'_{i_2}) \leq N_T.$$

$$\begin{aligned} d(v_{i_1} v_{i_2} v_{i_3} \dots v_{i_n}, v) &= d(v_{i_1} v_{i_2} v_{i_3} \dots v_{i_n}, v'_{i_n} v_{i_n}) \text{ (Since } v = v'_{i_n} v_{i_n} \text{)} \\ &\leq d(v_{i_1} v_{i_2} v_{i_3} \dots v_{i_n}, v'_{i_2} v_{i_2} v_{i_3} \dots v_{i_n}) + d(v'_{i_2} v_{i_2} v_{i_3} \dots v_{i_n}, v'_{i_3} v_{i_3} \dots v_{i_n}) \\ &\quad + \dots + d(v'_{i_{n-1}} v_{i_{n-1}} v_{i_n}, v'_{i_n} v_{i_n}) \text{ (Applying triangle inequality of } d \text{)} \\ &\leq n \cdot N_T \text{ (using Lemma 7.15)} \end{aligned}$$

In fact, on any input word, the distances between the outputs produced by \mathcal{D} and \mathcal{T} is less than or equal to $k \cdot N_T$, as there can be at most k switches between runs. Hence, we get that $d(\mathcal{D}, \mathcal{T})$ is bounded. \square

The characterisation of Lemma 7.16 also holds for series-sequential functions.

Lemma 7.17

Let $\mathcal{T} = \mathcal{D}_1 \dots \mathcal{D}_k$ be an **unambiguous transducer** where each \mathcal{D}_i is a **sequential trim** transducer for $i \in [k], k > 1$. The **series-sequential function** defined by \mathcal{T} is **approximately determinisable** w.r.t. a **metric** $d \in \{d_l, d_{lcs}, d_{dl}\}$ iff \mathcal{T} satisfies **ATP**.

Proof. (\rightarrow) follows from Proposition 7.14, and we prove (\leftarrow). Similarly to the proof in Lemma 7.16, we construct a sequential transducer \mathcal{D} using a subset construction of \mathcal{T} such that \mathcal{D} approximately determinises \mathcal{T} . Although \mathcal{T} is functional, and each \mathcal{D}_i , $i \in [k]$, is sequential, the nondeterminism arises between the transitions between one \mathcal{D}_i to the other. We assume an ordering for the states of \mathcal{T} for each \mathcal{D}_i . Intuitively, \mathcal{D} stores a subset of states of \mathcal{T} to capture all possible *active* runs on any input word. It produces the output of the active run of the smallest indexed sequential transducer, called the *producing* run. Upon termination of the producing run, it switches to the next active run of the smallest indexed sequential transducer. Since \mathcal{T} is unambiguous, there is at most one accepting run on any input. Formally, $\mathcal{D} = (Q, s_0, \delta, F, o)$ where

1. Q is the power set of the states in \mathcal{T} . Each set $S \in Q$ has exactly one state marked with a \bullet to denote the currently producing run.
2. s_0 is the initial state of \mathcal{T} and is marked with a \bullet .
3. The transition function $\delta : Q \times A \rightarrow Q \times B^*$ is defined as follows: $\delta(P, a) = (S, x)$ where S consists of all states q in \mathcal{T} reachable from a state $p \in P$ via transition

(p, a, q, x_{pq}) in \mathcal{T} . The output word x is set as follows. Let p be the \bullet marked state in P that belongs to the sequential transducer \mathcal{D}_i for some $i \in [k]$.

- a) If a transition (p, a, q, x_{pq}) in \mathcal{T} exists within the same sequential transducer \mathcal{D}_i , then $x = x_{pq}$ and q is \bullet marked in S .
- b) If no transition exists from p on a within \mathcal{D}_i , but a transition (p, a, q, x_{pq}) in \mathcal{T} exists to a different sequential transducer, i.e., q belongs to \mathcal{D}_{i+1} , then $x = x_{pq}$ and q is \bullet marked in S . Such a transition is called a *switch* (to \mathcal{D}_{i+1}).
- c) Otherwise, choose the smallest numbered state $p' \in P$ that belongs to the smallest indexed sequential transducer \mathcal{D}_j where $j \in [k]$ with a defined transition $(p', a, q', x_{p'q'})$ in \mathcal{T} , and set $x = x_{p'q'}$ and \bullet mark q' in S . Such a transition is also called a switch (to \mathcal{D}_j).

4. F is the set of all states $P \in Q$ such that P contains a final state of \mathcal{T} .
5. The output function $o : F \rightarrow B^*$ is defined, for $P \in F$, as $o(P) = o_{\mathcal{T}}(p)$ for some arbitrary final state $p \in P$ of \mathcal{T} , where $o_{\mathcal{T}}$ is the output function of \mathcal{T} .

The number of states of \mathcal{D} is exponential in the number of states of \mathcal{T} . We now argue that the number of switches in the run of \mathcal{D} between the active runs of \mathcal{T} on any input word is less than the number of states in \mathcal{T} , and hence finite. Let n_i be the number of states in \mathcal{D}_i for $i \in [k]$. Consider the run of \mathcal{D} on an arbitrary input word. Initially, only the initial state of \mathcal{D}_1 is active in \mathcal{D} . As the run proceeds, if P is the set of states of \mathcal{T} reached so far, then by construction of \mathcal{D} , the only \bullet marked state in P is the last state of the active run of \mathcal{T} that belongs to the smallest indexed \mathcal{D}_i . If this run eventually dies, then two cases can happen: (1) \mathcal{D} switches to another active run in \mathcal{D}_i , or, (2) if none exists, \mathcal{D} switches to some active run in \mathcal{D}_j , where $j > i$ is minimal. If case (2) happens, then no more states of \mathcal{D}_i are active, so \mathcal{D} will never switch again to \mathcal{D}_i in the future. The number of times case (1) can happen is bounded by n_i . Indeed, at most n_i states of \mathcal{D}_i can be active at any moment, and since \mathcal{D}_i is sequential, the number of active states in \mathcal{D}_i can only decrease.

Now, if \mathcal{D} eventually switches to \mathcal{D}_j , at most n_j states in \mathcal{D}_j are active, so at most n_j switches of type (1) can happen in \mathcal{D}_j , and so on until \mathcal{D} eventually terminates in some sequential transducer \mathcal{D}_l for $l \geq j$. Therefore, in the worst case, \mathcal{D} switches n_i

times in \mathcal{D}_i before switching to \mathcal{D}_{i+1} for all $i \in \{1, \dots, k-1\}$. So, the overall number of switches in the run of \mathcal{D} is at most $N = \sum_{i=1}^k n_i$, which is exactly the number of states in \mathcal{T} .

Now we show that $d(\mathcal{D}, \mathcal{T}) < \infty$ w.r.t. Levenshtein family. From the construction, it is clear that \mathcal{D} and \mathcal{T} have the same domain.

Consider an input word u accepted by \mathcal{T} . Since \mathcal{T} is unambiguous, there is exactly one run in \mathcal{T} that accepts u . Let $v = \mathcal{T}(u)$ and $v' = \mathcal{D}(u)$. From the construction of \mathcal{D} , v' can be decomposed into $v_{s_1} v_{s_2} \dots v_{s_n}$ accommodating $n \in [N]$ switches between the active runs of \mathcal{T} on the prefixes of u , where each $v_{s_i}, i \in [n]$ is the output produced by an active run r_i on the prefix of u . For each v_{s_i} , let v'_{s_i} denote the prefix of the output produced by the run r_i up to v_{s_i} .

Observe that $v'_{s_i} v_{s_i}$ and $v'_{s_{i+1}}$ (for $1 < i < n$) is the output produced by $\mathcal{D}_1 \dots \mathcal{D}_{j_i}$ and $\mathcal{D}_1 \dots \mathcal{D}_{j_{i+1}}$ on the same prefix of u where $j_i, j_{i+1} \in [k]$. By using Lemma 7.15, we obtain $d(v'_{s_i} v_{s_i}, v'_{s_{i+1}}) \leq N_T$. Similarly, since v_{s_1}, v'_{s_2} are the outputs of $\mathcal{D}_1 \dots \mathcal{D}_{j_1}$ and $\mathcal{D}_1 \dots \mathcal{D}_{j_2}$ on the same input prefix, it follows that $d(v_{s_1}, v'_{s_2}) \leq N_T$. As a consequence,

$$\begin{aligned} d(v_{s_1} v_{s_2} \dots v_{s_n}, v) &= d(v_{s_1} v_{s_2} \dots v_{s_n}, v'_{s_n} v_{s_n}) \quad (v = v'_{s_n} v_{s_n} \text{ since } r_{s_n} \text{ is the only accepting run of } u) \\ &\leq d(v_{s_1} v_{s_2} \dots v_{s_n}, v'_{s_2} v_{s_2} \dots v_{s_n}) + d(v'_{s_2} v_{s_2} \dots v_{s_n}, v'_{s_3} v_{s_3} \dots v_{s_n}) \\ &\quad + \dots + d(v'_{s_{n-1}} v_{s_{n-1}} v_{s_n}, v'_{s_n} v_{s_n}) \quad (\text{Applying triangle inequality of } d) \\ &\leq n \cdot N_T \quad (\text{using Lemma 7.15}) \end{aligned}$$

Therefore, $d(\mathcal{D}(u), \mathcal{T}(u)) \leq n \cdot N_T \leq N \cdot N_T$. This holds for any $u \in \text{dom}(\mathcal{T})$, and N being the number of states in \mathcal{T} , we get that $d(\mathcal{D}, \mathcal{T})$ is bounded. \square

We extend the characterisation to rational functions, where STP is also required to decompose the function into a finite union of series-sequential functions, which can then be transformed into a multi-sequential function using the properties of ATP.

Lemma 7.18

A rational function given by a trim transducer \mathcal{T} is approximately determinisable w.r.t. a metric $d \in \{d_l, d_{lcs}, d_{dl}\}$ iff \mathcal{T} satisfies both ATP and STP.

Proof. (\rightarrow) follows from Proposition 7.14.

(\leftarrow) Assume that the rational function is given by an unambiguous transducer \mathcal{T} with set of states Q . Disregarding the labels on transitions, decompose \mathcal{T} into a directed acyclic graph of maximal SCCs. Consider the set of paths Π of the form $\pi = S_1 t_1 S_2 \dots t_{n-1} S_n$ such that S_1 is an SCC which contains an initial state, S_n is an SCC which contains a final state, and for all $1 \leq i < n$, t_i is a transition of \mathcal{T} from a state of S_i to some state of S_{i+1} . For each path π , let \mathcal{T}_π denote the trim subtransducer of \mathcal{T} that removes all the transitions in \mathcal{T} except the transitions t_i ($1 \leq i < n$) and the transitions occurring in the SCCs S_i ($1 \leq i \leq n$).

Note that since the SCCs are maximal, the set Π is finite. Now, it is straightforward to see that $\mathcal{T} \equiv \mathcal{U} = \bigcup_{\pi \in \Pi} \mathcal{T}_\pi$. From [Lemma 7.6](#), we deduce \mathcal{U} satisfies both ATP as well as STP. Moreover, given \mathcal{T} 's unambiguity, each input accepted by \mathcal{T} is accepted by exactly one \mathcal{T}_π and, each SCC within a \mathcal{T}_π has a single entry and exit point.

Since \mathcal{U} satisfies STP, each SCC in \mathcal{T}_π satisfies TP, with the initial state being the unique entry point of the SCC. Hence we can determinise each SCC in \mathcal{T}_π and can obtain an unambiguous series-sequential transducer that is equivalent to \mathcal{T}_π and indeed satisfies ATP by [Lemma 7.6](#). From [Lemma 7.17](#), there exists a sequential transducer \mathcal{D}_π for each \mathcal{T}_π , such that $d(\mathcal{D}_\pi, \mathcal{T}_\pi)$ is finite.

Let $d(\mathcal{T}_\pi, \mathcal{D}_\pi) \leq k_\pi$ for some $k_\pi \in \mathbb{N}$. Consequently, the distance between \mathcal{T} and the new transducer $\mathcal{U}' = \bigcup_{\pi \in \Pi} \mathcal{D}_\pi$ is bounded where

$$d(\mathcal{T}, \mathcal{U}') = d(\mathcal{U}, \mathcal{U}') \leq \max \{k_\pi \mid \pi \in \Pi\}. \quad (7.6)$$

Further, since \mathcal{T} satisfies ATP, we deduce \mathcal{U}' also satisfies ATP by [Lemma 7.6](#). Being multi-sequential and satisfying ATP, \mathcal{U}' can be further approximately determinised using the construction outlined in [Lemma 7.16](#). Let \mathcal{D} be the sequential transducer such that $d(\mathcal{U}', \mathcal{D}) < \infty$. Since d is a metric, $d(\mathcal{T}, \mathcal{D}) \leq d(\mathcal{T}, \mathcal{U}') + d(\mathcal{U}', \mathcal{D})$. Since both $d(\mathcal{T}, \mathcal{U}')$ and $d(\mathcal{U}', \mathcal{D})$ are finite, $d(\mathcal{T}, \mathcal{D})$ is also finite. \square

7.4 Approximate Problems for Rational Relations

In this section, we consider three possible generalisations of the [approximate determinisation](#) problem to [rational relations](#). We describe those generalisations informally. The first one asks to decide whether a rational relation is [close](#) to some [rational function](#). We call it the approximate functionality problem. The second one, that we still call determinisation

problem, amounts to deciding whether a given rational relation is **close** to a sequential function. Finally, we consider an approximate uniformisation problem, which asks, given R , whether there exists a sequential function s which is close to a function f , whose graph is included in R . However, show that this problem is undecidable.

7.4.1 Approximate Functionality

Definition 7.19 (Approximate Functionalisation)

A rational relation R is **approximately functionalisable** w.r.t. a **metric** d if there exists a **rational function** f such that $d(R, f) < \infty$.

The **approximate functionality** problem asks, given R represented by a transducer, whether it is approximately functionalisable w.r.t. d . Towards this, we define the following value for a relation R and metric d , which measures how different are output words over the same input in R . More precisely, it is the maximal distance between any two output words over the same input word, in R :

$$\text{diff}_d(R) = \sup_{u \in \text{dom}(R)} \sup_{v_1, v_2 \in R(u)} d(v_1, v_2) \quad (7.7)$$

Lemma 7.20

For all **rational relation** R given as a transducer, $\text{diff}_d(R)$ are computable for all **metrics** given in Table 1.1.

Proof. Given a rational relation R , we can construct a rational relation R_o that consists of all pairs of output words of R on any input, i.e., $R_o = \{(v_1, v_2) \mid \exists u \in \text{dom}(R), (u, v_1), (u, v_2) \in R\}$. If R is a relation **defined** by a transducer \mathcal{T} , then the transducer obtained by $\mathcal{T} \times \mathcal{T}$ (**cartesian product** of \mathcal{T} by itself) by ignoring the input word is the **output product** transducer that defines the relation R_o . Observe that $\text{diff}_d(R)$ is equivalent to the **diameter** of R_o w.r.t. d . Since the diameter of a rational relation is computable for all metrics given in Table 1.1 (See Theorem 4.15), for those metrics $\text{diff}_d(R)$ is computable. \square

We now characterise rational relations which are approximately functionalisable.

Lemma 7.21

A rational relation R is approximately functionalisable w.r.t. a metric d if and only if $\text{diff}_d(R) < \infty$.

Proof. (\rightarrow) Assume that R is approximately functionalisable, i.e., there exists a rational function f such that $d(R, f) < \infty$. Therefore, $\text{dom}(R) = \text{dom}(f)$ and there exists an integer k such that for each input word $u \in \text{dom}(R)$, for each output word $v \in R(u)$, $d(v, f(u)) \leq k$ (since f is a function). By triangle inequality of metric d , the distance between any two arbitrary outputs words $v_1, v_2 \in R(u)$ on any input $u \in \text{dom}(R)$ is $d(v_1, v_2) \leq d(v_1, f(u)) + d(f(u), v_2) \leq 2k$. Since this holds for any input in the domain of R , we get $\text{diff}_d(R) = \sup_{u \in \text{dom}(R)} \sup_{v_1, v_2 \in R(u)} d(v_1, v_2) \leq 2k$.

(\leftarrow) Assume that $\text{diff}_d(R) < \infty$. Let $\text{diff}_d(R) \leq k$ for some $k \in \mathbb{N}$. We show that any uniformiser of the relation (a function of the same domain as the relation whose graph is included in the relation) is a function that approximately functionalise the relation. Given a relation $R \subseteq A^* \times B^*$, a uniformiser of R is a function $U : A^* \rightarrow B^*$ such that $\text{dom}(R) = \text{dom}(U)$ and for all $u \in \text{dom}(R)$, it holds that $(u, U(u)) \in R$. It is known that any rational relation admits a rational uniformiser (Kobayashi, 1969).

Let f be a uniformiser of R . We prove that $d(R, f) < \infty$. Since f is a uniformiser of R , $\text{dom}(R) = \text{dom}(f)$, and for all $u \in \text{dom}(R)$, it holds that $(u, f(u)) \in R$. Since $\text{diff}_d(R) \leq k$, the distance between the outputs of R on any input is less than or equal to k . Thus, for any input word $u \in \text{dom}(R)$, for each output word $v \in R(u)$, $d(v, f(u)) \leq k$ (since $f(u)$ is also an output of $R(u)$). Hence, $d(R, f) < \infty$, i.e., R is approximately functionalisable w.r.t. d . \square

Since $\text{diff}_d(R)$ is computable (see Lemma 7.20), we get the following result.

Theorem 7.22

The approximate functionality problem for rational relations given as transducers w.r.t. a metric given in Table 1.1 is decidable.

7.4.2 Approximate Determinisation

A rational relation R is said to be approximately determinisable for a metric d if it is almost a sequential function with respect to d . Formally, it means that there exists a

sequential function f such that $d(R, f) < \infty$. We show that the associated decision problem, that we call *approximate determinisation* problem, is decidable for Levenshtein family of distances. In fact, the characterisation for approximate determinisation of rational functions also holds for rational relations.

Lemma 7.23

A rational relation defined by a trim transducer \mathcal{T} is approximately determinisable w.r.t. Levenshtein family (d_l, d_{lcs}, d_{dl}) if and only if \mathcal{T} satisfies ATP and STP.

Proof. Let R be a rational relation given by a transducer \mathcal{T} , and let $d \in \{d_l, d_{lcs}, d_{dl}\}$. The proof of direction (\rightarrow) is the same as Proposition 7.14 when the function is replaced with a relation. For the other direction, assume that \mathcal{T} satisfies both ATP and STP. We first show that if R is approximately functionalisable then it is approximately determinisable when \mathcal{T} (that defines R) satisfies both ATP and STP. Let f be a rational function that approximately functionalises R w.r.t. Levenshtein distance, i.e., $d(R, f) < \infty$. Thus, a transducer \mathcal{F} that defines f satisfies $d(\mathcal{T}, \mathcal{F}) < \infty$. From Lemma 7.6, because \mathcal{T} satisfies both ATP and STP and $d(\mathcal{T}, \mathcal{F}) < \infty$, it follows that \mathcal{F} also satisfies ATP and STP. Using Lemma 7.18, the rational function f is approximately determinisable. Let g be a sequential function that approximately determinises f w.r.t. Levenshtein, i.e., $d(f, g) < \infty$. Since d is a metric on relations (see Proposition 4.4), $d(R, g) \leq d(R, f) + d(f, g)$. Since both $d(R, f)$ and $d(f, g)$ are finite, we get $d(R, g) < \infty$. Hence, the rational relation R is approximately determinisable.

Now it suffices to show that R is approximately functionalisable w.r.t. Levenshtein family of distance. Towards this, we prove that $\text{diff}_d(R) < \infty$ when \mathcal{T} satisfies ATP. Observe that $\text{diff}_d(R) = \text{dia}_d(R_o)$ where $R_o = \{(v_1, v_2) \mid \exists u \in \text{dom}(R), (u, v_1), (u, v_2) \in R\}$ is a rational relation that consists of all pairs of output words of R on any input. Using Lemma 7.15, $\text{dia}_d(R_o) \leq N_{\mathcal{T}}$ when \mathcal{T} satisfies ATP. Hence, $\text{diff}_d(R) = \text{dia}_d(R_o) < \infty$, and R is approximately functionalisable w.r.t. d by virtue of Lemma 7.21. \square

Since ATP and STP are decidable (Lemma 7.8), we obtain the following theorem.

Theorem 7.24

The approximate determinisation problem for rational relations given as transducers w.r.t. a metric $d \in \{d_l, d_{lcs}, d_{dl}\}$ is decidable.

7.4.3 Approximate Uniformisation

Given a relation $R \subseteq A^* \times B^*$, a *uniformiser* of R is a function $U : A^* \rightarrow B^*$ such that $\text{dom}(R) = \text{dom}(U)$ and for all $u \in \text{dom}(R)$, it holds that $(u, U(u)) \in R$. It is known that any rational relation admits a rational uniformiser (Kobayashi, 1969; Eilenberg, 1974), which is not true if we seek a sequential uniformiser. Moreover, the problem of deciding whether a given rational relation admits a sequential uniformiser is undecidable (Carayol and Löding, 2011). We consider an approximate variant of this problem.

Definition 7.25 (Approximate Uniformisation)

A rational relation R is *approximately uniformisable* w.r.t. a metric d if there exists a *uniformiser* U of R and a *sequential function* f such that $d(U, f) < \infty$. In that case, we say that R is *d-approximate uniformisable* by a sequential function.

In other words, R is d -approximate uniformisable by a sequential function f iff there exists an integer $k \in \mathbb{N}$ such that for all $u \in \text{dom}(R)$, there exists $(u, v) \in R$ with $d(v, f(u)) \leq k$.

Theorem 7.26

Checking whether a given rational relation is d -approximate uniformisable for $d \in \{d_l, d_{lcs}, d_{dl}\}$ is undecidable.

Proof. Let $\phi_1 : \{1, \dots, k\} \rightarrow B^*$ and $\phi_2 : \{1, \dots, k\} \rightarrow B^*$ be two *morphisms* defining an instance of the Post correspondence problem (PCP) (Post, 1946), which asks whether there exists $w \in \{1, \dots, k\}^+$ such that $\phi_1(w) = \phi_2(w)$.

Let $A = B \uplus \{1, \dots, i, a, b, \#\}$. Let R be the relation which as input takes any word of the form $w_1\#w_2\#\dots w_m\#X$ where $w_i \in \{1, \dots, k\}^*$ for $1 \leq i \leq m$ and $X \in \{a, b\}$. Let us define the outputs:

- If $X = a$, then the only output is $\phi_1(w_1)\#\phi_1(w_2)\#\dots\phi_1(w_m)\#$.
- If $X = b$, then any word of the form $v_1\#\dots v_m\#$ where $v_i \neq \phi_2(w_i)$ for all $1 \leq i \leq m$ is a valid output.

It can be shown that R is rational, recognizable by some transducer \mathcal{T} . We now show that R is approximately uniformisable iff PCP has no solution iff R is exactly uniformisable.

First, suppose that PCP has a solution w and R is approximately uniformisable by some sequential transducer \mathcal{D} , i.e. $d(\mathcal{T}, \mathcal{D}) \leq K$ for some K . Consider inputs of the form $u_\ell = (w\#)^\ell$ for $\ell \geq 0$, and let $\alpha_\ell, \alpha_a, \alpha_b$ be such that

$$q_0 \xrightarrow{u_\ell | \alpha_\ell}_{\mathcal{D}} q, q \xrightarrow{a | \alpha_a}_{\mathcal{D}} q_f \text{ and } q \xrightarrow{b | \alpha_b}_{\mathcal{D}} p_f,$$

where q_0 is the initial state, q is a state and q_f, p_f are final states of \mathcal{D} . Since $d(\mathcal{T}, \mathcal{D}) \leq K$, for all ℓ , $d((\Phi_1(w)\#)^\ell, \alpha_\ell \alpha_a) \leq K$ holds. Similarly, for all ℓ , there exist v_1, v_2, \dots, v_ℓ all different from $\Phi_2(w)$ such that

$$d(v_1\#v_2\#\dots v_\ell\#, \alpha_\ell \alpha_b) \leq K.$$

Since $d(\alpha_\ell \alpha_a, \alpha_\ell \alpha_b)$ is uniformly bounded for all ℓ by some M , by applying [triangle inequality](#) of d , we get that for all ℓ , there exist v_1, \dots, v_ℓ all different from $\Phi_2(w)$ s.t.

$$d((\Phi_1(w)\#)^\ell, v_1\#v_2\#\dots v_\ell\#) \leq 2K + M$$

Take $\ell = 4K + 2M + 2$, and fix a sequence of at most $2K + M$ edits from $(\Phi_1(w)\#)^\ell = (\Phi_1(w)\#\Phi_1(w)\#)^{2K+M+1}$ to $v_1\#v_2\#\dots v_\ell\#$. In this sequence, at least one copy of $(\Phi_1(w)\#\Phi_1(w)\#)$ is not edited by the sequence. Therefore, there exists some i such that $\#v_i\# = \#\Phi_1(w)\#$ and hence $v_i = \Phi_1(w)$. It is a contradiction since $v_i \neq \Phi_2(w) = \Phi_1(w)$. Therefore R is not approximately uniformisable.

Conversely, if there is no solution to PCP, then the following sequential function is an exact uniformiser of R : on any input of the form $w_1\#\dots\#w_mX$ it outputs $\Phi_1(w_1)\#\dots\#\Phi_1(w_m)$.

Since the PCP problem is undecidable ([Post, 1946](#)), the approximate uniformisation problem is undecidable for the Levenshtein family of distances. \square

7.5 Conclusion

In this chapter, we showed the decidability of [approximate determinisation](#) and [approximate functionality](#) for transducers. The approximate functionality problem reduces to computing the [diameter](#) of a [rational relation](#), while approximate determinisation for [Levenshtein family](#) is decided by approximate versions of [twinning property](#) — [ATP](#) and [STP](#). It is shown that STP is decidable in polynomial time, whereas ATP reduces

to deciding the [conjugacy of a rational relation](#), which requires exponential time (see [Chapter 5](#)). Consequently, the overall time complexity of approximate determinisation for a rational relation given as a transducer is doubly exponential. We conjecture that this is suboptimal, and leave as future work finding a better upper bound.

We showed that [approximate uniformisation](#) is undecidable for the Levenshtein family. However, our proof does not extend to the letter-to-letter setting, where both the given transducer and the required uniformiser process *and produce* a single letter on every transition. This problem is closely related to the standard Church synthesis for regular specifications, with the modification that the strategy to be synthesized is allowed to make a bounded number of errors. To our knowledge, this variant has not been studied in the literature.

Summary and Future Work

8.1 Summary of Contributions

We summarize the decidability results presented in this thesis in [Table 8.1](#).

Table 8.1. Summary of contributions.

| Input | Question | Result |
|--------------------------------|----------------------|---|
| functions f, g | $d(f, g)?$ | computable for rational functions w.r.t. d in Table 1.1 . |
| functions f, g | $d(f, g) < \infty?$ | decidable for rational functions w.r.t. d in Table 1.1 . |
| integer k , functions f, g | $d(f, g) \leq k?$ | decidable for rational functions w.r.t. d in Table 1.1 . |
| relation R | $dia_d(R)?$ | computable for rational relations w.r.t. d in Table 1.1 . |
| relation R | $dia_d(R) < \infty?$ | decidable for rational relations w.r.t. d in Table 1.1 . |

| Input | Question | Result |
|--------------------------------|--|---|
| integer k , relation R | $dia_d(R) \leq k$? | decidable for rational relations w.r.t. d in Table 1.1. |
| relations R, S | $Index(R, S)$? | computable when R is rational and S is d -metrizable rational for d in Table 1.1. |
| relations R, S | $Index(R, S) < \infty$? | decidable when R is rational and S is d -metrizable rational for d in Table 1.1. |
| integer k , relations R, S | $Index(R, S) \leq k$? | decidable when R is rational and S is d -metrizable rational for d in Table 1.1. |
| relation R | is R conjugate? | decidable for rational relations. |
| rational relation R | is R approximately functionalisable w.r.t. d ? | decidable for d in Table 1.1. |
| rational relation R | is R approximately determinisable w.r.t. d ? | decidable for rational relations for $d \in \{d_l, d_{lcs}, d_{dl}\}$. |
| rational relation R | is R approximately uniformisable w.r.t. d ? | undecidable for rational relations for $d \in \{d_l, d_{lcs}, d_{dl}\}$. |

8.2 Future Work

Metrics over words are extended to relations and functions, and we have established the decidability of the problems listed in Table 8.1. While these results provide a foundational understanding, several intriguing directions remain open for future research:

Computational Complexity and Efficient Algorithms. While we have shown decidability, determining the precise computational complexity of the problems in Table 8.1 remains open, and finding efficient algorithms to compute them is an interesting problem. In particular, deciding [conjugacy of a rational relation](#) relies on analysing [rational expressions](#), and an interesting direction for future research is to explore an automata-theoretic approach to this problem.

Parameterized Approximation and Extensions. A natural extension is to study the [approximate determinisation](#) and [approximate functionality](#) of [rational relations](#) up to distance k , where k is given as input. Approximate determinisation is studied for the [Levenshtein family](#) in the thesis, and extending it to other metrics in Table 1.1 is open. Furthermore, approximate decision problems could be explored in the context of other classical transducer problems, such as approximate minimisation and active learning.

Generalisation to more Expressive Transducers. Extending the problems discussed in this thesis to more general classes of transducers, such as *finite-valued transducers* and *two-way transducers*, presents an exciting avenue for future research. *Finite-valued transducers* are *k-valued transducers* for some $k \in \mathbb{N}$. Even though the distance between general rational relations is uncomputable due to the undecidability of their [equivalence](#) problem, we believe that computing the distance between finite-valued transducers remains feasible, as their equivalence has been shown to be decidable ([Culík II and Karhumäki, 1986](#); [Weber, 1993](#); [de Souza, 2008](#)). The transducers considered in this thesis are one-way [transducers](#), in the sense that they process input from left to right. *Two-way transducers* are those that allow bidirectional input reading while producing output in a streaming fashion. Notably, the equivalence problem for functional two-way transducers is known to be decidable ([Gurari and Ibarra, 1983](#); [Culík II and Karhumäki, 1987](#)), making their distance computation and deciding conjugacy, natural problems to study.

References

- M. Ackroyd. Isolated word recognition using the weighted levenshtein distance. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(2):243–244, 1980. <https://doi.org/10.1109/TASSP.1980.1163382>.
- A. V. Aho and T. G. Peterson. A minimum distance error-correcting parser for context-free languages. *SIAM Journal on Computing*, 1(4):305–312, 1972. <https://doi.org/10.1137/0201022>.
- C. Allauzen and M. Mohri. Linear-space computation of the edit-distance between a string and a finite automaton. In *London Algorithmics 2008: Theory and Practice*. 2008. <https://doi.org/10.48550/arXiv.0904.4686>.
- B. Aminof, O. Kupferman, and R. Lampert. Rigorous approximated determinization of weighted automata. *Theoretical Computer Science*, 480:104–117, 2013. <https://doi.org/10.1016/J.TCS.2013.02.005>.
- H. Bai, F. Franek, and W. F. Smyth. The new periodicity lemma revisited. *Discrete Applied Mathematics*, 212:30–36, 2016. <https://doi.org/10.1016/j.dam.2016.05.003>.
- M. Béal and O. Carton. Determinization of transducers over finite and infinite words. *Theoretical Computer Science*, 289(1):225–251, 2002. [https://doi.org/10.1016/S0304-3975\(01\)00271-7](https://doi.org/10.1016/S0304-3975(01)00271-7).
- M.-P. Béal, O. Carton, C. Prieur, and J. Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292(1):45–63, 2003. [https://doi.org/10.1016/S0304-3975\(01\)00214-6](https://doi.org/10.1016/S0304-3975(01)00214-6).

- J. Berstel. *Transductions and context-free languages*, volume 38 of *Teubner Studienbücher : Informatik*. Teubner, 1979. ISBN 3519023407.
- M. Blattner and T. Head. Single-valued a-transducers. *Journal of Computer and System Sciences*, 15(3):310–327, 1977. [https://doi.org/10.1016/S0022-0000\(77\)80033-0](https://doi.org/10.1016/S0022-0000(77)80033-0).
- U. Boker and T. A. Henzinger. Approximate determinization of quantitative automata. In *32nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012*, volume 18 of *LIPICs*, pages 362–373. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. <https://doi.org/10.4230/LIPICS.FSTTCS.2012.362>.
- U. Boker and T. A. Henzinger. Exact and approximate determinization of discounted-sum automata. *Logical Methods in Computer Science*, 10(1), 2014. [https://doi.org/10.2168/LMCS-10\(1:10\)2014](https://doi.org/10.2168/LMCS-10(1:10)2014).
- K. Bringmann and M. Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1216–1235. SIAM, 2018. <https://doi.org/10.1137/1.9781611975031.79>.
- A. L. Buchsbaum, R. Giancarlo, and J. R. Westbrook. An approximate determinization algorithm for weighted finite-state automata. *Algorithmica*, 30(4):503–526, 2001. <https://doi.org/10.1007/S00453-001-0026-6>.
- M.-P. Béal and O. Carton. Determinization of transducers over finite and infinite words. *Theoretical Computer Science*, 289(1):225–251, 2002. ISSN 0304-3975. [https://doi.org/10.1016/S0304-3975\(01\)00271-7](https://doi.org/10.1016/S0304-3975(01)00271-7).
- A. Carayol and C. Löding. Uniformization in automata theory. In *14th International Congress of Logic, Methodology and Philosophy of Science*, pages 153–178. London: College Publications, 2011.
- J. Cassaigne, J. Karhumäki, and J. Manuch. On conjugacy of languages. *RAIRO Theoretical Informatics and Applications*, 35(6):535–550, 2001. <https://doi.org/10.1051/ita:2001130>.

- C. Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theoretical Computer Science*, 5(3): 325–337, 1977. [https://doi.org/10.1016/0304-3975\(77\)90049-4](https://doi.org/10.1016/0304-3975(77)90049-4).
- C. Choffrut and J. Karhumäki. Combinatorics of words. In *Handbook of Formal Languages*, volume 1, pages 329–438. Springer-Verlag, 1997. https://doi.org/10.1007/978-3-642-59136-5_6.
- C. Choffrut and G. Pighizzini. Distances between languages and reflexivity of relations. *Theoretical Computer Science*, 286(1):117–138, 2002. [https://doi.org/10.1016/S0304-3975\(01\)00238-9](https://doi.org/10.1016/S0304-3975(01)00238-9).
- C. Choffrut and M. P. Schützenberger. Décomposition de fonctions rationnelles. In *3rd Annual Symposium on Theoretical Aspects of Computer Science, STACS 1986*, volume 210 of *Lecture Notes in Computer Science*, pages 213–226. Springer, 1986. https://doi.org/10.1007/3-540-16078-7_78.
- V. A. Cicirello. Kendall tau sequence distance: Extending kendall tau from ranks to sequences. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, 7(23):e1, 2020. <https://doi.org/10.4108/EAI.13-7-2018.163925>.
- A. Cohen and J. Collard. Instance-wise reaching definition analysis for recursive programs using context-free transductions. In *22nd International Conference on Parallel Architectures and Compilation Techniques, PACT 1998*, pages 332–339. IEEE Computer Society, 1998. <https://doi.org/10.1109/PACT.1998.727269>.
- T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *36th Internatilonal Colloquium Automata on Languages and Programming,ICALP 2009*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009. https://doi.org/10.1007/978-3-642-02930-1_12.
- T. Colcombet. Regular cost functions, part i: logic and algebra over words. *Logical Methods in Computer Science*, 9, 2013. [https://doi.org/10.2168/LMCS-9\(3:3\)2013](https://doi.org/10.2168/LMCS-9(3:3)2013).

- T. Colcombet. The factorisation forest theorem. In J. Pin, editor, *Handbook of Automata Theory*, pages 653–693. European Mathematical Society Publishing House, Zürich, Switzerland, 2021. <https://doi.org/10.4171/Automata-1/18>.
- T. Colcombet, D. Kuperberg, A. Manuel, and S. Toruńczyk. Cost functions definable by min/max automata. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*, volume 47 of *LIPICs*, pages 29:1–29:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. <https://doi.org/10.4230/LIPICS.STACS.2016.29>.
- K. Culík II and J. Karhumäki. The equivalence of finite valued transducers (on HDTOL languages) is decidable. *Theoretical Computer Science*, 47(3):71–84, 1986. [https://doi.org/10.1016/0304-3975\(86\)90134-9](https://doi.org/10.1016/0304-3975(86)90134-9).
- K. Culík II and J. Karhumäki. The equivalence problem for single-valued two-way transducers (on NPDTOL languages) is decidable. *SIAM Journal on Computing*, 16(2): 221–230, 1987. <https://doi.org/10.1137/0216018>.
- F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964. <https://doi.org/10.1145/363958.363994>.
- R. de Souza. On the decidability of the equivalence for k-valued transducers. In *12th International Conference on Developments in Language Theory, DLT 2008*, volume 5257 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2008. https://doi.org/10.1007/978-3-540-85780-8_20.
- S. Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. ISBN 0122340019. <https://www.worldcat.org/oclc/310535248>.
- C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and development*, 9(1):47–68, 1965. <https://doi.org/10.1147/rd.91.0047>.

- D. Eppstein, Z. Galil, and R. Giancarlo. Efficient algorithms with applications to molecular biology. In *Sequences: Combinatorics, Compression, Security, and Transmission*, pages 59–74. Springer, 1990. https://doi.org/10.1007/978-1-4612-3352-7_5.
- E. Filiot and P.-A. Reynier. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19, 2016. <https://doi.org/10.1145/2984450.2984453>.
- E. Filiot, I. Jecker, C. Löding, and S. Winter. On equivalence and uniformisation problems for finite transducers. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55 of *LIPICs*, pages 125:1–125:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. <https://doi.org/10.4230/LIPICS.ICALP.2016.125>.
- E. Filiot, N. Mazzocchi, and J. Raskin. A pattern logic for automata with outputs. *International Journal of Foundations of Computer Science*, 31(6):711–748, 2020. <https://doi.org/10.1142/S0129054120410038>.
- N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965. <https://doi.org/10.2307/2034009>.
- O. Finkel, V. Halava, T. Harju, and E. Sahla. On bi-infinite and conjugate post correspondence problems. *RAIRO Theoretical Informatics and Applications*, 57:7, 2023. <https://doi.org/10.1051/ITA/2023008>.
- P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2(1):88–101, 1968. [https://doi.org/10.1016/S0022-0000\(68\)80006-6](https://doi.org/10.1016/S0022-0000(68)80006-6).
- C. Frougny. Fibonacci numeration systems and rational functions. In *12th International Symposium on Mathematical Foundations of Computer Science, MFCS 1986*, volume 233 of *Lecture Notes in Computer Science*, pages 350–359. Springer, 1986. <https://doi.org/10.1007/BFB0016259>.
- C. Frougny. Representations of numbers and finite automata. *Mathematical Systems Theory*, 25(1):37–60, 1992. <https://doi.org/10.1007/BF01368783>.

- C. Frougny and J. Sakarovitch. Rational relations with bounded delay. In *8th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1991*, volume 480 of *Lecture Notes in Computer Science*, pages 50–63. Springer, 1991. <https://doi.org/10.1007/BFB0020787>.
- S. Ginsburg and G. F. Rose. A characterization of machine mappings. *Journal of Symbolic Logic*, 33(3):468–468, 1968. <https://doi.org/10.2307/2270340>.
- T. V. Griffiths. The unsolvability of the equivalence problem for λ -free nondeterministic generalized machines. *Journal of the ACM*, 15(3):409–413, 1968. <https://doi.org/10.1145/321466.321473>.
- E. M. Gurari and O. H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Mathematical systems theory*, 16(1):61–66, 1983. <https://doi.org/10.1007/BF01744569>.
- R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. <https://doi.org/10.1002/j.1538-7305.1950.tb00463.x>.
- Y.-S. Han, S.-K. Ko, and K. Salomaa. Computing the edit-distance between a regular language and a context-free language. volume 24, pages 85–96, 08 2012. ISBN 978-3-642-31652-4. https://doi.org/10.1007/978-3-642-31653-1_9.
- K. Hashiguchi. A decision procedure for the order of regular events. *Theoretical Computer Science*, 8:69–72, 1979. [https://doi.org/10.1016/0304-3975\(79\)90057-4](https://doi.org/10.1016/0304-3975(79)90057-4).
- T. Henzinger, J. Otop, and R. Samanta. Lipschitz robustness of finite-state transducers. *Leibniz International Proceedings in Informatics, LIPIcs*, 29, 03 2014. <https://doi.org/10.4230/LIPIcs.FSTTCS.2014.431>.
- D. S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, 1977. <https://doi.org/10.1145/322033.322044>.
- O. H. Ibarra. The unsolvability of the equivalence problem for epsilon-free ngsm’s with unary input (output) alphabet and applications. *SIAM Journal on Computing*, 7(4): 524–532, 1978. <https://doi.org/10.1137/0207042>.

- I. Jecker and E. Filiot. Multi-sequential word relations. *International Journal of Foundations of Computer Science*, 29(2):271–296, 2018. <https://doi.org/10.1142/S0129054118400075>.
- J. H. Johnson. Rational equivalence relations. *Theoretical Computer Science*, 47(3): 39–60, 1986. [https://doi.org/10.1016/0304-3975\(86\)90132-5](https://doi.org/10.1016/0304-3975(86)90132-5).
- J. Karhumäki. Combinatorial and computational problems on finite sets of words. In *3rd International Conference on Machines, Computations, and Universality, MCU 2001*, volume 2055 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2001. https://doi.org/10.1007/3-540-45132-3_4.
- R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *25th Annual ACM Symposium on Theory of Computing, STOC, 1993*, pages 278–285. ACM, 1993. <https://doi.org/10.1145/167088.167170>.
- M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 06 1938. ISSN 0006-3444. <https://doi.org/10.1093/biomet/30.1-2.81>.
- D. Kirsten and S. Lombardy. Deciding unambiguity and sequentiality of polynomially ambiguous min-plus automata. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*, volume 3 of *LIPICs*, pages 589–600. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. <https://doi.org/10.4230/LIPICS.STACS.2009.1850>.
- D. E. Knuth, J. H. M. Jr., and V. R. Pratt. Fast Pattern Matching in Strings. *SIAM Journal on Computing*, 6(2):323–350, 1977. <https://doi.org/10.1137/0206024>.
- K. Kobayashi. Classification of formal languages by functional binary transductions. *Information and Control*, 15(1):95–109, 1969. ISSN 0019-9958. [https://doi.org/10.1016/S0019-9958\(69\)90651-2](https://doi.org/10.1016/S0019-9958(69)90651-2).
- S. Konstantinidis. Computing the edit distance of a regular language. *Information and Computation*, 205:1307–1316, 09 2007. <https://doi.org/10.1016/j.ic.2007.06.001>.
- D. Kozen. *Automata and computability*. Undergraduate texts in computer science. Springer, 1997. ISBN 978-0-387-94907-9.

- J. B. Kruskal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review*, 25(2):201–237, 1983. <https://doi.org/10.1137/1025045>.
- H. Leung and V. Podolskiy. The limitedness problem on distance automata: Hashiguchi’s method revisited. *Theoretical Computer Science*, 310(1-3):147–158, 2004. [https://doi.org/10.1016/S0304-3975\(03\)00377-3](https://doi.org/10.1016/S0304-3975(03)00377-3).
- V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.
- M. Lothaire. *Combinatorics on words, Second Edition*. Cambridge mathematical library. Cambridge University Press, 1997. ISBN 978-0-521-59924-5.
- R. C. Lyndon and M.-P. Schützenberger. The equation $a^M = b^N c^P$ in a free group. *Michigan Mathematical Journal*, 9(4):289–298, 1962. <https://doi.org/10.1307/mmj/1028998766>.
- M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997. <https://aclanthology.org/J97-2003.pdf>.
- M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003. <https://doi.org/10.1142/S0129054103002114>.
- M. Mohri, F. Pereira, and M. Riley. *Speech Recognition with Weighted Finite-State Transducers*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-49127-9. https://doi.org/10.1007/978-3-540-49127-9_28.
- A. Muscholl and G. Puppis. The Many Facets of String Transducers (Invited Talk). In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019*, volume 126 of *LIPIcs*, pages 2:1–2:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. <https://doi.org/10.4230/LIPICS.STACS.2019.2>.
- M. Nivat. *Transduction des langages de Chomsky*. PhD thesis, Annales de l’Institut Fourier, 1968. <https://doi.org/10.5802/aif.287>.

- T. Okuda, E. Tanaka, and T. Kasai. A method for the correction of garbled words based on the Levenshtein metric. *IEEE Transactions on Computers*, C-25(2):172–178, 1976. <https://doi.org/10.1109/TC.1976.5009232>.
- D. Peifer. Max dehn and the origins of topology and infinite group theory. *The American Mathematical Monthly*, 122(3):217–233, 2015. <https://doi.org/10.4169/AMER.MATH.MONTHLY.122.03.217>.
- E. L. Post. A variant of a recursively unsolvable problem. *Bulletion of the American Mathematical Society*, 52:264–268, 1946. <https://doi.org/10.1090/s0002-9904-1946-08555-9>.
- G. N. Raney. Sequential functions. *Journal of the ACM*, 5(2):177–180, 1958. ISSN 0004-5411. <https://doi.org/10.1145/320924.320930>.
- C. Reutenauer and M.-P. Schützenberger. Minimization of rational word functions. *SIAM Journal on Computing*, 20(4):669–685, 1991. <https://doi.org/10.1137/0220042>.
- R. Samanta, J. V. Deshmukh, and S. Chaudhuri. Robustness analysis of string transducers. In D. V. Hung and M. Ogawa, editors, *11th International Symposium on Automated Technology for Verification and Analysis, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, volume 8172 of *Lecture Notes in Computer Science*, pages 427–441. Springer, 2013. https://doi.org/10.1007/978-3-319-02444-8_30.
- M. P. Schützenberger. Sur les relations rationnelles. In *2nd GI Conference on Automata Theory and Formal Languages, 1975*, volume 33 of *Lecture Notes in Computer Science*, pages 209–213. Springer, 1975. https://doi.org/10.1007/3-540-07407-4_22.
- M. P. Schützenberger. Sur une variante des fonctions séquentielles. *Theoretical Computer Science*, 4(1):47–57, 1977. [https://doi.org/10.1016/0304-3975\(77\)90055-X](https://doi.org/10.1016/0304-3975(77)90055-X).
- I. Simon. Limited subsets of a free monoid. In *19th Annual Symposium on Foundations of Computer Science, SFCS, 1978*, pages 143–150. IEEE Computer Society, 1978. <https://doi.org/10.1109/SFCS.1978.21>.

- B. Smyth. Special issue - computing patterns in strings: foreword. *Fundamenta Informaticae*, 56(1,2), 2002. ISSN 0169-2968.
- R. E. Stearns and H. B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985. <https://doi.org/10.1137/0214044>.
- L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *5th Symposium on Theory of Computing, STOC 1973*, pages 1–9. ACM, 1973. <https://doi.org/10.1145/800125.804029>.
- J. Ullman. Near-optimal, single-synchronization-error-correcting code. *IEEE Transactions on Information Theory*, 12(4):418–424, 1966. <https://doi.org/10.1109/TIT.1966.1053920>.
- R. Wagner. Order-n correction for regular languages. *Communications of the ACM*, 17: 265–268, 05 1974. <https://doi.org/10.1145/360980.360995>.
- R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974. ISSN 0004-5411. <https://doi.org/10.1145/321796.321811>.
- A. Weber. Decomposing finite-valued transducers and deciding their equivalence. *SIAM Journal on Computing*, 22(1):175–202, 1993. <https://doi.org/10.1137/0222014>.
- A. Weber and R. Klemm. Economy of description for single-valued transducers. *Information and Computation*, (2):327–340, 1995. <https://doi.org/10.1006/INCO.1995.1071>.

Index

- d -metrizable relation, 53, 54, 56, 164
- k -bounded diameter problem, 48, 50, 51
- k -bounded index problem, 51, 53, 54
- k -closeness problem, 45, 46, 50, 56, 114, 115, 130, 132, 133
- k -valued, 13, 14, 165
- ATP, 138–140, 145, 147, 149, 150, 152, 154, 158, 160
- STP, 138–140, 145, 147, 149, 154, 158, 160
- alphabet, 9–11, 14, 42, 49, 138
- approximate determinisation, 146, 147, 149, 150, 152, 154, 155, 157, 158, 160, 164, 165
- approximate functionality, 156, 157, 160, 164, 165
- approximate uniformisation, 159, 161, 164
- automaton, 10, 11, 13, 114, 121, 130
- bounded delay, 48, 53, 121
- bounded diameter problem, 48, 50, 51, 53, 116, 119–121, 133
- bounded index problem, 51, 53, 54, 56, 57
- cartesian product, 12, 14, 47, 49, 61, 111, 114, 116, 146, 150, 156
- closeness problem, 45, 46, 50, 56, 114, 115, 117, 124, 128, 130, 131, 133, 136, 155, 156
- common inner witness, 68–70, 75, 90, 93, 96
- common outer witness, 68–70, 75, 90, 93, 96
- common witness, 68, 70, 72, 73, 75, 76, 78, 79, 84–86, 90, 93, 99, 105, 106, 108, 109, 118–120
- composition closure, 50, 51, 53, 54, 56
- conjugacy distance, 38, 39, 55, 116, 119, 121, 130, 133
- conjugate, 26–31, 33, 47, 60, 62–64, 66, 72, 73, 75, 76, 78, 79, 86, 90, 99, 105, 106, 116, 119, 120, 136, 138, 150, 161, 164, 165
- cut, 29–31, 33, 67, 68, 78, 80, 86, 106
- cyclic shift, 23, 38
- Damerau-Levenshtein distance, 38, 39, 47, 55, 116, 117, 130, 147, 149, 150, 152, 154, 158, 159, 164

- delay, 23, 24, 43, 122, 123, 126, 127, 136, 138, 142
- determinisation, 15, 60, 61, 111, 135
- diameter of a relation, 48, 50, 56, 57, 115, 116, 150, 156, 160, 163, 164
- distance between transducers, 44, 45, 114, 116
- distance problem, 45–47, 49, 53, 56, 113, 115, 133, 148
- domain, 9, 11, 14, 42, 43, 46, 114, 124, 128, 130, 151
- edit distance, 36, 39, 49, 55, 61, 133, 136
- equivalence, 10, 14, 44, 45, 50, 122, 165
- factor, 9, 23, 28, 64, 81, 116, 124
- function, 10, 42, 50, 57, 136, 145, 163, 164
- functionality, 14, 135
- Hamming distance, 36, 39, 45, 48, 54, 55, 57, 116, 123, 124, 130–132, 136
- Hausdorff distance, 43
- index of a relation, 51, 53, 54, 56, 164
- inner witness, 66–68, 112
- interior, 123, 126, 127
- language, 10, 16, 41, 47, 49, 62, 68, 115, 117, 121, 130
- left-border, 123, 127
- Levenshtein distance, 37, 39, 44, 47, 48, 55, 57, 116, 117, 120, 130, 136, 147, 149, 150, 152, 154, 158, 159, 164
- Levenshtein family, 116, 121, 133, 136, 146, 147, 158, 160, 165
- lomonon common subsequence distance, 37, 39, 47, 55, 116, 117, 130, 147, 149, 150, 152, 154, 158, 159, 164
- metric, 26, 36, 39, 42–48, 50, 53, 54, 56, 57, 113–116, 118, 121, 130, 136, 146, 149, 150, 152, 154, 156–158, 160, 164
- metric on functions, 42, 146, 159, 163
- metric on relations, 43, 156
- monoid, 16, 17, 62
- morphism, 49, 159
- multi-sequential, 147, 150
- nondeterministic, 10, 11, 60
- outer witness, 66–68, 112
- power, 24, 140
- prefix, 9, 11, 23, 28, 81, 86, 121, 130, 138, 141, 150
- primitive, 24, 27, 29–31, 33, 78, 106
- primitive root, 24–28, 67–70, 72, 75, 78–80, 90, 93, 106, 116, 117, 139, 140
- rational expression, 16, 18, 62–64, 68, 73, 117–121, 133, 165
- rational function, 12–14, 42–44, 46, 47, 49, 61, 111, 113–115, 124, 128, 130, 133, 135, 146, 147, 149, 154–156, 163

-
- rational relation, 11, 13, 16, 42, 44, 48–51, 53, 54, 56, 57, 60, 62, 115, 116, 120, 121, 137, 155–160, 163–165
 - rationalizable distance, 54–56
 - redux, 74, 75, 85, 93
 - relation, 9, 10, 42–44, 50, 51, 61, 163, 164
 - right-border, 123, 127
 - run, 10, 11, 123, 138, 150
 - sequential function, 13, 15, 57, 61, 114, 121, 130, 135–137, 146–148, 152, 159
 - series-sequential, 148, 152
 - singleton redux, 74, 76, 96, 99, 105
 - suffix, 9, 24, 81, 86
 - sumfree expression, 17, 63, 106, 108, 109, 118
 - sumfree normal form, 18
 - sumfree set, 65, 66, 73–76, 85, 96, 105
 - transducer, 11, 13, 16, 42, 54, 56, 60, 111, 116, 130, 135, 137–140, 145, 149, 150, 154, 157, 158, 165
 - transposition distance, 37, 39, 55, 116, 126, 128, 130–132
 - twinning pattern, 138, 139
 - twinning property, 15, 61, 111, 136–138, 160
 - unambiguous, 10, 12, 15, 57, 114, 132, 152
 - uniformiser, 159
 - witness, 66