Ph.D. Dissertation

Learning and Equivalence of One-Counter Systems

Prince Mathew

183312002

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

December, 2024

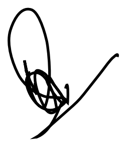
Certificate of Approval

The Ph.D. Defence Committee recommends to accept the dissertation entitled **Learning and Equivalence of One-Counter Systems**, submitted by **Prince Mathew** 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 on 28 April, 2025.

Advisor:

Dr. Sreejith A.V.

Ph.D. Defence Committee Members:



Chairperson: Prof. Deepak D'Souza

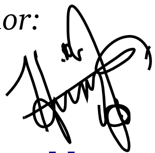
Internal Examiner: Dr. Amaldev Manuel

Declaration

I, PRINCE MATHEW, hereby testify that the work embodied in this dissertation titled "Learning and Equivalence of One-Counter Systems", carried out under the supervision of Dr. Sreejith A.V., is the result of bonafide research undertaken during the period 2019-2024 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:



PRINCE MATHEW

Advisor:

Dr. Sreejith A.V.

Dedications

To my Family.

Acknowledgements

This thesis is the culmination of years of hard work, and it would not have been possible without the help and support of many individuals and institutions. It stands as much a testament to their contributions as to my efforts. I am deeply grateful to everyone who has contributed to this journey, and I take this opportunity to acknowledge their invaluable support.

First and foremost, I would like to express my heartfelt gratitude to my advisor, Dr. A.V. Sreejith, for his constant motivation, support, and guidance throughout my Ph.D. journey. This thesis is as much a result of his hard work as it is of mine, and I am deeply indebted to his dedication and mentorship, which have directly influenced my efforts and achievements. This thesis would never have materialised without his unwavering dedication and mentorship. He always made himself available whenever I needed his advice, irrespective of the day or time. His insights and encouragement have been instrumental in shaping this work, and I thank him wholeheartedly for his invaluable guidance.

I would also like to extend my sincere thanks to my collaborators, Dr. Vincent Penelle and Dr. Prakash Saivasan, for their significant contributions since the beginning of my Ph.D. Their input has been crucial in ensuring that my research met its goals. I am immensely grateful for the time and effort they have dedicated to guiding me and their valuable insights that have enriched this thesis.

I am deeply thankful to the Department of Computer Science, IIT Goa, and its faculty for their constant motivation, support, and guidance. The department has been an encouraging and nurturing environment that has played a vital role in my academic growth.

I extend my gratitude to CMI (Chennai Mathematical Institute) for allowing me to undertake coursework and for hosting my occasional visits. I want to thank Prof.

Madhavan Mukund, Dr. C. Aiswarya, and Prof. Deepak Khemani for their kindness and support, which made my stay in Chennai both productive and memorable. I extend my heartfelt thanks to all the faculty, the B.Sc., M.Sc., and Ph.D. scholars at CMI for their invaluable companionship, support, and guidance during my time there.

I am grateful to Dr. Prakash Saivasan and the Institute of Mathematical Sciences (IMSc) for facilitating my visits and making all the necessary arrangements. I would also like to thank Prof. Véronique Bruyère, Prof. Guillermo A. Pérez, Prof. Jean-François Raskin and Prof. Christof Löding for inviting me to the University of Mons, University of Antwerp, Université Libre de Bruxelles and RWTH Aachen University respectively. I am grateful for their efforts in ensuring smooth arrangements during these visits. I want to thank Mr. P.M. Rane, my house owner in Goa, for making all the necessary arrangements over the past few years that made my stay in Goa a smooth one. His assistance and thoughtfulness were instrumental in providing me with a conducive environment for work and research.

I also wish to express my sincere thanks to Dr. Anup Basil Mathew and Dr. A. Baskar from BITS Pilani Goa for being part of the discussions during the initial stages of my Ph.D. I would also like to express my gratitude to Prof. Somenath Biswas and Dr. Amaldev Manuel for their valuable inputs and support during the initial stages of my Ph.D. Their guidance and assistance were instrumental in shaping the foundation of my research. I want to take a moment to remember and thank the late Mr. Ajay Kumar for being a wonderful companion during my coursework at IIT Goa. I am also grateful to all my friends at IIT Goa – B.Tech, M.Tech, and Ph.D. scholars – who made my Ph.D. life memorable and enjoyable. A special note of thanks goes to Dr. Rahul C S for all his intuitive suggestions and for lending us his ears.

I want to thank all the anonymous reviewers who reviewed my research papers over the years. The ideas presented in those papers form a significant part of this thesis. Their constructive feedback has greatly influenced how this work is presented and has helped polish it to its current form. I would also like to thank Dr. Clint P. George, Dr. Lok Pati Tripathi, Dr. Arpita Korwar, Dr. Sudakshina Dutta, Dr. Amaldev Manuel, Dr. Vincent Penelle, Dr. Prakash Saivasan and Dr. A.V. Sreejith who have been part of my doctoral committee at various stages. Their valuable inputs and suggestions have been instrumental in shaping my research.

I would like to sincerely thank Prof. Stefan Göller (University of Kassel, Germany) and Prof. Deepak D'Souza (IISc Bangalore, India) for serving as the reviewers of my thesis. I sincerely appreciate the time and effort they have devoted to carefully reviewing my work and providing thoughtful and constructive feedback. Their insights have been invaluable in enhancing both the scientific rigour and the overall presentation of this thesis. I am genuinely grateful for their contributions, which have helped improve the quality of this thesis.

A special note of thanks goes to my wife and colleague, Mrs. Saina Sunny, who has been my pillar of support throughout this journey. She has been by my side through all the ups and downs, patiently listening to my ideas, providing feedback, and reading my work. Her sacrifices, encouragement, and care over the past five years have been invaluable, and I am deeply thankful for her unwavering support.

I extend my heartfelt gratitude to my family for their unwavering support and encouragement throughout this journey. I am especially grateful to my parents, Mr. Mathew Kurian and Mrs. Jayamol T U, whose unwavering belief in me has been a constant source of strength and inspiration. I would also like to express my sincere thanks to my in-laws, Mr. T U Sunny and Mrs. Jolly Sunny, for their understanding and motivation, which have been invaluable throughout this journey.

Finally, I thank the Almighty for keeping me safe and for all the countless blessings, both visible and invisible, that have supported me through this endeavour. I would also like to extend my gratitude to all my teachers who have guided me over the years, helping shape me into who I am today.

I want to express my heartfelt thanks to everyone who has been part of my journey, directly or indirectly. This thesis stands as a tribute to your contributions, and I couldn't have reached this point without your support.

Abstract

This thesis contributes to the study of active learning and equivalence of one-counter systems. The first part of this thesis focuses on developing a new approach for active learning of deterministic real-time one-counter automata (DROCAs). Traditional approaches to learning DROCAs often involve learning an initial portion of an infinite behavioural graph. The aim is to identify a repetitive structure in the graph that defines the overall behaviour of the DROCA. However, this repetitive structure often requires constructing an exponentially large graph, leading to an exponential runtime and number of queries. In contrast, our SAT-based method reduces the number of queries to a polynomial number, significantly improving the efficiency of learning algorithms for DROCAs. Furthermore, it learns a minimal counter-synchronous DROCA recognising the language making it more feasible for practical applications. We also introduce improved equivalence checking algorithms for counter-synchronised one-counter automata that is used in our learning algorithm. This helps to obtain much smaller counter-examples on equivalence queries. Additionally, we present an even faster equivalence check for the special case of visibly one-counter automata.

The second part of the thesis introduces real-time one-deterministic-counter automata (RODCAs). These are real-time one-counter automata with the property of counter-determinacy, meaning that all paths labelled by a given word starting from the initial configuration have the same counter-effect. We prove that the equivalence of weighted RODCAs over fields is in P. This is a step towards resolving the open question of the equivalence problem of weighted one-counter automata. We also considered boolean RODCAs and showed that the equivalence problem for (nondeterministic) boolean RODCAs is in PSPACE. In contrast, it is undecidable for (nondeterministic) boolean one-counter automata. Additionally, a faster equivalence-checking method is proposed for the special case of deterministic weighted real-time one-counter automata. The thesis also explores the regularity and covering problems for weighted RODCAs, and shows that they are in P.

Keywords: One-counter automata, Equivalence, Active Learning, Weighted automata, Reachability, SAT solver, Regularity, Covering.

Author: PRINCE MATHEW

Advisor: Dr. Sreejith A.V.

List of Publications

1. Prince Mathew, Vincent Penelle, Prakash Saivasan, and A.V Sreejith. “Weighted One-Deterministic-Counter Automata”. In Patricia Bouyer and Srikanth Srinivasan, editors, *43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023)*, volume 284 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1-39:23, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
   `LIPIcs.FSTTCS.2023.39`, `doi:10.4230/LIPIcs.FSTTCS.2023.39`.

2. Prince Mathew, Vincent Penelle, Prakash Saivasan, and A.V Sreejith. “Equivalence of Deterministic Weighted Real-Time One-Counter Automata”. *In the proceedings of the 11th Indian Conference on Logic and its Applications (ICLA 2025)- To appear*. `CoRR, abs/2411.03066, 2024`.

3. Prince Mathew, Vincent Penelle, and A.V Sreejith. Learning Real-Time One-Counter Automata Using Polynomially Many Queries, *In the proceedings of the 31st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2025)- To appear*. `CoRR, abs/2411.08815, 2024`.

4. Prince Mathew, Vincent Penelle, and A.V Sreejith. Learning Deterministic One-Counter Automata in Polynomial Time, *In the proceedings of the 40th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2025)- To appear*. `CoRR, abs/2503.04525, 2025`.

Contents

List of Tables

List of Figures

List of Algorithms

CHAPTER 1

Introduction

1.1 Thesis Overview

Automata are mathematical models for systems that can describe sequences of operations or events, making them valuable for applications such as formal verification of software, model checking, and system analysis. A one-counter automaton (OCA) extends a finite-state automaton with a non-negative integer counter. The counter can be incremented, decremented, and tested for zero on a transition. They are strictly more expressive than finite-state systems and can capture the behaviour of certain infinite-state systems. A visibly one-counter automaton (VOCA) is an OCA with the visibly constraint, where the counter actions depend only on the letter.

Automata learning focuses on the automatic construction of automata from observations, such as input/output samples. Automata learning is particularly important for software verification as it allows for the modelling and analysing of systems where exhaustive testing may not be feasible. With suitable algorithms, automata learning can enable software verification tools to automatically construct models from system traces, which can then be analysed for errors, security vulnerabilities, and performance issues. Despite theoretical advancements, the practical applicability of automata learning algorithms, especially for extensions of finite automata (e.g., one-counter automata, pushdown automata), is hindered by inefficient equivalence checks and learning processes. This limits the broader applicability of automata learning algorithms for complex systems.

In the first half of this thesis, we primarily focus on active learning and equivalence of real-time one-counter systems. The model is called real-time as there are no ε-transitions. We discuss the state-of-the-art and our contributions to learning and equivalence of one-counter systems separately.

In the second half of the thesis, we examine weighted one-counter systems and introduce a model called the weighted real-time one-deterministic counter automaton (weighted RODCA). These are weighted one-counter automata in which the counter actions are deterministic. Our main focus is on the equivalence, regularity, and covering of weighted RODCAs over fields. Additionally, we demonstrate the existence of a more efficient equivalence check for a subclass of weighted RODCAs. We also study both deterministic and nondeterministic RODCAs and show that they are expressively equivalent to DROCAs.

1.1.1 Active Learning and Equivalence of DROCAs

Active Learning

The objective of learning algorithms is to identify a model that best fits a given set of observations or data. However, this is a computationally challenging task. For instance, finding the minimal deterministic finite automaton (DFA) that accepts a given set of positive samples and rejects a given set of negative samples is NP-complete (Gold, 1978). Angluin (1987) proposed an active learning framework involving a learner and a teacher to overcome this challenge. The learner constructs an automaton through a structured process of queries to the teacher (see Figure 1.1). Membership queries determine whether a given word is accepted by the target automaton, while equivalence queries verify if a hypothesised model matches the target and returns a counter-example if it doesn't. Angluin (1987) showed that DFAs can be learned using membership and equivalence queries in polynomial time. Angluin's algorithm, known as the L^* algorithm, provided a theoretical foundation for efficient learning of DFAs by allowing the learner to learn a minimal DFA in time polynomial in the size of the DFA. However, one cannot extend the L^* algorithm directly for more complex

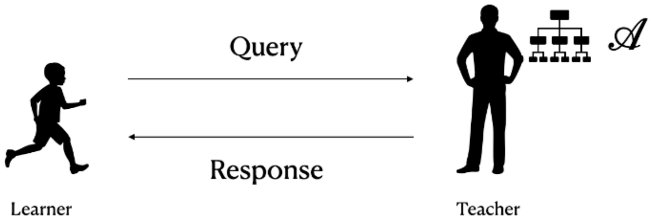


Fig. 1.1. Active learning framework.

automata, such as one-counter automata and pushdown automata, which have infinitely many equivalence classes.

While state-of-the-art algorithms effectively learn finite-state models, extending these methods to systems with resources like stacks or counters poses significant challenges. These extended models are highly desirable as they capture richer and more expressive behaviour, enabling the formal verification of complex properties. However, current computational and algorithmic limitations hinder the effective learning of such systems, emphasising the need for novel approaches to bridge this gap in automata learning.

Recent advancements in automata learning, especially in deterministic real-time one-counter automaton (DROCA), have shown promising potential for practical applications (Bruyère et al., 2022). However, the scalability remains a concern. Current learning algorithms for DROCAs, such as those by Fahmy and Roos (1995) and Bruyère et al. (2022), still suffer from exponential time complexity in both the runtime and the number of queries. This is also the case with the algorithm for learning VOCAs by Neider and Löding (2010). This exponential cost is a critical limitation that restricts the applicability of these algorithms in real-world scenarios where scalability is a priority.

Our Contributions to Learning DROCA

To address limitations in the learning process, we focused on a new method for learning deterministic real-time one-counter automaton (DROCA) with the help of a SAT solver that requires only polynomially many queries (see Chapter 4). We follow an active learning framework similar to that by Angluin (1987) that uses membership, equivalence and counter value queries to infer an unknown DROCA from a teacher. The DROCA learnt will be counter-synchronous with the DROCA under learning, meaning that for any word, the counter-value reached on reading that word is the same on both machines.

In our learning algorithm, we use ideas from the L^* algorithm by Angluin (1987) and the idea of computing a minimal separating DFA with the help of a SAT solver (Dell'Erba et al., 2024). This sets it apart from existing techniques that rely on identifying repetitive patterns in behaviour graphs of exponential size. We show that learning a minimal counter-synchronous DROCA with respect to the teachers DROCA can be done using polynomially many membership, equivalence, and counter-value queries. This improves the existing result by Bruyère et al. (2022) that needed an exponential number of queries, space and time for learning DROCAs using membership, equivalence, counter-value and partial-equivalence queries.

We can adapt the algorithm for learning DROCAs to learn VOCAs with the minimal number of states without using counter-value queries. It is important to note that there does not exist an Angluin-style learning algorithm for VOCAs that learns a minimal VOCA in polynomial time. If there is an L^*-type algorithm to learn a minimal VOCA in polynomial time, then it can be used to minimise VOCAs in polynomial time. However, Michaliszyn and Otop (2022) have pointed out that the problem of minimising VOCAs is NP-Complete. This implies that knowing the counter-values will not help us get a better learning algorithm for DROCAs. Hence, learning DROCAs with the help of counter-value queries (even with counter-observability) is also a difficult problem.

We implemented the proposed algorithm and tested it on randomly generated DROCAs. Our evaluations show that the proposed method outperforms the existing technique by Bruyère et al. (2022) on the test set.

Equivalence

The equivalence problem for machine models is an important problem in formal verification and learning. It asks whether two given machine models are equivalent. However, the equivalence check is particularly challenging for complex models, such as one-counter automata and pushdown automata, due to their decision complexities. The study on weighted versions of pushdown or one-counter machines is limited. Probabilistic pushdown automaton (pPDA) is a weighted pushdown automata variant that has been studied in the literature. The decidability of equivalence of pPDAs is a long-standing open problem and is as complex as the multiplicity equivalence of context-free grammars (Forejt et al., 2014). The latter problem is not known to be decidable. These highlight the intrinsic difficulty of handling weighted and probabilistic extensions in pushdown automata models. Since the equivalence problem for pPDAs is unknown, the natural question to ask is the equivalence problem for probabilistic one-counter automata. However, this problem is also unresolved.

Our Contributions to the Equivalence of One-Counter Systems

The equivalence of DROCAs is in P Böhm and Göller (2011). However, it cannot be used in practice because of its high computational costs. Given two DROCAs with number of states less than some integer n, the equivalence check takes $\mathcal{O}(n^{26})$ time. To overcome this, we introduced the concept of counter-synchronous DROCAs, which allows for a faster equivalence check that runs in $\mathcal{O}(n^5)$ time (see Chapter 7). Two DROCAs are counter-synchronous if, for any word, the counter-value reached on reading that word is the same on both machines. We use this equivalence check of counter-synchronous DROCAs in our learning algorithm for DROCAs in Chapter 4. We also have discovered an even faster $\mathcal{O}(n^3)$ algorithm for checking the equivalence of two visibly one-counter automata.

1.1.2 Weighted One-Counter Systems

In the second half of the thesis, we identify a subclass of probabilistic one-counter automata called weighted real-time one-deterministic-counter automaton (weighted RODCA) (see Chapter 6) and prove that its equivalence is in P (see Chapter 7). These are weighted real-time one-counter automata with the property of counter-determinacy, meaning that all paths labelled by a given word starting from the initial configuration have the same counter-effect. This property allows for a polynomial time equivalence checking for this model, whereas the problem is not known to be decidable for weighted OCAs in general. Counter-determinacy can be seen as a relaxation of the *visibly* constraint on one-counter automata, as the counter actions are no longer input-driven but are deterministic. These are strictly more expressive than weighted VOCAs, which are weighted OCAs where the input alphabet determines the counter action.

Our primary focus was on the equivalence problem for weighted RODCAs where the weights are from a field. This field can be infinite, such as the set of rationals $\mathbb{Q}$ with the standard addition and multiplication operations. We first introduced two problems for weighted RODCAs, called the co-VS reachability problem and co-VS coverability problem and prove that they are in P (see Chapter 7). The co-VS reachability and the co-VS coverability problem are crucial, along with the ideas developed by Böhm and Göller (2011); Böhm et al. (2014) and Valiant and Paterson (1975) in proving that the equivalence problem of weighted RODCAs is in P. Our positive result on weighted RODCAs forms a foundation for developing their learning algorithms, creating new possibilities for efficient verification tools.

We have also studied deterministic weighted real-time one-counter automaton (DWROCA) (see Chapter 5), which is a subclass of weighted RODCAs and proved that the equivalence of this model can be checked faster than that of general weighted RODCAs. All these results are positive steps towards solving the equivalence of weighted pushdown systems. We also considered deterministic RODCAs and nondeterministic RODCAs in Chapter 6. We showed that deterministic RODCAs can be converted to DROCAs in polynomial time, and hence, their equivalence is in P. Nondeterministic RODCAs can be transformed to DROCAs that are exponentially larger, and hence, their equivalence is in PSPACE. In contrast, it is undecidable for nondeterministic one-counter automata.

1.1.3 Other Results

We have also considered the regularity, covering and coverable equivalence problems of weighted RODCAs in Chapter 7. First, we consider the regularity problem of weighted RODCAs – the problem of deciding whether a weighted RODCA is equivalent to some weighted automaton. The crucial idea in proving regularity is to check for the existence of infinitely many equivalence classes. The proof technique is adapted from the ideas developed by Böhm et al. (2014) in the context of real-time OCAs. We prove that the regularity problem of weighted RODCAs (weights from a field) is in P.

Finally, we look at covering and coverable equivalence problems. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two weighted RODCAs without initial distributions. We say $\mathcal{A}_2$ *covers* $\mathcal{A}_1$ if for all initial configurations of $\mathcal{A}_1$ there exists an initial configuration of $\mathcal{A}_2$ that makes them equivalent. We say $\mathcal{A}_1$ and $\mathcal{A}_2$ are *coverable equivalent* if $\mathcal{A}_1$ covers $\mathcal{A}_2$, and $\mathcal{A}_2$ covers $\mathcal{A}_1$. We show that the covering and coverable equivalence problems for weighted RODCAs are decidable in polynomial time. The proof relies on the algorithm to check the equivalence of two weighted RODCAs.

A summary of our results is given in Table 1.1.

<table>
<tr><th></th><th>Deterministic RODCAs</th><th>Nondeterministic RODCAs</th><th>weighted RODCAs over a field</th></tr>
<tr><td>Reachability</td><td rowspan="2">P (Chistikov et al., 2019)</td><td rowspan="2">P (Chistikov et al., 2019)</td><td rowspan="6">P</td></tr>
<tr><td>Coverability</td></tr>
<tr><td>Equivalence</td><td rowspan="2">P (Böhm and Göller, 2011)</td><td rowspan="2">PSPACE (Böhm and Göller, 2011)</td></tr>
<tr><td>Regularity</td></tr>
<tr><td>Covering</td><td rowspan="2">P</td><td rowspan="3">PSPACE</td></tr>
<tr><td>Coverable equivalence</td></tr>
<tr><td>Learning</td><td>P[1]</td><td>open</td></tr>
</table>

Table 1.1. Summary of results.

1.2 Organisation of the Thesis

The rest of the thesis is organised into four main parts, each focusing on distinct aspects of learning and equivalence of one-counter automata and its variants. Below is an overview:

Part I: Preliminaries

This part establishes the foundational concepts necessary for the rest of the thesis.

- **Chapter 2: Foundations**, introduces mathematical notations and core topics like linear algebra, complexity classes and one-counter automata. The chapter explains various OCA variants, including deterministic, deterministic real-time, and visibly one-counter automata. It also explores the concepts of learning automata.

[1]One can obtain a P^{NP} algorithm for learning Deterministic RODCAs from the results presented in Chapter 4. However, in recent work Mathew et al. (2025c), we have proved the existence of a polynomial time algorithm for learning deterministic one-counter automata.

Part II: Deterministic Real-Time One-Counter Automata (DROCA)

The second part investigates equivalence and learning of DROCAs.

• **Chapter 3: Equivalence of DROCAs** gives faster equivalence checks for counter-synchronised and visibly one-counter automata.

• **Chapter 4: Learning DROCAs Using Polynomially Many Queries** presents our learning algorithm for DROCAs using SAT solver. The chapter explains the learning algorithm, implementation details and experimental results.

Part III: Weighted One-Counter Automata

This part presents our work on weighted one-counter systems.

• **Chapter 5: Deterministic Weighted Real-Time One-Counter Automata** introduces deterministic weighted real-time one-counter automaton (DWROCA) and provides an algorithm for checking its equivalence.

• **Chapter 6: Real-Time One-Deterministic Counter Automata** introduces weighted real-time one-deterministic counter automaton (weighted RODCA) and discusses our results for nondeterministic RODCAs.

• **Chapter 7: Equivalence of Weighted RODCAs Over Fields**, focuses on weighted RODCAs over fields and studies their reachability, equivalence, regularity and covering problems.

Part IV: Conclusions

This part presents our conclusions and future directions.

• **Chapter 8: Summary and Future Work**, summarises our contributions and identifies future research directions.

Note to Readers of the PDF Version of This Thesis

This document incorporates numerous hyperlinks to enhance usability in its electronic format. In addition to the cross-references generated by LaTeX, most of the terms and concepts defined within the document are directly linked to their definitions. This feature, made possible by the knowledge package by Thomas Colcombet, applies to both text and mathematical expressions. Keep in mind that a single expression can contain multiple links. For example, the expression $\mathsf{ce}(T)|_1 > \mathsf{K}^2\mathrm{poly}_3(\mathsf{K})$ includes links to three distinct concepts: ce, K and poly_3.

Part I

Preliminaries

CHAPTER 2

Foundations

Contents

2.1 Mathematical Notations

For any set S, we use $|S|$ to denote the number of elements in S. We denote by $\mathbb{N}$ the set $\{0, 1, 2, 3, ...\}$ and by $[i, j]$ the interval $\{i, i+1, \ldots, j\}$. For any $d \in \mathbb{N}$, the sign of d, denoted by $sign(d)$, is defined as

$$sign(d) = \begin{cases} 0, & \text{if } d = 0 \\ 1, & \text{otherwise.} \end{cases}$$

An alphabet is a finite non-empty set of letters. In this thesis, we denote the alphabet by Σ. We use Σ^* to denote the set of all finite-length words over Σ. For all $l \in \mathbb{N}$, we use $\Sigma^{\leq l}$ (resp. Σ^l) to denote the set of all words over Σ having a length less than or equal to l (resp. exactly equal to l). Let $w = \sigma_1\sigma_2 \ldots \sigma_n \in \Sigma^*$, where $\sigma_1, \sigma_2, \cdots, \sigma_n \in \Sigma$. We use $|w|$ to denote the length of the word w. For $j, k \in [1, n]$, with $j < k$, we use $w[j]$ to denote the letter σ_j and $w_{[j \cdots k]}$ to denote the factor $\sigma_j\sigma_{j+1} \cdots \sigma_k$. A word $u = \sigma_1 \cdots \sigma_k$ is a *subword* of a word w, if $w = u_0\sigma_1u_1\sigma_2 \cdots \sigma_ku_k$, where $\sigma_i \in \Sigma$, $u_j \in \Sigma^*$ for all $i \in [1, k]$ and $j \in [0, k]$. The word u is a *proper subword* of w if $u \neq w$. We say that a word u is a *prefix* of a word w if there exists $v \in \Sigma^*$ such that $w = uv$. Similarly, a word v is a *suffix* of a word w if there exists $u \in \Sigma^*$ such that $w = uv$. A set S of words is *prefix-closed*, if for all $s \in S$, any prefix s' of s is also in S. Similarly, a set S of words is *suffix-closed*, if for all $s \in S$, any suffix s' of s is also in S.

Given a tuple $e = (e_1, e_2)$, we use $e|_1$ to denote e_1 and $e|_2$ to denote e_2.

A group is a triple $(S, \circ, e)$ where S is a non-empty set, $\circ$ is an associative binary operation and $e \in S$ is an identity element. The set S is closed under the operation $\circ$ and for all $s \in S$, $e \circ s = s \circ e = s$. For all $s \in S$ there exists $s^{-1} \in S$ such that $s \circ s^{-1} = s^{-1} \circ s = e$.

A *field* $\mathcal{F} = (S, +, \times, 0_e, 1_e)$ is a set S with two commutative operations $+$ and $\times$ and distinguished elements 0_e and 1_e such that

- $(S, +, 0_e)$ and $(S \setminus \{0_e\}, \times, 1_e)$ are groups.

- $\times$ distributes over $+$ (i.e., for all $s, t, r \in S$, $s \times (t + r) = s \times t + s \times r$).

2.1.1 Linear Algebra

In this thesis, we use $\mathbf{x}, \mathbf{y}, \mathbf{z}$ to denote vectors over a field $\mathcal{F}$, and $\mathbb{A}, \mathbb{B}, \mathbb{M}$ to denote matrices over a field $\mathcal{F}$. We use $\mathcal{U}, \mathcal{V}$ to denote vector spaces. Given a vector $\mathbf{x} = [x_0, \ldots, x_{n-1}] \in \mathcal{F}^n$ for some $n \in \mathbb{N}$, we use $\mathbf{x}[i]$ to denote x_i for all $i \in [0, n-1]$. We recall the following fundamental theorems of vector spaces. An interested reader can refer to the book on linear algebra by Strang (2006) for more details.

Lemma 2.1

The following are true for a field $\mathcal{F}$.

1. For any set X of n vectors in $\mathcal{F}^k$ with $n > k$, there exists a vector $\mathbf{x} \in X$ that is a linear combination of the other vectors in X.

2. Given a set B of n vectors in $\mathcal{F}^k$ and a vector $\mathbf{x} \in \mathcal{F}^k$, we can check if $\mathbf{x}$ is a linear combination of vectors in B in time polynomial in k and n.

The above lemma holds irrespective of whether the values k and n are given in unary or binary notation.

Lemma 2.2

Let $\mathcal{V}$ be a vector space, $k \in \mathbb{N}$ and for all $n \in [0, k]$, $\mathbf{z}_n \in \mathcal{F}^k$ and $\mathbb{M}_n \in \mathcal{F}^{k \times k}$. Then, there exists an $i \in [1, k]$ such that the following conditions are true:

1. $\mathbf{z}_i$ is a linear combination of $\mathbf{z}_0, \ldots \mathbf{z}_{i-1}$, and

2. if $\mathbf{z}_i \mathbb{M}_i \notin \mathcal{V}$, then there exists $j < i$ such that $\mathbf{z}_j \mathbb{M}_i \notin \mathcal{V}$.

Proof. Let $k \in \mathbb{N}, n \in [0, k]$, $\mathbf{z}_n \in \mathcal{F}^k, \mathbb{M}_n \in \mathcal{F}^{k \times k}$ be matrices and $\mathcal{V}$ be a vector space.

1. Consider the set $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k\}$ of $k+1$ vectors of dimension k. It follows from Lemma 2.1 that there are at most k independent vectors of dimension k, and hence not all elements of the set can be independent.

2. Let $i \in [1, k]$ be such that $\mathbf{z}_i$ is a linear combination of $\mathbf{z}_0, \ldots \mathbf{z}_{i-1}$ and $\mathbf{z}_i\mathbb{M}_i \notin \mathcal{V}$. Let us assume for contradiction that $\mathbf{z}_j\mathbb{M}_i \in \mathcal{V}$ for all $j \in [0, i-1]$. Since $\mathbf{z}_i$ is a linear combination on $\mathbf{z}_0, \ldots \mathbf{z}_{i-1}$, there exists $s_0, \ldots s_{i-1} \in \mathcal{F}$ such that

$$\mathbf{z}_i = s_0 \cdot \mathbf{z}_0 + s_1 \cdot \mathbf{z}_1 + \cdots + s_{i-1} \cdot \mathbf{z}_{i-1}$$

Since $\mathbf{z}_i\mathbb{M}_i = \left(\sum_{j=0}^{i-1} s_j \cdot \mathbf{z}_j\right)\mathbb{M}_i$ and $\mathcal{V}$ is closed under linear combinations, we get that $\mathbf{z}_i\mathbb{M}_i \in \mathcal{V}$ contradicting our initial assumption. □

Lemma 2.3

Let $\mathcal{V}$ be a vector space, $k \in \mathbb{N}$ and for all $n \in [0, k^2]$, $\mathbb{A}_n, \mathbb{M}_n, \mathbb{B}_n \in \mathcal{F}^{k \times k}$. Then, there exists an $i \in [1, k^2]$ such that the following conditions are true:

1. $\mathbb{M}_i$ is a linear combination of $\mathbb{M}_0, \ldots, \mathbb{M}_{i-1}$, and
2. for all $\mathbf{x} \in \mathcal{F}^k$, if $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i \notin \mathcal{V}$, then there exists a $j < i$ such that $\mathbf{x}\mathbb{A}_i\mathbb{M}_j\mathbb{B}_i \notin \mathcal{V}$.

Proof. Let $\mathbb{A}_n, \mathbb{M}_n, \mathbb{B}_n \in \mathcal{F}^{k \times k}$ for $n \in [0, k^2]$, be matrices over $\mathcal{F}$ and $\mathcal{V}$ a vector space.

1. Consider the set $\{\mathbb{M}_0, \mathbb{M}_1, \ldots, \mathbb{M}_{k^2}\}$ of k^2+1 matrices of dimension k^2. It follows from Lemma 2.1 that there are at most k^2 independent vectors of dimension k^2, and hence not all elements of this set can be independent.

2. Let $i \in [1, k^2]$ be such that $\mathbb{M}_i$ is a linear combination of $\mathbb{M}_0, \ldots, \mathbb{M}_{i-1}$ and $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i \notin \mathcal{V}$. Since $\mathbb{M}_i$ is dependent on $\mathbb{M}_0, \ldots, \mathbb{M}_{i-1}$, we prove that there exists $j < i$ such that $\mathbf{x}\mathbb{A}_i\mathbb{M}_j\mathbb{B}_i \notin \mathcal{V}$. Let us assume for contradiction that this is not the case. Since $\mathbb{M}_i$ is a linear combination on $\mathbb{M}_0, \ldots \mathbb{M}_{i-1}$, there exists $s_0, \ldots s_{i-1} \in \mathcal{F}$ such that

$$\mathbb{M}_i = s_0 \cdot \mathbb{M}_0 + s_1 \cdot \mathbb{M}_1 + \cdots + s_{i-1} \cdot \mathbb{M}_{i-1}$$

Since $\mathbf{x}\mathbb{A}_i\mathbb{M}_j\mathbb{B}_i \in \mathcal{V}$ for all $j \in [0, i-1]$ we get that $\mathbf{x}\mathbb{A}_i\mathbb{M}_i\mathbb{B}_i = \sum_{j=0}^{i-1} s_j \cdot \mathbf{x}\mathbb{A}_i\mathbb{M}_j\mathbb{B}_i \in \mathcal{V}$, which is a contradiction. □

Lemma 2.4

Let $k \in \mathbb{N}, \mathbb{A} \in \mathcal{F}^{k \times k}$ and $\mathcal{V} \subseteq \mathcal{F}^k$ be a vector space. Then, the following set is a vector space,

$$\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^k \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}.$$

Proof. To prove that $\mathcal{U}$ is a vector space, it suffices to show that it is closed under vector addition and scalar multiplication. First, we prove that $\mathcal{U}$ is closed under vector addition. Let $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{U}$ be two vectors, since $\mathbf{z}_1\mathbb{A}, \mathbf{z}_2\mathbb{A} \in \mathcal{V}$, $(\mathbf{z}_1 + \mathbf{z}_2)\mathbb{A} = \mathbf{z}_1\mathbb{A} + \mathbf{z}_2\mathbb{A} \in \mathcal{V}$. Therefore, $\mathbf{z}_1 + \mathbf{z}_2 \in \mathcal{U}$. Now, we prove that $\mathcal{U}$ is closed under scalar multiplication. For any vector $\mathbf{z}_1 \in \mathcal{U}$, we know that $\mathbf{z}_1\mathbb{A} \in \mathcal{V}$. Since $\mathcal{V}$ is a vector space, for any scalar $s \in \mathcal{F}$, $(s \cdot \mathbf{z}_1)\mathbb{A} \in \mathcal{V}$, and therefore $s \cdot \mathbf{z}_1 \in \mathcal{U}$. This concludes the proof. $\square$

In particular, the above lemma holds for the vector space $\{\mathbf{0} \in \mathcal{F}^k\}$.

2.2 Complexity Classes

In this section, we will briefly introduce and discuss the complexity classes that we use in this thesis. These complexity classes categorise problems based on the computational resources required to solve them. These resources typically include time and space, measured as functions of the input size.

The class P consists of all decision problems that can be solved efficiently using an algorithm that runs in polynomial time with respect to the input size. i.e., problem is said to be in P if there exists an algorithms that solves it in $\mathcal{O}(n^k)$ time for some constant $k \in \mathbb{N}$, where n is the input size.

The class NP contains all decision problems for which a given solution can be verified in polynomial time. However, finding a solution to a problem in this class might be hard. Every problem that can be solved in polynomial time can also be verified in polynomial time. Therefore, it follows that P $\subseteq$ NP. However, the question of whether NP $\subseteq$ P remains an open problem. A problem is NP-*complete* if it is in NP and is at least as hard as any other problem in NP. This means that any problem in NP can be transformed into it using a polynomial-time reduction.

If there is a polynomial time algorithm for solving any NP-complete problem, then all problems in NP can also be solved in polynomial time, implying P=NP.

The class NL consists of all decision problems that can be solved using a logarithmic amount of memory. It is known that all NL problems can be solved in polynomial time, implying NL⊆ P. A problem is NL-*complete* if it belongs to NL and is at least as hard as any other problem in NL. This means that any problem in NL can be transformed into it using a log-space reduction.

The class PSPACE consists of all problems that can be solved using a polynomial amount of memory. Every NP problem is in PSPACE implying NP ⊆ PSPACE. A problem is PSPACE-*complete* if it belongs to PSPACE and is at least as hard as any other problem in PSPACE. This means that any problem in PSPACE can be transformed to it using a polynomial-time reduction.

2.3 Finite Automata

Finite automata are models of computation used for recognising regular languages. They consist of finite states and transitions between them based on the input symbols. There are two types of finite automata: nondeterministic finite automata (NFA) and deterministic finite automata (DFA).

2.3.1 Nondeterministic Finite Automata

Definition 2.5 (Nondeterministic finite automata)

A nondeterministic finite automaton (NFA), is defined by a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, where

- Q is a finite set of states,
- Σ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta : Q \times \big(\Sigma \cup \{\varepsilon\}\big) \to 2^Q$ is the transition function, and
- $F \subseteq Q$ is the set of accepting states.

In an NFA, the range of the transition function δ is the powerset 2^Q. Therefore, on reading a symbol from a state, the next state is not given by a single element of Q, but rather a subset of it. This subset defines all the possible states that can be reached on taking that transition. We also allow ε-transitions, where the NFA can move to a new state without consuming any input.

Given a word $w = a_1 a_2 \cdots a_n$, where each $a_1, \ldots, a_n \in \big(\Sigma \cup \{\varepsilon\}\big)$ and states $q, q' \in Q$, we write $q \xrightarrow{w} q'$, if there exists states $q_1, \ldots, q_{n+1}$ such that $q_1 = q, q_{n+1} = q'$ and for all $i \in [1, n]$, $q_{i+1} \in \delta(q_i, a_i)$. A word $w \in \Sigma^*$ is accepted by an NFA $\mathcal{A}$ if there exists a state $q' \in F$ such that $q_0 \xrightarrow{w} q'$. The language of an NFA, denoted as $\mathcal{L}(\mathcal{A})$, is the set of all words accepted by $\mathcal{A}$.

2.3.2 Deterministic Finite Automata

Definition 2.6 (Deterministic finite automata)

A deterministic finite automata (DFA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, where

- Q is a finite set of states,
- Σ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta : Q \times \Sigma \to Q$ is the transition function, and
- $F \subseteq Q$ is the set of accepting states.

The transition function δ can be extended inductively for words as follows: $\delta(q, \sigma w) = \delta(\delta(q, \sigma), w)$ for any $q \in Q$, $\sigma \in \Sigma$, and $w \in \Sigma^*$. We use $q \xrightarrow{w} q'$ to denote that $\delta(q, w) = q'$. A word w is said to be *accepted* by the DFA $\mathcal{A}$ if $q_0 \xrightarrow{w} q'$ such that $q' \in F$. The word is said to be *rejected* otherwise. The language of the DFA, denoted as $\mathcal{L}(\mathcal{A})$, is the set of all words accepted by $\mathcal{A}$. Two DFAs $\mathcal{A}$ and $\mathcal{B}$ are said to be equivalent if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.

2.4 One-Counter Automata

A one-counter automaton (OCA) can be viewed as a finite automaton equipped with an integer counter (refer Figure 2.1). The transitions within this machine depend on the current state and whether the counter's value is positive or zero. The automaton can increment or decrement the counter during transitions, but the counter's value must always stay non-negative. Unlike finite state machines that can recognise only regular languages (languages that can be expressed using regular expressions), the counter aids OCAs in recognising non-regular languages (e.g., $\{a^n b^n \mid n \geq 0\}$).

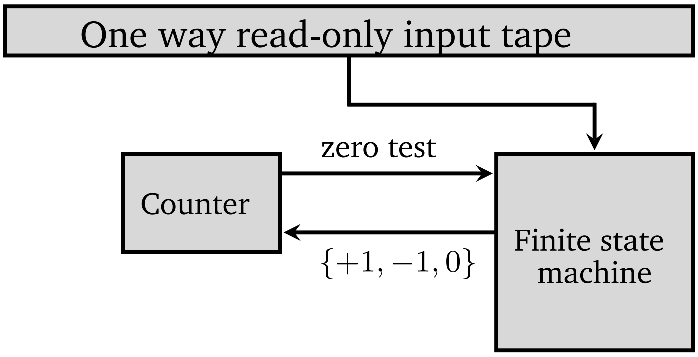


Fig. 2.1. One-counter automata.

An OCA is called deterministic or nondeterministic based on how the transitions of the finite state machine are defined. In this thesis, our focus is on deterministic one-counter automata.

2.4.1 Variants of One-Counter Automata

2.4.1.1 Nondeterministic One-Counter Automata

Definition 2.7 (Nondeterministic one-counter automata)
A nondeterministic one-counter automaton (NOCA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta_0, \delta_1, F)$, where

- Q is a finite nonempty set of states,
- Σ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta_0 : Q \times \big(\Sigma \cup \{\varepsilon\}\big) \to 2^{Q\times\{0,+1\}}$ and $\delta_1 : Q \times \big(\Sigma \cup \{\varepsilon\}\big) \to 2^{Q\times\{0,+1,-1\}}$ are functions that defines the transitions, and
- $F \subseteq Q$ is the set of final states.

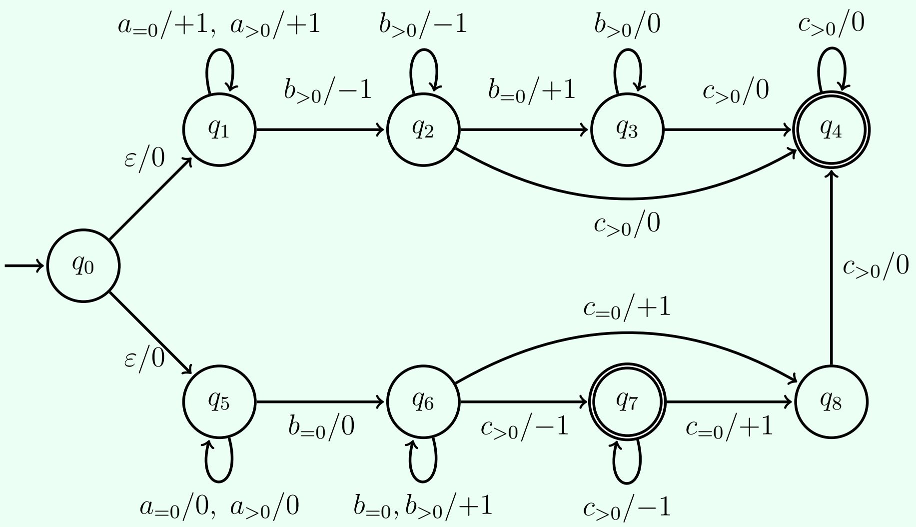
A configuration of an NOCA is a pair (q, n), where q is a state and n is a counter value. The transition function δ_0 (resp. δ_1) can only be taken from a configuration with zero (resp. positive) counter value. Each transition can change the counter value by at most one. Given a configuration $(q, n) \in Q \times \mathbb{N}$ and $\sigma \in \Sigma \cup \{\varepsilon\}$, there is transition from the configuration (q, n) to another configuration (q', n') on reading σ if and only if $(q', e) \in \delta_{sign(n)}(q, \sigma)$, where $e \in \{-1, 0, +1\}$ and $n' = n+e$. We denote this by $(q, n) \xrightarrow{\sigma} (q', n')$. A *run* of a word starting from a configuration c_0 and ending in c_n on reading the word w (denoted as $c_0 \xrightarrow{w} c_n$) is specified by the transition rules that leads from c_0 to c_n and in the process, reads the word w. Note that in an NOCA, a word can have multiple runs. A run $\pi_1 = c_0 \xrightarrow{w_1} c_i$ is called a *sub-run* of the run $\pi = c_0 \xrightarrow{w} c_n$, if $w = w_1 w_2$ for some $w_1, w_2 \in \Sigma^*$ and there exists a run $\pi_2 = c_i \xrightarrow{w_2} c_n$ such that that runs π and $\pi_1\pi_2$ are the same. A word w is accepted by an NOCA if there exist a run $(q_0, 0) \xrightarrow{w} (q_F, n)$, for some $q_F \in F$ and $n \in \mathbb{N}$. The language of an NOCA is the set of all words accepted by it.

Example 2.8 (`MatchOrSkip`)

An NOCA recognising the language `MatchOrSkip` defined as

$$\texttt{MatchOrSkip} = \{a^n b^m c^k \mid n, m, k > 0 \text{ with } n \neq m \text{ or } m \neq k\}.$$

If a transition from state q_i to q_j is labelled with $\sigma_{=0}/e$ (resp. $\sigma_{>0}/e$) for some $\sigma \in \left(\Sigma \cup \{\varepsilon\}\right)$ and $e \in \{-1, 0, +1\}$, then it can be taken upon reading the symbol σ from state q_i when the current counter value is zero (resp. positive). The value e will be added to the current counter value in both cases. Assume that all transitions not shown in the figure go to a non-final sink state while incrementing the counter by 1.



2.4.1.2 Deterministic One-Counter Automata

Definition 2.9 (Deterministic one-counter automata)

A deterministic one-counter automaton (DOCA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta_0, \delta_1, F)$, where

- Q is a finite nonempty set of states,
- Σ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta_0 : Q \times \big(\Sigma \cup \{\varepsilon\}\big) \to Q \times \{0, +1\}$ and $\delta_1 : Q \times \big(\Sigma \cup \{\varepsilon\}\big) \to Q \times \{0, +1, -1\}$ are partial functions that defines the transitions, and
- $F \subseteq Q$ is the set of final states.

A deterministic one-counter automaton (DOCA), was first introduced by Valiant and Paterson (1975). A configuration of a DOCA is a pair (q, n), where q is a state and n is a counter value. The transition function δ_0 (resp. δ_1) can only be taken from a configuration with zero (resp. positive) counter value. The transition function δ_0 (resp. δ_1) can have an ε-transition from a state q (i.e., can be taken without reading any input) if $\delta_0(q, a)$ (resp. $\delta_1(q, a)$) is not defined for all $a \in \Sigma$. Each transition can change the counter value by at most one. Given a configuration $(q, n) \in Q \times \mathbb{N}$ and $\sigma \in \Sigma \cup \{\varepsilon\}$, there is transition from the configuration (q, n) to another configuration (q', n') on reading σ if and only if $\delta_{sign(n)}(q, \sigma) = (q', e)$, where $e \in \{-1, 0, +1\}$ and $n' = n + e$. We denote this by $(q, n) \xrightarrow{\sigma} (q', n')$. A run of a word starting from a configuration c_0 and ending in c_n on reading the word w (denoted as $c_0 \xrightarrow{w} c_n$) is specified by the transition rules that leads from c_0 to c_n and in the process, reads the word w. A word w is accepted by the DOCA if $(q_0, 0) \xrightarrow{w} (q_F, n)$, for some $q_F \in F$ and $n \in \mathbb{N}$. According to the definition by Valiant and Paterson (1975), there are no ε-transitions possible from a final state, and the counter value can never go below zero.

There are different ways of defining a DOCA. For instance, Böhm et al. (2013) defines it as a machine with reset moves instead of having ε-transitions. The set of states is partitioned into states that have a reset transition and states that don't. The reset move from a state resets the counter to zero without reading any input and takes the machine to a state based on the current state and the current counter value modulo some period. They show that this model is equivalent to the definition of DOCA with ε-transitions and is equi-succint.

Two DOCAs are equivalent if the set of words accepted by both of them are the same. Valiant and Paterson (1975) proved that the equivalence of DOCAs is decidable, and an analysis of their proof shows that it is in PSPACE. Based on these ideas, Böhm et al. (2013) conducted a thorough analysis, which proved that the equivalence of DOCAs is NL-complete.

Theorem 2.10 (Theorem 3, Böhm et al. (2013))

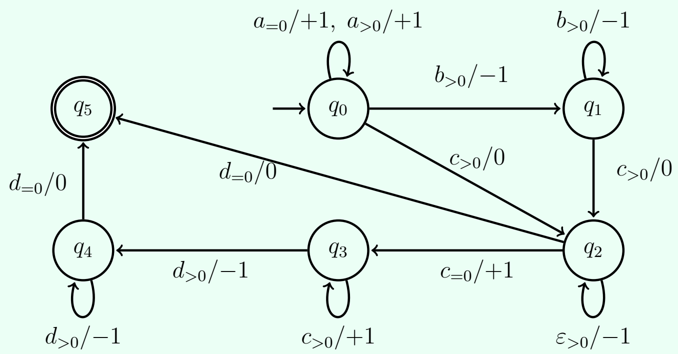
Equivalence of deterministic one-counter automata is NL-complete.

Example 2.11 (LeadMatch)

A DOCA recognising the language LeadMatch as follows

$$\texttt{LeadMatch} = \{a^m b^n\ c^t d^t\ \mid m, n, t \in \mathbb{N} \text{ and } m > n, t > 0\}.$$

Assume that all transitions not shown in the figure go to a non-final sink state while incrementing the counter by 1. Note that you cannot have a transition on $a_{>0}$ or $b_{>0}$ from q_2 since the machine is deterministic.



2.4.1.3 Deterministic Real-Time One-Counter Automata

Definition 2.12 (Deterministic real-time one-counter automata)
A deterministic real-time one-counter automaton (DROCA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta_0, \delta_1, F)$, where

- Q is a finite nonempty set of states,
- Σ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta_0 : Q \times \Sigma \to Q \times \{0, +1\}$ and $\delta_1 : Q \times \Sigma \to Q \times \{0, +1, -1\}$ are the transition functions, and
- $F \subseteq Q$ is the set of final states.

In a deterministic real-time one-counter automaton (DROCA), the transition function is deterministic, meaning each state and counter value pair has a unique transition on reading a symbol. Additionally, "real-time" indicates that there are no ε-transitions in this system.

We use $|\mathcal{A}|$ to denote the size of $\mathcal{A}$, which we consider to be $|Q|$. A configuration c of a DROCA is a pair $(q, n) \in Q \times \mathbb{N}$, where $q \in Q$ denotes the current state and $n \in \mathbb{N}$ is the counter value. The configuration $c_0 = (q_0, 0)$ is called the *initial configuration* of $\mathcal{A}$. Let $p, q \in Q, n \in \mathbb{N}, e \in \{-1, 0, +1\}$ and $\sigma \in \Sigma$. A *transition* between two configurations (p, n) and $(q, n + e)$ on the symbol σ is defined, if $\delta_{sign(n)}(p, \sigma) = (q, e)$. We use $(p, n) \xrightarrow{\sigma} (q, n + e)$ to denote this. A *run* over a word $w = \sigma_1 \ldots \sigma_{n-1}$ from a configuration (p_1, m_1) is the sequence of transitions $(p_1, m_1) \xrightarrow{\sigma_1} (p_2, m_2) \xrightarrow{\sigma_2} (p_3, m_3) \xrightarrow{\sigma_3} \ldots \xrightarrow{\sigma_{n-1}} (p_n, m_n)$ where $p_i \in Q$ and $m_i \in \mathbb{N}$. In this case, we denote the run by $(p_1, m_1) \xrightarrow{w} (p_n, m_n)$. Note that the counter values always stay non-negative during a run, implying that you cannot perform a decrement operation on the counter from a configuration with zero counter value. The run $(p_1, m_1) \xrightarrow{\sigma_1} (p_2, m_2) \xrightarrow{\sigma_2} (p_3, m_3) \xrightarrow{\sigma_3} \ldots \xrightarrow{\sigma_{n-1}} (p_n, m_n)$ is called *floating* if for all $i \in [1, n-1], m_i > 0$. The run is called *non-floating* otherwise.

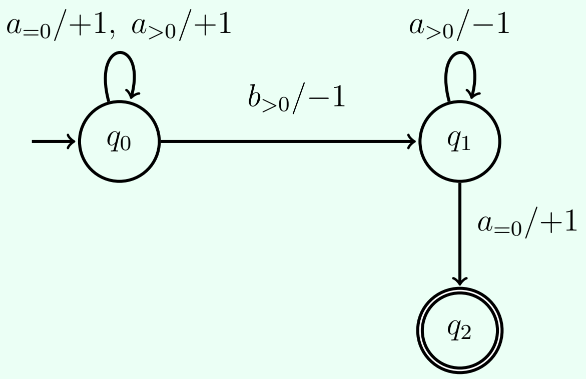
Let $q \in Q, n \in \mathbb{N}$, and $w \in \Sigma^*$ with $(q_0, 0) \xrightarrow{w} (q, n)$. We use $\mathsf{ce}_{\mathcal{A}}(w) = n$ to denote the counter value reached on reading w from the initial configuration and call it the *counter-effect* of w. We denote by $height_{\mathcal{A}}(w)$ the maximal counter-effect of the prefixes of w in $\mathcal{A}$. We drop the subscript $\mathcal{A}$ when the DROCA under consideration is evident. A word w is *accepted* by $\mathcal{A}$ if and only if $(q_0, 0) \xrightarrow{w} (q_f, m)$ for some $q_f \in F$ and $m \in \mathbb{N}$. The language of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of all words accepted by $\mathcal{A}$. For convenience, we use $\mathcal{A}(w) = 1$ if $w \in \mathcal{L}(\mathcal{A})$ and $\mathcal{A}(w) = 0$ otherwise. Two configurations $\mathtt{c}_1$ and $\mathtt{c}_2$ of a DROCA are not equivalent to each other if there exists $w \in \Sigma^*$ with $\mathtt{c}_1 \xrightarrow{w} (p_1, m_1)$, $\mathtt{c}_2 \xrightarrow{w} (p_2, m_2)$ for some $p_1, p_2 \in Q$ and $m_1, m_2 \in \mathbb{N}$ such that exactly one among p_1 and p_2 is a final state. We use $\mathtt{c}_1 \not\equiv \mathtt{c}_2$ to denote this. Otherwise, we say that $\mathtt{c}_1$ and $\mathtt{c}_2$ are equivalent and use $\mathtt{c}_1 \equiv \mathtt{c}_2$ to denote it.

Example 2.13 (`FlipBalance`)

A DROCA recognising the language `FlipBalance` defined as

$$\texttt{FlipBalance} = \{a^n b a^n \mid n > 0\}.$$

Assume that all transitions not shown in the figure go to a non-final sink state while incrementing the counter by 1.



Note that there are no ε-transitions in a DROCA. Given two DROCAs $\mathcal{A}$ and $\mathcal{B}$, we say that they are equivalent if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$. The equivalence of DROCAs was shown to be NL-complete by Böhm and Göller (2011).

Theorem 2.14 (Theorem 3, Böhm and Göller (2011))

Equivalence of deterministic real-time one-counter automata is NL-complete.

Now, we define the configuration graph of a DROCA up to a given counter value. Recall that in a DROCA there are no ε-transitions.

Definition 2.15 (Configuration graph up to counter value k)

Given a DROCA $\mathcal{A} = (Q, \Sigma, q_0, \delta_0, \delta_1, F)$ and a counter value k, the *configuration graph* of $\mathcal{A}$ up to counter value k is a DFA $G_{\mathcal{A}} = (Q', \Sigma, q_{init}, \Delta, F')$, where

- $Q' = \{(q, n) \mid n \leq k\}$, is the set of states.
- $q_{init} = (q_0, 0)$ is the initial state.
- $\Delta : Q' \times \Sigma \to Q'$ is the transition function defined as follows: For $q, q' \in Q$, $n \in \mathbb{N}$, and $d \in \{-1, 0, 1\}$

$$\Delta((q,n),a) = (q', n+d), \text{ if } n = 0 \text{ and } \delta_0(q,a) = (q', d) \text{ or}$$
$$\text{if } n > 0 \text{ and } \delta_1(q,a) = (q', d).$$

- $F' = \{(q, n) \mid n \leq k \text{ and } q \in F\}$, is the set of final states.

We now define the notion of counter-synchronous DROCAs. Two DROCAs are *counter-synchronous* if, for any word, the counter values reached by both the DROCAs on reading that word are the same.

Definition 2.16 (Counter-synchronous DROCAs)

Two DROCAs $\mathcal{A}$ and $\mathcal{B}$ are *counter-synchronised* if for all $w \in \Sigma^*$, $\mathsf{ce}_{\mathcal{A}}(w) = \mathsf{ce}_{\mathcal{B}}(w)$.

Given a DROCA $\mathcal{A}$, the DROCA $\mathcal{A}^c$ obtained by making its final states as non-final and its non-final states as final will give us a DROCA that is counter-synchronous with $\mathcal{A}$. However, $\mathcal{A}$ and $\mathcal{A}^c$ are not equivalent. Given two DROCAs $\mathcal{A}$ and $\mathcal{B}$ that are counter-synchronous and equivalent, we call $\mathcal{B}$ a *minimal counter-synchronous* DROCA with respect to $\mathcal{A}$, if $\mathcal{B}$ has the minimal number of states among all DROCAs that are equivalent and counter-synchronous to $\mathcal{A}$. Figure 2.2 shows two DROCAs that are counter-synchronous and equivalent. These DROCAs

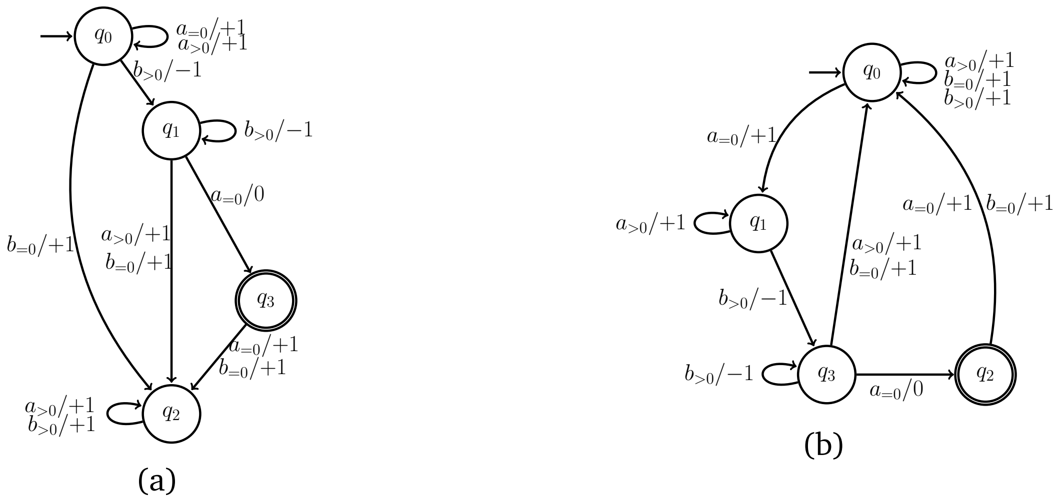


Fig. 2.2. Two DROCAs that are counter-synchronous and equivalent recognising the language $\texttt{MatchAB} = \{a^n b^n a \mid n > 0\}$.

recognise the language $\{a^n b^n a \mid n > 0\}$ and contains the minimal number of states among all counter-synchronous DROCAs recognising this language. However, they are not isomorphic to one another.

2.4.1.4 Visibly One-Counter Automata

Definition 2.17 (Visibly one-counter automata)

A visibly one-counter automaton (VOCA) is a tuple $\mathcal{A} = (Q, \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}, q_0, \delta_0, \delta_1, F)$, where

- Q is a finite nonempty set of states,
- $\Sigma = \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta_0 : Q \times \Sigma \to Q \times \{0, +1\}$ and $\delta_1 : Q \times \Sigma \to Q \times \{0, +1, -1\}$ are the transition functions, and
- $F \subseteq Q$ is the set of final states.

A VOCA is a DROCA where the input alphabet Σ can be written as a union of three disjoint sets $\Sigma_{call}, \Sigma_{ret}$, and Σ_{int}. The VOCA has to increment (resp. decrement) its counter on reading a symbol from Σ_{call} (resp. Σ_{ret}). The counter value remains unchanged on reading a symbol from Σ_{int}. We call $(\Sigma_{call}, \Sigma_{ret}, \Sigma_{int})$ the *pushdown alphabet*. The notions of counter-effect, runs and transitions for VOCAs remain the same as that of DROCAs. Note that we have defined VOCAs to be deterministic.

In the case of VOCAs, we use $\mathtt{ce}(w)$ to denote the effect on the counter on reading a word $w \in \Sigma^*$. The counter-effect of a transition is solely based on the input alphabet, making the starting state and counter value irrelevant.

$$\text{For } \sigma \in \Sigma, \mathtt{ce}(\sigma) = \begin{cases} 1, & \text{if } \sigma \in \Sigma_{call} \\ -1, & \text{if } \sigma \in \Sigma_{ret}, \text{ and} \\ 0, & \text{if } \sigma \in \Sigma_{init} \end{cases}$$

If $w = \sigma_1\sigma_2 \ldots \sigma_n$ for some $n \in \mathbb{N}$ and $\sigma_1, \ldots, \sigma_n \in \Sigma$, then $\mathtt{ce}(w) = \mathtt{ce}(\sigma_1) + \mathtt{ce}(\sigma_2) + \ldots + \mathtt{ce}(\sigma_n)$. Since the counter value can never go below zero in a one-counter automaton, there will be words that do not have a valid run in a VOCA. These words will be considered as rejected by the VOCA. For any VOCA, there exists an equivalent DROCA. A VOCA recognising the language $\{a^n b^n a \mid n > 0\}$ defined over the pushdown alphabet $(\{a\}, \{b\}, \emptyset)$ is given in Example 2.19.

Since VOCAs are a subclass of DROCAs, their equivalence is also decidable in polynomial time, and hence we get the following corollary. The lower bound comes from the emptiness checking of DFAs.

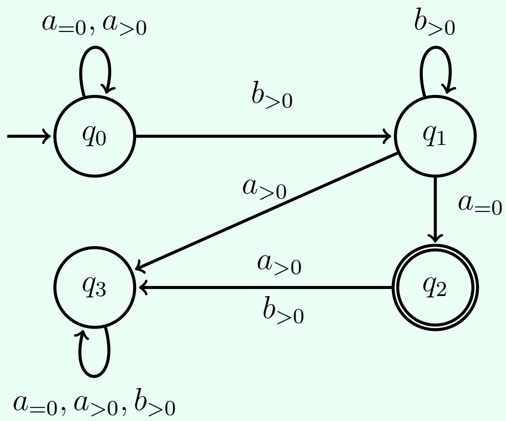
Theorem 2.18
Equivalence of visibly one-counter automata is NL-complete.

Similar to that of DROCA*s*, VOCA*s* also have two sets of transition functions: one for counter value zero and the other for positive counter value. The transition function is chosen based on a zero test on the counter value. The notion of acceptance in a VOCA is with a final state. One can also think of a model where the notion of acceptance is with a final state and counter value zero. There are languages that VOCA*s* accept with final state that cannot be recognised by any VOCA that accept with final state and zero counter value (eg., $\{a^n b^m \mid m < n\}$).

Also, there are languages recognised by VOCAs that accept with final state and counter value zero that cannot be recognised by any VOCA that accept with final state (e.g., $\{a^n b^n \mid n > 0\}$). The equivalence results that we present for VOCAs will hold true even if the notion of acceptance is with a final state and counter value zero.

Example 2.19 (`MatchAB`)

A VOCA recognising the language `MatchAB` $= \{a^n b^n a \mid n > 0\}$ is given in the figure below. The counter value is always incremented on reading an a and is decremented on reading a b. If a transition from state q_i to q_j is labelled with $\sigma_{=0}$ (resp. $\sigma_{>0}$) for some $\sigma \in \Sigma$, then it can be taken upon reading the symbol σ from state q_i when the current counter value is zero (resp. positive).



Visibly one-counter automata (VOCAs) are a subclass of visibly pushdown automata introduced by Alur and Madhusudan (2004) and were first studied by Bárány et al. (2006). Now, we compare our model with the definition of visibly one-counter automata with a threshold m (called m-VCAs) by Bárány et al. (2006), where the machine can test for counter value up to m. The notion of acceptance by m-VCAs is by final state and counter value zero. As stated in Bárány et al. (2006), for any m $\in \mathbb{N}$, (m+1)-VCAs are more expressive than m-VCAs. A VOCA that accepts with final state and counter value zero is equivalent to a 0-VCA and is less expressive than m-VCAs with $m > 0$. For instance, the language $\{a^n b^n \mid n > 0\}$ cannot be recognised by a VOCA. Also, there are languages recognised by a VOCA that accept with final state that cannot be recognised by any m-VCA (e.g., $\{a^n b^m \mid m < n\}$).

2.4.2 Pushdown Automata

A pushdown automaton (PDA) is a computational model that extends finite automata by using a stack as an auxiliary storage device. This additional storage space allows them to recognise the broader class of context-free languages. A one-counter automaton can be seen as a PDA that can store only one type of symbol in the stack.

PDAs can be classified into nondeterministic and deterministic variants based on their transition functions. We briefly discuss both these models.

2.4.2.1 Nondeterministic Pushdown Automata

Definition 2.20 (Nondeterministic pushdown automata)

A nondeterministic pushdown automaton (NPDA) is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$, where

- Q is a finite set of states,
- Σ is the finite set of input alphabet,
- Γ is the finite set of stack alphabet,
- $\delta : Q \times \left(\Sigma \cup \{\varepsilon\}\right) \times \Gamma \to 2^{Q \times \Gamma^*}$ is the transition function,
- $q_0 \in Q$ is the initial state,
- $\perp \in \Gamma$ is the unique bottom of the stack symbol, and
- $F \subseteq Q$ is the set of final states.

A configuration of an NPDA is of the form $(q, z) \in Q \times \Gamma^*$ that denotes the current state and current contents of the stack. The configuration $(q_0, \perp)$ is the initial configuration of $\mathcal{A}$. Given a configuration $(q, z\gamma)$ for some $q \in Q, \gamma \in \Gamma$ and $z \in \Gamma^*$, and $\sigma \in \Sigma \cup \{\varepsilon\}$, there is transition from the configuration $(q, z\gamma)$ to another configuration (q', zz') for some $z' \in \Gamma^*$ on reading the symbol σ, if and only if $(q', z') \in \delta(q, \sigma, \gamma)$. We denote this by $(q, z\gamma) \xrightarrow{\sigma} (q', zz')$. A run of a word starting from a configuration c_0 and ending in c_n on reading the word w (denoted as $c_0 \xrightarrow{w} c_n$) is specified by the transition rules that leads from c_0 to c_n and in the process, reads the word w. Note that in an NPDA, a word can have multiple runs. A word w is accepted by an NPDA if there exist a run $(q_0, \perp) \xrightarrow{w} (q_F, z)$, for some $q_F \in F$ and $z \in \Gamma^*$. The language of an NPDA is the set of all words accepted by it.

2.4.2.2 Deterministic Pushdown Automata

Definition 2.21 (Deterministic pushdown automata)

A deterministic pushdown automaton (DPDA) is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$, where

- Q is a finite set of states,
- Σ is the finite set of input alphabet,
- Γ is the finite set of stack alphabet
- $\delta : Q \times \big(\Sigma \cup \{\varepsilon\}\big) \times \Gamma \to Q \times \Gamma^*$ is the transition function,
- $q_0 \in Q$ is the initial state,
- $\perp \in \Gamma$ is the unique bottom of the stack symbol, and
- $F \subseteq Q$ is the set of final states.

A DPDA is a pushdown automaton where the transition function is deterministic. i.e., for every state, input symbol and stack symbol at most one transition is possible. Also, $\delta(q, \varepsilon, \gamma)$ is defined for some $q \in Q$ and $\gamma \in \Gamma$ only if $\delta(q, \sigma, \gamma)$ is not defined for all $\sigma \in \Sigma$. The stack alphabet represents the finite set of symbols that a DPDA can store in the stack.

A configuration of a DPDA is of the form $(q, z) \in Q \times \Gamma^*$ that denotes the current state and current contents of the stack. The configuration $(q_0, \perp)$ is the initial configuration of $\mathcal{A}$. Given a configuration $(q, z\gamma)$ for some $q \in Q, \gamma \in \Gamma$

and $z \in \Gamma^*$, and $\sigma \in \Sigma \cup \{\varepsilon\}$, there is transition from the configuration $(q, z\gamma)$ to another configuration (q', zz') for some $z' \in \Gamma^*$ on reading the symbol σ, if and only if $\delta(q, \sigma, \gamma) = (q', z')$. We denote this by $(q, z\gamma) \xrightarrow{\sigma} (q', zz')$. Let $w = a_1, \ldots a_n$, where $a_i \in \Sigma \cup \{\varepsilon\}$ be a word. There is a run on w starting from a configuration c_0 and ending in a configuration c_n, if there exists configurations $c_1, \ldots, c_{n-1}$ such that $c_{i-1} \xrightarrow{a_i} c_i$ for all $i \in [1, n]$. We use $c_0 \xrightarrow{w} c_n$ to denote this. A word w is accepted by a DPDA if $(q_0, \perp) \xrightarrow{w} (q_F, z)$, for some $q_F \in F$ and $z \in \Gamma^*$. The language of a DPDA is the set of all words accepted by it.

2.4.3 Expressive Power and Comparisons

We now compare the expressive power of NPDAs, DPDAs, NOCAs, DOCAs, DROCAs, VOCAs, and DFAs. Let $\mathcal{L}(\text{NPDA}s), \mathcal{L}(\text{DPDA}s), \mathcal{L}(\text{NOCA}s), \mathcal{L}(\text{DOCA}s), \mathcal{L}(\text{DROCA}s), \mathcal{L}(\text{DOCA}s)$ and $\mathcal{L}(\text{DFA}s)$ respectively denote the set of languages recognised by NPDAs, DPDAs, NOCAs, DOCAs,DROCAs, VOCAs, and DFAs respectively.

Theorem 2.22

1. $\mathcal{L}(\text{DFA}s) \subsetneq \mathcal{L}(\text{VOCA}s) \subsetneq \mathcal{L}(\text{DROCA}s) \subsetneq \mathcal{L}(\text{DOCA}s) \subsetneq \mathcal{L}(\text{DPDA}s)$,

2. $\mathcal{L}(\text{DOCA}s) \subsetneq \mathcal{L}(\text{NOCA}s) \subsetneq \mathcal{L}(\text{NPDA}s)$ and $\mathcal{L}(\text{DPDA}s) \subsetneq \mathcal{L}(\text{NPDA}s)$.

Proof. We first prove that $\mathcal{L}(\text{DFA}s) \subsetneq \mathcal{L}(\text{VOCA}s) \subsetneq \mathcal{L}(\text{DROCA}s) \subsetneq \mathcal{L}(\text{DOCA}s) \subsetneq \mathcal{L}(\text{DPDA}s)$. We prove these inclusions individually.

Claim 1. $\mathcal{L}(\text{DFA}s) \subsetneq \mathcal{L}(\text{VOCA}s)$.

Proof. By definition, every DFA is a VOCA. A DFA can be seen as a VOCA that does not use the counter. Whereas, we show that the language $\texttt{MatchAB} = \{a^n b^n a \mid n > 0\}$ can be recognised by a VOCA (see Example 2.19), but not by a DFA. This can be proved using a pumping argument. Assume for contradiction that there is $k > 0$ and a DFA $\mathcal{A}$ with k states that recognises this language. Consider the word $w = a^{k+1} b^{k+1}$. The word w is accepted by this DFA. Now consider the run of this word w on $\mathcal{A}$. Since the DFA has only k states, by the pigeon-hole principle, at least one state repeats during the run of the word a^{k+1} in $\mathcal{A}$. Let q be this state. The run of the word w can be written as $q_0 \xrightarrow{a^{i_1}} q \xrightarrow{a^{i_2}} q \xrightarrow{a^{i_3}b^{k+1}a} q_F$, where $i_2 > 0$,$i_1 + i_2 + i_3 = k + 1$, q_0 is the initial state and q_F is the final state of $\mathcal{A}$. The words $a^{i_1}a^{i_3}b^{k+1}a$ will also be accepted by this DFA. However, since $i_1 + i_3 < k+1$, this word is not in the language `MatchAB`. This contradicts our initial assumption that there is a DFA with k states that recognises the language `MatchAB`. $\square_{Claim:1}$

Claim 2. $\mathcal{L}$(VOCAs) $\subsetneq$ $\mathcal{L}$(DROCAs).

Proof. By definition, every VOCA is a DROCA. We show that there exists languages recognised by a DROCA that cannot be recognised by a VOCA. Consider the language `FlipBalance` $= \{a^n b a^n \mid n > 0\}$. A DROCA recognising this language is given in Example 2.13. We show that this language cannot be recognised by a VOCA.

Assume for contradiction that this language is recognised by a VOCA $\mathcal{A}$ with k states. Consider the the word $w = a^{k+2}ba^{k+2}$ accepted by $\mathcal{A}$. There are three cases to consider here.

Case -1: The VOCA decrements its counter on reading an a.
This is not possible since the VOCA will have to decrement its counter from counter value zero on reading the first a in w. Since the counter cannot go below zero, the machine cannot decrement its counter on reading an a.

Case-2: The VOCA doesnot modify its counter on reading an a.
Since the machine doesnot modify its counter on reading an a, there is a configuration with counter value zero that occurs twice during the run of the prefix a^{k+1} of w from the initial configuration c_0 of $\mathcal{A}$. The run on the word w in $\mathcal{A}$ can be written as $c_0 \xrightarrow{a^{i_1}} (q, 0) \xrightarrow{a^{i_2}} (q, 0) \xrightarrow{a^{i_3}ba^{k+2}} (q_F, m)$ where $i_2 > 0$, $i_1 + i_2 + i_3 = k+2$, $m \in \mathbb{N}$ and q_F is a final state of $\mathcal{A}$. Now consider the word $a^{i_1}a^{i_3}ba^{k+2}$. This word will also be accepted by this VOCA, since its run on $\mathcal{A}$ reaches a final state. However, it is not in the language `FlipBalance`. This contradicts our assumpton that $\mathcal{A}$ recognises the language `FlipBalance`.

Case-3: The VOCA increments its counter on reading an a.
Since the machine increments its counter on reading an a, there is a state that occurs twice during the run of the word a^{k+1} from the initial configuration c_0 of $\mathcal{A}$. The run on the word w in $\mathcal{A}$ can be written as $c_0 \xrightarrow{a^{i_1}} (q, m_1) \xrightarrow{a^{i_2}} (q, m_1 + d) \xrightarrow{a^{i_3}ba^{k+2}} (q_F, m_2)$ where $i_2 > 0$, $i_1 + i_2 + i_3 = k + 2$, $d < k, m_1, m_2 \in \mathbb{N}$

and q_F is a final state of $\mathcal{A}$. Now consider the word $a^{i_1}a^{i_3}ba^{k+2}$. Theis word is also accepted by this VOCA, since its run on $\mathcal{A}$ is $c_0 \xrightarrow{a^{i_1}} (q, m_1) \xrightarrow{a^{i_3}ba^{k+2}} (q_F, m_2 - d)$ also reaches a final state. However, this word is not in the language `FlipBalance`. This contradicts our assumpton that $\mathcal{A}$ recognises the language `FlipBalance`. Therefore, no matter what pushdown alphabet we choose, a VOCA cannot recognise the given language `FlipBalance`. $\square_{Claim:2}$

Claim 3. $\mathcal{L}$(DROCAs) $\subsetneq$ $\mathcal{L}$(DOCAs).

Proof. By definition, every DROCA is a DOCA. We now give an example of a language recognised by a DOCA, but not by any DROCA. Consider the language `LeadMatch` $= \{a^m b^n\ c^t d^t \ \mid m, n, t \in \mathbb{N}$ and $m > n, t > 0\}$. A DOCA recognising this language is given in Example 2.11. We prove that a DROCA cannot recognise this language.

Assume for contradiction that some DROCA $\mathcal{A}$ with k states recognise the language `LeadMatch`. Let $w = a^{5k^2} b^{5k^2-1} cd$. The word $w \in$ `LeadMatch`, is accepted by $\mathcal{A}$. Let c_0 denote the initial configuration of $\mathcal{A}$ and $\pi = c_0 \xrightarrow{w} (q_F, m)$ denote the run of w in $\mathcal{A}$, where q_F is a final state of $\mathcal{A}$ and $m \in \mathbb{N}$.

First, we prove that $height_{\mathcal{A}}(a^{5k^2}) > 4k$. Assume for contradiction that $height_{\mathcal{A}}(a^{5k^2}) \leq 4k$. Since there are only $4k^2$ distinct configurations of $\mathcal{A}$ with counter values less than or equal to $4k$, by the pigeon-hole principle, there is a configuration that repeats during the run of the word a^{5k^2} from c_0. The subword between those configurations can be removed to get a word that is accepted by $\mathcal{A}$ but not in the given language. This is a contradiction. Therefore, $height_{\mathcal{A}}(a^{5k^2}) > 4k$.

Since $height_{\mathcal{A}}(a^{5k^2}) > 4k$, there exist an $t \leq 5k^2$ such that $c_0 \xrightarrow{a^t} (p_1, 4k+1)$ for some state p_1 of $\mathcal{A}$. Consider the run of the word $w' = a^t c^{k+1} d^{k+1}$ in $\mathcal{A}$. We already know that $c_0 \xrightarrow{a^t} (p_1, 4k+1)$. Therefore, no configuration with counter value less than $2k-1$ is encountered during the run $(p_1, 4k+1) \xrightarrow{c^k d^k} (q''_F, m_1)$ for some final state q''_F of $\mathcal{A}$ and $m_1 \in \mathbb{N}$ with $m_1 \geq 2k-1$. Let $e_0, e_1, \ldots, e_k$ respectively denote the configurations reached on reading $\varepsilon, c, \ldots, c^k$ from $(p_1, 4k+1)$. By the pigeon-hole principle, there exists $i, j \in [0, k]$, such that configurations e_i and e_j have the same state. The sub-run between these configurations can be removed to get an accepting run for the word $a^t c^{k-(j-i)} d^k$, that is not in the language `LeadMatch`.

This contradicts our initial assumption that $\mathcal{A}$ recognises the language `LeadMatch`. Therefore, there is no DROCA that recognises the language `LeadMatch`.

$\square_{Claim:3}$

Claim 4. $\mathcal{L}(\text{DOCA}s) \subsetneq \mathcal{L}(\text{DPDA}s)$.

Proof. By definition every DOCA is a DPDA. Since DOCAs are deterministic pushdown automata (DPDAs) with a singleton stack alphabet, any language recognised by a DOCA can be recognised by a pushdown automata. We now prove that DOCAs are less expressive than DPDAs. Consider following the language `MatchTwice` $= \{a^m b^n c^n d^m \mid m, n > 0\}$. This language can be recognised by a pushdown automaton by using the stack to check whether the number of a and d (resp. b and c) are equal. However, prove that this language cannot be recognised by a DOCA because the counter cannot keep track of the number of as and the number of bs simultaneously. We provide a formal proof below.

Assume for contradiction that there is a DOCA $\mathcal{A}$ with k states recognising this language. Let $w = a^{3k^3+1} b^{4k^4} c^{4k^4} d^{3k^3+1}$. The word w is accepted by $\mathcal{A}$ since it is in the given language. Consider the run $\pi = c_0 \xrightarrow{w} (q_f, t)$ in $\mathcal{A}$ where c_0 is the intial configuration of $\mathcal{A}$, $t \in \mathbb{N}$ and q_f is a final state of $\mathcal{A}$. Let $\pi = c_0 \xrightarrow{a^{3k^3+1}} (p_3, m') \xrightarrow{b^{4k^4} c^{4k^4} d^{3k^3+1}} (q_f, t)$ where p_3 is a state of $\mathcal{A}$ and $m' \in \mathbb{N}$. Consider the run $\pi' = c_0 \xrightarrow{a^{3k^3+1}} (p_3, m')$. We first prove that $m' > 2k^2$.

Assume for contradiction that $m' \leq 2k^2$. Since $\mathcal{A}$ has only k states, the maximum counter value encountered during the run π' must be greater than $3k^2$. If not, then by the pigeon-hole principle, there is a configuration that repeats during this run, and the subword between those configurations can be pumped to get a word that is accepted by $\mathcal{A}$ but not in the given language. Observe that π does not have loops on ε-transitions that increases the counter value. If it does, then it will get stuck in that loop and will never be able to get out. The run π' can be written as $c_0 \xrightarrow{a^{i_1}} (p_1, m') \xrightarrow{a^{i_2}} (p_2, 3k^2 + 1) \xrightarrow{a^{i_3}} (p_3, m')$, where p_1, p_2 are states of $\mathcal{A}$ and $i_1, i_2, i_3 \in \mathbb{N}$ such that $i_1 + i_2 + i_3 = 3k^3 + 1$. For any $i \in [m', 3k^2 + 1]$, we denote by e_i and e'_i the configurations with counter value i that is encountered for the last (resp. first) time before (resp. after) reaching counter value $3k^2 + 1$ during the run π'. Consider the pairs of configurations $(e_{m'}, e'_{m'}), (e_{m'+1}, e'_{m'+1}), \ldots, (e_{3k^2}, e'_{3k^2})$. Since $m' \leq 2k^2$, by the pigeonhole principle, there exist two states r, s of $\mathcal{A}$, and indices $i, j \in [m', 3k^2]$ such that, the states of e_i and e_j is r and the states of e'_i and e'_j is s. Let $j_1, j_2, j_3, j_4, j_5 \in \mathbb{N}$ such that $\pi' = c_0 \xrightarrow{a^{i_1}} (p_1, m') \xrightarrow{a^{j_1}} e_i \xrightarrow{a^{j_2}} e_j \xrightarrow{a^{j_3}} e'_j \xrightarrow{a^{j_4}} e'_i \xrightarrow{a^{j_5}} (p_3, m')$. We can now remove the subruns $e_i \xrightarrow{a^{j_2}} e_j$ and $e'_j \xrightarrow{a^{j_5}} e'_i$ to get a run $\pi' = c_0 \xrightarrow{a^{i_1}} (p_1, m') \xrightarrow{a^{j_1}} e_i \xrightarrow{a^{j_3}} e'_i \xrightarrow{a^{j_5}} (p_3, m')$. Since $(p_3, m') \xrightarrow{b^{4k^4} c^{4k^4} d^{3k^3+1}} (q_f, t)$, the word $a^{i_1+j_1+j_3+j_5} b^{4k^4} c^{4k^4} d^{3k^3+1}$ is also accepted by $\mathcal{A}$. Since $i_1 + j_1 + j_3 + j_5 < 3k^3 + 1$, it contradicts our assumption that $\mathcal{A}$ recognises the language `MatchTwice`. Therefore, $m' > 2k^2$.

Using a similar argument, we can show that $\pi'' = c_0 \xrightarrow{a^{3k^3+1}} (p_3, m') \xrightarrow{b^{4k^4}} (p_4, m'')$ is a run that reaches a state p_4 with counter value $m'' > 4k^3$ and no configuration with counter value less than or equal to k^2 is encountered during the run $(p_3, m') \xrightarrow{b^{4k^4}} (p_4, m'')$. The run π can hence be written as $\pi = c_0 \xrightarrow{a^{3k^3+1} b^{4k^4}} (p_4, m'') \xrightarrow{c^{i_1}} (p_5, m'' - k^2) \xrightarrow{c^{i_2} d^{3k^3+1}} (q_f, t)$ for some $i_1, i_2 \in \mathbb{N}$ and state p_5 of $\mathcal{A}$ such that $i_1 + i_2 = 4k^4$ and no configuration with counter value greater than or equal to $m'' - k^2$ is encountered during the run $(p_5, m'' - k^2) \xrightarrow{c^{i_2} d^{3k^3+1}} (q_f, t)$. If there is no such i_1, i_2, then again, using an argument similar to the one used above, we can show that $\mathcal{A}$ accepts a word that is not in the language `MatchTwice`.

Now, let $e_0, \ldots, e_{k^2}$ respectively denote the last configurations with counter value $0, \ldots, k^2$ encountered during the sub-run $c_0 \xrightarrow{a^{3k^3+1}} (p_3, m')$ and $e'_{k^2}, \ldots, e'_0$ respectively denote the first configurations with counter values $m'', \ldots, m'' - k^2$ encountered during the run $(p_4, m'') \xrightarrow{c^{i_1}} (p_5, m'' - k^2)$. By the pigeonhole principle, there exist two states r, s of $\mathcal{A}$, and indices $i, j \in [0, k^2]$ such that, the states of e_i and e_j is r and the states of e'_i and e'_j is s. Let $j_1, j_2, j_3, j_4, j_5, j_6 \in \mathbb{N}$ such that π can be written as $\pi = c_0 \xrightarrow{a^{j_1}} e_i \xrightarrow{a^{j_2}} e_j \xrightarrow{a^{j_3} b^{4k^4}} (p_4, m'') \xrightarrow{c^{j_4}} e'_j \xrightarrow{c^{j_5}} e'_i \xrightarrow{c^{j_6} d^{3k^3+1}} (q_f, t)$, where $j_1 + j_2 + j_3 = 3k^3 + 1$ and $j_4 + j_5 + j^6 = 4k^4$. By removing the portions $e_i \xrightarrow{a^{j_2}} e_j$ and $e'_j \xrightarrow{c^{j_5}} e'_i$ from this run, we get that the word $w' = a^{j_1+j_3} b^{4k^4} c^{j_4+j_6} d^{3k^3+1}$ has a run $c_0 \xrightarrow{w'} (q_f, t)$. However, w' is not in the language `MatchTwice`. This contradicts our assumption that $\mathcal{A}$ recognises the language `MatchTwice`. Therefore, we can conclude that the language `MatchTwice` cannot be recognised by a DOCA. $\square_{Claim:4}$

Item 1 follows from Claims 1 to 4.

Now we prove that $\mathcal{L}(\text{DOCA}s) \subsetneq \mathcal{L}(\text{NOCA}s) \subsetneq \mathcal{L}(\text{NPDA}s)$ and $\mathcal{L}(\text{DPDA}s) \subsetneq \mathcal{L}(\text{NPDA}s)$. We prove each of these inclusions one by one.

Claim 5. $\mathcal{L}(\text{DOCA}s) \subsetneq \mathcal{L}(\text{NOCA}s)$.

Proof. A DOCA, by definition, is also an NOCA. Therefore, every language recognised by a DOCA is recognised by an NOCA. However, we prove that DOCAs are less expressive than NOCAs. Consider the language $\texttt{MatchOrSkip} = \{a^n b^m c^k \mid n, m, k > 0 \text{ with } n \neq m \text{ or } m \neq k\}$. An NOCA recognising this function is given in Example 2.8. However, we show that this language cannot be recognised using a DOCA.

Assume for contradiction that there is a DOCA $\mathcal{A}$ that recognises the language $\texttt{MatchOrSkip}$. Since DOCAs are DPDAs, it is recognised by a DPDA also. Since DPDAs are closed under complementation (Sipser, 1997), the complement of the language $\texttt{MatchOrSkip}$ must also be recognised by a DPDA. Intersecting this complement language with the regular language $\{a^* b^* c^*\}$ should give us a context-free language (Hopcroft and Ullman, 1979).However, the language that we obtain is $\{a^t b^t c^t \mid t > 0\}$, which is a well-known non-context-free language. This is a contradiction. Therefore, there is no DPDA that recognises the language $\texttt{MatchOrSkip}$. Hence, there does not exist a DOCA that recognises this language. $\square_{Claim:5}$

Claim 6. $\mathcal{L}(\text{NOCA}s) \subsetneq \mathcal{L}(\text{NPDA}s)$.

Proof. By definition, every NOCA is an NPDA. Since NOCAs are nondeterministic pushdown automata (NPDAs) with a singleton stack alphabet, any language recognised by NOCAs can be recognised by NPDAs. Now, we show that NOCAs are less expressive than NPDAs. Consider the language $\texttt{MatchTwice} = \{a^m b^n c^n d^m \mid m, n > 0\}$. As we have observed in proof of Claim 4 in Theorem 2.22, there is a DPDA that recognises this language. Since every DPDA is an NPDA, this language is recognised by an NPDA also. However, there does not exist an NOCA that recognises this language.

Assume for contradiction that there is a NOCA $\mathcal{A}$ with k states recognising this language. Let $w = a^{3k^3+1} b^{4k^4} c^{4k^4} d^{3k^3+1}$. The word w is accepted by $\mathcal{A}$ since it is in the given language. Consider the run $\pi = c_0 \xrightarrow{w} (q_f, t)$ in $\mathcal{A}$ where c_0 is the intial configuration of $\mathcal{A}$, $t \in \mathbb{N}$ and q_f is a final state of $\mathcal{A}$. Since w is accepted by $\mathcal{A}$, such a run exists. We can assume that there are no loops on ε-transitions with inverse counter effects between two configurations with counter value zero during the run π. If there are such loops, then we can remove them to get a shorter run on w that reaches a final state. Now, similar to the proof of Claim 4, we can show that the $\mathcal{A}$ accepts a word that is not in the language `MatchTwice`. Hence, an NOCA cannot recognise the language `MatchTwice`. $\square_{Claim:6}$

Claim 7. $\mathcal{L}(\text{DPDA}s) \subsetneq \mathcal{L}(\text{NPDA}s)$.

Proof. A DPDA, by definition, is also an NPDA. Therefore, every language recognised by a DPDA is recognised by an NPDA. However, DPDAs are less expressive than NPDAs. Consider the language `MatchOrSkip` $= \{a^n b^m c^k \mid n, m, k > 0 \text{ with } n \neq m \text{ or } m \neq k\}$. An NOCA which is also an NPDA that recognises this language is given in Example 2.8. However, as proven in Claim 5, there is no DPDA that recognises this language. $\square_{Claim:7}$

Item 2 follows from Claim 5, Claim 6, and Claim 7. $\square$

The relation between the expressive power of these automata models is depicted in Figure 2.3.

$$\begin{tikzcd}
\mathcal{L}(\text{DFA}s) \arrow[r, "\subsetneq"] & \mathcal{L}(\text{VOCA}s) \arrow[r, "\subsetneq"] & \mathcal{L}(\text{DROCA}s) \arrow[r, "\subsetneq"] & \mathcal{L}(\text{DOCA}s) \arrow[r, "\subsetneq"] \arrow[dr, "\subsetneq"'] & \mathcal{L}(\text{DPDA}s) \arrow[dr, "\subsetneq"] & \\
& & & & \mathcal{L}(\text{NOCA}s) \arrow[r, "\subsetneq"] & \mathcal{L}(\text{NPDA}s)
\end{tikzcd}$$

Fig. 2.3. Expressive power of some automata models.

From the above proof it can be observed that $\mathcal{L}(\text{NOCA}s)$ and $\mathcal{L}(\text{DPDA}s)$ are incomparable. From the proof of Claim 5, we know that the language `MatchOrSkip` is recognised by an NOCA but not by any DPDA. Also, from the proof of Claim 4, we know that the language `MatchTwice` is recognised by a DPDA but not by any NOCA.

For additional examples on languages recognised by these machines, readers can refer to the Ph.D. thesis by Staquet (2024).

2.5 Learning Finite Automata

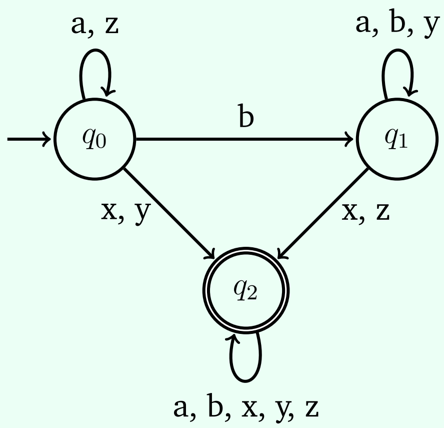
Automata learning focuses on constructing models from the observed behaviours of a system. Automata learning frameworks can be broadly divided into passive and active learning frameworks. We briefly discuss both of these below.

2.5.1 Passive Learning

In a passive learning framework, the aim is to identify a model based on a given set of observations. Here, we can't obtain any additional information regarding the system to aid us in the process of learning. The minimal separating DFA problem was one among the first to be considered. In the *minimal separating* DFA problem, two finite disjoint sets of words, Pos and Neg, are given as inputs. The aim is to find the minimal DFA that accepts all words in Pos and rejects all words in Neg (refer Example 2.23). This problem was shown to be NP-complete by Gold (1978).

Example 2.23

The minimal separating DFA for the set $Pos = \{x, ay, bz\}$ and $Neg = \{\varepsilon, a, b, az, by\}$.



Under the assumption that P $\neq$ NP, Pitt and Warmuth (1993) proved that given two disjoint sets of labelled samples Pos and Neg, and a constant $n \in \mathbb{N}$, no polynomial time algorithm can guarantee to produce a DFA with fewer than

K^n states that accepts all words in Pos and rejects all words in Neg, where K is the size of the minimal separating DFA for the sets Pos and Neg.

There are various passive automata learning techniques, such as the evidence-driven state merging (EDSM) algorithm and regular positive and negative inference (RPNI) algorithm, that in practice, identify a DFA of a reasonable size consistent with a given set of positive and negative samples in polynomial time.

2.5.2 Active Learning

The problem of identifying a minimal model that agrees with a given set of samples is difficult. Therefore, efficient learning algorithms assume that the learner has access to some additional information. Angluin (1987), proposed an active learning framework for learning DFAs that involves a learner and a teacher. The learner exactly learns a target language using polynomially many queries to the teacher. We will now discuss Angluin's algorithm (known as the $\mathtt{L}^*$ algorithm) for learning DFAs in detail.

$\mathtt{L}^*$ Algorithm

In an active learning framework, we have a learner and a teacher. The teacher is assumed to know a regular set R. The teacher can answer the following two types of queries by the learner.

1. *membership queries* MQ_R: the learner provides a word $w \in \Sigma^*$. The teacher returns 1 if $w \in R$, and 0 if $w \notin R$.

2. *equivalence queries* EQ_R: the learner provides a DFA $\mathcal{A}$ and asks whether $\mathcal{L}(\mathcal{A}) = R$. The teacher returns *yes* if they are the same. Otherwise, the teacher provides a counter-example $z \in \Sigma^*$ such that z is in exactly one among R and $\mathcal{L}(\mathcal{A})$.

The learner learns the regular language using polynomially many membership and equivalence queries with respect to the size of the longest counter example returned by the teacher and the number of states in the minimal DFA that accepts the language R. Towards this purpose, the learner maintains an observation table $C = (\mathcal{P}, \mathcal{S}, Mem)$ where $\mathcal{P} \subset \Sigma^*$ is a set of prefix-closed words, $\mathcal{S} \subset \Sigma^*$ is a set of suffix-closed words and $Mem : (\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S} \to \{0, 1\}$, is a function that maps every word in $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$ to its corresponding membership in R. i.e., for $w \in (\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$, $Mem(w) = 1$ if $w \in R$ and is 0 otherwise.

An observation table can be visualised as a table with rows indexed by $\mathcal{P} \cup \mathcal{P}\Sigma$ and columns indexed by $\mathcal{S}$. For $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, the entry in row p and column s denotes $Mem(ps)$. Given $p_1, p_2 \in \mathcal{P} \cup \mathcal{P}\Sigma$, we say that $row(p_1) = row(p_2)$ if and only if for all $s \in \mathcal{S}$, $Mem(p_1 s) = Mem(p_2 s)$.

Definition 2.24 (Closed)

An observation table is *closed* if for all $p' \in \mathcal{P}\Sigma$ there exists $p \in \mathcal{P}$, such that $row(p) = row(p')$. The observation table is said to be *not closed* otherwise.

Definition 2.25 (Consistent)

An observation table is *consistent* if for all $p_1, p_2 \in \mathcal{P}$, $row(p_1) = row(p_2)$ implies that for all $\sigma \in \Sigma$, $row(p_1\sigma) = row(p_2\sigma)$. We say that the observation table is *not consistent* otherwise.

Constructing a DFA from a closed and consistent observation table.

Given a closed and consistent observation table C, we define a corresponding DFA $M_C = (Q, \Sigma, q_0, \delta, F)$, where

- $Q = \{row(p) \mid p \in \mathcal{P}\}$ is the set of states,
- $q_0 = \{row(\varepsilon)\}$ is the initial state,
- Σ is the input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function, defined as follows:

$$\text{for all } p \in \mathcal{P} \text{ and } \sigma \in \Sigma,\ \ \delta(row(p), \sigma) = row(p\sigma).$$

- $F = \{row(p) \mid Mem(p) = 1\}$ is the set of final states.

The algorithm L*, proposed by Angluin for learning regular sets using queries and counter-examples is given in Algorithm 1.

Algorithm 1: L*: DFA Learning algorithm.

Require : The teacher knowing a regular language R
Ensure : A DFA accepting the language R is returned

Initialise $\mathcal{P}$ and $\mathcal{S}$ to $\{\varepsilon\}$.
Initialise the observation table $C = (\mathcal{P}, \mathcal{S}, Mem)$ using membership queries.
repeat
    while *C is not closed or not consistent* **do**
        if *C is not closed* **then**
            Find $p \in \mathcal{P}$, $\sigma \in \Sigma$ such that $row(p\sigma) \neq row(p')$ for all $p' \in \mathcal{P}$.
            Add $p\sigma$ to $\mathcal{P}$.
        end
        if *C is not consistent* **then**
            Find $p, q \in \mathcal{P}$, $\sigma \in \Sigma$, $s \in \mathcal{S}$ such that $row(p) = row(q)$, and $Mem(p\sigma s) \neq Mem(q\sigma s)$.
            Add σs to $\mathcal{S}$.
        end
        Extend Mem to $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$, using membership queries.
    end
    Construct a DFA M_C from C.
    Ask an equivalence query $\mathrm{EQ}_R(M_C)$.
    if *teacher gives a counter-example z* **then**
        Add z and all its prefixes to $\mathcal{P}$.
        Extend Mem to $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$ using membership queries.
    end
until *teacher replies yes to an equivalence query*;
Halt and output M_C.

To have a better understanding of how the L* algorithm works, let us look at an example where the learner learns a regular language from the teacher with the help of membership and equivalence queries, as discussed.

Example L*.

We assume that the teacher has the language $R = \{ba^* + a(a+b)^*b\}$ in their mind, and the learner uses the L* algorithm to learn this language. The learner initialises $\mathcal{P}$ and $\mathcal{S}$ to $\{\varepsilon\}$ and constructs the initial observation table $C = (\mathcal{P}, \mathcal{S}, Mem)$ (see Table 2.1) using membership queries. This observation table is not closed since for all $p \in \mathcal{P}$, $row(b) \neq row(p)$. Therefore, we add b to $\mathcal{P}$ to make the table closed and fill the observation table using membership queries. The observation table obtained after this step is shown in Table 2.2.

	ε
ε	0
a	0
b	1

Table 2.1. Initial observation table with $\mathcal{P} = \mathcal{S} = \{\varepsilon\}$.

	ε
ε	0
b	1
a	0
ba	1
bb	1

Table 2.2. Closed and consistent observation table with $\mathcal{P} = \{\varepsilon, b\}$ and $\mathcal{S} = \{\varepsilon\}$.

This observation table is both closed and consistent. Therefore, we construct the DFA shown in Figure 2.4 from this observation table.

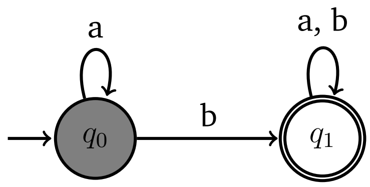


Fig. 2.4. DFA constructed from the observation table in Table 2.2.

The learner now asks an equivalence query EQ_R with this automaton as input. Since, this is not the right DFA, the teacher returns a counter-example. Let's assume that the counter-example returned by the teacher is aba. We add aba and all it prefixes (i.e., a, ab) to $\mathcal{P}$ and extend Mem to $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$ using membership queries. The observation table obtained after this step is shown in Table 2.3. However, this observation table is not consistent since $row(b) = row(a)$ but $b \cdot a \cdot \varepsilon \neq a \cdot a \cdot \varepsilon$. Therefore, we add a to $\mathcal{S}$ and extend Mem to $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$ using membership queries. The observation table obtained after this step is shown in Table 2.4.

	ε
ε	0
b	1
a	0
ab	1
aba	0
ba	1
bb	1
aa	0
abb	1
abaa	0
abab	1

Table 2.3. Observation table with $\mathcal{P} = \{\varepsilon, b, a, ab, aba\}$ and $\mathcal{S} = \{\varepsilon\}$ that is not consistent.

	ε	a
ε	0	0
b	1	1
a	0	0
ab	1	0
aba	0	0
ba	1	1
bb	1	1
aa	0	0
abb	1	0
abaa	0	0
abab	1	0

Table 2.4. Observation table with $\mathcal{P} = \{\varepsilon, b, a, ab, aba\}$ and $\mathcal{S} = \{\varepsilon, a\}$ that is not consistent.

This table is also not consistent since $row(\varepsilon) = row(a)$ but $\varepsilon \cdot b \cdot a \neq a \cdot b \cdot a$. Therefore, we add ba to $\mathcal{S}$ and extend Mem to $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$ using membership queries. The observation table obtained after this step is shown in Table 2.5. Each distinct colour in the table represents a distinct Myhill-Nerode equivalence class.

	ε	a	ba
ε	0	0	1
b	1	1	1
a	0	0	0
ab	1	0	0
aba	0	0	0
ba	1	1	1
bb	1	1	1
aa	0	0	0
abb	1	0	0
abaa	0	0	0
abab	1	0	0

Table 2.5. Closed and consistent observation table with $\mathcal{P} = \{\varepsilon, b, a, ab, aba\}$ and $\mathcal{S} = \{\varepsilon, a, ba\}$.

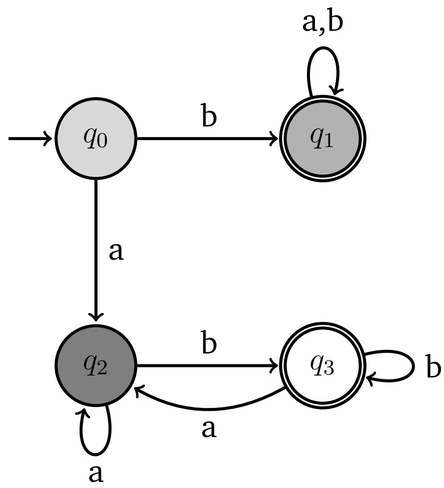


Fig. 2.5. DFA constructed from the observation table in Table 2.5.

This observation table is both closed and consistent. Therefore, the learner constructs the automaton shown in Figure 2.5 from this observation table and asks an equivalence query. Since this DFA accepts the language $R = ba^* + a(a + b)^*b$, the teacher replies *yes* to the equivalence query. The learner has now successfully learnt the target language, and the $\mathtt{L}^*$ algorithm outputs this learnt DFA and halts.

Motivated by the $\mathtt{L}^*$ algorithm, various other active learning algorithms were later proposed. The TTT Algorithm by Isberner et al. (2014) is such an algorithm that needs a lesser number of queries to learn a target language (also see Isberner (2015)). Recently, Vaandrager et al. (2022) proposed the $\mathtt{L}^\#$ algorithm that operates directly on tree-shaped automata rather than using observation tables. Instead of trying to identify the distinct Myhill-Nerode equivalence classes, $\mathtt{L}^\#$ tries to distinguish states having different behaviours.

2.6 Conclusion

This chapter has laid the foundational groundwork for understanding this thesis. We introduced the notations used throughout the thesis and discussed some basic concepts from linear algebra. Additionally, we presented various one-counter automata models and compared their expressive powers. A brief discussion on automata learning algorithms was included, along with an overview of the L^* algorithm for learning DFA. This foundation will support the subsequent chapters, which will delve into the theoretical and practical aspects of one-counter systems.

Part II

Deterministic Real-Time One-Counter Automata (DROCA)

- The results presented in Chapter 3 and Chapter 4 will be published in the proceedings of **TACAS 2025**. The full version of the paper is available on arXiv under the title "**Learning Real-Time One-Counter Automata Using Polynomially Many Queries**" [https://arxiv.org/abs/2411.08815].

This is a joint work with *Dr. Vincent Penelle*, and *Dr. Sreejith A.V*.

CHAPTER 3

Equivalence of DROCAs

In this chapter, we show that the equivalence of counter-synchronous DROCAs and that of VOCAs can be checked faster than that of general DROCAs. The equivalence of DROCAs was shown to be NL-complete by Böhm and Göller (2011). They prove that if two DROCAs with number of states less than $\mathsf{K} \in \mathbb{N}$ are not equivalent, then there is an $\mathcal{O}(\mathsf{K}^{26})$ length word[1] that distinguishes them. However, this polynomial is too large for practical applications. First, in Section 3.1, we examine the reachability and coverability problems of DROCAs and show that they are in P. Next, in Section 3.2, we introduce an algorithm that solves the following two problems: (1) Counter synchronicity: check if two DROCAs are counter-synchronous, and (2) Equivalence: check if two counter-synchronous DROCAs are equivalent. Finally, Section 3.3 presents an even more efficient algorithm for equivalence checking of VOCAs.

Contents

[1]This polynomial is obtained from the equivalence result of DROCA from Chapter 5.

3.1 Reachability and Coverability Problems of DROCAs

In this section, we define the *reachability* problem and the *coverability* problem for DROCAs and show that they are in P. The proofs are straightforward and use standard techniques. These results presented in this section are folklore and are presented here for the sake of completeness.

REACHABILITY PROBLEM
INPUT: a DROCA $\mathcal{A}$, two configurations $\mathtt{c}$ and $\mathtt{d}$.
OUTPUT: *Yes*, if there exists a run $\mathtt{c} \xrightarrow{*} \mathtt{d}$ in $\mathcal{A}$. *No*, otherwise.

COVERABILITY PROBLEM
INPUT: a DROCA $\mathcal{A}$, a configuration $\mathtt{c}$, and a state p.
OUTPUT: *Yes*, if there exists a run $\mathtt{c} \xrightarrow{*} (p, n)$ in $\mathcal{A}$ for some $n \in \mathbb{N}$. *No*, otherwise.

The reachability and coverability problems of one-counter automata are well-studied in the literature. Chistikov et al. (2019) studied the length of the shortest accepted word by one-counter automata. They proved that for a DROCA of size n and states p, q, if $(p, 0) \xrightarrow{*} (q, 0)$ then the length of the shortest word w such that $(p, 0) \xrightarrow{w} (q, 0)$ is not greater than $14n^2$. This upper bound was previously conjectured by Wojtczak (2009). Chistikov et al. (2019) also proved that whenever there is a path for a configuration (p, m) to another configuration (q, k) where $p, q \in Q$ and $m, k \in \mathbb{N}$, there also exists a path from (p, m) to (q, k) that has length at most $14n^2 + n \cdot max(m, k)$. One can obtain polynomial time algorithms (even an NL upper bound) for reachability and coverability using their results. However, we present proofs for both reachability and coverability for completeness.

Given a DROCA $\mathcal{A}$, and two configurations $\mathtt{c}$ and $\mathtt{d}$ as inputs, we say that a word w is a *reachability witness* for $(\mathtt{c}, \mathtt{d})$ if $\mathtt{c} \xrightarrow{w} \mathtt{d}$. The word w is called a *minimal reachability witness* for $(\mathtt{c}, \mathtt{d})$ if it is a reachability witness and for all $w' \in \Sigma^*$ if $\mathtt{c} \xrightarrow{w'} \mathtt{d}$ then $|w'| \geq |w|$. Similarly, given a coverability problem with a DROCA $\mathcal{A}$, a configuration $\mathtt{c}$, and a state q as inputs, we say that a word w is a *coverability witness* for $(\mathtt{c}, q)$ if $\mathtt{c} \xrightarrow{w} (q, n)$ for some $n \in \mathbb{N}$. The word w is called a *minimal* *coverability witness* for (c, q) if it is a coverability witness for (c, q) and for all $w' \in \Sigma^*$ if $\mathsf{c} \xrightarrow{w'} (q, n')$ for some $n' \in \mathbb{N}$, then $|w'| \geq |w|$. In the following lemma, we show that if the run of a minimal coverability witness is a floating run, then the last counter value is linearly bounded in the initial counter value and the size of the automaton.

Lemma 3.1

Given a DROCA $\mathcal{A}$, a state p and a configuration (q, k), if $(q, k) \xrightarrow{*} (p, n)$ for some $n \in \mathbb{N}$ is a floating run, then there exists a word w and $m < k + |\mathcal{A}|$ such that $(q, k) \xrightarrow{w} (p, m)$ and is a floating run.

Proof. Let p be a state of a DROCA $\mathcal{A}$, $\mathsf{c} = (q, k)$ be the given configuration and $\mathsf{c} \xrightarrow{*} (p, n)$ for some $n \in \mathbb{N}$ is a floating run. Let $w \in \Sigma^*$ be a word such that $\mathsf{c} \xrightarrow{w} (p, m)$ for some $m \in \mathbb{N}$ and for all $w' \in \Sigma^*$, if $\mathsf{c} \xrightarrow{w'} (p, n)$ for some $n \in \mathbb{N}$ then $n \geq m$. We show that the counter value $m < k + |\mathcal{A}|$.

Assume for contradiction that $m \geq k + |\mathcal{A}|$. For any $i \in [k, m]$, we denote by g_i the configuration with counter value i encountered for the last time during the run of the word w from the given configuration c.

Consider the configurations $\mathsf{g}_k, \mathsf{g}_{k+1}, \mathsf{g}_{k+2}, \ldots, \mathsf{g}_m$. Since $m \geq k + |\mathcal{A}|$, by the pigeonhole principle, there exists i, j with $k \leq i < j \leq m$, such that g_i and g_j have the same state. Let $w = w_1 w_2 w_3$ for some $w_1, w_2, w_3 \in \Sigma^*$ such that $\mathsf{c} \xrightarrow{w_1} \mathsf{g}_i \xrightarrow{w_2} \mathsf{g}_j \xrightarrow{w_3} (p, m)$. The configurations g_i and g_j have the same state, and since g_j is the configuration with counter value j encountered for the last time during the run of the word w from c, the counter values encountered during the run $\mathsf{g}_j \xrightarrow{w_3} (p, m)$ is greater than j. Therefore, the run $\mathsf{c} \xrightarrow{w_1} \mathsf{g}_i \xrightarrow{w_3} (p, m-(j-i))$ is a valid floating run since it does not introduce any new zero tests. This contradicts our initial assumption that m is the smallest possible counter value where we encounter the state p in a floating run starting from c.

$\square$

In the following lemma, we show that the maximum counter value encountered during the run of a minimal reachability witness is linearly bounded in the input counter values and quadratically bounded in the size of the DROCA.

Lemma 3.2

Given a DROCA $\mathcal{A}$, and two configurations (q,k) and (p,n), if $(q,k) \xrightarrow{*} (p,n)$, then there exists $w \in \Sigma^*$, such that $(q,k) \xrightarrow{w} (p,m)$ and the maximum counter value encountered during this run is less than $max\{k,m\} + |\mathcal{A}|^2$.

Proof. Let (q,k) and (p,n) be two configurations of a DROCA $\mathcal{A}$ and $w \in \Sigma^*$ be a minimal reachability witness for $((q,k),(p,n))$. Let h denote the maximum counter value encountered during the run $(q,k) \xrightarrow{w} (p,n)$. We show that $h < max\{k,m\} + |\mathcal{A}|^2$.

Let $t = max\{k,m\}$. Assume for contradiction that $h \geq t + |\mathcal{A}|^2$. For any $i \in [t,h]$, we denote by e_i and e'_i the configurations with counter value i that is encountered for the last (resp. first) time before (resp. after) reaching counter value h during the run of the word w. Consider the pairs of configurations $(\mathsf{e}_t,\mathsf{e}'_t),(\mathsf{e}_{t+1},\mathsf{e}'_{t+1}),\ldots,(\mathsf{e}_{h-1},\mathsf{e}'_{h-1})$. Since $h > t + |\mathcal{A}|^2$, by the pigeonhole principle, there exist two states r,s, and indices $i,j \in [t,h-1]$ such that, the states of e_i and e_j is r and the states of e'_i and e'_j is s. Let $w = u_1u_2u_3u_4u_5$ for some $u_1,u_2,u_3,u_4,u_5 \in \Sigma^*$ such that $(q,k) \xrightarrow{u_1} \mathsf{e}_i \xrightarrow{u_2} \mathsf{e}_j \xrightarrow{u_3} \mathsf{e}'_j \xrightarrow{u_4} \mathsf{e}'_i \xrightarrow{u_5} (p,n)$. The run $(q,k) \xrightarrow{u_1} \mathsf{e}_i \xrightarrow{u_3} \mathsf{e}'_i \xrightarrow{u_5} (p,n)$ is a valid run since the sub-runs on u_2 and u_4 are loops with inverse counter effects. Note that removing these sub-runs does not introduce any zero tests since the counter values encountered during the runs $\mathsf{e}_i \xrightarrow{u_2} \mathsf{e}_j$ and $\mathsf{e}'_j \xrightarrow{u_4} \mathsf{e}'_i$ are greater than i. This contradicts the minimality of w. Therefore, $h < max\{k,m\} + |\mathcal{A}|^2$. □

Now, we prove that if there exists a minimal coverability witness for a coverability problem, then the final counter value is linearly bounded in the input counter value and the size of the automaton.

Lemma 3.3

Given a DROCA $\mathcal{A}$, a configuration (q,k), and a state p, if $(q,k) \xrightarrow{*} (p,n)$ for some $n \in \mathbb{N}$, then there exists $m \in \mathbb{N}$ with $m < k + |\mathcal{A}|$ such that $(q,k) \xrightarrow{*} (p,m)$.

Proof. Let $\mathcal{A}$ be a DROCA, (q,k) a configuration of $\mathcal{A}$, and p a state of $\mathcal{A}$. Let $w \in \Sigma^*$ denote the minimal coverability witness such that $(q,k) \xrightarrow{w} (p,m)$ in $\mathcal{A}$ for some $m \in \mathbb{N}$. We prove that the counter value $m < k + |\mathcal{A}|^2$.

If $(q, k) \xrightarrow{w} (p, m)$ is a floating run, then from lemma 3.1, we get that $m < k + |\mathcal{A}|$. Let us now assume that $(q, k) \xrightarrow{w} (p, m)$ is non-floating run. Let e_0 denote the last configuration with counter value encountered during the run $(q, k) \xrightarrow{w} (p, m)$. Let $w_1, w_2 \in \Sigma^*$ such that $w = w_1 w_2$ and $(q, k) \xrightarrow{w_1} \mathsf{e}_0 \xrightarrow{w_2} (p, m)$. Since w is the minimal coverability witness, by applying Lemma 3.1 on the run $\mathsf{e}_0 \xrightarrow{w_2} (p, m)$, we get that $m < |\mathcal{A}|$. □

Using Lemma 3.1, Lemma 3.2, and Lemma 3.3, we prove that both the reachability and coverability problems are in P.

Theorem 3.4 (Chistikov et al. (2019))
Given a DROCA $\mathcal{A}$, a configuration (q, k), a state p, and $t \in \mathbb{N}$,

1. if $(q, k) \xrightarrow{*} (p, n)$ for some $n \in \mathbb{N}$, then a minimal coverability witness for $((q, k), p)$ in $\mathcal{A}$ can be found in $(max\{k, |\mathcal{A}|\} + |\mathcal{A}|^2)|\mathcal{A}|$ time.

2. if $(q, k) \xrightarrow{*} (p, t)$, then a minimal reachability witness for $((q, k), (p, n))$ in $\mathcal{A}$ can be found in $(max\{k, t\} + |\mathcal{A}|^2)|\mathcal{A}|$ time.

Proof. First, we show that the maximum counter value encountered during the run of a minimal reachability witness and a minimal coverability witness is polynomially bounded by the size of the automaton and the input counter values.

Let $\mathcal{A}$ be a DROCA, (q, k) a configuration, p a state of $\mathcal{A}$ and $t \in \mathbb{N}$. First, consider the coverability problem. From Lemma 3.3, we get that if $(q, k) \xrightarrow{*} (p, n)$ for some $n \in \mathbb{N}$, then there exists $m \in \mathbb{N}$ with $m < k + |\mathcal{A}|$ such that $(q, k) \xrightarrow{*} (p, m)$. Let w be the minimal reachability witness for $(q, k) \xrightarrow{*} (p, m)$. From Lemma 3.2, the maximum counter value encountered during this run is $max\{k, |\mathcal{A}|\} + |\mathcal{A}|^2$. Next, we consider the reachability problem. From Lemma 3.2, we get that if $(q, k) \xrightarrow{*} (p, t)$, then the maximum counter value encountered during this run is $max\{k, t\} + |\mathcal{A}|^2$. Therefore, in either case, the maximum counter value encountered is polynomially bounded in $|\mathcal{A}|$ and the input counter values. Therefore, both the reachability and coverability problems are reduced to checking reachability between a set of configurations whose counter value is polynomially bounded in the size of the automaton and the input counter values.

Let r be the maximum counter value encountered during the run of a minimal reachability witness. To check reachability, we construct the configuration graph of $\mathcal{A}$ up to counter value r. The configuration graph of $\mathcal{A}$ up to counter value r will contain at most $r \times |\mathcal{A}|$ many states. The reachability from one state to another in this graph can be done in $r \times |\mathcal{A}|$ time using standard breadth-first search. Therefore, coverability can be checked in $(max\{k, |\mathcal{A}|\} + |\mathcal{A}|^2)|\mathcal{A}|$ time and reachability in $(max\{k, t\} + |\mathcal{A}|^2)|\mathcal{A}|$ time. □

3.2 Equivalence of counter-synchronised DROCAs

In this section, we prove that the equivalence of two counter-synchronous DROCAs can be checked significantly faster than that of general DROCAs.

> COUNTER-SYNCHRONICITY OF DROCAS
> INPUT: Two DROCAs $\mathcal{A}$ and $\mathcal{B}$.
> OUTPUT: *Yes*, if $\mathcal{A}$ and $\mathcal{B}$ are counter-synchronised.
> *No*, otherwise.

> EQUIVALENCE OF COUNTER-SYNCHRONISED DROCAS
> INPUT: Two counter-synchronised DROCAs $\mathcal{A}$ and $\mathcal{B}$.
> OUTPUT: *Yes*, if $\mathcal{A}$ and $\mathcal{B}$ are equivalent.
> *No*, otherwise.

Given two DROCAs with K states, we give $\mathcal{O}(\alpha(\mathsf{K}^5)\mathsf{K}^5)$ algorithms for the following: (1) Check whether they are counter-synchronised and (2) verify their equivalence if they are counter-synchronised. This is stated in Theorem 3.5.

Hopcroft and Karp (1971) gave an algorithm for checking the equivalence of two DFAs. An analysis of this algorithm by Almeida et al. (2010) shows that given two DFAs with size n, the algorithm runs in $\mathcal{O}(\alpha(n)n)$ time. Here, α is a slow-growing function and is related to the functional inverse of the Ackermann function (Almeida et al., 2010). The paper does not provide a closed-form expression for α. However $\alpha(n) = o(n)$ and for every $j \in [0, 2^{2^{2^{16}}}]$, $\alpha(j) \leq 4$. Therefore, for all practical applications, one can consider α as a constant function. The function α in Theorem 3.5 and Theorem 3.6 comes from this algorithm for checking the equivalence of two DFAs.

Theorem 3.5

Given two DROCAs $\mathcal{A}$ and $\mathcal{B}$ with $|\mathcal{A}|, |\mathcal{B}| \leq \mathsf{K}$ for some $\mathsf{K} \in \mathbb{N}$,

1. if $\mathcal{A}$ and $\mathcal{B}$ are not counter-synchronised then there is a word $w \in \Sigma^*$ with $|w| \leq 2\mathsf{K}^5$, $height_{\mathcal{A}}(w) \leq \mathsf{K}^4$ and, $height_{\mathcal{B}}(w) \leq \mathsf{K}^4$ such that, $\mathtt{ce}_{\mathcal{A}}(w) \neq \mathtt{ce}_{\mathcal{B}}(w)$. There is an $\mathcal{O}(\mathsf{K}^6)$ time algorithm to output this word if it exists.

2. if $\mathcal{A}$ and $\mathcal{B}$ are counter-synchronised and not equivalent, then there is a word $w \in \Sigma^*$ with $|w| \leq 2\mathsf{K}^5$, and $height_{\mathcal{A}}(w) = height_{\mathcal{B}}(w) \leq \mathsf{K}^4$ such that $\mathcal{A}(w) \neq \mathcal{B}(w)$. There is an $\mathcal{O}(\alpha(\mathsf{K}^5)\mathsf{K}^5)$ time algorithm to output this word if it exists.

Proof. Let $\mathcal{A} = (Q_1, \Sigma, p_0, \delta_0^1, \delta_1^1, F_1)$ and $\mathcal{B} = (Q_2, \Sigma, q_0, \delta_0^2, \delta_1^2, F_2)$ be two DROCAs.

Case-1: $\mathcal{A}$ and $\mathcal{B}$ are not counter-synchronised.

Let w be a word such that $\mathtt{ce}_{\mathcal{A}}(w) \neq \mathtt{ce}_{\mathcal{B}}(w)$ and for all $w' \in \Sigma^*$ with $\mathtt{ce}_{\mathcal{A}}(w') \neq \mathtt{ce}_{\mathcal{B}}(w')$, either $|w'| > |w|$ or $height_{\mathcal{A}}(w') > height_{\mathcal{A}}(w)$.

Let $w = w_1 a$ for some $w_1 \in \Sigma^*$ and $a \in \Sigma$. We know that for all strict prefixes w' of w, $\mathtt{ce}_{\mathcal{A}}(w') = \mathtt{ce}_{\mathcal{B}}(w')$. Since the counter values remain the same for all strict prefixes of w, the synchronous run on the word w_1 on the two machines can be seen as the run of a DROCA $\mathcal{C}$ with $|\mathcal{A}| \times |\mathcal{B}|$ states as defined below.

$$\mathcal{C} = (Q_1 \times Q_2, \Sigma, (p_0, q_0), \delta_0, \delta_1, F_1 \times F_2), \text{ where}$$

$\delta_0 : (Q_1 \times Q_2) \times \Sigma \to (Q_1 \times Q_2) \times \{0, +1\}$ and $\delta_1 : (Q_1 \times Q_2) \times \Sigma \to (Q_1 \times Q_2) \times \{0, +1, -1\}$ are the transition functions defined as follows: For $q_1, q_1' \in Q_1, q_2, q_2' \in Q_2$, $\delta_0((q_1, q_2), a) = ((q_1', q_2'), e)$ if $\delta_0^1(q_1, a) = (q_1', e)$ and $\delta_0^2(q_2, a) = (q_2', e)$ and is undefined otherwise. Similarly, for $q_1, q_1' \in Q_1, q_2, q_2' \in Q_2$, $\delta_1((q_1, q_2), a) = ((q_1', q_2'), e)$ if $\delta_1^1(q_1, a) = (q_1', e)$ and $\delta_1^2(q_2, a) = (q_2', e)$ and is undefined otherwise.

The states of this machine $\mathcal{C}$ will be of the form $(p, q) \in Q_1 \times Q_2$. The synchronous run of w_1 can be represented as $((p_0, q_0), 0) \xrightarrow{w_1} ((p, q), m)$ for some $(p, q) \in Q_1 \times Q_2$ and $m \in \mathbb{N}$.

Claim 1. $height_{\mathcal{A}}(w_1) = height_{\mathcal{B}}(w_1) < (|\mathcal{A}| \times |\mathcal{B}|)^2$.

Proof. Assume for contradiction that $height_{\mathcal{A}}(w_1) = height_{\mathcal{B}}(w_1) \geq (|\mathcal{A}| \times |\mathcal{B}|)^2$. If $m > 0$ (resp. $m = 0$), then in order to find a shorter word that distinguishes $\mathcal{A}$ and $\mathcal{B}$, we only need to find a shorter word to a configuration $((p, q), m')$ of $\mathcal{C}$ for some $m' > 0$ (resp $m' = 0$). From Lemma 3.2 and Lemma 3.3, we get that there exists $w' \in \Sigma^*$ with $height_{\mathcal{C}}(w') < (|\mathcal{A}| \times |\mathcal{B}|)^2$ such that $((p_0, q_0), 0) \xrightarrow{w'} ((p, q), m')$, where $m' = 0$ if $m = 0$ and $m' > 0$ otherwise. Therefore, $(p_0, 0) \xrightarrow{w'} (p, m')$ in $\mathcal{A}$, $(q_0, 0) \xrightarrow{w'} (q, m')$ in $\mathcal{B}$ and $height_{\mathcal{A}}(w') = height_{\mathcal{B}}(w') < (|\mathcal{A}| \times |\mathcal{B}|)^2$. Now consider the word $w'a$. The machines $\mathcal{A}$ and $\mathcal{B}$ reach different counter values on reading this word. This contradicts our initial assumption regarding the minimality of w.
$\square_{Claim:1}$

From Claim 1, we know that both $height_{\mathcal{A}}(w_1)$ and $height_{\mathcal{B}}(w_1)$ is less than $(|\mathcal{A}| \times |\mathcal{B}|)^2$. Hence, the number of distinct counter values encountered during this run is $(|\mathcal{A}| \times |\mathcal{B}|)^2$.

Let $G_{\mathcal{A}}$ denote the configuration graph of $\mathcal{A}$ up to counter value $(|\mathcal{A}| \times |\mathcal{B}|)^2$ and $G_{\mathcal{B}}$ denote the configuration graph of $\mathcal{B}$ up to counter value $(|\mathcal{A}| \times |\mathcal{B}|)^2$. The number of states in $G_{\mathcal{A}}$ is $|\mathcal{A}| \times (|\mathcal{A}| \times |\mathcal{B}|)^2$ and that of $G_{\mathcal{B}}$ is $|\mathcal{B}| \times (|\mathcal{A}| \times |\mathcal{B}|)^2$. Let $G = (Q', \Sigma, q_{init}, \delta, F)$ be a DFA. Where $F = \{q_F\}$ is a singleton set containing a final state, $Q' = \left(Q_1 \times Q_2 \times (|\mathcal{A}| \times |\mathcal{B}|)^2\right) \cup F$, $q_{init} = (p_0, q_0, 0)$ is the initial state, $\delta : Q' \times \Sigma \to Q'$ is a partial transition function defined as follows: for $p \in Q_1, q \in Q_2, n \in [0, (|\mathcal{A}| \times |\mathcal{B}|)^2]$ and $a \in \Sigma$, if $\delta^1_{sign(n)}(p, a) = (p', e)$ and $\delta^2_{sign(n)}(q, a) = (q', e)$ for some $e \in \{-1, 0, +1\}$ with $n + e \leq (|\mathcal{A}| \times |\mathcal{B}|)^2$ then, $\delta((p, q, n), a) = (p', q', n + e)$. if $\delta^1_{sign(n)}(p, a) = (p', e)$ and $\delta^2_{sign(n)}(q, a) = (q', e')$ for some $e, e' \in \{-1, 0, +1\}$ with $e \neq e'$ then, $\delta((p, q, n), a) = q_F$. The number of states in G is less than $(|\mathcal{A}| \times |\mathcal{B}|)^3$. Now, checking whether $\mathcal{A}$ and $\mathcal{B}$ are counter-synchronous reduces to checking whether the final state f is reachable in G. Using breadth-first search on G, we get that the smallest word, if it exists, that satisfies this property is less than $(|\mathcal{A}| \times |\mathcal{B}|)^3$ and can be found in $\mathcal{O}((|\mathcal{A}| \times |\mathcal{B}|)^3)$ time.

Case-2: $\mathcal{A}$ and $\mathcal{B}$ are counter-synchronised but not equivalent.

Assume $\mathcal{A}$ and $\mathcal{B}$ are counter-synchronised DROCAs. If $\mathcal{A}$ and $\mathcal{B}$ are not equivalent, then there exists a $w \in \Sigma^*$ such that $\mathcal{A}(w) \neq \mathcal{B}(w)$. Without loss of generality, let us assume that $\mathcal{A}(w) = 1$ and $\mathcal{B}(w) = 0$.

Now consider the run of the word w on the DROCA $\mathcal{A} \times \mathcal{B}$ obtained by taking the product of $\mathcal{A}$ and $\mathcal{B}$. Similar to the proof of the previous case, we can show that if w is a minimal witness, then $height_{\mathcal{A}}(w), height_{\mathcal{B}}(w) \leq (|\mathcal{A}| \times |\mathcal{B}|)^2$. Now consider the configuration graphs of $\mathcal{A}$ and $\mathcal{B}$ upto counter value $(|\mathcal{A}| \times |\mathcal{B}|)^2$. In order to check the equivalence of $\mathcal{A}$ and $\mathcal{B}$, it suffices to check the equivalence of these initial portions of the configuration graphs. The total number of states in the configuration graphs of $\mathcal{A}$ and $\mathcal{B}$ up to counter value $(|\mathcal{A}| \times |\mathcal{B}|)^2$ is $(|\mathcal{A}| + |\mathcal{B}|)(|\mathcal{A}| \times |\mathcal{B}|)^2$. Therefore, the length of w is bounded by $(|\mathcal{A}| + |\mathcal{B}|) \times (|\mathcal{A}| \times |\mathcal{B}|)^2$. Also, this equivalence check can be done in $\mathcal{O}(\alpha((|\mathcal{A}| + |\mathcal{B}|) \times (|\mathcal{A}| \times |\mathcal{B}|)^2)(|\mathcal{A}| + |\mathcal{B}|) \times (|\mathcal{A}| \times |\mathcal{B}|)^2)$ time using the algorithm for checking the equivalence of two DFAs Hopcroft and Karp (1971) that returns a minimal word that is accepted in one machine and rejected in the other.

Note that in both these cases, the returned word w is such that $|w| \leq (|\mathcal{A}| + |\mathcal{B}|) \times (|\mathcal{A}| \times |\mathcal{B}|)^2$, $height_{\mathcal{A}}(w) \leq (|\mathcal{A}| \times |\mathcal{B}|)^2$, $height_{\mathcal{B}}(w) \leq (|\mathcal{A}| \times |\mathcal{B}|)^2$ and can be found in $\mathcal{O}(\alpha((|\mathcal{A}| + |\mathcal{B}|) \times (|\mathcal{A}| \times |\mathcal{B}|)^2)(|\mathcal{A}| + |\mathcal{B}|) \times (|\mathcal{A}| \times |\mathcal{B}|)^2)$ time. Since $|\mathcal{A}|, |\mathcal{B}| \leq \mathsf{K}$, the theorem follows. $\square$

3.3 Equivalence of Visibly One-Counter Automata

In the special case of VOCAs, we show that there is a faster algorithm running in $\mathcal{O}(\alpha(\mathsf{K}^3)\mathsf{K}^3)$ for checking the equivalence of two VOCAs with number of states less than or equal to K. Given two VOCAs over the same pushdown alphabet $(\Sigma_{call}, \Sigma_{ret}, \Sigma_{int})$, the counter is always incremented (resp. decremented) on reading a symbol from Σ_{call} (resp. Σ_{ret}) and is left unchanged on reading a symbol from Σ_{int}. Therefore, the counter value reached on reading a word is dependent only on the word and the pushdown alphabet. For two VOCAs over the same pushdown alphabet, the counter value reached on both the VOCAs on the same word is, therefore, the same.

EQUIVALENCE OF VOCAS
INPUT: Two VOCAs $\mathcal{A}$ and $\mathcal{B}$.
OUTPUT: *Yes*, if $\mathcal{A}$ and $\mathcal{B}$ are equivalent.
No, otherwise.

Similar to Theorem 3.5, the function α in the following theorem is a function that grows very slowly Almeida et al. (2010) and can be considered a constant function for all practical purposes. It comes from the algorithm by Hopcroft and Karp (1971) for checking the equivalence of two DFAs. The ideas in proving Theorem 3.6 is similar to those used in Section 7.2 in the context of weighted one-counter automata.

Theorem 3.6

Given two VOCAs $\mathcal{A}$ and $\mathcal{B}$ over the same pushdown alphabet with $|\mathcal{A}|, |\mathcal{B}| \leq \mathsf{K}$ for some $\mathsf{K} \in \mathbb{N}$, if $\mathcal{A}$ and $\mathcal{B}$ are not equivalent, then there exists a minimal word w such that $\mathcal{A}(w) \neq \mathcal{B}(w)$ with $|w| \leq 4\mathsf{K}(\mathsf{K}+\mathsf{K}^2)$, and $height_{\mathcal{A}}(w) = height_{\mathcal{B}}(w) \leq 2(\mathsf{K}+\mathsf{K}^2)$. There is an $\mathcal{O}(\alpha(\mathsf{K}^3)\mathsf{K}^3)$ algorithm to find this word if it exists.

Proof. Let $\mathcal{A} = (Q_1, \Sigma, p_{init}, \delta_0^1, \delta_1^1, F_1)$ and $\mathcal{B} = (Q_2, \Sigma, q_{init}, \delta_0^2, \delta_1^2, F_2)$ be two VOCAs that are not equivalent. Since $\mathcal{A}$ and $\mathcal{B}$ are VOCAs, for any $w \in \Sigma^*$, $\mathtt{ce}_{\mathcal{A}}(w) = \mathtt{ce}_{\mathcal{B}}(w)$. We use $\mathtt{ce}(w)$ to denote this value. Similary, we use $height(w)$ to denote $height_{\mathcal{A}}(w) = height_{\mathcal{B}}(w)$. Let $Q_1 = \{p_1, p_2, \ldots, p_{|\mathcal{A}|}\}$ and $Q_2 = \{q_1, q_2, \ldots, q_{|\mathcal{B}|}\}$. For a pair of states $p_i \in Q_1$ and $q_j \in Q_2$, we define the row vector $\mathbf{x}_{(p_i,q_j)} \in \{0,1\}^{|\mathcal{A}|+|\mathcal{B}|}$ as follows: $\mathbf{x}_{(p_i,q_j)}[k] = 1$ if and only if $k = i$ or $k = |\mathcal{A}| + j$ for $k \in [1, |\mathcal{A}| + |\mathcal{B}|]$. We also define the row vector $\boldsymbol{\eta} \in \{0,1\}^{|\mathcal{A}|+|\mathcal{B}|}$ such that for $k \in [1, |\mathcal{A}| + |\mathcal{B}|]$

$$\boldsymbol{\eta}[k] = \begin{cases} 1, \text{ if } k \leq |\mathcal{A}| \text{ and } p_k \in F_1 \\ -1, \text{ if } k > |\mathcal{A}| \text{ and } q_{k-|\mathcal{A}|} \in F_2 \\ 0, \text{ otherwise.} \end{cases}$$

Therefore, $\mathbf{x}_{(p,q)}\boldsymbol{\eta}^\top \neq 0$, if exactly one among p and q is a final state.

We consider the synchronous run of $\mathcal{A}$ and $\mathcal{B}$. A configuration pair is denoted by $(\mathbf{x}_{(p,q)}, n)$ where $p \in Q_1, q \in Q_2$, and n is a counter value. The initial configuration pair is denoted by $c_{init} = (\mathbf{x}_{(p_{init},q_{init})}, 0)$. Given two configuration pairs $c_1 = (\mathbf{x}_{(p,q)}, n)$ and $c_2 = (\mathbf{x}_{(p',q')}, m)$, we use the notation $c_1 \xrightarrow{u} c_2$ to denote that $(p, n) \xrightarrow{u} (p', m)$ and $(q, n) \xrightarrow{u} (q', m)$. We define the transition matrix of u from $(\mathbf{x}_{(p,q)}, n)$ as the matrix $\mathbb{M} \in \{0,1\}^{(|\mathcal{A}|+|\mathcal{B}|)^2}$ such that for $i, j \in [1, |\mathcal{A}| + |\mathcal{B}|]$,

$\mathbb{M}[i,j] = 1$ if and only if $(p_i, n) \xrightarrow{u} (p_j, m)$ in $\mathcal{A}$ or $(q_{i-|\mathcal{A}|}, n) \xrightarrow{u} (q_{j-|\mathcal{A}|}, n)$ in $\mathcal{B}$. Therefore, $\mathbf{x}_{(p,q)}\mathbb{M} = \mathbf{x}_{(p',q')}$. Since $\mathcal{A}$ and $\mathcal{B}$ are VOCAs, Claim 1 and Claim 2 directly follow from the definition of the transition matrix.

Claim 1. *For any $p, p' \in P$, $q, q' \in Q$ and word w, the transition matrix of w from $(\mathbf{x}_{(p,q)}, n)$ is the same as the transition matrix of w from $(\mathbf{x}_{(p',q')}, n)$.*

Claim 2. *Let w be such that $\mathtt{ce}(w') > 0$ for all prefixes w' of w. Then, the transition matrix of w from $(\mathbf{x}_{(p,q)}, n)$ is the same as the transition matrix of w from $(\mathbf{x}_{(p',q')}, m)$ for any $p, p' \in P$ and $q, q' \in Q$ and $m, n > 0$.*

We use the fact that the machines are VOCAs to obtain Claim 1 and Claim 2.

Claim 3. *Any set of $|\mathcal{A}|^2 + |\mathcal{B}|^2 + 1$ transition matrices is linearly dependent.*

Proof. Given any transition matrix $\mathbb{M}$, a non-zero value can occur only in positions $i, j \in [1, |\mathcal{A}| + |\mathcal{B}|]$ with both $i, j < |\mathcal{A}|$ or both $i, j > |\mathcal{A}|$. Therefore, there are at most $|\mathcal{A}|^2 + |\mathcal{B}|^2$ positions in $\mathbb{M}$ where a non-zero value can occur. A transition matrix can be seen as a vector of size $|\mathcal{A}|^2 + |\mathcal{B}|^2$, discarding the portions where a non-zero value can never occur. From the fundamental theorem of vector spaces (see Lemma 2.2), if we have more than $|\mathcal{A}|^2 + |\mathcal{B}|^2$ transition matrices, there will be at least one that is dependent on the others (Strang, 2006). $\square_{Claim:3}$

Let w be a minimal word such that $\mathcal{A}(w) \neq \mathcal{B}(w)$. Let $c_\ell = (\mathbf{x}_\ell, n_\ell)$ denote the configuration pair such that $c_{init} \xrightarrow{w} c_\ell$. Therefore, $\mathbf{x}_\ell \boldsymbol{\eta}^\top \neq 0$.

Claim 4. *No configuration pair repeats during the synchronous run $c_{init} \xrightarrow{w} c_\ell$.*

Proof. Assume for contradiction that there is a configuration pair that repeats during the synchronous run on w. Let d denote this configuration pair and w_1, w_2, w_3 be words such that $w = w_1 w_2 w_3$ and $c_{init} \xrightarrow{w_1} d \xrightarrow{w_2} d \xrightarrow{w_3} c_\ell$. Since $c_{init} \xrightarrow{w_1} d \xrightarrow{w_3} c_\ell$, the word $w_1 w_3$ is a shorter word that distinguishes $\mathcal{A}$ and $\mathcal{B}$. This contradicts the minimality of w. $\square_{Claim:4}$

Claim 5. $height(w) \leq n_\ell + |\mathcal{A}|^2 + |\mathcal{B}|^2$.

Proof. Assume for contradiction that $height(w) = m$ and $m > n_\ell + |\mathcal{A}|^2 + |\mathcal{B}|^2$. Then there exists $w_1, w_2 \in \Sigma^*$ and configuration pair d such that $w = w_1 w_2$ and $c_{init} \xrightarrow{w_1} d \xrightarrow{w_2} c_\ell$ such that $\mathtt{ce}(w_1) = m$. For an $i \in [n_\ell, m]$, let $(\mathbf{y}_i, i)$ (resp. $(\mathbf{y}'_i, i)$)

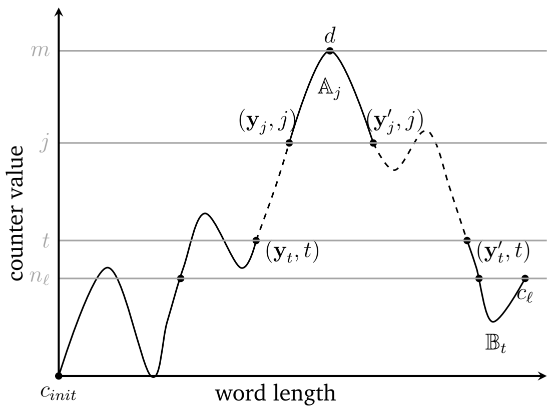


Fig. 3.1. The figure shows the synchronous run of a word on two VOCAs such that it reaches a final state in one VOCA and a non-final state in the other. Configuration pairs c_{l_j} and c_{l_t} (resp. c_{g_j} and c_{g_t}) are where the counter values j and t are encountered for the last (resp. first) time before (resp. after) reaching m. The dashed line denotes the part of the synchronous run that can be removed to get a smaller word that distinguishes the VOCAs.

be the configuration pair such that the counter value i is encountered for the last (resp. first) time during the run $c_{init} \xrightarrow{w_1} d$ (resp. $d \xrightarrow{w_2} c_\ell$); let $\mathbb{A}_i$ and $\mathbb{B}_i$ denote the transition matrices such that $\mathbf{y}_i\mathbb{A}_i = \mathbf{y}'_i$ and $\mathbf{y}'_i\mathbb{B}_i = \mathbf{x}_\ell$ (see Figure 7.1); and let x_i, y_i and z_i be such that $c_{init} \xrightarrow{x_i} (\mathbf{y}_i, i) \xrightarrow{y_i} (\mathbf{y}'_i, i) \xrightarrow{z_i} c_\ell$ and $w = x_i y_i z_i$.

Consider the matrices $\mathbb{A}_{m-1}, \mathbb{A}_{m-2}, \ldots, \mathbb{A}_{n_\ell}$ in order. From Claim 3, it follows that there exists $t \in [n_\ell, m-1]$ such that $\mathbb{A}_t$ is linearly dependent on matrices $\mathbb{A}_{m-1}, \cdots, \mathbb{A}_{t+1}$. Since $x_\ell \boldsymbol{\eta}^\top \neq 0$, it follows that $\mathbf{y}_t\mathbb{A}_t\mathbb{B}_t\boldsymbol{\eta}^\top \neq 0$. Therefore, $\mathbf{y}_t(r_{t+1}\mathbb{A}_{t+1} + \cdots + r_{m-1}\mathbb{A}_{m-1})\mathbb{B}_t\boldsymbol{\eta}^\top \neq 0$ for some integers $r_{t+1}, \ldots, r_{m-1}$ and hence there is a $j > t$ such that $\mathbf{y}_t\mathbb{A}_j\mathbb{B}_t\boldsymbol{\eta}^\top \neq 0$. We conclude the proof, by saying that $\hat{w} = x_t y_j z_t$ is a word accepted by exactly one of $\mathcal{A}$ or $\mathcal{B}$ contradicting the minimality of w. It suffices to show $c_{init} \xrightarrow{x_t} (\mathbf{y}_t, t) \xrightarrow{y_j} (\mathbf{y}_t\mathbb{A}_j, t) \xrightarrow{z_t} (\mathbf{y}_t\mathbb{A}_j\mathbb{B}_t, n_\ell)$. From Claim 2, we get that the transition matrix of y_j from $(\mathbf{y}_t, t)$ is $\mathbb{A}_j$. From Claim 1, we get that the transition matrix of z_t from $(\mathbf{y}_t\mathbb{A}_t, t)$ is $\mathbb{B}_t$. $\square_{Claim:5}$

Claim 6. $n_\ell \leq |\mathcal{A}| + |\mathcal{B}|$.

Proof. Assume for contradiction that $n_\ell > |\mathcal{A}| + |\mathcal{B}|$. For $i \in [1, n_\ell]$, let $(\mathbf{z}_i, i)$ denote the configuration pair such that the counter value i is encountered for the last time during the run $c_{init} \xrightarrow{w} c_\ell$. For all $i \in [1, n_\ell]$, let $x_i, z_i \in \Sigma^*$ such that

$c_{init} \xrightarrow{x_i} (\mathbf{z}_i, i) \xrightarrow{z_i} c_\ell$ and $w = x_i z_i$. For all $i \in [1, n_\ell - 1]$, let $\mathbb{C}_i$ denote the transition matrix such that $\mathbf{z}_i \mathbb{C}_i = \mathbf{x}_\ell$ with $\mathbf{z}_i \mathbb{C}_i \boldsymbol{\eta}^\top \neq 0$. Consider the vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{n_\ell}$ in order. From the fundamental theorem of vector spaces (see Lemma 2.2), it follows that there exists $t \leq (|\mathcal{A}| + |\mathcal{B}|) + 1$ such that $\mathbf{z}_t$ is a linear combination of $\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}$. Since $\mathbf{z}_t \mathbb{C}_t \boldsymbol{\eta}^\top \neq 0$, there exists $j < t$ such that $\mathbf{z}_j \mathbb{C}_t \boldsymbol{\eta}^\top \neq 0$. From Claim 2, we get that the transition matrix of z_t from $(\mathbf{z}_t, t)$ is $\mathbb{C}_t$. The word $x_j z_t$ contradicts the minimality of w. $\square_{Claim:6}$

Let $\mathsf{K} \in \mathbb{N}$ such that $|\mathcal{A}|, |\mathcal{B}| < \mathsf{K}$. From Claim 5 and Claim 6, we get that $height(w) \leq 2(\mathsf{K} + \mathsf{K}^2)$. We construct DFA $\mathcal{A}'$ (resp. $\mathcal{B}'$) of size $\mathcal{O}(\mathsf{K}^3)$ corresponding to the configuration graph of $\mathcal{A}$ (resp. $\mathcal{B}$) up to counter value $2(\mathsf{K} + \mathsf{K}^2)$. By using the Hopcroft-Karp algorithm Hopcroft and Karp (1971) for checking equivalence of DFAs, we get a distinguishing word with length less than $4\mathsf{K}(\mathsf{K} + \mathsf{K}^2)$ if $\mathcal{A}$ and $\mathcal{B}$ are not equivalent. $\square$

The above proof will not work for the case of DROCAs, since Claim 1 and Claim 2 in the above theorem don't hold for DROCAs. For VOCAs, the counter actions only depend on the input alphabet. Whereas, for DROCAs, the counter actions depend on the input alphabet, the current counter value and the current state of the DROCA. The transition matrix of a word changes based on its state and the counter value from which it is read.

3.4 Conclusion

In this chapter, we developed a specialised equivalence checking algorithm for counter-synchronous DROCAs. For VOCAs, we optimised this process and proposed an even faster equivalence check. Additionally, we presented the reachability and coverability problems of DROCAs and showed that they are in P. A summary of the results presented in this chapter is given in the folllowing table.

Problem	**Complexity**
Coverability from a configuration (q, k) to a state p of a DROCA with size K	$\mathcal{O}((\max\{k, \mathsf{K}\} + \mathsf{K}^2)\mathsf{K})$ (Chistikov et al., 2019)
Reachability from a configuration (q, k) to another configuration (p, t) of a DROCA with size K	$\mathcal{O}((\max\{k, t\} + \mathsf{K}^2)\mathsf{K})$ (Chistikov et al., 2019)
Counter-synchronicity of two DROCAs of size K	$\mathcal{O}(\mathsf{K}^6)$
Equivalence of two counter-synchronised DROCAs of size K	$\mathcal{O}(\alpha(\mathsf{K}^5)\mathsf{K}^5)$
Equivalence of two VOCAs of size K	$\mathcal{O}(\alpha(\mathsf{K}^3)\mathsf{K}^3)$

Table 3.1. Summary of results presented in Chapter 3.

CHAPTER 4

Learning DROCAs Using Polynomially Many Queries

In this chapter, we introduce a novel method for active learning of deterministic real-time one-counter automaton (DROCA). The existing techniques for learning a DROCA rely on observing the behaviour of the DROCA up to exponentially large counter values. Our algorithm eliminates this need and requires only a polynomial number of queries. Additionally, our method differs from existing techniques as we learn a minimal counter-synchronous DROCA, resulting in much smaller counter-examples on equivalence queries. Learning a minimal counter-synchronous DROCA cannot be done in polynomial time unless $\mathsf{P} = \mathsf{NP}$, even in the case of visibly one-counter automata (VOCAs). We use a SAT solver to overcome this difficulty. The solver is used to compute a minimal separating DFA from a given set of positive and negative samples.

We implemented the proposed learning algorithm and tested it on randomly generated DROCAs. Our evaluations show that the proposed method outperforms the existing techniques on the test set. We use the equivalence result of counter-synchronous DROCAs from Section 3.2 in our learning algorithm.

Contents

4.1 Introduction

Automata learning is a subfield of theoretical computer science that focuses on the automatic construction of automata from observations, such as input/output samples. This process plays a crucial role in fields like software verification and machine learning. Automata are mathematical models for systems that can describe sequences of operations or events, making them valuable for applications such as formal verification of software, model checking, and system analysis. Automata learning is particularly important for software verification as it allows for the modelling and analysing systems where exhaustive testing may not be feasible. With suitable algorithms, automata learning can enable software verification tools to automatically construct models from system traces, which can then be analysed for errors, security vulnerabilities, and performance issues. Despite theoretical advancements, the practical applicability of automata learning algorithms, especially for automata with resources (e.g., one-counter automata, pushdown automata), is hindered by inefficient equivalence checks and learning processes. This limits the broader applicability of automata learning algorithms for complex systems.

The objective of learning algorithms is to identify a model that best fits a given set of observations or data. However, this is a computationally challenging task. For instance, finding a minimal separating DFA – a DFA that accepts a given set of positive samples and rejects a given set of negative samples – is known to be NP-complete (Gold, 1978). Angluin (1987) introduced an active learning framework involving a learner and a teacher to overcome this challenge (see Figure 1.1). The learner constructs an automaton through a structured process of queries to the teacher. She proved that DFA can be learned using membership and equivalence queries in polynomial time. Angluin's algorithm, known as the L^* algorithm, provided a theoretical foundation for efficient DFA learning by allowing the learner to learn a minimal DFA in time polynomial in the size of the DFA by identifying all distinct equivalence classes. However, one cannot extend the L^* algorithm directly for more complex automata, such as OCA, which have infinitely many equivalence classes.

In this chapter, we are interested in active learning of a deterministic real-time one-counter automata (DROCAs). The counter adds expressive power, enabling a DROCA to recognise certain context-free languages (e.g., $\{a^n b^n \mid n > 0\}$) that cannot be recognised with a DFA. However, current computational and algorithmic limitations hinder the effective learning of such systems, emphasising the need for novel approaches to bridge this gap in automata learning.

4.2 Active Learning DROCAs

In this section, we explore active learning of DROCAs. In this framework, we have a $learner$ and a $teacher$. The learner aims to construct a DROCA that recognises the same language as the teacher's DROCA (call it $\mathcal{A}$). We discuss the various types of queries used in the learning process. We also provide a brief discussion of existing learning algorithms and compare the proposed method with these techniques, highlighting their differences.

4.2.1 Types of Queries

We list the various types of queries commonly used in the literature.

- membership queries $\mathsf{MQ}_{\mathcal{A}}$: the learner provides a word $w \in \Sigma^*$. The teacher returns 1 if $w \in \mathcal{L}(\mathcal{A})$, and 0 if $w \notin \mathcal{L}(\mathcal{A})$.
- equivalence queries $\mathsf{EQ}_{\mathcal{A}}$: the learner asks whether a DROCA $\mathcal{C}$ is equivalent to $\mathcal{A}$. The teacher returns *yes* if $\mathcal{C}$ and $\mathcal{A}$ are equivalent. Otherwise, the teacher provides a counter-example $z \in \Sigma^*$ such that $\mathcal{C}(z) \neq \mathcal{A}(z)$.
- counter value queries $\mathsf{CV}_{\mathcal{A}}$: the learner asks the counter value reached on reading a word w in $\mathcal{A}$. The teacher returns the corresponding counter value. i.e., $\mathtt{ce}_{\mathcal{A}}(w)$.
- partial equivalence queries $\mathsf{EQ}_{\mathcal{A}}$: the learner gives a DROCA $\mathcal{C}$ and an integer t and asks whether for all $z \in \Sigma^{\leq t}$, $\mathcal{C}(z) = \mathcal{A}(z)$. The teacher returns *yes* if this condition is true. Otherwise, the teacher provides a counter-example $z \in \Sigma^{\leq t}$ such that $\mathcal{C}(z) \neq \mathcal{A}(z)$.

The membership query and the equivalence query are similar to the ones used in L^*. The membership query returns the membership of a given word, and the equivalence query checks whether two DROCAs are equivalent and returns a counter-example if they are not. The counter value query, when given a word, returns the counter value reached on reading that word from the initial configuration. A partial equivalence query takes a DROCA and a limit and determines whether the set of all words whose length does not exceed the given limit accepted by the given DROCA and the target language are the same. It returns a word of length less than the limit that distinguishes them otherwise. We also introduce an additional query called the synchronous-equivalence query for learning DROCAs.

- *synchronous-equivalence queries* $\mathsf{SEQ}_{\mathcal{A}}$: the learner asks whether a DROCA $\mathcal{C}$ is equivalent and counter-synchronous to $\mathcal{A}$. The teacher returns *yes* if $\mathcal{C}$ and $\mathcal{A}$ are counter-synchronous and equivalent. Otherwise, the teacher provides a counter-example $z \in \Sigma^*$ such that $\mathcal{C}(z) \neq \mathcal{A}(z)$ or $\mathtt{ce}_{\mathcal{A}}(z) \neq \mathtt{ce}_{\mathcal{C}}(z)$.

One can also consider minimal versions of equivalence and synchronous-equivalence queries, that returns the smallest counterexample rather than returning an arbitrary one. These are outlined below.

- minimal equivalence queries $\mathsf{MEQ}_{\mathcal{A}}$: the learner asks whether a DROCA $\mathcal{C}$ is equivalent to $\mathcal{A}$. The teacher returns *yes* if $\mathcal{C}$ and $\mathcal{A}$ are equivalent. Otherwise, the teacher provides a *minimal* counter-example $z \in \Sigma^*$ such that $\mathcal{C}(z) \neq \mathcal{A}(z)$.

- minimal synchronous-equivalence queries $\mathsf{MSQ}_{\mathcal{A}}$: the learner asks whether a DROCA $\mathcal{C}$ is equivalent and counter-synchronous to $\mathcal{A}$. The teacher returns *yes* if $\mathcal{C}$ and $\mathcal{A}$ are counter-synchronous and equivalent. Otherwise, the teacher provides a *minimal* counter-example $z \in \Sigma^*$ such that $\mathcal{C}(z) \neq \mathcal{A}(z)$ or $\mathtt{ce}_{\mathcal{A}}(z) \neq \mathtt{ce}_{\mathcal{C}}(z)$.

In the case of synchronous-equivalence queries and *minimal synchronous-equivalence queries*, the teacher only returns the counter-example z and does not provide any additional information.

4.2.2 Existing Methods

In the context of learning one-counter systems, studies have been conducted by Berman and Roos (1987), Fahmy and Roos (1995), Neider and Löding (2010), and Bruyère et al. (2022). In this section, we provide a brief discussion of each of these works.

Berman and Roos (1987): Learning One-Counter Languages in Polynomial Time

P. Berman and R. S. Roos, in an extended abstract, claim to have proved that both equivalence and learning of deterministic one-counter automata is in P. However, as pointed out by Böhm et al. (2014), the results cannot be verified due to a lack of precise formulation and detailed proofs. It was later proved by Böhm et al. (2013), that the equivalence of determinisitc one-counter automata is indeed in P (NL-complete). However, the existence of a polynomial time algorithm for learning deterministic one-counter automata remained an open problem[1].

Fahmy and Roos (1995): Efficient Learning of Real-Time One-Counter Automata

A. F. Fahmy and R. S. Roos. proposed a learning algorithm for DROCAs using membership and equivalence queries. This work was proposed as a faster algorithm than Berman and Roos (1987) for learning DROCAs and can be considered as the first algorithm for learning DROCAs. They showed that a DROCA can be learned by first learning an initial segment of its behaviour graph. Unlike the behaviour graph of VOCAs, this behaviour graph is defined based on the standard Myhill-Nerode congruence. Additionally, they proved that the behaviour graph of a DROCAs exhibits a repeating structure. To learn a DROCA, it is sufficient to identify this repeating structure. However, to identify this repeating structure, one needs to learn an exponentially large behaviour graph. Recognising this repetitive structure of the behaviour graph lies at the core of all existing learning algorithms for learning DROCAs. This work also motivated Bárány et al. (2006) to prove that checking whether a given visibly pushdown automaton is equivalent to some VOCA is decidable. However, Bruyère et al. (2022) raises concerns about the precision of Fahmy and Roos' proofs for learning DROCAs, indicating that their method did not yield the expected results in some instances.

[1]We have recently proved that active learning of DOCAs is also in P (Mathew et al., 2025c).

Neider and Löding (2010): Learning Visibly One-Counter Automata in Polynomial Time

D. Neider and C. Löding, in a technical report, combined the techniques in Fahmy and Roos (1995) and Bárány et al. (2006) to propose an algorithm for learning VOCAs, employing an additional partial equivalence query. Their algorithm learns the initial portion of a behaviour graph defined based on a refined Myhill-Nerode congruence. Two words belong to the same equivalence class under this congruence if they reach the same counter value when read from the initial configuration. Note that in a VOCA, the input alphabet determines the counter-actions. Similar to the work by Fahmy and Roos (1995), the algorithm works by identifying a repetitive structure of this behaviour graph. They prove that their algorithm for learning VOCAs runs in polynomial time with respect to some characteristic parameters of the target language. However, these parameters can be exponential in size with respect to the number of states of a "minimal" VOCA recognising the language. Hence, even the existing algorithms for learning VOCAs have exponential time and query complexity with respect to the number of states of a minimal VOCA recognising the language.

Bruyère et al. (2022): Learning Real-Time One-Counter Automata

Recent work by V. Bruyère, G. A. Pérez, and G. Staquet focuses on learning DROCAs and introduces an additional counter value query along with the partial equivalence query. They use the counter value queries and employ the techniques introduced by Neider and Löding (2010) to learn DROCAs, but using exponential space, time, and number of queries. Their learning algorithm will be referred to as *BPS* from this point forward. This algorithm also works by identifying the repetitive structure of the behaviour graph defined based on the refined Myhill-Nerode congruence introduced by Neider and Löding (2010). They evaluated an implementation of their learning algorithm on random benchmarks and used it for JSON-stream validation.

4.2.2.1 Limitations of Existing Methods

From a complexity theoretical perspective, DROCAs can be learned with polynomial space and exponential time with a straightforward brute-force approach. This method entails enumerating all conceivable DROCAs, starting with a one-state DROCA, and submitting equivalence queries for each. This approach, without a doubt, entails an exponential number of equivalence queries.

All existing algorithms for learning DROCAs, including the algorithm by Fahmy and Roos (1995), the algorithm by Bruyère et al. (2022), and the algorithm for learning VOCA by Neider and Löding (2010) require exponential time and an exponential number of queries with respect to the number of states of a minimal DROCA recognising the language. All these algorithms share the idea of learning the initial portion of an infinite behaviour graph and then seek to identify a repetitive structure in it. However, in the worst-case scenario, this repetitive structure becomes apparent only after learning an exponentially large portion of the graph (see Figure 4.3). In this case, the learnt DROCA will be exponentially large. Consequently, learning this exponential-sized behaviour necessitates exponentially many queries. Moreover, the equivalence queries also run on these exponentially large DROCAs, making it even more infeasible. Section 4.2.3 gives the formal definition of a behaviour graph and discusses its repetitive structure. We prove the existence of DROCAs for which repetitive structure of the behaviour graph becomes apparent only after observing an exponentially large segment of the behaviour graph. Additionally, we present a language recognised by a VOCA that exhibits this phenomenon.

4.2.3 Behavioural Graph and its Repetitive Structure

The existing algorithms for learning one-counter automaton rely on learning the initial portion of an infinite canonical automaton called the behaviour graph. It has a periodic structure and thus can be represented using a finite object. The current algorithms seek to identify this repetitive pattern from an initial portion of the behaviour graph. However, one needs to learn the behaviour of the automaton up to exponentially large counter values to observe this pattern. We will now define a refined Myhill-Nerode congruence that defines a behaviour graph.

Definition 4.1 (Definition 2 in Neider and Löding (2010))
Given a DROCA $\mathcal{A}$, we define a *refined Myhill-Nerode congruence* $\simeq \subseteq \Sigma^* \times \Sigma^*$ as follows: for $u, v \in \Sigma^*$, $u \simeq v$ if and only if for all $z \in \Sigma^*$, $\mathcal{A}(uz) = \mathcal{A}(vz)$ and $\mathtt{ce}_{\mathcal{A}}(uz) = \mathtt{ce}_{\mathcal{A}}(vz)$. For all $w \in \Sigma^*$, the equivalence class of w under $\simeq$ is defined as $[w] = \{u \in \Sigma^* \mid u \simeq w\}$.

In addition to the standard conditions enforced by the Myhill-Nerode congruence, this refinement also ensures that all words that belong to the same equivalence class under it have the same counter value. The behaviour graph of a DROCA is an infinite state automaton induced by this refined congruence $\simeq$. Each equivalence class under this refined congruence corresponds to a state of the behaviour graph, and the transitions between them are defined based on the equivalence class of the resultant words.

Definition 4.2 (Definition 3 in Neider and Löding (2010))
Given a DROCA $\mathcal{A}$ and the refined Myhill-Nerode congruence $\simeq$, the *behaviour graph* $\mathtt{BG}_{\mathcal{A}}$ of $\mathcal{A}$ is a machine defined as $\mathtt{BG}_{\mathcal{A}} = (P, \Sigma, p_0, \Delta, H)$ where,

- $P = \{[u] \mid u \in \Sigma^*\}$ is the set of states.
- $p_0 = [\varepsilon]$ is the initial state.
- $\Delta : P \times \Sigma \to P$ is the transition function and is defined as follows. For $u \in \Sigma^*$ and $a \in \Sigma$, $\Delta([u], a) = [ua]$.
- $H = \{[u] \mid \mathcal{A}(u) = 1\}$ is the set of final states.

Given a DROCA $\mathcal{A} = (Q, \Sigma, q_0, \delta_0, \delta_1, F)$ and it's behaviour graph $\mathtt{BG}_{\mathcal{A}} = (P, \Sigma, p_0, \Delta, H)$ we define a mapping $cMap$ between the configurations of $\mathcal{A}$ and the states of the behaviour graph $\mathtt{BG}_{\mathcal{A}}$ similar to that by Fahmy and Roos (1995). Note that the behaviour graph can be obtained from the configuration graph of $\mathcal{A}$ by merging the states corresponding to the configurations that belong to the same refined Myhill-Nerode equivalence class. For a configurations c of $\mathcal{A}$, we say that $cMap(c) = [u]$ for some $u \in \Sigma^*$ if and only if $(q_0, 0) \xrightarrow{u} c$ in $\mathcal{A}$. We extend this mapping $cMap$ to the infinite configuration graph $G_{\mathcal{A}} = (Q', \Sigma, q_0, \Delta', F')$ of $\mathcal{A}$ as follows: given a configuration c of $\mathcal{A}$ and $a \in \Sigma$, $cMap(\Delta'(c, a)) = \Delta(cMap(c), a)$. A subgraph of the configuration graph is now mapped into a subgraph of the behavioural graph using the mapping $cMap$.

Let S denote a set of configurations and $G_{\mathcal{A}}(S)$ denote the portion of the configuration graph containing S and all the transitions between the states in S. Let $S_{m,n}$ denote the set of configurations $\{(p, i) \mid p \in Q, m \leq i < n\}$. We say that the behaviour graph has a repetitive structure with *offset* o and *period* per for some $o, per \in \mathbb{N}$, if segments of the configuration graph $B_{\mathcal{A}}(S_{o,o+per}), B_{\mathcal{A}}(S_{o+per,o+2*per}), \ldots$ all have isomorphic images under the mapping $cMap$ and for all $o' < o$ and $per' < per$ this condition does not hold. Either all these images coincide in the behaviour graph $\mathtt{BG}_{\mathcal{A}}$, or they are all distinct subgraphs of $\mathtt{BG}_{\mathcal{A}}$. For any DROCA $\mathcal{A}$, the behaviour graph of $\mathcal{A}$ has a repetitive structure with an offset greater than zero and period greater than or equal to zero (Bruyère et al. (2022) (Theorem 1), Neider and Löding (2010)(Theorem 1)).

Example 4.3 (PrimeMatch)

Let S be a finite set of prime numbers. Consider the language

$$\mathtt{PrimeMatch}(S) = \bigcup_{i \in S} \left\{ a^n \mathfrak{p}_i b^{n-1} a \mid i \text{ divides } n \right\}$$

A VOCA over the pushdown alphabet $\Sigma = (\{a\}, \{b\} \cup \bigcup_{i \in S}\{\mathfrak{p}_i\}, \emptyset)$ recognising the language $\mathtt{PrimeMatch}(\{2, 3\})$ is given in Figure 4.1. Here, each $\mathfrak{p}_i$ denotes a unique symbol. The initial portion of the configuration graph and behaviour graph corresponding to this VOCA is given in Figure 4.2 and Figure 4.3, respectively. The behaviour graph can be obtained by merging equivalent states having the same counter value. Observe that the segment of the behaviour graph from counter value 3 to 8 repeats infinitely. In Figure 4.3, the behaviour graph shown has an offset 2 and a period $2 * 3 = 6$.

Every state in a behaviour graph has an associated counter value that reflects the counter value reached on reading the words in that equivalence class. The behaviour graph of a DROCA has a repetitive structure, as mentioned in Example 4.3. It consists of an initial part followed by a repeating segment that repeats indefinitely. The existing algorithm for learning DROCAs using counter value queries (Bruyère et al., 2022) and the learning algorithm for VOCAs (Neider and Löding, 2010) works by finding the repetitive structure of this behaviour graph. However, there are VOCAs for which the repetitive structure of the behaviour graph becomes evident only after observing the behaviour graph up to some exponentially large counter value. We will now prove this fact by providing a class of languages for which this is true.

Lemma 4.4 (Exponentially large period of behaviour graph)
There exist languages recognised by VOCAs, for which the period of the behaviour graph is exponential in the size of the VOCAs.

Proof. Consider a VOCA that recognises the language $\texttt{PrimeMatch}(S)$ (see Example 4.3), where S contains the first n prime numbers. It is easy to observe from Figure 4.1 that there is a VOCA $\mathcal{B}$ recognising $\texttt{PrimeMatch}(S)$ defined over the pushdown alphabet $\Sigma = (\Sigma_{call}, \Sigma_{ret}, \Sigma_{init})$ where $\Sigma_{call} = \{a\}, \Sigma_{ret} = \{b\} \cup \bigcup_{i \in S}\{\mathfrak{p}_i\}$ and $\Sigma_{init} = \emptyset$ with $|\mathcal{B}| = \sum_{i \in S} i + 2$. By the prime number theorem, the sum of the first n prime number is less than $n^2 \log n$. Therefore, we get that $|\mathcal{B}| < n^2 \log n$. Let c_0 denote the initial configuration of $\mathcal{B}$.

Let $j \in S$ and m be a positive integer such that m is divisible by j. For all $i \in S$, let s_i denote the state of $\mathcal{B}$ such that $c_0 \xrightarrow{a^m \mathfrak{p}_i} (s_i, m-1)$. For $i \in S$, $(s_i, m-1) \equiv (s_j, m-1)$ if and only if m is divisible by both i and j. Since $b^{m-1}a$ is the only word accepted from $(s_j, m-1)$, it should be accepted from $(s_i, m-1)$ also. If m is not divisible by i then, $(s_i, m-1) \not\equiv (s_j, m-1)$ since $b^{m-1}a$ is accepted from $(s_j, m-1)$ but not from $(s_i, m-1)$. Therefore, the set of configurations $\{(s_i, m-1) \mid i \in S\}$ are equivalent to each other if and only if $m = c \cdot \prod_{i \in S} i$ for some $c > 0$. i.e., m is divisible by all elements in S. This is in fact, the factor which defines the period of the behaviour graph. Similar to Figure 4.3, the behaviour graph of $\texttt{PrimeMatch}(S)$ will have an offset 2 and a period $\prod_{i \in S} i$, which is exponential in n. $\square$

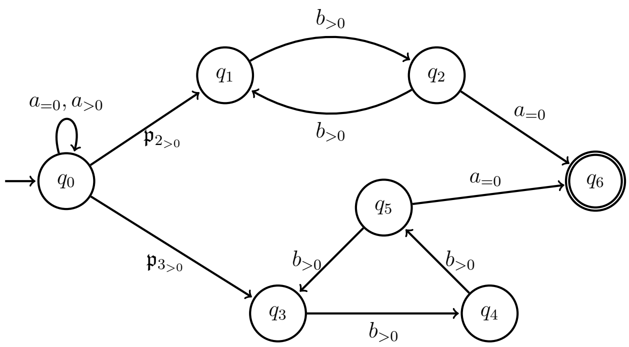


Fig. 4.1. A VOCA over the pushdown alphabet $\Sigma = (\{a\}, \{b\} \cup \bigcup_{i \in S}\{\mathfrak{p}_i\}, \emptyset)$ for the language `PrimeMatch`(S) for $S = \{2, 3\}$. Note that $(q_1, 6) \equiv (q_3, 6)$ but $(q_1, 3) \not\equiv (q_3, 3)$. Transitions not shown in the figure go to a non-final sink state q_7.

4.2.4 `MinOCA`: An overview

This chapter introduces a novel approach for active learning of DROCAs. The algorithm, hereafter referred to as `MinOCA`, learns a DROCA using a polynomial number of queries with respect to the size of the teacher's DROCA. However, the active learning framework differs from that introduced by Angluin (1987) in a few crucial aspects (see Section 4.3 for details). Similar to the work by Bruyère et al. (2022), we use an additional query type called *counter value query*. This allows the learner to ask for the counter value reached on reading a word in the DROCA. Furthermore, the learner has access to a minimal synchronous-equivalence query on the DROCA. The teacher returns true for this equivalence query if the learnt DROCA is counter-synchronous and equivalent to the teacher's DROCA. Otherwise, it returns a minimal word that violates this property. We consider only complete DROCAs in this chapter. Every incomplete DROCA can be made complete without changing its language by directing all the undefined transitions to a new non-final sink state.

In this framework, we give an algorithm that learns a minimal counter-synchronous DROCA. A key innovation in our approach is the use of a SAT solver for solving the NP-hard problem of finding a minimal separating DFA from a set of positive and negative samples. The solver, in conjunction with a modified version of `L*`, learns a characteristic DFA (see Definition 4.6). Subsequently, we use this characteristic DFA to construct a minimal counter-synchronous DROCA.

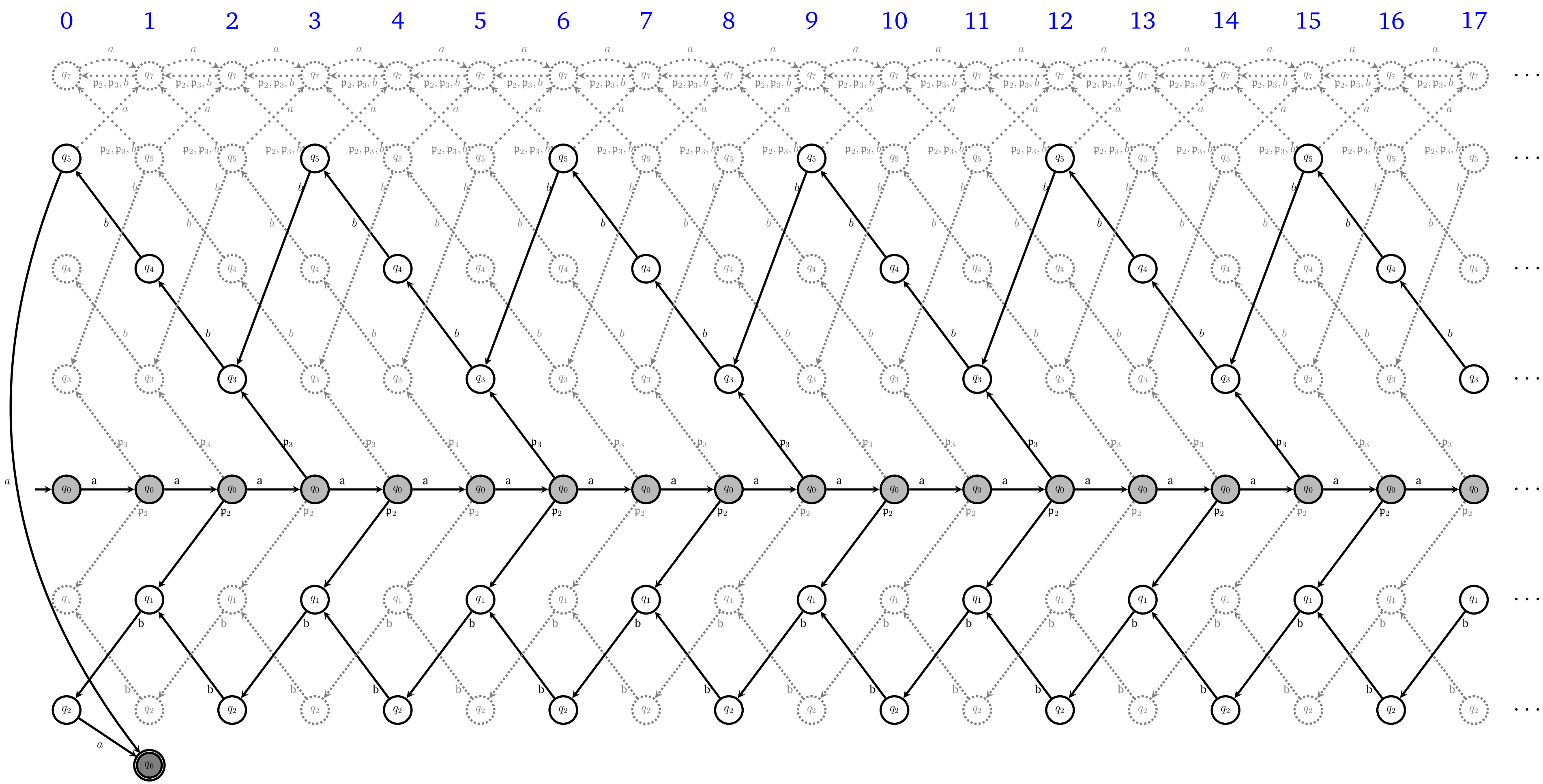


Fig. 4.2. The initial portion of the configuration graph of VOCA shown in Figure 4.1. The counter values of configurations are given at the top. Transitions not shown in the configuration graph lead to state q_7 with the appropriate counter value. For a given counter value, all configurations having the same color are equivalent. Configurations drawn with dotted circles do not have a word that is accepted from them.

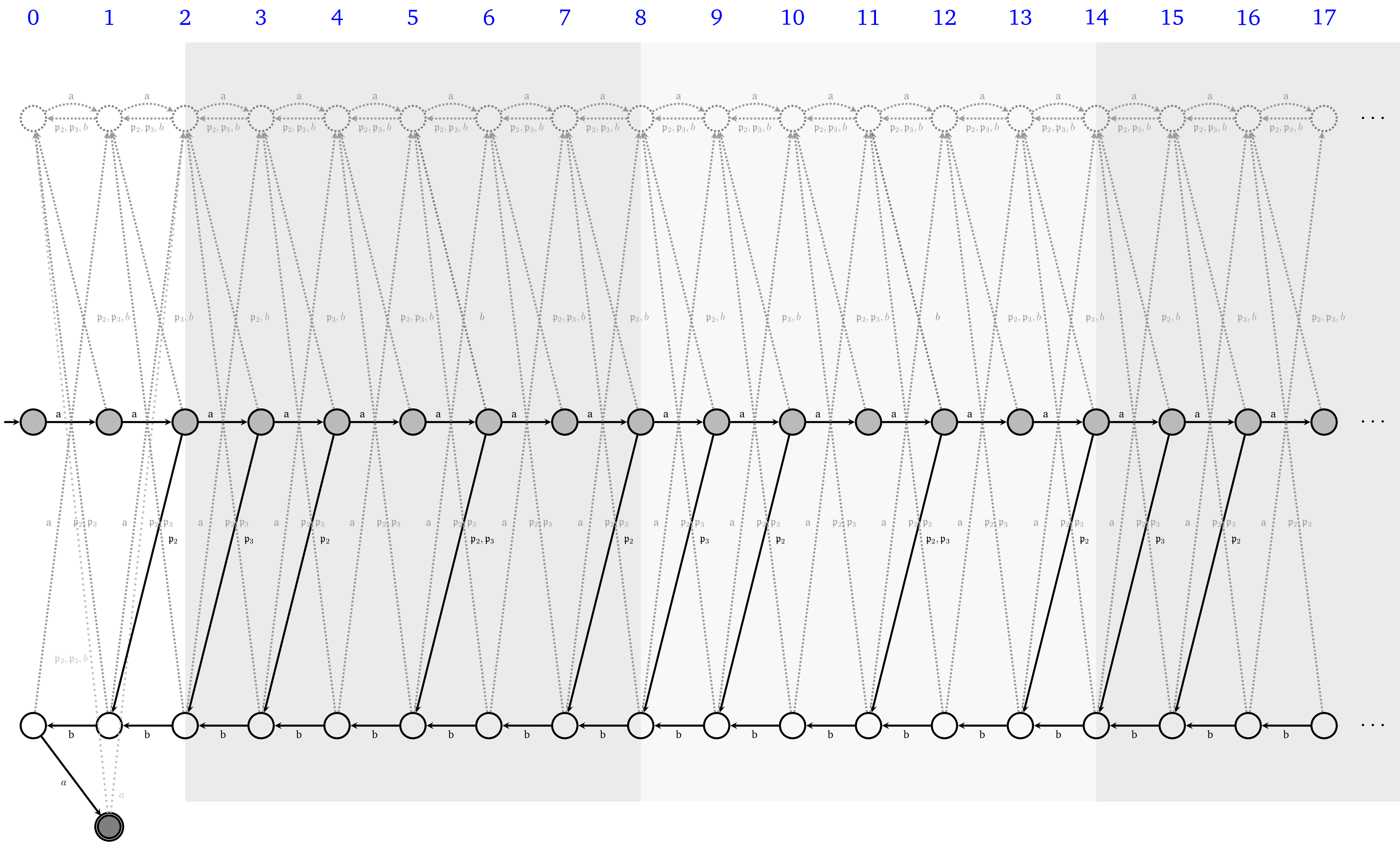


Fig. 4.3. The initial portion of the behaviour graph of VOCA shown in Figure 4.1. Each state in the behaviour graph represents an equivalence class of the refined Myhill-Nerode congruence. i.e., equivalent configurations having the same counter value in the configuration graph will be a single state in the behaviour graph. The counter values corresponding to each state are given at the top. States drawn with dotted circles do not have a word that is accepted from them.

Observe that in Figure 4.3, all states drawn with dotted circles belong to the same equivalence class under the standard Myhill-Nerode congruence.

Justification for Using SAT Solver

Let us first consider the problem of minimisation of VOCAs. The existence of a minimal finite automata is a classic result in finite automata theory. It is also well known that DFA minimisation can be done in polynomial time. However, VOCAs differs from DFAs in this aspect. For a language recognised by a VOCA, a canonical minimal VOCA recognising this language does not exist. Thus, the problem of minimisation of VOCAs is to produce at least one VOCA of minimal size that recognises the target language. Minimisation and learning of VOCAs are related. Consider an algorithm that learns a minimal VOCA. This algorithm can then be used for minimising VOCA. Hence, a polynomial time algorithm to learn a minimal VOCA implies a polynomial time algorithm for minimisation.

However, it was observed by Michaliszyn and Otop (2022) that given a VOCA $\mathcal{A}$, and an $n > 0$, deciding whether there exists a VOCA that is equivalent to $\mathcal{A}$ with at most n states is NP-complete (Proposition 1, Michaliszyn and Otop (2022)). This follows from the result by Gauwin et al. (2020) that minimising visibly pushdown automata is NP-complete. Therefore, unless $\mathsf{P} = \mathsf{NP}$, learning a minimal visibly one-counter automaton (VOCA) cannot be done in polynomial time.

Justification for Using Minimal Synchronous-Equivalence Query

One significant bottleneck in learning DROCAs is the equivalence test by the teacher. Given two DROCAs with number of states less than some $\mathsf{K} \in \mathbb{N}$, the best-known algorithm for equivalence check takes $\mathcal{O}(\mathsf{K}^{26})$ time[2]. This is impractical for real-world applications. Bruyère et al. (2022) were the first to pursue a practical application of learning DROCAs. Due to the difficulty in checking the equivalence of DROCAs, they used an incomplete equivalence that checks in their implementation. Their equivalence check works as follows. First, they will check whether the configuration graphs up to counter value $t = \mathsf{K}^4$ of both the DROCAs

[2]This polynomial is obtained from the equivalence result of DROCAs from Chapter 5.

are equivalent. If they are equivalent, then with probability $\frac{1}{2}$, the algorithm performs an equivalence check of the configuration graphs up to counter value $t = t + k$ for some fixed $k \in \mathbb{N}$ and with probability $\frac{1}{2}$, outputs that the given DROCAs are equivalent. This process repeats until the algorithm finds a counter-example or outputs that the given DROCAs are equivalent. This procedure might say that two non-equivalent DROCAs are equivalent if the length of the minimal counter-example is large.

To mitigate this, we use the minimal synchronous-equivalence queries that run in $\mathcal{O}(\alpha(\mathsf{K}^5)\mathsf{K}^5)$ time[3]. Using this equivalence check, we obtain significantly smaller counter-examples on equivalence queries if the learnt DROCA is not the right one. Our equivalence queries are also on models whose size is less than or equal to a minimal counter-synchronised DROCA. On an equivalence query, if the DROCA presented by the learner is not equivalent to the teacher's DROCA, then the teacher returns a word z that satisfies one of the following conditions: (1) The run on z reaches configurations with different counter values in the learnt DROCA and the teacher's DROCA or (2) The word z is accepted by exactly one among the learnt DROCA and the teacher's DROCA.

4.2.5 Comparison with Existing Methods

Our approach (`MinOCA`) differs fundamentally from the existing methods by eliminating the need to observe the automaton's behaviour up to exponentially large counter values that require exponentially many queries. We propose an algorithm for learning DROCAs using only a polynomial number of queries. Furthermore, unlike existing techniques that learn exponentially large DROCAs, our algorithm always learns an equivalent counter-synchronous DROCA with the minimal number of states.

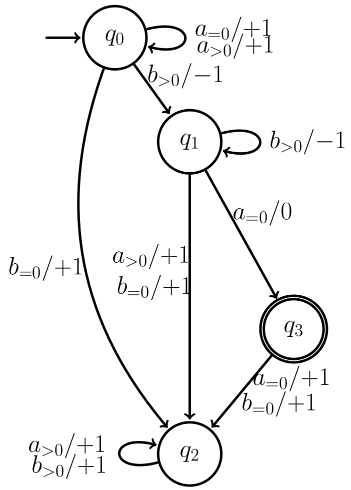
Now, we provide two examples of DROCAs learnt using `MinOCA` and `BPS`. In Figure 4.4, the DROCA learnt by `MinOCA` is a minimal one that is counter-synchronised and equivalent to the input DROCA. The DROCA learnt by `BPS` is equivalent but is neither counter-synchronous with respect to the input nor

[3]This polynomial is obtained from the equivalence result of counter-synchronised DROCAs from Chapter 3.

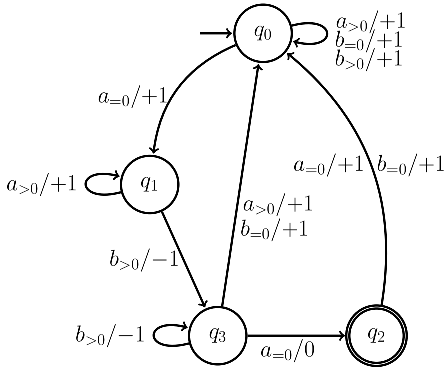
minimal. It is also not complete. However, the equivalence check used by BPS is less powerful in the sense that they cannot check whether the learnt DROCA is counter-synchronous with the teacher's DROCA. In Figure 4.5, the number of states in the output DROCA is smaller for BPS compared to the output of MinOCA. However, the DROCA learnt by MinOCA is a minimal one that is counter-synchronised and equivalent to the input DROCA. The DROCA learnt by BPS is equivalent but not counter-synchronous with respect to the input DROCA.

Observations

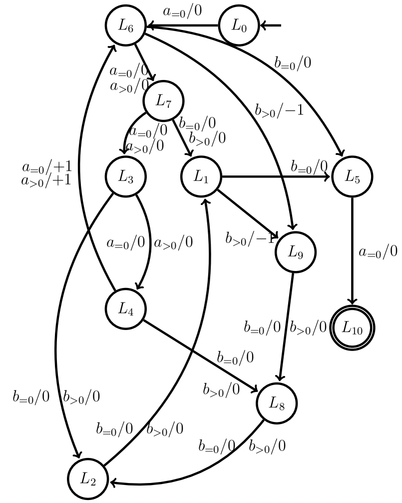
We have observed that minimal synchronous-equivalence queries can be replaced with minimal equivalence queries. However, in that case, the number of queries needed for learning will depend on the polynomial bound obtained from the result by Böhm and Göller (2011) on the length of the minimal counter-example that distinguishes two DROCAs. The minimal synchronous-equivalence queries are much faster than equivalence queries due to our result on the equivalence of counter-synchronous DROCAs (see Theorem 3.5). Similarly, a minimal equivalence query can be replaced with a partial equivalence query. One can simulate the teacher returning a minimal counter-example in polynomial time by asking partial equivalence queries starting from limit 0 and incrementing the limit until the teacher returns a counter-example. Similarly, one can replace a partial equivalence query with a minimal equivalence query. The assumption that the teacher consistently offers the minimal counter-example obviates the necessity for partial equivalence queries, as this information can be deduced from the counter-example length.



(a) Input DROCA.

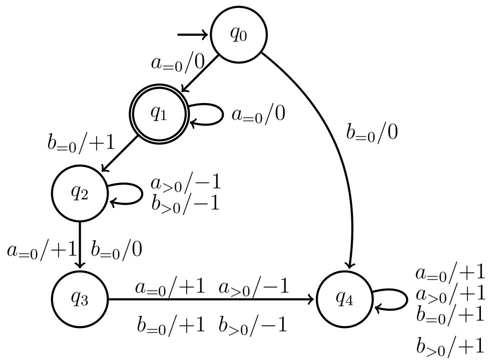


(b) Learnt by MinOCA.

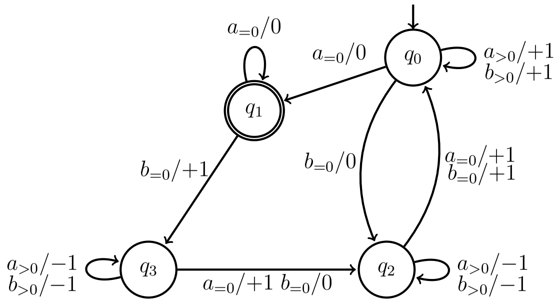


(c) Learnt by BPS.

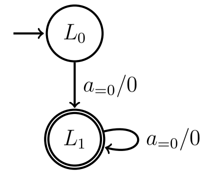
Fig. 4.4. The input DROCA recognises the language $\{a^n b^n a \mid n > 0\}$.



(a) Input DROCA.



(b) Learnt by MinOCA.



(c) Learnt by BPS.

Fig. 4.5. The input DROCA recognises the language $\{w \in \{a, b\}^* \mid w$ does not contain a $b\}$.

4.2.6 Experiments

We evaluate an implementation of our algorithm against randomly generated DROCAs and compare the results obtained with the existing technique by Bruyère et al. (2022). Experiments were conducted on randomly generated DROCAs with number of states ranging from 2 to 15 and the input alphabet size varying from 2 to 5. The results indicate that `MinOCA` outperforms BPS in terms of the number of successfully learnt languages within the given timeout.

The remainder of this chapter is organised as follows: Section 4.3 details our learning algorithm for DROCAs, Section 4.4 covers the implementation details and presents our experimental results. Finally, Section 4.5 summarises our work and suggests future research directions.

4.3 `MinOCA`: The Proposed Method

In this section, we give an L*-like algorithm for active learning of deterministic real-time one-counter automaton (DROCA). In this framework, we have a *learner* and a *teacher*. The learner aims to construct a DROCA that recognises the same language as the teacher's DROCA (call it $\mathcal{A}$). The teacher can answer membership queries, counter value queries and minimal synchronous-equivalence queries by the learner.

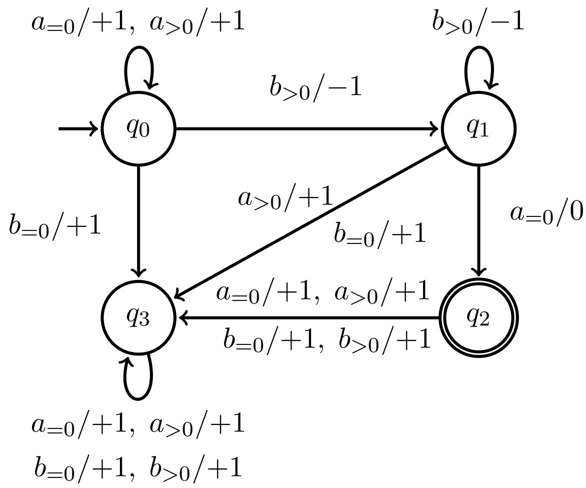


Fig. 4.6. A DROCA recognising the language $\{a^n b^n a \mid n > 0\}$.

4.3.1 Observation Table

Our algorithm maintains an observation table $C = (\mathcal{P}, \mathcal{S}, Mem, \mathsf{ce}\restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, Actions)$ over the input alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ for some $k \in \mathbb{N}$ where $\mathcal{P} \subseteq \Sigma^*$ is a nonempty prefix-closed set of strings, $\mathcal{S} \subseteq \Sigma^*$ is a nonempty suffix-closed set of strings, $Mem : (\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S} \to \{0,1\}$ is a function that indicates whether words belong to the language, $\texttt{ce} \restriction_{\mathcal{P}\cup\mathcal{P}\Sigma} : \mathcal{P} \cup \mathcal{P}\Sigma \to \mathbb{N}$ is the function $\texttt{ce}$ with domain restricted to the set $\mathcal{P} \cup \mathcal{P}\Sigma$, and $Actions : (\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S} \to \{0,1\} \times \{0,1,-1\}^k$ is a function representing the sign of the counter value reached and the counter-actions on every letter after reading a word. Given $w \in (\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$, $Mem(w)$ is equal to 0 (resp. 1) if $\mathcal{A}(w)$ is equal to 0 (resp. 1), and $Actions(w) = (sign(\texttt{ce}(w)), \texttt{ce}(w\sigma_1) - \texttt{ce}(w), \ldots, \texttt{ce}(w\sigma_k) - \texttt{ce}(w))$. Given $w_1, w_2 \in (\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$ we say that $Actions(w_1)$ is not similar to $Actions(w_2)$ if $sign(\texttt{ce}(w_1)) = sign(\texttt{ce}(w_2))$ and $Actions(w_1) \neq Actions(w_2)$. We use $Actions(w_1) \not\sim Actions(w_2)$ to denote this. We say that $Actions(w_1)$ is similar to $Actions(w_2)$ otherwise. The observation table initially has $\mathcal{P} = \mathcal{S} = \{\varepsilon\}$ and is augmented as the algorithm runs.

An observation table can be viewed as a two-dimensional array with rows labelled with elements of $\mathcal{P} \cup \mathcal{P}\Sigma$, columns labelled by elements of $\mathcal{S}$ and an additional column labelled $\texttt{ce}$. The column $\texttt{ce}$ contains the counter value reached on reading the word labelling a row (see Table 4.1). For any $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, the entry in row p and column s is equal to $(Mem(ps), Actions(ps))$ and $cell(p)$ denotes the finite function f_p from $\mathcal{S}$ to $\{0,1\} \times \{0,1\} \times \{0,1,-1\}^k$ defined by $f_p(s) = (Mem(ps), Actions(ps))$. We use $row(p)$ to denote $(\texttt{ce}(p), cell(p))$. For $p, p' \in \mathcal{P} \cup \mathcal{P}\Sigma$, we say $row(p)$ is equal to $row(p')$ (denoted by $row(p) = row(p')$), if $cell(p) = cell(p')$ and $\texttt{ce}(p) = \texttt{ce}(p')$.

Now, we introduce the notion of d-closed and d-consistent observation tables for any $d \in \mathbb{N}$. This is similar to the notion of closed and consistency used by Angluin (1987), but it also takes into account the counter values.

Definition 4.5 (d-closed and d-consistent)

Let $d \in \mathbb{N}$ and $(\mathcal{P}, \mathcal{S}, Mem, \texttt{ce} \restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, Actions)$ be an observation table.

1. The observation table is said to be *d-closed* if for all $p' \in \mathcal{P}\Sigma$ with $\texttt{ce}(p') \leq d$ there exists $p \in \mathcal{P}$ such that $row(p) = row(p')$. The observation table is otherwise said to be not d-closed.

2. The observation table is said to be *d-consistent* if for any $p, q \in \mathcal{P}$, $\texttt{ce}(p) = \texttt{ce}(q) \leq d$ and $row(p) = row(q)$ implies that for all $a \in \Sigma$, $row(pa) = row(qa)$. We say that the observation table is not d-consistent otherwise.

Consider the observation table given in Table 4.1. This table is 1-closed but not 2-closed. This is because of the presence of the words aa, ba and bb in $\mathcal{P}\Sigma$. The

		ce	ε		a	
			Mem	*Actions*	Mem	*Actions*
$\mathcal{P}$	ε	0	0	$(0,+1,+1)$	0	$(1,+1,-1)$
	a	1	0	$(1,+1,-1)$	0	$(1,+1,-1)$
	ab	0	0	$(0,0,+1)$	1	$(0,+1,+1)$
	aba	0	1	$(0,+1,+1)$	0	$(1,+1,+1)$
	b	1	0	$(1,+1,+1)$	0	$(1,+1,+1)$
$\mathcal{P}\Sigma$	aa	2	0	$(1,+1,-1)$	0	$(1,+1,-1)$
	abb	1	0	$(1,+1,+1)$	0	$(1,+1,+1)$
	abaa	1	0	$(1,+1,+1)$	0	$(1,+1,+1)$
	abab	1	0	$(1,+1,+1)$	0	$(1,+1,+1)$
	ba	2	0	$(1,+1,+1)$	0	$(1,+1,+1)$
	bb	2	0	$(1,+1,+1)$	0	$(1,+1,+1)$

Table 4.1. An observation table corresponding to the DROCA given in Figure 4.6 recognising the language $\{a^n b^n a \mid n > 0\}$. Here, $\mathcal{P} = \{\varepsilon, a, ab, aba, b\}$ and $\mathcal{S} = \{\varepsilon, a\}$.

given table is trivially 1-consistent as there are no equal rows in $\mathcal{P}$.

4.3.2 Constructing a DROCA from an Observation Table

We introduce the notion of a characteristic DFA. Given an alphabet Σ, we define the modified alphabet $\widetilde{\Sigma} = \bigcup_{a\in\Sigma}\{a^0, a^1\}$. Note that a^0 and a^1 are special symbols and should not be confused with ε and a respectively. For a DROCA $\mathcal{A}$, we define the function $\texttt{Enc}_{\mathcal{A}} : \Sigma^* \to \widetilde{\Sigma}^*$ as follows: For $w \in \Sigma^+$, $\texttt{Enc}_{\mathcal{A}}(w) = \tilde{w}$, such that for all $i \in [0, |w|-1]$, $\tilde{w}[i] = w[i]^{sign(\texttt{ce}_{\mathcal{A}}(w_{[0\cdots i-1]}))}$. Also $\texttt{Enc}_{\mathcal{A}}(\varepsilon) = \varepsilon$.

Definition 4.6 (Characteristic DFA)
Let $\mathcal{A} = (Q, \Sigma, q_0, \delta_0, \delta_1, F)$ be a DROCA. The *characteristic* DFA $\mathcal{D}_{\mathcal{A}}$ of $\mathcal{A}$ over the modified alphabet $\widetilde{\Sigma}$ is $\mathcal{D}_{\mathcal{A}} = (Q, \widetilde{\Sigma}, q_0, \delta, F)$ where, for all $q \in Q$ and $a \in \Sigma$, $\delta(q, a^0) = p$ (resp. $\delta(q, a^1) = p$) if and only if $\delta_0(q, a) = (p, c)$ (resp. $\delta_1(q, a) = (p, c)$) for some $p \in Q$ and $c \in \{0, 1, -1\}$.

Figure 4.7 shows the characteristic DFA over the modified alphabet $\widetilde{\Sigma}$ corresponding to the DROCA given in Figure 4.6. We can construct a DROCA from an characteristic DFA if we have access to the counter-actions.

4.3.2.1 Constructing a Characteristic DFA using a SAT solver

Let $C = (\mathcal{P}, \mathcal{S}, Mem, \texttt{ce}\restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, Actions)$ be an observation table. We provide C as input to the procedure `ConstructAutomaton` (see Algorithm 2) and obtain a

Algorithm 2: Algorithm to construct a characteristic DFA from an observation table.

Procedure *ConstructAutomaton*()

Input : Observation table $C = (\mathcal{P}, \mathcal{S}, Mem, \texttt{ce}\restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, Actions)$

Output : characteristic DFA $\mathcal{D}_C$

Initialise $Pos = \emptyset,\ Neg = \emptyset$,

$Operations = \{Actions(w) \mid w = p.s$ for some $p \in \mathcal{P}$ and $s \in \mathcal{S}\}$.

foreach p *in* $\mathcal{P}$ **do**

  foreach s *in* $\mathcal{S}$ **do**

    if $Mem(ps) = 1$ **then** add $\texttt{Enc}(ps)$ to Pos.

    else add $\texttt{Enc}(ps)$ to Neg.

    add $\texttt{Enc}(ps) \cdot Actions(ps)$ to Pos.

    foreach $op \in Operations$ **do**

      if $Actions(ps) \not\sim op$ **then** add $\texttt{Enc}(ps) \cdot op$ to Neg.

    end

  end

end

$\mathcal{B}$= `find_dfa`(Pos, Neg).

Remove transitions on $Operations$ from $\mathcal{B}$ to obtain a DFA $\mathcal{D}_C$.

return $\mathcal{D}_C$.

end

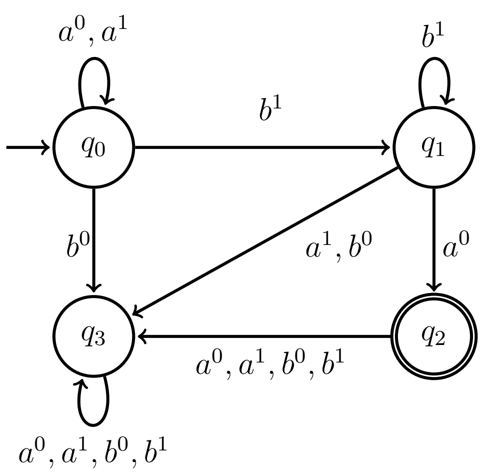


Fig. 4.7. The characteristic DFA corresponding to the DROCA shown in Figure 4.6.

DFA $\mathcal{D}_C$ over the modified alphabet $\widetilde{\Sigma}$. The DFA $\mathcal{D}_C$ satisfies the following two properties:

1. for all $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, $\texttt{Enc}_{\mathcal{A}}(ps) \in \mathcal{L}(\mathcal{D}_C)$ if and only if $Mem(ps) = 1$.

2. for any $p_1, p_2 \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s_1, s_2 \in \mathcal{S}$, if the run on $\texttt{Enc}_{\mathcal{A}}(p_1 s_1)$ and $\texttt{Enc}_{\mathcal{A}}(p_2 s_2)$ reaches the same state in $\mathcal{D}_C$ then $Actions(p_1 s_1)$ is similar to $Actions(p_2 s_2)$.

We create two sets of words, Pos and Neg. The DFA $\mathcal{D}_C$ will accept all words in Pos and reject all words in Neg. For any $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, we add $\texttt{Enc}_{\mathcal{A}}(ps)$ to Pos (resp. Neg) if and only if $Mem(ps) = 1$ (resp. 0). This ensures condition 1. To ensure condition 2, we add words over a larger alphabet $\widetilde{\Sigma} \cup Operations$, where $Operations = \{Actions(w) \mid w = ps$ for some $p \in \mathcal{P}$ and $s \in \mathcal{S}\}$. For any $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, we add $\texttt{Enc}_{\mathcal{A}}(ps) \cdot Actions(ps)$ to Pos and for all $op \in Operations$ where $Actions(ps) \not\sim op$, we add $\texttt{Enc}_{\mathcal{A}}(ps) \cdot op$ to Neg. We find the minimal separating DFA for the sets Pos and Neg using the function `find_dfa`. We remove the transitions labelled by the letters from $Operations$ in this minimal separating DFA to obtain $\mathcal{D}_C$. Given an observation table, the sets Pos and Neg can be constructed in polynomial time. Every state of $\mathcal{A}$ can add two elements to the set $Operations$ – one corresponding to the counter-actions on reading letters from counter value zero and one for counter-actions on reading letters from positive counter value. Hence the cardinality of this set is at most $2|\mathcal{A}|$. In our implementation, we use the algorithm by Dell'Erba et al. (2024) that uses a SAT solver to learn a minimal separating DFA. However, any algorithm that finds a minimal separating DFA will suffice (Heule and Verwer, 2010; Leucker and Neider, 2012; Neider, 2012).

Consider the following example of constructing a characteristic DFA from the observation table given in Table 4.1. In this case, the set of distinct $Actions$ in the observation table is $Operations = \{(0, +1, +1), (1, +1, -1), (0, 0, +1), (1, +1, +1)\}$. For convenience, let us use c, d, g, h to denote $(0, +1, +1), (1, +1, -1), (0, 0, +1)$ and $(1, +1, +1)$ repectively. Given two tuples $x, y \in Operations$, we say that x is not similar to y if $x[0] = y[0]$ but either $x[1] \neq y[1]$ or $x[2] \neq y[2]$. If the counter-actions of a row $p \in \mathcal{P}$ is some $t \in Operations$, then we add the string $\texttt{Enc}(p) \cdot t$ to the set Pos and for all $x \in Operations$ with x not similar to t, we add $\texttt{Enc}(p) \cdot t$ to the set Neg. In Table 4.1, since the counter-action of ε is c, we add the string $\texttt{Enc}(\varepsilon) \cdot c$ to the set Pos and since c is not similar to g we add the string $\texttt{Enc}(\varepsilon) \cdot g$ to the set Neg. We repeat the same procedure for all the rows in the table based on their counter-actions. The set Pos and Neg obtained from the observation table in Table 4.1 is given below.

$Pos = \{a^0b^1a^0, c, a^0d, a^0b^1g, a^0b^1a^0c, b^0h, a^0a^1d, a^0b^1b^0h, a^0b^1a^0a^0h, a^0b^1a^0b^0h\}$, and

$Neg = \{\varepsilon, a^0, a^0b^1, b^0, a^0a^1, a^0b^1b^0, a^0b^1a^0a^0, a^0b^1a^0b^0, g, a^0h, a^0b^1c, a^0b^1a^0g, b^0d, a^0a^1h,$ $a^0b^1b^0d, a^0b^1a^0a^0d, a^0b^1a^0b^0d\}$.

We can now use a SAT solver to find the minimal separating DFA for the the set Pos and Neg. In the next subsection, we observe that $\mathcal{D}_C$ is a characteristic DFA.

4.3.2.2 Constructing a DROCA from a Characteristic DFA

Let $C = (\mathcal{P}, \mathcal{S}, Mem, \mathtt{ce}\restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, Actions)$ be an observation table over the input alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$. The following lemma states that we can construct a DROCA $\mathcal{B}_C$ from C such that it agrees with the observation table C. The idea is to use the observation table to assign counter-actions to transitions of $\mathcal{D}_C$.

Lemma 4.7
Given an observation table C of a DROCA $\mathcal{A}$, we can construct a DROCA $\mathcal{B}_C$ with $|\mathcal{B}_C| \le |\mathcal{A}|$ such that for all $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, $\mathcal{B}_C(ps) = Mem(ps)$ and $\mathtt{ce}_{\mathcal{B}_C}(ps) = \mathtt{ce}_{\mathcal{A}}(ps)$.

Proof. Let $\mathcal{D}_C = (Q, \widetilde{\Sigma}, q_0, \delta, F)$ be a DFA obtained by giving C as input to the function $\mathtt{ConstructAutomaton}$. For all $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S} \cup \mathcal{S}\Sigma$, we can find $\mathtt{ce}(ps)$ from C. We define the DROCA $\mathcal{B}_C = (Q, \Sigma, q_0, \delta_0, \delta_1, F)$ where δ_0 and δ_1 are specified as follows. For all $q \in Q$, $a \in \Sigma$, $\delta_0(q, a) = (\delta(q, a^0), c)$ for some $c \in \{0, 1\}$, if there exists $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$ such that $\mathcal{D}_C$ on reading $\mathtt{Enc}_{\mathcal{A}}(ps)$ reaches the state q with $\mathtt{ce}(ps) = 0$ and $\mathtt{ce}(psa) = c$. Similarly, for all $q \in Q$, $a \in \Sigma$, $\delta_1(q, a) = (\delta(q, a^1), c)$ for some $c \in \{0, 1, -1\}$, if there exists $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$ such that $\mathcal{D}_C$ on reading $\mathtt{Enc}_{\mathcal{A}}(ps)$ reaches the state q with $\mathtt{ce}(ps) > 0$ and $\mathtt{ce}(psa) = c$. If there are any transitions that do not have a counter-action, it means that there is no word in the observation table that took that transition. One can assign any arbitrary counter-action for these transitions. Note that since the function $\mathtt{ConstructAutomaton}$ returns a minimal separating DFA, the number of states in $\mathcal{B}_C$ is at most $|\mathcal{A}|$. Otherwise, it contradicts the minimality of $\mathcal{D}_C$, since the characteristic DFA of $\mathcal{A}$ contains only $|\mathcal{A}|$ many states.

We know that for any word $w, w' \in \Sigma^*$, if $\mathcal{D}_C$ on reading w and w' reaches the same state, then $Actions(w)$ is similar to $Actions(w')$. We assign counter-actions to transitions from a state in $\mathcal{B}_C$ based on the counter-actions of words that reach that state in $\mathcal{D}_C$. Since all words that reach a state have similar counter-actions, this assignment will be consistent. By construction, for all $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, $\mathtt{ce}_{\mathcal{B}_C}(ps) = \mathtt{ce}_{\mathcal{A}}(ps)$. This also ensures that $\mathtt{Enc}_{\mathcal{A}}(ps) = \mathtt{Enc}_{\mathcal{B}_C}(ps)$. Note that, $\mathcal{D}_C$ on reading $\mathtt{Enc}_{\mathcal{A}}(ps)$ reaches a final state if and only if $ps \in \mathcal{L}(\mathcal{A})$. Since $\mathtt{Enc}_{\mathcal{A}}(ps) = \mathtt{Enc}_{\mathcal{B}_C}(ps)$, we get that $\mathcal{B}_C$ reaches a final state on reading ps if and only if $ps \in \mathcal{L}(\mathcal{A})$. Therefore, for all $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, $\mathcal{B}_C(ps) = 1$ if and only if $Mem(ps) = 1$. □

4.3.3 `MinOCA`: The Learning Algorithm

The algorithm to learn a DROCA that accepts the same language as a given DROCA $\mathcal{A}$ is described in Algorithm 3. Initially, it sets up an observation table using empty strings and incrementally refines this table to distinguish states of the unknown DROCA. The process iteratively increases an integer value d and uses membership and counter value queries to construct a d-closed and d-consistent observation table $C = (\mathcal{P}, \mathcal{S}, Mem, \texttt{ce}\restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, Actions)$ (see lines 3 – 13 of Algorithm 3). This part resembles the $\texttt{L}^*$ algorithm. Making the observation table d-closed will result in the addition of new rows to $\mathcal{P}$, and making it d-consistent will result in the addition of new columns to $\mathcal{S}$. We construct a DROCA from a d-closed and d-consistent observation table C using Lemma 4.7 and ask a minimal synchronous-equivalence query. If the teacher provides a counter-example, then all its prefixes are added to $\mathcal{P}$, and the value of d is updated to the height of the counter-example, if it is more than d. The table is then extended until it becomes d-closed and d-consistent. This process continues until the correct DROCA is learnt.

4.3.4 Example: `MinOCA` in Action.

We assume the learner is trying to learn the DROCA given in Figure 4.8 from the teacher. The learner uses Algorithm 3 (`MinOCA`) to learn this language. First, the learner initialises $\mathcal{P}$ and $\mathcal{S}$ to $\{\varepsilon\}$ and d to 1. The initial observation table $C = (\mathcal{P}, \mathcal{S}, Mem, \texttt{ce}\restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, Actions)$ is built as shown in Table 4.2 using membership and counter value queries.

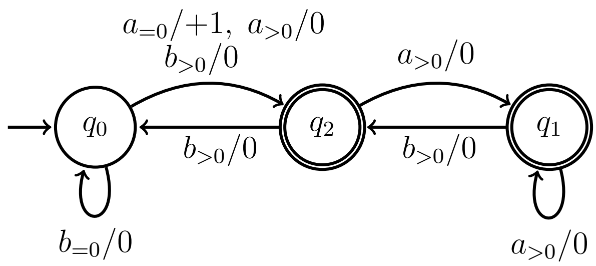


Fig. 4.8. Example: The DROCA under learning.

Algorithm 3: *MinOCA*: DROCA Learning algorithm.

Require : The teacher knowing a DROCA $\mathcal{A}$

Ensure : A DROCA accepting the same language as $\mathcal{A}$ is returned

Initialise $\mathcal{P}$ and $\mathcal{S}$ to $\{\varepsilon\}$, d to 1 and C to $(\mathcal{P}, \mathcal{S}, \mathit{Mem}, \mathsf{ce}\restriction_{\mathcal{P}\cup\mathcal{P}\Sigma}, \mathit{Actions})$.
repeat
  while *C is not d-closed or not d-consistent* **do**
    if *C is not d-closed* **then**
      Find $p \in \mathcal{P}$, $a \in \Sigma$ such that $\mathsf{ce}(pa) \leq d$, $\mathit{row}(pa) \neq \mathit{row}(p')$ for all $p' \in \mathcal{P}$.
      Add pa to $\mathcal{P}$.
    end
    if *C is not d-consistent* **then**
      Find $p, q \in \mathcal{P}$, $a \in \Sigma$, $s \in \mathcal{S}$ such that $\mathsf{ce}(p) = \mathsf{ce}(q) \leq d$, $\mathit{row}(p) = \mathit{row}(q)$, and $\mathit{Mem}(pas) \neq \mathit{Mem}(qas)$ or $\mathit{Actions}(pas) \neq \mathit{Actions}(qas)$).
      Add as to $\mathcal{S}$.
    end
    Extend Mem and $\mathit{Actions}$ to $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$, using MQ and CV queries.
  end
  Construct a DROCA $\mathcal{B}_C$ from C using Lemma 4.7.
  Ask minimal synchronous-equivalence query $\mathsf{MSQ}_{\mathcal{A}}(\mathcal{B}_C)$.
  if *teacher gives a counter-example z* **then**
    Add z and all its prefixes to $\mathcal{P}$.
    Extend Mem and $\mathit{Actions}$ to $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$ using MQ and CV queries.
    if $\mathit{height}_{\mathcal{A}}(z) > d$ **then** $d = \mathit{height}_{\mathcal{A}}(z)$.
  end
until *teacher replies* yes *to a minimal synchronous-equivalence query*;
Halt and output $\mathcal{B}_C$.

	ce	ε	
		Mem	*Actions*
ε	0	0	$(0,+1,0)$
a	1	1	$(1,0,0)$
b	0	0	$(0,+1,0)$

Table 4.2. Initial observation table with $\mathcal{P} = \mathcal{S} = \{\varepsilon\}$.

This observation table is not d-closed for $d = 1$, since $\mathsf{ce}(a) = d$ and $\mathit{row}(a) \neq$

$row(p)$ for all $p \in \mathcal{P}$ with $\texttt{ce}(p) = d$. Therefore, we add a to $\mathcal{P}$ and fill the observation table using membership and counter value queries (see Table 4.3).

	ce	ε	
		Mem	*Actions*
ε	0	0	$(0, +1, 0)$
a	1	1	$(1, 0, 0)$
b	0	0	$(0, +1, 0)$
aa	1	1	$(1, 0, 0)$
ab	1	0	$(1, 0, 0)$

Table 4.3. An observation table with $\mathcal{P} = \{\varepsilon, a\}$ and $\mathcal{S} = \{\varepsilon\}$ that is not 1-closed.

However, the observation table is still not d-closed since $\texttt{ce}(ab) = d$ and $row(ab) \neq row(p)$ for all $p \in \mathcal{P}$, with $\texttt{ce}(p) = d$. Therefore, we add ab to $\mathcal{P}$ and fill the observation table using membership and counter value queries. The observation table after this step is shown in Table 4.4.

	ce	ε	
		Mem	*Actions*
ε	0	0	$(0, +1, 0)$
a	1	1	$(1, 0, 0)$
ab	1	0	$(1, 0, 0)$
b	0	0	$(0, +1, 0)$
aa	1	1	$(1, 0, 0)$
aba	1	1	$(1, 0, 0)$
abb	1	1	$(1, 0, 0)$

Table 4.4. A 1-closed and 1-consistent observation table with $\mathcal{P} = \{\varepsilon, a, ab\}$ and $\mathcal{S} = \{\varepsilon\}$.

This observation table is both d-closed and d-consistent. We use the procedure `ConstructAutomaton` using this observation table as input to obtain a characteristic DFA. Let x denote the tuple $(0, +1, 0)$ and y denote the tuple $(1, 0, 0)$. The tuple x (resp. y) represents the counter-actions on reading symbols from zero (resp positive) counter value. The sets Pos and Neg created inside the function `ConstructAutomaton` are given below.

$$Pos = \{x, a^0, a^0y, b^0x, a^0a^1, a^0a^1y, a^0b^1y, a^0b^1a^1, a^0b^1a^1y, a^0b^1b^1, a^0b^1b^1y\}.$$
$$Neg = \{\varepsilon, b^0, a^0b^1\}.$$

The characteristic DFA returned by `ConstructAutomaton` is shown in Figure 4.9.

A DROCA is constructed from this characteristic DFA by removing from it the transitions on x and y, and by assigning counter-actions to the rest of the transitions

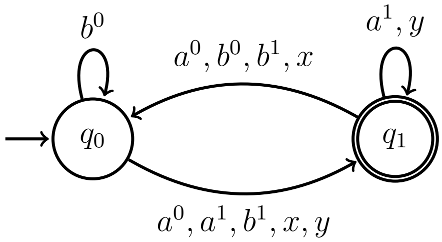


Fig. 4.9. The characteristic DFA obtained from ConstructAutomaton using Table 4.4 as input.

based on the tuples x and y. For instance, the tuple $x = (0, +1, 0)$ is accepted from state q_0. This means that the counter-action on a transition on a (resp. b) from counter value 0 from this state has counter-action $+1$ (resp. 0). The resultant DROCA is shown in Figure 4.10.

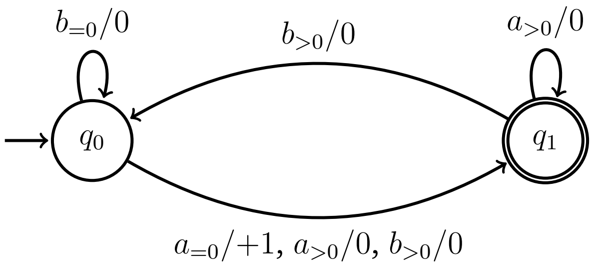


Fig. 4.10. The DROCA obtained from the characteristic DFA in Figure 4.9.

The learner now asks a minimal synchronous-equivalence query with this DROCA as input. This DROCA is counter-synchronous with the teachers DROCA but not equivalent. Therefore, the teacher returns a minimal counter-example. Let us assume that the counter-example returned by the teacher is aab. We add aab and all its prefixes to $\mathcal{P}$ and extend the observation table using membership and counter value queries. The observation table obtained after this step is shown in Table 4.5. Since $height(aab) = 1 = d$, we keep the value of d unchanged.

This observation table is not d-consistent since $row(aa) = row(aab)$ but $aa \cdot b \cdot \varepsilon \neq aab \cdot b \cdot \varepsilon$. Therefore, we add b to $\mathcal{S}$ and extend the table using membership and counter value queries. The observation table obtained after this step is shown in Table 4.6. This observation table is both d-closed and d-consistent. Therefore, the learner uses the procedure ConstructAutomaton using this observation table as input to obtain a characteristic DFA. The sets Pos and Neg created during this

	ce	ε	
		Mem	*Actions*
ε	0	0	$(0, +1, 0)$
a	1	1	$(1, 0, 0)$
ab	1	0	$(1, 0, 0)$
aa	1	1	$(1, 0, 0)$
aab	1	1	$(1, 0, 0)$
b	0	0	$(0, +1, 0)$
aba	1	1	$(1, 0, 0)$
abb	1	1	$(1, 0, 0)$
aaa	1	1	$(1, 0, 0)$
aaba	1	1	$(1, 0, 0)$
aabb	1	0	$(1, 0, 0)$

Table 4.5. An observation table with $\mathcal{P} = \{\varepsilon, a, ab, aa, aab\}$ and $\mathcal{S} = \{\varepsilon\}$ that is not 1-closed.

	ce	ε		b	
		Mem	*Actions*	Mem	*Actions*
ε	0	0	$(0, 1, 0)$	0	$(0, +1, 0)$
a	1	1	$(1, 0, 0)$	0	$(1, 0, 0)$
ab	1	0	$(1, 0, 0)$	1	$(1, 0, 0)$
aa	1	1	$(1, 0, 0)$	1	$(1, 0, 0)$
aab	1	1	$(1, 0, 0)$	0	$(1, 0, 0)$
b	0	0	$(0, 1, 0)$	0	$(0, +1, 0)$
aba	1	1	$(1, 0, 0)$	0	$(1, 0, 0)$
abb	1	1	$(1, 0, 0)$	0	$(1, 0, 0)$
aaa	1	1	$(1, 0, 0)$	1	$(1, 0, 0)$
aaba	1	1	$(1, 0, 0)$	1	$(1, 0, 0)$
aabb	1	0	$(1, 0, 0)$	1	$(1, 0, 0)$

Table 4.6. A 1-closed and 1-consistent observation table with $\mathcal{P} = \{\varepsilon, a, ab, aa, aab\}$ and $\mathcal{S} = \{\varepsilon, b\}$.

process are given below.

$$\begin{aligned}
Pos = \{&x, a^0, a^0y, b^0x, a^0a^1, a^0a^1y, a^0b^1y, a^0b^1a^1, a^0b^1a^1y, a^0b^1b^1, a^0b^1b^1y,\\
&a^0a^1a^1, a^0a^1a^1y, a^0a^1b^1, a^0a^1b^1y, a^0a^1b^1a^1, a^0a^1b^1a^1y, a^0a^1b^1b^1y, b^0b^0x,\\
&a0b1a1b1y, a0b1b1b1y, a0a1a1b1, a0a1a1b1y, a0a1b1a1b1, a0a1b1a1b1y,\\
&a^0a^1b^1b^1b^1, a^0a^1b^1b^1b^1y\}.\\
Neg = \{&\varepsilon, b^0, a^0b^1, a^0a^1b^1b^1, b^0b^0, a^0b^1a^1b^1, a^0b^1b^1b^1\}.
\end{aligned}$$

The characteristic DFA returned is shown in Figure 4.11. A DROCA is constructed from this characteristic DFA by removing from it the transitions on x and y, and by

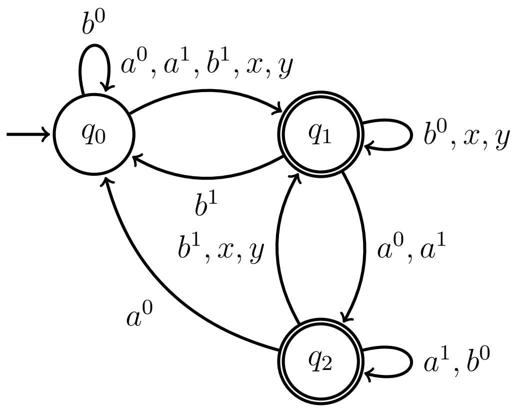


Fig. 4.11. The characteristic DFA obtained from `ConstructAutomaton` using Table 4.6 as input.

assigning counter-actions to the remaining transitions based on the tuples x and y. The resultant DROCA is shown in Figure 4.12.

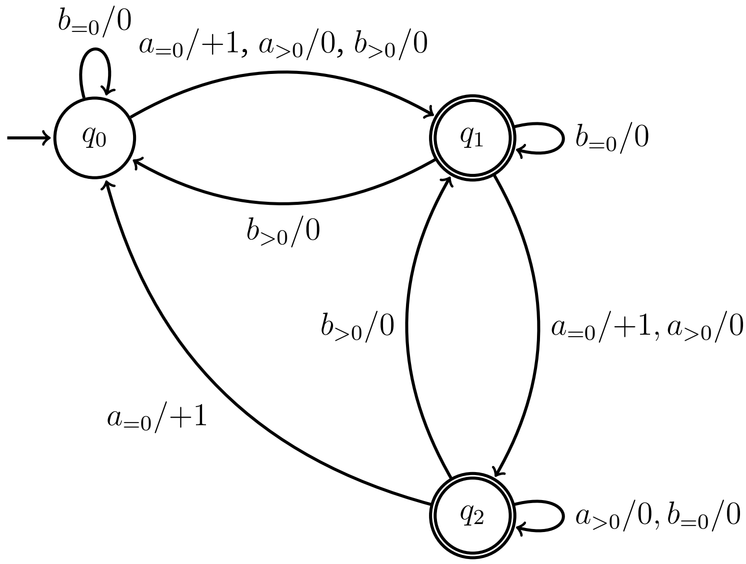


Fig. 4.12. The DROCA obtained from the characteristic DFA in Figure 4.11.

The learner now asks a minimal synchronous-equivalence query with this DROCA as input. Since this is equivalent and counter-synchronous with the teachers DROCA, the teacher replies yes to the minimal synchronous-equivalence query. The algorithm halts by outputting the learnt DROCA shown in Figure 4.12.

4.3.5 Analysis of `MinOCA`

The correctness of Algorithm 3 follows from the fact that whenever the teacher replies *yes* to a minimal synchronous-equivalence query, the learnt DROCA recognises the target language. Now, we show that this algorithm will terminate after polynomially many membership, counter value and minimal synchronous-equivalence queries.

Recall that the behaviour graph (see Definition 4.2) of a DROCA is an infinite state automaton induced by the refined Myhill-Nerode congruence $\simeq$. Each equivalence class under this refined congruence corresponds to a state of the behaviour graph, and the transitions between them are defined based on the equivalence class of the resultant words. The state corresponding to the equivalence class of a word will be marked as a final state in the behaviour graph if and only if that word is accepted by the DROCA.

Let $\simeq|_C$ denote the restriction of $\simeq$ to the entries in the observation table C. i.e., for all $p, p' \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s, s' \in \mathcal{S}$, $ps \simeq|_C p's'$ if and only if for all $z \in \Sigma^*$ where both psz and $p's'z$ are in $(\mathcal{P} \cup \mathcal{P}\Sigma)\mathcal{S}$, $\mathcal{A}(psz) = \mathcal{A}(p's'z)$ and $\mathtt{ce}_{\mathcal{A}}(psz) = \mathtt{ce}_{\mathcal{A}}(p's'z)$. Given an observation table C, we can construct a partial behavioural graph BG_C induced by $\simeq|_C$ in a similar fashion. Note that for all $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, the counter value corresponding to the equivalence class reached on reading ps and the membership of ps in BG_C is the same as the counter value reached and membership of ps in C. The following observation follows from the definition of $\simeq$.

Proposition 4.8

Given $p, p' \in \mathcal{P}$, if $p \simeq p'$, then $row(p) = row(p')$.

In the next proposition, let C be an observation table and $\mathcal{B}_C$ denote the DROCA learnt by the learner from C (using Line 14 of Algorithm 3).

Proposition 4.9

Let $d \in \mathbb{N}$ and z be a counter-example returned by MSQ$(\mathcal{B}_C)$ on Line 15, with $height_{\mathcal{A}}(z) = d$. Let C' be the observation table obtained by adding the prefixes of z to $\mathcal{P}$ in C and making it d-closed and d-consistent. Then the number of distinct rows with counter value less than or equal to d in C' is more than that of C.

Proof. Let $d \in N$ and $z =$MSQ$(\mathcal{B}_C)$ be a counter-example with $height_{\mathcal{A}}(z) = d$. Let C' be the observation table obtained by adding the prefixes of z to $\mathcal{P}$ in C and making it d-closed and d-consistent. Assume for contradiction that the number of distinct rows with counter value less than or equal to d in C' is the same as that of C. From Proposition 4.8, we get that BG_C and $BG_{C'}$ are the same in this case. From Lemma 4.7 for all $p \in \mathcal{P} \cup \mathcal{P}\Sigma$ and $s \in \mathcal{S}$, the counter values reached and membership of ps are the same in $\mathcal{B}_C$ and C. We also know that the counter value corresponding to the equivalence class reached on reading ps and membership of ps

in BG_C is the same as the counter value reached and membership of ps in C. Since $BG_C = BG_{C'}$, the membership and counter values reached by all prefixes of z in $\mathcal{A}$ should also match $\mathcal{B}_C$ contradicting the assumption that z is a counter-example. $\square$

Proposition 4.10

For any $d \in \mathbb{N}$, at most $d \times |\mathcal{A}|$ many counter-examples of height less than or equal to d are returned by the minimal synchronous-equivalence query.

Proof. Fix a $d' \in \mathbb{N}$. There are at most $|\mathcal{A}|$ many configurations of $\mathcal{A}$ with counter value d'. We know that for $p, p' \in \mathcal{P}$, if $row(p) \neq row(p')$, then $p \not\approx p'$. Hence, there are at most $|\mathcal{A}|$ distinct rows in the observation table with counter value d'. Consequently, there are at most $d \times |\mathcal{A}|$ distinct rows with counter values $d' \leq d$. From Proposition 4.9, we get that every counter-example with height less than or equal to d will increase the number of distinct rows with counter value $d' \leq d$ in the observation table. The proposition follows. $\square$

Proposition 4.11

At most $|\mathcal{A}|^5+1$ many minimal synchronous-equivalence queries are executed during the run of Algorithm 3.

Proof. Assume for contradiction that more than $|\mathcal{A}|^5+1$ many minimal synchronous-equivalence queries (Line 15 of Algorithm 3) are executed. Therefore, more than $|\mathcal{A}|^5$ many counter-examples are returned by these queries. From Proposition 4.10, it follows that at least one counter-example is of height greater than $|\mathcal{A}|^4$. However, from Theorem 3.5, we know that the height of the minimal counter-example that distinguishes two DROCAs of size $|\mathcal{A}|$ is at most $|\mathcal{A}|^4$. This is a contradiction, since by Lemma 4.7, $|\mathcal{B}_C| \leq |\mathcal{A}|$ for any observation table C during the run of the algorithm. $\square$

This gives us the following theorem.

Theorem 4.12

Given a DROCA $\mathcal{A}$, a minimal counter-synchronous DROCA recognising the same language can be learnt with $|\mathcal{A}|^5 + 1$ queries to the SAT solver, $|\mathcal{A}|^5 + 1$ minimal synchronous-equivalence queries and polynomially many membership and counter value queries.

The minimality of the DROCA learnt follows from Lemma 4.7 and the fact that the observation table corresponding to a DROCA $\mathcal{A}$ and a minimal counter-synchronous DROCA $\mathcal{A}'$ is the same.

A visibly one-counter automata (VOCA) is a DROCA where the decision to increment, decrement or not change the current counter value on a transition is determined only by the input alphabet and not by state or current counter value. We can adapt the learning algorithm for DROCAs for the special case of VOCAs and use the equivalence check from Theorem 3.6. This will learn a minimal VOCA using at most $2|\mathcal{A}|^3+1$ equivalence queries. Note that for VOCAs, we do not require the counter value queries.

Corollary 4.13

Given a VOCA $\mathcal{A}$, a minimal VOCA recognising the same language can be learnt using $\mathcal{O}(|\mathcal{A}|^3)$ queries to the SAT solver, $\mathcal{O}(|\mathcal{A}|^3)$ minimal synchronous-equivalence queries and polynomially many membership queries.

The counter value reached upon reading any word from the initial configuration of a VOCA can be determined without querying the teacher. Hence, we do not require the counter value queries for learning VOCA. For VOCA, some words do not have a valid run. i.e., the counter goes below zero during the run. These words will not have a corresponding entry in the observation table and will be treated as don't care. Also, we don't have to create the set $Operations$ to encode the counter-actions, as the counter actions are determined by the input alphabet.

4.4 Implementation

The proposed method was implemented in Python[4] and was tested against randomly generated DROCAs. This section discusses the implementation details and compares the performance of our method with BPS Bruyère et al. (2022).

4.4.1 Equivalence Query

Even though there is a polynomial time algorithm to check the equivalence of two DROCAs (Böhm and Göller, 2011), the required polynomial is so large that it is not suitable for practical applications. Hence the implementation by Bruyère et al. (2022) uses an approximate equivalence query that may say two non-equivalent DROCAs to be equivalent.

In our implementation, we construct a DROCA that is counter-synchronised with the DROCA to be learnt. Their equivalence can be checked by running a breadth-

[4]The implementation of MinOCA, the datasets used, and the complete results generated can be found in the following link: https://doi.org/10.5281/zenodo.14604419.

first search on the configuration graph of their union up to the counter value and length obtained from Theorem 3.5. The minimal synchronous-equivalence query either returns a word for which the constructed DROCA and DROCA to be learnt reach different counter values, or returns a word that is accepted by one and rejected by the other. In our implementation, after each equivalence query, we increment the value of d for which d-closed and d-consistency have to be checked.

A major distinction between `MinOCA` and `BPS` is that the latter employs an approximate equivalence query, while the former uses an exact equivalence query. Therefore, there is a possibility that the DROCA returned by `BPS` is incorrect. On the other hand, the DROCA returned by `MinOCA` is always the correct one.

4.4.2 Finding a Minimal-Separating DFA

We utilise the Python library `DFAMiner` by Dell'Erba et al. (2024) that uses a SAT solver to learn a minimal separating DFA from a given set of positive and negative samples. The Python library `dfa-identify` by Vazquez-Chanlatte et al. (2021) performs the same function. Our experiments showed that `DFAMiner` outperforms `dfa-identify` with respect to the time taken for identifying a minimal separating DFA. Various other techniques for computing a minimal separating automaton using SAT solvers are discussed by Leucker and Neider (2012), and Neider (2012).

4.4.3 Random Generation of DROCAs

We follow a procedure similar to that by Bruyère et al. (2022) to randomly generate DROCAs with a given number of states.

Let $n \in \mathbb{N}$ be the number of states of the DROCA to be generated. The procedure `GenerateDROCA` used to generate random DROCAs is as follows. First, we initialise the set of states $Q = \{q_1, q_2, \ldots, q_n\}$. For all $q \in Q$, we make q a final state with probability 0.5. If $Q = F$ or $F = \emptyset$ after this step, then we restart the procedure. Otherwise, for all $q \in Q$ and $a \in \Sigma$, we assign $\delta_0(q, a) = (p, c)$ (resp. $\delta_1(q, a) = (p, c)$), with p a random state in Q and c a random counter operation in $\{0, +1\}$ (resp. $\{0, +1, -1\}$). The constructed DROCA $\mathcal{A} = (Q, \Sigma, \{q_1\}, \delta_0, \delta_1, F)$. If the number of reachable states of $\mathcal{A}$ from the initial configuration is not n, then we discard $\mathcal{A}$ and restart the whole procedure. Otherwise, we output $\mathcal{A}$. The exact procedure is given in Algorithm 4.

Algorithm 4: Algorithm to generate a random DROCA with n states.

```
Procedure GenerateDROCA()
    Input  : An integer n
    Output : A random DROCA with n reachable states

    repeat
        Initialise Q = {1, 2, ..., n}, F = ∅.
        foreach p in Q add p to F with probability 0.5.
        foreach p in Q and a ∈ Σ assign δ0(q, a) = (p, c) (resp.
          δ1(q, a) = (p, c)), with p a random state in Q and c a random
          counter operation in {0, +1} (resp. {0, +1, −1}).
        Initialise DROCA A = (Q, Σ, {1}, δ0, δ1, F).
    until F ≠ ∅, F ≠ Q and reachable(A) = n;
    return A.
end
```

To ensure that the generated DROCA has n reachable states, we use the procedure `reachable` – this procedure takes a DROCA as input and returns the number of distinct states visited. It performs a breadth-first search on the configuration graph up to counter value n^2 starting from the initial configuration. From Lemma 3.2, we know that if a state q is reachable from the initial configuration, then there exists a word w that takes us from that initial configuration to a configuration with state q and the maximum counter value encountered during the run on w is less than n^2. Since the configuration graph up to counter value n^2 contains only n^3 configurations, this can be done in $\mathcal{O}(n^3)$ time. Two datasets, each with 5600 DROCAs, were generated, varying input alphabet sizes from 2 to 5 and states from 2 to 15, with 100 random DROCAs for each combination.

Dataset$_1$**: Dataset for comparing `MinOCA` *and* BPS.** The notion of acceptance in BPS is by final state and zero counter value, whereas `MinOCA` employs the notion of acceptance with the final state only. This is the widely accepted notion of acceptance for DROCAs and is the one used by Böhm and Göller (2011) while proving that the equivalence of DROCAs is in P. For comparing the two methods, we generated random DROCAs where the notion of acceptance by final state and counter value zero and the notion of acceptance by final state are the same. For this, we use the procedure `GenerateDROCA` with an added condition. This mandates that the final states can only be reached by transitions that read a symbol from counter value of zero and do not modify the counter value. The results of experiments conducted using $\texttt{Dataset}_1$ on MinOCA and BPS are shown in Figures 4.13-4.18.

$\texttt{Dataset}_2$: Dataset for evaluating the performance of MinOCA on general DROCAs. The performance of MinOCA was evaluated for general DROCAs, where the notion of acceptance is with final states only. Random DROCAs were generated using the procedure GenerateDROCA for this purpose. The results of experiments conducted using $\texttt{Dataset}_2$ on MinOCA are shown in Figures 4.19-4.24.

4.4.4 Experimental Results

We implemented MinOCA in Python and used the Java implementation of BPS. The computations were performed on an Apple M1 chip with 8GB of RAM, running macOS Sonoma Version 14.3.

4.4.4.1 Comparing MinOCA and BPS Using $\texttt{Dataset}_1$.

A timeout of 5 minutes was allotted for both BPS and MinOCA for learning each DROCA in $\texttt{Dataset}_1$. If the algorithms cannot successfully learn a language within the timeout, we discard that sample and process the next one. The number of languages successfully learned by MinOCA and BPS for different input sizes is depicted in Figure 4.13, and the exact values are provided in Table 4.8. Figure 4.14 shows the average length of the longest counter-example obtained during learning. In all cases, MinOCA provides a smaller counter-example on average. Figure 4.15 shows the average number of states in the learnt DROCA. The number of states of the automaton learnt by MinOCA is always less than or equal to the input size. Figure 4.16 shows the average number of equivalence queries used for successfully learning the input DROCAs. The number of equivalence queries is smaller for MinOCA in all cases. The data used to plot Figures 4.13- 4.18 is given in Table 4.7.

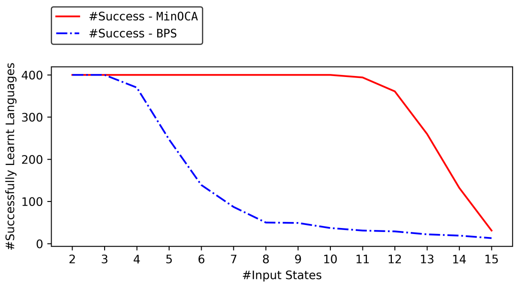


Fig. 4.13. Number of successfully learnt languages (out of 400) by MinOCA and BPS for Dataset_1.

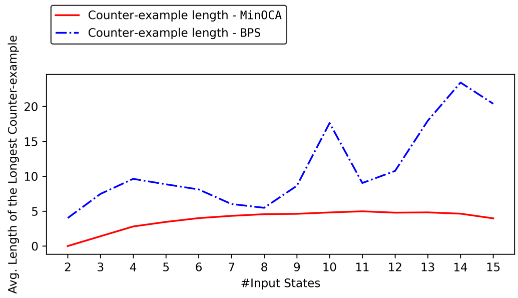


Fig. 4.14. The average length of the longest counter-example obtained by MinOCA and BPS for Dataset_1.

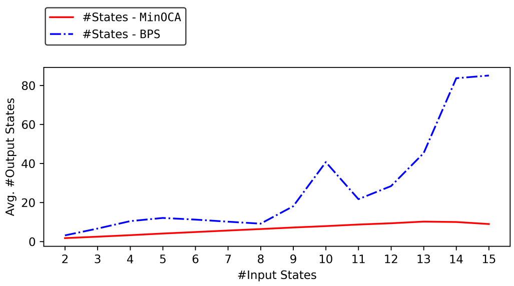


Fig. 4.15. Average number of states in the learnt DROCA obtained by MinOCA and BPS for Dataset_1.

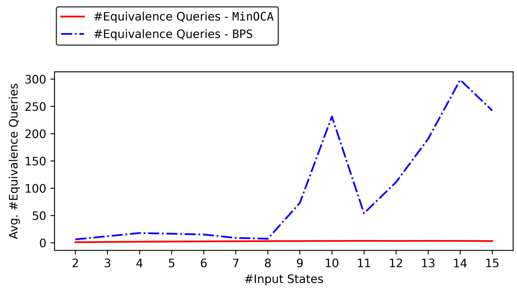


Fig. 4.16. Average number of equivalence queries used for learning by MinOCA and BPS for Dataset_1.

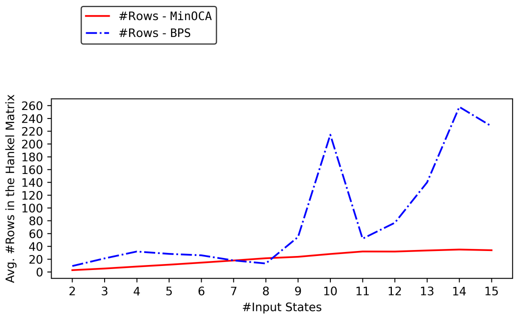


Fig. 4.17. Average number of rows in the observation table obtained by MinOCA and BPS for Dataset_1.

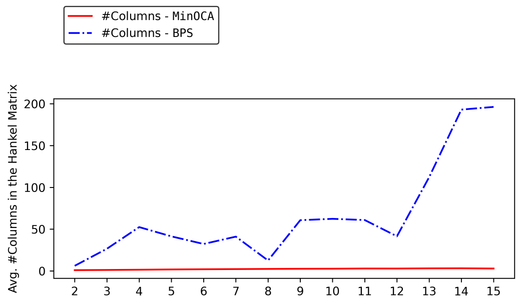


Fig. 4.18. Average columns in the observation table for MinOCA and BPS on Dataset_1.

#States	Success1	Success2	States1	States2	LongestCE1	LongestCE2	EqQ1	EqQ2	Row1	Row2	Col1	Col2
2	400	400	1.75	3.15	0.00	4.02	1.00	6.19	2.78	9.22	1.00	6.18
3	400	400	2.49	6.65	1.40	7.47	1.43	11.93	5.28	21.13	1.21	26.59
4	400	370	3.26	10.45	2.81	9.63	1.91	17.79	8.38	31.86	1.51	52.49
5	400	247	4.11	12.09	3.46	8.85	2.21	16.55	11.30	28.20	1.83	41.39
6	400	139	4.87	11.24	4.00	8.11	2.50	15.11	14.52	25.96	2.06	32.29
7	400	87	5.64	10.17	4.33	6.02	2.74	8.93	17.84	17.95	2.25	41.08
8	400	50	6.40	9.16	4.56	5.48	2.98	7.32	21.36	13.18	2.46	12.56
9	400	49	7.20	18.08	4.62	8.63	3.09	73.01	23.62	54.61	2.64	60.73
10	400	37	7.91	40.70	4.81	17.62	3.31	231.46	28.03	214.43	2.72	62.38
11	394	31	8.71	21.71	4.98	9.03	3.51	53.97	31.92	51.94	2.95	60.94
12	361	29	9.37	28.41	4.78	10.76	3.32	111.10	31.83	76.79	2.91	41.48
13	260	22	10.22	45.32	4.82	17.95	3.47	190.36	33.50	139.86	3.12	112.18
14	132	19	10.00	83.68	4.64	23.42	3.38	298.58	34.98	257.89	3.20	193.00
15	31	13	8.97	85.08	3.97	20.38	3.06	242.31	33.90	227.62	2.94	196.23

Table 4.7. The table shows the data used to plot Figures 4.13- 4.18. The column *#States* denotes the number of states in the input DROCA. The columns $Sucess1$, $States1$, $LongestCE1$, $EqQ1$, $Row1$, $Col1$ (resp. $Sucess2$, $States2$, $LongestCE2$, $EqQ2$, $Row2$, $Col2$) respectively denote the number of successfully learnt languages, the average number of states in the learnt DROCA, the average length of the longest counter-example, the average number of equivalence queries used, the average number of rows in the final observation table and average number of columns in the final observation table for `MinOCA` (resp. `BPS`) for $\texttt{Dataset}_1$.

	$\lvert\Sigma\rvert$ \ $\lvert Q\rvert$	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MinOCA	2	100	100	100	100	100	100	100	100	100	95	85	54	27	7
	3	100	100	100	100	100	100	100	100	100	99	91	61	22	1
	4	100	100	100	100	100	100	100	100	100	100	91	79	38	9
	5	100	100	100	100	100	100	100	100	100	100	94	66	45	14
BPS	2	100	100	98	80	61	48	39	32	32	24	23	18	15	9
	3	100	100	95	67	31	17	5	12	2	5	5	2	2	4
	4	100	100	90	48	25	13	4	3	3	2	0	1	2	0
	5	100	100	87	52	22	9	2	2	0	0	1	1	0	0

Table 4.8. Number of successfully learnt samples (out of 100) by `MinOCA` and BPS for $\texttt{Dataset}_1$ with the number of states ranging from 2 to 15 and the size of the alphabet ranging from 2 to 5.

4.4.4.2 Evaluating the Performance of `MinOCA` on $\texttt{Dataset}_2$.

We used $\texttt{Dataset}_2$ to evaluate the performance of `MinOCA` on DROCAs that accept with final state. A timeout of 5 minutes was allocated for learning each language. Figure 4.19 shows the number of successfully learnt languages by `MinOCA`. One can see that as the number of states exceed 10, there is a decrease in the number of samples learnt in 5 minutes.

#States	Success	States	LongestCE	EqQ	MaxEqQ	Time	SAT	Row	Col
2	400	2.00	0.34	1.10	2.00	1.26	1.26	3.97	1.00
3	400	2.99	2.44	1.68	4.00	1.93	1.93	8.21	1.12
4	400	3.98	3.60	2.08	5.00	2.44	2.44	12.53	1.27
5	400	4.97	4.35	2.45	6.00	2.92	2.91	17.35	1.50
6	400	5.96	4.64	2.71	7.00	3.33	3.31	22.38	1.65
7	400	6.94	4.83	2.91	7.00	3.91	3.86	27.28	1.77
8	400	7.94	4.92	3.19	9.00	5.11	5.01	33.56	1.92
9	400	8.91	4.97	3.31	12.00	7.62	7.42	38.66	1.98
10	400	9.90	5.10	3.59	14.00	16.60	16.22	45.53	2.11
11	385	10.88	5.38	3.71	13.00	45.34	44.69	51.05	2.13
12	321	11.84	5.27	3.77	10.00	112.92	111.84	55.81	2.16
13	113	12.72	5.31	3.41	7.00	169.28	167.69	53.22	2.27
14	22	13.00	5.68	3.64	6.00	184.53	182.56	55.14	2.36
15	4	12.50	4.00	3.00	3.00	135.98	135.00	43.00	2.75

Table 4.9. The table shows the data used to plot Figures 4.19- 4.24. The column *#States* denotes the number of states in the input DROCA. The columns $Sucess$, $States$, $LongestCE$, EqQ, $MaxEqQ$, $Time$, SAT, Row, Col respectively denote the number of successfully learnt languages, the average number of states in the learnt DROCA, the average length of the longest counter-example, the average number of equivalence queries used, the maximum number of equivalence queries used, the average time taken for learning, the average time taken by the SAT solver in finding a minimal separating DFA, the average number of rows in the final observation table and average number of columns in the final observation table for $\texttt{Dataset}_2$.

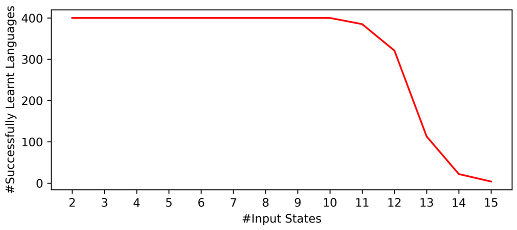


Fig. 4.19. Number of successfully learnt languages (out of 400) by MinOCA for $\mathtt{Dataset}_2$.

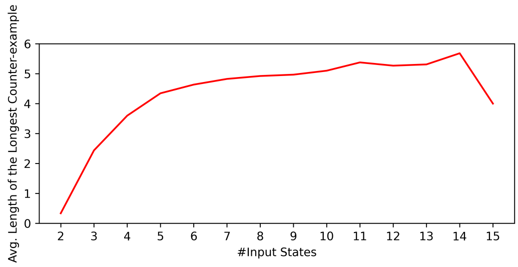


Fig. 4.20. The average length of the longest counter-example obtained by MinOCA for $\mathtt{Dataset}_2$.

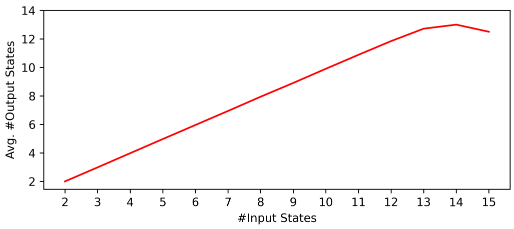


Fig. 4.21. Average number of states in the learnt DROCA obtained by MinOCA for $\mathtt{Dataset}_2$.

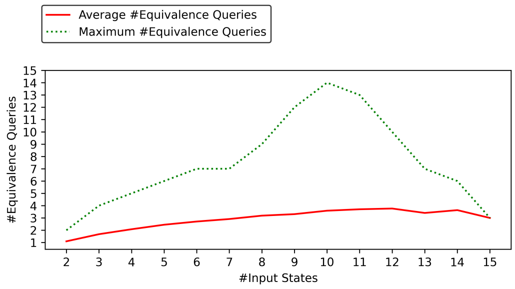


Fig. 4.22. Average and maximum number of equivalence queries used for learning by MinOCA for $\mathtt{Dataset}_2$.

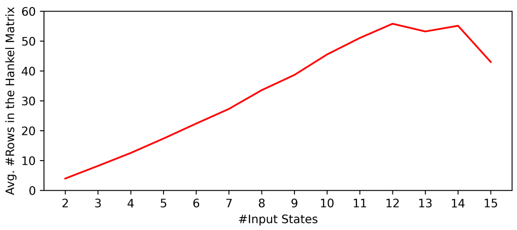


Fig. 4.23. The average number of rows in the observation table obtained by MinOCA for $\mathtt{Dataset}_2$.

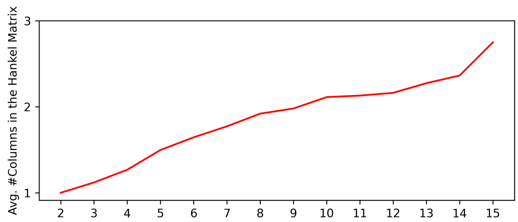


Fig. 4.24. The average number of columns in the observation table obtained by MinOCA for $\mathtt{Dataset}_2$.

The number of languages successfully learned by MinOCA for different input sizes is depicted in Figure 4.19, and the exact values are provided in Table 4.10. Figure 4.20 shows the average length of the longest counter-example obtained during learning. Figure 4.21 shows the average number of states in the learnt DROCA. Figure 4.22 shows the average and maximum number of equivalence queries used for successfully learning the input DROCAs. The data used to plot Figures 4.19- 4.24 is given in Table 4.9.

MinOCA spends most of the time in finding a minimal separating DFA (see Figure 4.25) using the SAT solver. The scalability of our algorithm is hence dependent on the scalability of finding a minimal separating DFA.

$\lvert\Sigma\rvert$ \ $\lvert Q\rvert$	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	100	100	100	100	100	100	100	100	100	94	77	34	10	3
3	100	100	100	100	100	100	100	100	100	91	74	20	3	1
4	100	100	100	100	100	100	100	100	100	100	83	33	4	0
5	100	100	100	100	100	100	100	100	100	100	87	26	5	0

Table 4.10. Number of successfully learnt samples (out of 100) by MinOCA for $\texttt{Dataset}_2$ with the number of states ranging from 2 to 15 and the size of the alphabet ranging from 2 to 5.

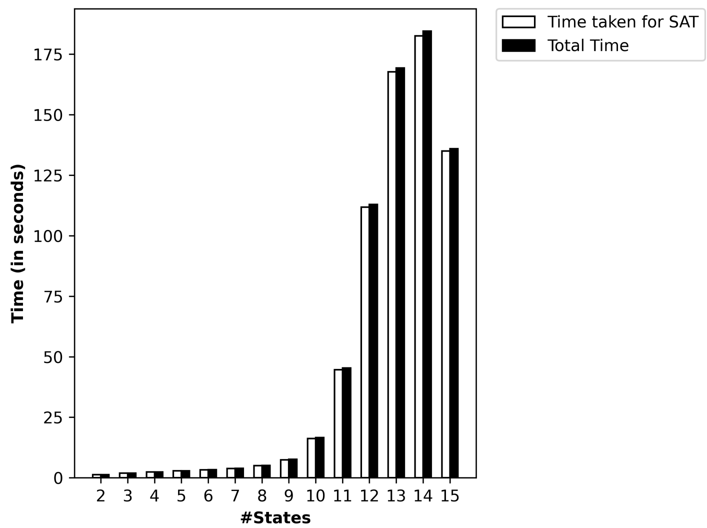


Fig. 4.25. Total time taken compared to the time taken by the SAT solver to find a minimal separating DFA when tested on general DROCAs.

4.5 Conclusion

In this chapter, we presented a novel approach for active learning of DROCAs. We showed that a DROCAs can be learnt using polynomially many queries with the help of a SAT solver in contrast to the existing techniques that require exponentially many queries. Our algorithm learns a minimal counter-synchronous DROCA, which results in significantly smaller counter-examples in equivalence queries. We evaluated an implementation of our algorithm against randomly generated DROCAs and compared the results with the existing technique (Bruyère et al., 2022). The results show that the proposed method significantly outperforms the existing one.

In future work, the proposed algorithm can be improved by finding better methods for identifying the minimal separating DFA. The algorithm can also be applied to learn VOCAs, and the scalability of this approach warrants further investigation. Extending these ideas to learn more general models, such as visibly pushdown automata, represents a valuable direction for further research. Another open problem that remains is to determine whether active learning of DROCAs can be done in polynomial time. This problem is open, even in the case of VOCAs. However, learning a minimal VOCA cannot be done in polynomial time unless $\mathsf{P} = \mathsf{NP}$. Exploring the possibility of finding a polynomial-time algorithm to obtain a polynomial approximation is worthy of further study.

Part III

Weighted One-Counter Automata

- The results presented in Chapter 5 will be published in the proceedings of **ICLA 2025** under the title "**Equivalence of Deterministic Weighted Real-time One-Counter Automata**". The full version of the paper is available on arXiv [https://arxiv.org/abs/2411.03066].
- The results presented in Chapter 6 and Chapter 7 have been published in the proceedings of **FSTTCS 2024** under the title "**Weighted One-Deterministic-Counter Automata**" [https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.FSTTCS.2023.39].

Both of these are joint works with *Dr. Vincent Penelle*, *Dr. Prakash Saisavan*, and *Dr. Sreejith A.V.*

CHAPTER 5

Deterministic Weighted Real-Time One-Counter Automata

This chapter introduces deterministic weighted real-time one-counter automaton (DWROCA). A DWROCA is a deterministic real-time one-counter automaton(DROCA) whose transitions are assigned a weight from a field $\mathcal{F}$. Two DWROCAs are equivalent if the weight assigned to every word by one is the same as the weight assigned to every word by the other. This chapter gives a polynomial-time algorithm for checking the equivalence of two DWROCAs.

In Chapter 5, Chapter 6, and Chapter 7, we assume that the field involved and its elements have some finite representation. For example, rational numbers can be represented as a pair of integers (p, q) where $q \neq 0$, representing the fraction $\frac{p}{q}$. The field operations are then implemented as algorithms - addition requires finding common denominators, multiplication involves multiplying numerators and denominators separately, and so on. This representation is exact and complete, allowing us to work with the infinite field of rational numbers using only finite representations of each individual element.

Contents

5.1 Introduction

In this chapter, we give a polynomial-time algorithm to check the equivalence of two deterministic weighted real-time one-counter automata (DWROCAs) where the weights are from a field (possibly infinite). DWROCAs are weighted one-counter automata with the "deterministic" condition — at most one transition on a letter from a state, and the counter is modified by at most *one* on a transition. Hence, for a word over a finite alphabet, there is at most one run starting from the initial configuration that determines its accepting weight (see Definition 5.2 for a formal definition). Additionally, "real-time" indicates that there are no ε-transitions in this system. The weight assigned to a word by a given DWROCA is the product of the weights of the transitions along its run starting from the initial configuration.

We say that two DWROCAs are equivalent if the weight assigned to every word by one of the automata is the same as the weight assigned by the other. We show that if two DWROCAs are not equivalent, then there exists a small word w that got assigned with different weights by the two machines. It suffices to show that the length of w is polynomially bounded in the size of the DWROCAs under consideration. Two weighted automata that "simulate" the run of the DWROCAs up to that bound can then be used to check equivalence. To prove a bound on the length of w, we give a polynomial that bounds the counter values during the run of w on both machines. Our proof strategy borrows the "belt technique" developed in the context of deterministic real-time one-counter automata by

Böhm and Göller (2011) (also see Böhm et al. (2010),Böhm et al. (2014)). We introduce a pumping technique (see Theorem 5.3) to adapt their proof strategy for checking the equivalence of two DWROCAs.

5.1.1 Related Work

One-counter automata are pushdown automata with a singleton stack alphabet. It is well known that the equivalence problem for nondeterministic pushdown automata is undecidable. On the other hand, Sénizergues (1997) proved that it is decidable for deterministic pushdown automata. It was later proved that there is a primitive recursive algorithm for this (Stirling, 2002). The equivalence problem for deterministic one-counter automata is NL-complete (Böhm et al., 2013). Studies on probabilistic pushdown automata conducted by Forejt et al. (2014) showed that the equivalence problem of probabilistic pushdown automata is equivalent to the multiplicity equivalence of context-free grammars. The latter problem is not known to be decidable. The decidability of equivalence is known for some special sub-classes of probabilistic pushdown automata. The equivalence problem is at least as hard as polynomial identity testing (Forejt et al., 2014) even when the input alphabet contains only one letter. For the case of visibly probabilistic pushdown automata, there is a randomised polynomial time algorithm for checking equivalence (Kiefer et al., 2013). However, there is a polynomial time reduction from polynomial identity testing to this problem and is hence not likely to be in P.

5.2 Definitions

5.2.1 Deterministic Weighted Automata

Definition 5.1 (Deterministic weighted automata)

A deterministic weighted automaton (DWA) over a field $\mathcal{F} = (S, \oplus, \circ, 0_e, 1_e)$ is a tuple $\mathcal{A} = (Q, \Sigma, q_0, s_0, \delta, \eta_F)$, where

- Q is a finite, nonempty set of states.
- Σ is the input alphabet.
- $q_0 \in Q$ is the initial state.
- $s_0 \in \mathcal{F}$ is the initial weight.
- $\delta : Q \times \Sigma \to Q \times \mathcal{F}$ is the transition function.
- $\eta_F : Q \to \mathcal{F}$ is a function that assigns an output weight to each state.

We use $|\mathcal{A}|$ to denote the size of $\mathcal{A}$, which we consider to be $|Q|$. A *configuration* of a DWA is a pair $c = (q, t)$, with $q \in Q$ and $t \in \mathcal{F}$. The configuration $(q_0, s_0) \in Q \times \mathcal{F}$ is called the *initial configuration* of $\mathcal{A}$. A *transition* is a tuple $\tau = (p_\tau, a_\tau, q_\tau, s_\tau)$ where $p_\tau, q_\tau \in Q$ are states, $a_\tau \in \Sigma$, and $s_\tau \in \mathcal{F}$ such that $\delta(p_\tau, a_\tau) = (q_\tau, s_\tau)$. Given a transition $\tau = (p_\tau, a_\tau, q_\tau, s_\tau)$ and a configuration $c = (q_c, s_c)$, we denote the application of τ to c as $\tau(c) = (q_\tau, s_c \circ s_\tau)$ if $q_c = p_\tau$ and is undefined otherwise.

Given a sequence of transitions $T = \tau_0 \cdots \tau_{\ell-1}$, we denote $word(T) = a_{\tau_0} \cdots a_{\tau_{\ell-1}}$ the *word labelling it* and $\mathtt{we}(T) = s_{\tau_0} \circ \cdots \circ s_{\tau_{\ell-1}}$ its *weight-effect*. For all $0 \leq i < j \leq |\ell - 1|$, we use $T_{i \cdots j}$ to denote the sequence of transitions $\tau_i \cdots \tau_j$ and $|T|$ to denote ℓ.

A *run* π is an alternate sequence of configurations and transitions denoted as $\pi = c_0\tau_0c_1 \cdots \tau_{\ell-1}c_\ell$ such that for every i, $c_{i+1} = \tau_i(c_i)$. Given a sequence of transitions T and a configuration c, we denote $T(c)$ the run obtained by applying T to c sequentially (if it is defined). The word labelling it, its length, and weight-effect are those of its underlying sequence of transitions. A *sub-run* is a (syntactic) sub-word of a run that is also a run.

Since the machine is deterministic, for any word w, there is at most one run labelled by w starting in a given configuration c_0. We denote this run by $\pi(w, c_0)$. We use $\pi(w)$ to denote the run of starting the from the initial configuration labelled by w. A run $\pi(w, c_0) = c_0\tau_0 c_1 \cdots \tau_{\ell-1} c_\ell$ is also represented as $c_0 \xrightarrow{w} c_\ell$. We use the notation $c_0 \rightarrow^* c_\ell$ to denote the existence of some word w such that $c_0 \xrightarrow{w} c_\ell$. The *weight assigned* to a word w by $\mathcal{A}$ along the run $\pi(w, c_0)$ is denoted by $f_{\mathcal{A}}(w, c_0) = s_{c_0} \circ \texttt{we}(\pi(w, c_0)) \circ \eta_F(q_{c_\ell})$. We use the notation $f_{\mathcal{A}}(w)$ to denote $f_{\mathcal{A}}(w, (q_0, s_0))$.

Let $\mathcal{A}$ and $\mathcal{B}$ be two DWAs. Consider the configurations c_1 of $\mathcal{A}$ and c_2 of $\mathcal{B}$. We say that $c_1 \equiv_l c_2$ if and only if for all $w \in \Sigma^{\leq l}$, $f_{\mathcal{A}}(w, c_1) = f_{\mathcal{B}}(w, c_2)$. We say that the configurations c_1 and c_2 are *equivalent* if and only if $c_1 \equiv_l c_2$ for all $l \in \mathbb{N}$, and we denote this by $c_1 \equiv c_2$. We say that $\mathcal{A}$ and $\mathcal{B}$ are *equivalent* if for all $w \in \Sigma^*$, $f_{\mathcal{A}}(w) = f_{\mathcal{B}}(w)$. A word $w \in \Sigma^*$ is called a *non-equivalence witness* for $\mathcal{A}$ and $\mathcal{B}$ if and only if $f_{\mathcal{A}}(w) \neq f_{\mathcal{B}}(w)$. A *minimal witness* is a non-equivalence witness of minimal length.

An *uninitialised* DWA over a field $\mathcal{F} = (S, \oplus, \circ, 0_e, 1_e)$ is a tuple $\mathcal{C} = (Q, \Sigma, \delta, \eta_F)$, where Q is a finite, nonempty set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \rightarrow Q \times \mathcal{F}$ is the transition function and $\eta_F : Q \rightarrow \mathcal{F}$ is a function that assigns an output weight to each state. Given an unitialised DWA $\mathcal{C} = (Q, \Sigma, \delta, \eta_F)$, a state $q_0 \in Q$, and a weight $s_0 \in \mathcal{F}$, we can get a DWA $\mathcal{A} = (Q, \Sigma, q_0, s_0\delta, \eta_F)$. A configuration of $\mathcal{C}$ is a pair $(q, s) \in Q \times \mathcal{F}$. The set of all configurations of $\mathcal{C}$ is the set $\{(q, s) \mid q \in Q \text{ and } s \in \mathcal{F}\}$. Given a configuration (q, s) of $\mathcal{C}$, and a word $w \in \Sigma^*$, we define $f_{\mathcal{C}}(w, (q, s))$ as $f_{\mathcal{B}}(w)$, where $\mathcal{B} = (Q, \Sigma, q, s, \delta, \eta_F)$ is the DWA obtained by making q as the initial state and s as the initial weight.

5.2.2 Deterministic Weighted Real-Time One-Counter Automata

Definition 5.2 (DWROCA)

A deterministic weighted real-time one-counter automaton (DWROCA) over a field $\mathcal{F} = (S, \oplus, \circ, 0_e, 1_e)$ is a tuple $\mathcal{A} = (Q, \Sigma, q_0, s_0, \delta_0, \delta_1, \eta_F)$, where

- Q is a finite, nonempty set of states.
- Σ is the input alphabet.
- $q_0 \in Q$ is the initial state.
- $s_0 \in \mathcal{F}$ is the initial weight.
- $\delta_0 : Q \times \Sigma \to Q \times \{0, +1\} \times \mathcal{F}$ and $\delta_1 : Q \times \Sigma \to Q \times \{-1, 0, +1\} \times \mathcal{F}$ are the transition functions.
- $\eta_F : Q \to \mathcal{F}$ is a function that assigns an output weight to each state.

We use $|\mathcal{A}|$ to denote the size of $\mathcal{A}$, which we consider to be $|Q|$. A configuration of a DWROCA is a triple $c = (q, n, s)$, with $q \in Q$, $n \in \mathbb{N}$ and $s \in \mathcal{F}$. Given a configuration $c = (q, n, s)$, we use q_c, n_c,and s_c respectively to denote q, n and s. The configuration $(q_0, 0, s_0) \in Q \times \mathbb{N} \times \mathcal{F}$ is called the initial configuration of $\mathcal{A}$. A transition is a sextuple $\tau = (p_\tau, d_\tau, a_\tau, \mathsf{ce}_\tau, s_\tau, q_\tau)$ where $p_\tau, q_\tau \in Q$ are states, $d_\tau \in \{0, 1\}$ specifies which among δ_0 and δ_1 is used, $a_\tau \in \Sigma$, $\mathsf{ce}_\tau \in \{-1, 0, 1\}$ is the *counter-effect*, and $s_\tau \in \mathcal{F}$ such that $\delta_{d_\tau}(p_\tau, a_\tau) = (q_\tau, \mathsf{ce}_\tau, s_\tau)$. Intuitively, δ_0 is used when you have a zero-test, and δ_1 otherwise. Given a transition τ and a configuration $c = (q_c, n_c, s_c)$, we denote the application of τ to c as $\tau(c) = (q_\tau, n_c + \mathsf{ce}_\tau, s_c \circ s_\tau)$ if $q_c = p_\tau$ and $d_\tau = 0$ if and only if $n_c = 0$, and is undefined otherwise. Note that the counter values always stay non-negative, implying that you cannot perform a decrement operation on the counter from a configuration with counter value zero.

Given a sequence of transitions $T = \tau_0 \cdots \tau_{\ell-1}$, we denote *word*$(T) = a_{\tau_0} \cdots a_{\tau_{\ell-1}}$ the word labelling it, $\mathtt{we}(T) = s_{\tau_0} \circ \cdots \circ s_{\tau_{\ell-1}}$ its weight-effect, and $\mathtt{ce}(T) = \mathsf{ce}_{\tau_0} + \cdots + \mathsf{ce}_{\tau_{\ell-1}}$ its counter-effect. For all $0 \leq i < j \leq |\ell - 1|$, we use $T_{i \cdots j}$ to denote the sequence of transitions $\tau_i \cdots \tau_j$ and $|T|$ to denote ℓ.

We call a sequence of transitions non-floating if there is an i such that $d_{\tau_i} = 0$ and floating otherwise. We denote $\min_{\texttt{ce}}(T) = \min_i(\texttt{ce}(\tau_0 \cdots \tau_i))$ as the minimal counter-effect of its prefixes and $\max_{\texttt{ce}}(T) = \max_i(\texttt{ce}(\tau_0 \cdots \tau_i))$ as the maximal counter-effect of its prefixes. We say that the sequence of transitions T is *valid* if for every $i \in [0, \ell - 2]$, $q_{\tau_i} = p_{\tau_{i+1}}$.

A run π is an alternate sequence of configurations and transitions denoted as $\pi = c_0\tau_0c_1 \cdots \tau_{\ell-1}c_\ell$ such that for every i, $c_{i+1} = \tau_i(c_i)$. Given a sequence of transitions T and a configuration c, we denote $T(c)$ the run obtained by applying T to c sequentially (if it is defined). The word labelling it, its length, weight-effect and counter-effect are those of its underlying sequence of transitions. A *sub-run* is a (syntactic) sub-word of a run that is also a run. The run $\pi = c_1\tau_1c_2\tau_2 \ldots c_i$ is called a *simple cycle*, if $i \leq |\mathcal{A}|$, $q_{c_1} = q_{c_i}$ and for all $j, k \in [1, i-1]$, $q_{c_j} = q_{c_k}$ if and only if $j = k$. The *efficiency* of a simple cycle $\mathfrak{p} = \frac{-\texttt{ce}(\pi)}{|\pi|}$, is the counter loss per length of the cycle.

Observe that, for a valid floating sequence of transitions, $T(c)$ is defined if and only if $n_c > -\min_{\texttt{ce}}(T)$, and for a valid non-floating sequence of transition, $T(c)$ is defined if and only if $n_c = -\min_{\texttt{ce}}(T)$ and for every i, $d_{\tau_i} = 0$ if and only if $\texttt{ce}(\tau_0 \cdots \tau_{i-1}) = \min_{\texttt{ce}}(T)$. In particular, observe that if a valid floating sequence of transitions T is applicable to a configuration (q, n, t), then for every $n' \geq n$ and weight $s' \in \mathcal{F}$, it is applicable to (q, n', s').

Since the machine is deterministic, for any word w, there is exactly one run labelled by w starting in a given configuration c_0. We denote this run $\pi(w, c_0)$. A run π is called an execution if $c_0 = (q_0, 0, s_0)$ is the initial configuration of $\mathcal{A}$. We use $\pi(w)$ to denote the execution labelled by w. A run $\pi(w, c_0) = c_0\tau_0c_1 \cdots \tau_{\ell-1}c_\ell$ is also represented as $c_0 \xrightarrow{w} c_\ell$. We use the notation $c_0 \rightarrow^* c_\ell$ to denote the existence of some word w such that $c_0 \xrightarrow{w} c_\ell$. The *weight assigned* to a word w by $\mathcal{A}$ along the run $\pi(w, c_0)$ is denoted by $f_{\mathcal{A}}(w, c_0) = s_{c_0} \circ \texttt{we}(\pi(w, c_0)) \circ \eta_F(q_{c_\ell})$. We use the notation $f_{\mathcal{A}}(w)$ to denote $f_{\mathcal{A}}(w, (q_0, 0, s_0))$.

Let $\mathcal{A}$ and $\mathcal{B}$ be two DWROCAs. Consider the configurations c_1 of $\mathcal{A}$ and c_2 of $\mathcal{B}$. We say that $c_1 \equiv_l c_2$ if and only if for all $w \in \Sigma^{\leq l}$, $f_{\mathcal{A}}(w, c_1) = f_{\mathcal{B}}(w, c_2)$. We say that the configurations c_1 and c_2 are equivalent if and only if $c_1 \equiv_l c_2$ for all $l \in \mathbb{N}$, and we denote this by $c_1 \equiv c_2$. We say that $\mathcal{A}$ and $\mathcal{B}$ are equivalent if for all

$w \in \Sigma^*$, $f_{\mathcal{A}}(w) = f_{\mathcal{B}}(w)$. A word $w \in \Sigma^*$ is called a *non-equivalence witness* (or simply a $witness$) for $\mathcal{A}$ and $\mathcal{B}$ if and only if $f_{\mathcal{A}}(w) \neq f_{\mathcal{B}}(w)$. A minimal witness is a non-equivalence witness of minimal length.

Given a set of words $w, w_0, \ldots, w_n \in \Sigma^*$ such that $w = w_0 w_1 \cdots w_n$, and a set I of indices in $[0, n]$: $I = \{i_0, \ldots, i_k\}$, we call w_{-I} the word obtained by removing from w all w_ℓ with $\ell \in I$.

Theorem 5.3

Let w be a non-equivalence witness for two DWROCAs $\mathcal{A}$ and $\mathcal{B}$ such that $w = w_0 \cdots w_n$, where $w_i \in \Sigma^*$ for all $i \in [0, n]$, and I, J be two disjoint set of indices in $[0, n]$. If there exist $s_I, s_J, t_I, t_J \in \mathcal{F}$, such that the following hold:

- $f_{\mathcal{A}}(w_{-I}) = f_{\mathcal{A}}(w) \circ s_I$, $f_{\mathcal{B}}(w_{-I}) = f_{\mathcal{B}}(w) \circ t_I$,
- $f_{\mathcal{A}}(w_{-J}) = f_{\mathcal{A}}(w) \circ s_J$, $f_{\mathcal{B}}(w_{-J}) = f_{\mathcal{B}}(w) \circ t_J$,
- $f_{\mathcal{A}}(w_{-(I \cup J)}) = f_{\mathcal{A}}(w) \circ s_I \circ s_J$, and $f_{\mathcal{B}}(w_{-(I \cup J)}) = f_{\mathcal{B}}(w) \circ t_I \circ t_J$.

then, either $f_{\mathcal{A}}(w_{-I}) \neq f_{\mathcal{B}}(w_{-I})$, or $f_{\mathcal{A}}(w_{-J}) \neq f_{\mathcal{B}}(w_{-J})$, or $f_{\mathcal{A}}(w_{-(I \cup J)}) \neq f_{\mathcal{B}}(w_{-(I \cup J)})$.

Proof. Since w is a non-equivalence witness for $\mathcal{A}$ and $\mathcal{B}$, we know that $f_{\mathcal{A}}(w) \neq f_{\mathcal{B}}(w)$. Now, assume for contradiction that $f_{\mathcal{A}}(w_{-I}) = f_{\mathcal{B}}(w_{-I})$, $f_{\mathcal{A}}(w_{-J}) = f_{\mathcal{B}}(w_{-J})$, $f_{\mathcal{A}}(w_{-(I \cup J)}) = f_{\mathcal{B}}(w_{-(I \cup J)})$, and the condition in the lemma holds. Therefore, $s_I \neq t_I$ (if not we will get that $f_{\mathcal{A}}(w_{-I}) \neq f_{\mathcal{B}}(w_{-I})$). Since $f_{\mathcal{A}}(w_{-J}) = f_{\mathcal{B}}(w_{-J})$, $f_{\mathcal{A}}(w_{-(I \cup J)}) = f_{\mathcal{A}}(w_{-J}) \circ s_I$, and $f_{\mathcal{B}}(w_{-(I \cup J)}) = f_{\mathcal{B}}(w_{-J}) \circ t_I$, we get that $f_{\mathcal{A}}(w_{-(I \cup J)}) \neq f_{\mathcal{B}}(w_{-(I \cup J)})$. This is a contradiction. Therefore either $f_{\mathcal{A}}(w_{-I}) \neq f_{\mathcal{B}}(w_{-I})$, or $f_{\mathcal{A}}(w_{-J}) \neq f_{\mathcal{B}}(w_{-J})$, or $f_{\mathcal{A}}(w_{-(I \cup J)}) \neq f_{\mathcal{B}}(w_{-(I \cup J)})$. □

Now, we will define some notions on DWROCAs, that will help us in the equivalence check. We will first define the notion of an underlying weighted automata.

5.2.3 Underlying Weighted Automata

Floating runs of a DWROCA are isomorphic to runs of the deterministic automaton obtained by ignoring counter values. We formalise this so-called notion of *underlying uninitialised weighted automaton* here. This is analogous to the notion of an underlying DFA defined in Böhm and Göller (2011).

Definition 5.4 (Underlying uninitialised weighted automaton)
The *underlying uninitialised weighted automaton* of a DWROCA $\mathcal{A} = (Q, \Sigma, q_0, s_0, \delta_0, \delta_1, \eta_F)$ is the uninitialised deterministic weighted automaton given by $\mathrm{U}(\mathcal{A}) = (Q, \Sigma, \delta'_1, \eta_F)$, where δ'_1 is a transition function from $Q \times \Sigma \to Q \times \mathcal{F}$ and is defined as follows:

$$\delta'_1(q_1, a) = (q_2, s) \text{ iff } \delta_1(q_1, a) = (q_2, o, s), \text{for some } o \in \{-1, 0, +1\}.$$

A configuration c of a DWROCA $\mathcal{A}$ is said to be k-equivalent to a configuration β of an uninitialised weighted automata $\mathcal{B}$, denoted $c \sim_k \beta$, if for all $w \in \Sigma^{\leq k}$, $f_{\mathcal{A}}(w, c) = f_{\mathcal{B}}(w, \beta)$.

Given an uninitialised weighted automaton $\mathcal{B}$ and $k > 0$, we partition the configurations of the DWROCA $\mathcal{A}$ into two parts: $\mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k)$ are those configurations that are k-equivalent to some configuration of $\mathcal{B}$, and $\mathsf{notEqConfig}(\mathcal{A}, \mathcal{B}, k)$ are those that are not.

$$\mathsf{notEqConfig}(\mathcal{A}, \mathcal{B}, k) = \{c \in Q \times \mathbb{N} \times \mathcal{F} \mid \forall \beta \in Q \times \mathcal{F},\ c \not\sim_k \beta\}.$$
$$\mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k) = \{c \in Q \times \mathbb{N} \times \mathcal{F} \mid \exists \beta \in Q \times \mathcal{F},\ c \sim_k \beta\}.$$

The following lemma shows that membership in $\mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k)$ is independent of the weight of the configuration. This can be observed immediately from the definition of $\mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k)$. We prove this by taking advantage of the existence of multiplicative inverses in the field and the commutativity of multiplication.

Lemma 5.5
For all $s, \bar{s} \in \mathcal{F} \setminus \{0_e\}$, $p \in Q$, $m \in \mathbb{N}$, $(p, m, s) \in \mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k)$ if and only if $(p, m, \bar{s}) \in \mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k)$.

Proof. Let us assume that $(p, m, s) \in \mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k)$. Hence, there exists a configuration $(q, t) \in Q \times \mathcal{F}$ of $\mathcal{B}$ such that for all $w \in \Sigma^{\leq k}$, $f_{\mathcal{A}}(w, (p, m, s)) = f_{\mathcal{B}}(w, (q, t))$. Multiplying with $\bar{s} \circ s^{-1}$ on both sides we get that for all $w \in \Sigma^{\leq k}$, $\bar{s} \circ s^{-1} \circ f_{\mathcal{A}}(w, (p, m, s)) = \bar{s} \circ s^{-1} \circ f_{\mathcal{B}}(w, (q, t))$. Let $\bar{t} = \bar{s} \circ s^{-1} \circ t$. Due to the commutativity of multiplication, we get that for all $w \in \Sigma^{\leq k}$, $f_{\mathcal{A}}(w, (p, m, \bar{s})) = f_{\mathcal{B}}(w, (q, \bar{t}))$. Therefore, $(p, m, \bar{s}) \in \mathsf{EqConfig}(\mathcal{A}, \mathcal{B}, k)$. $\square$

The distance of a configuration c of a DWROCA $\mathcal{A}$ to a set of configurations C of $\mathcal{A}$ is the length of a minimal word that takes you from c to a configuration in C and is defined as

$$\mathrm{dist}(c, C) = \min\{|w| \mid \exists c' \in C \text{ with } c \xrightarrow{w} c'\}.$$

Notice that $\mathrm{dist}(c, C) < \infty$ if and only if there exists $c' \in C$ such that $c \to^* c'$ in $\mathcal{A}$. We denote this by $c \to^* C$. By abuse of notation, we denote $c \xrightarrow{w} C$ if there exists a configuration $c' \in C$ such that $c \xrightarrow{w} c'$. The notion of distance will play a key role in determining which parts of the run of a non-equivalence witness can be pumped out if it is not minimal. Consider a configuration c such that $c \xrightarrow{*} \mathsf{notEqConfig}(\mathcal{A}, \mathcal{B}, k)$. The following lemma identifies a special word u such that $c \xrightarrow{u} \mathsf{notEqConfig}(\mathcal{A}, \mathcal{B}, k)$. The proof of the lemma is similar to that of the non-weighted case presented in Valiant and Paterson (1975) (Lemma 2) and Böhm and Göller (2011) (Lemma 6). Their proof ideas can be directly used to prove this without much modifications due to Lemma 5.5. However, we provide proof for the sake of completeness.

Lemma 5.6

Given a configuration c of a DWROCA $\mathcal{A}$, and a weighted automaton $\mathcal{B}$, if $c \xrightarrow{*}$ notEqConfig$(\mathcal{A}, \mathcal{B}, k)$ then there exists a word $u = u_1 u_2^r u_3$ (with $r \geq 0$) such that $c \xrightarrow{u}$ notEqConfig$(\mathcal{A}, \mathcal{B}, k)$ and the following conditions hold:

1. for all $w \in \Sigma^*$, if $c \xrightarrow{w}$ notEqConfig$(\mathcal{A}, \mathcal{B}, k)$, then $|w| \geq |u|$.
2. $|u_1 u_3| \leq 3|\mathcal{A}|^3$.
3. $|u_2| \leq |\mathcal{A}|$.
4. either $u_2 = \varepsilon$ or u_2 is a simple cycle of counter loss greater than zero.
5. $|u| \leq (max\{|\mathcal{A}|, n_c\} + |\mathcal{A}|^2)|\mathcal{A}|$ and the maximum counter value encountered during the run $c \xrightarrow{u}$ notEqConfig$(\mathcal{A}, \mathcal{B}, k)$ is less than $max\{|\mathcal{A}|, n_c\} + |\mathcal{A}|^2$.

Proof. Let us fix u to be a minimal length word such that $\pi = c \xrightarrow{u} c' \in$ notEqConfig$(\mathcal{A}, \mathcal{B}, k)$ for the rest of the proof. First, we prove Item 5, let us assume that the maximum counter value M encountered during the run $c \xrightarrow{u}$ notEqConfig$(\mathcal{A}, \mathcal{B}, k)$ is greater than $max\{|\mathcal{A}|, n_c\} + |\mathcal{A}|^2$. By the pigeon-hole principle, we can find configurations c_i, c_j, c'_i, c'_j and $d < |\mathcal{A}|$ such that $u = u_1 u_2 u_3 u_4 u_5$ and

$$c \xrightarrow{u_1} c_i \xrightarrow{u_2} c_j \xrightarrow{u_3} c'_j \xrightarrow{u_4} c'_i \xrightarrow{u_5} c'.$$

where $n_{c_i} = n_{c'_i}$, $n_{c_j} = n_{c'_j}$, $n_{c_j} = n_{c_i} + d$, $q_{c_i} = q_{c_j}$ and $q_{c'_i} = q_{c'_j}$. Also, the configurations are chosen in such a way that c_i and c_j (resp. c'_i and c'_j) are chosen to be the configurations where the counter values n_{c_i} and n_{c_j} (resp. $n_{c'_i}$ and $n_{c'_j}$) occurs for the last (resp. first) time during the run before (resp. after) reaching counter value M for the last time during the run.

Now, consider the run $c \xrightarrow{u_1 u_3 u_5} c'' = (q_{c'}, n_{c'}, s_{c''})$. From Lemma 5.5, we get that $c'' \in$ notEqConfig$(\mathcal{A}, \mathcal{B}, k)$ contradicting the minimality of u. We can now prove that $|u| \leq (max\{|\mathcal{A}|, n_c\} + |\mathcal{A}|^2)|\mathcal{A}|$. Assume that this is not the case. By the pigeon-hole principle, we can find configurations c_i, c_j with $q_{c_i} = q_{c_j}$ and $n_{c_i} = n_{c_j}$ such that $u = v_1 v_2 v_3$ and

$$c \xrightarrow{v_1} c_i \xrightarrow{v_2} c_j \xrightarrow{v_3} c' \in \text{notEqConfig}(\mathcal{A}, \mathcal{B}, k).$$

Now, consider the run $c \xrightarrow{v_1 v_3} c'' = (q_{c'}, n_{c'}, s_{c''})$. Again from Lemma 5.5, we get that $c'' \in \text{notEqConfig}(\mathcal{A}, \mathcal{B}, k)$ contradicting the minimality of u.

Now, we prove Items 1-4. Assume that $|n_{c'} - n_c| < |\mathcal{A}|^2$ then by Item 5 the lemma will trivially be true with $u_2 = \varepsilon$. So let us assume that $n_{c'} \geq n_c + |\mathcal{A}|^2$. First we find a simple loop $\pi' = c_1\tau_1 c_2 \tau_2 \ldots c_i$ in π with the maximum efficiency ϱ. Let $v_2 = \mathit{word}(\pi')$ denote the corresponding word. Now, we remove disjoint loops with maximal length from the run π to get another run π'' such that the sum of the counter-effects of the loops removed is a multiple of ϱ and preserve at least one occurrence of state q_{c_1} in π''. Now if $|\pi''| \geq |\mathcal{A}|^3$, then by applying the pigeon-hole principle, we can find more loops in π'' such that the sum of their counter-effect is a multiple of ϱ contradicting our maximality condition.

Now $\pi'' = c \xrightarrow{v_1} c'_1 \xrightarrow{v_2} e$ where $c'_1 = (q_{c_1}, n_{c'_1}, t_{c'_1})$ and $e = (q_{c'}, n_{c'} + r\varrho, t_e)$ for some $n_{c'_1}, r \in \mathbb{N}$, $t_{c'_1}, t_e \in \mathcal{F}$ and $v_1, v_3 \in \Sigma^*$ with $|v_1 v_3| < |\mathcal{A}|^2$. Since $|n_{c'} - n_c| < |\mathcal{A}|^2$ we get that $r > 0$. Therefore, $c \xrightarrow{v_1} c'_1 \xrightarrow{v_2^r} c'_2 \xrightarrow{v_3} e'$ where $c'_2 = (q_{c_1}, n_{c'_1} + r\varrho, t_{c'_2})$ and $e' = (q_{c'}, n_{c'}, t_{e'})$ for some $t_{c'_2}, t_{e'} \in \mathcal{F}$. Now, from Lemma 5.5, we get that $e' \in \text{notEqConfig}(\mathcal{A}, \mathcal{B}, k)$. Note that since we have replaced loops in π with ones having at least the same efficiency, the length of the string $v_1 v_2^r v_3$ is still minimal. $\square$

5.3 Equivalence

EQUIVALENCE PROBLEM
INPUT: Two DWROCAs $\mathcal{A}$ and $\mathcal{B}$ over a field $\mathcal{F}$.
OUTPUT: *Yes*, if $\mathcal{A}$ and $\mathcal{B}$ are equivalent. *No*, otherwise.

In the remainder of the section, we fix two DWROCAs, $\mathcal{A}_1$ and $\mathcal{A}_2$. We also fix $\mathsf{K} = |\mathcal{A}_1| + |\mathcal{A}_2|$. To simplify the reasoning, we will reason on the synchronised runs on pairs of configurations. Given two DWROCAs, we consider a *configuration pair* $c = \langle \chi_c, \psi_c \rangle$ where χ_c is a configuration of $\mathcal{A}_1$ and ψ_c is a configuration of $\mathcal{A}_2$. We similarly consider *transition pairs* of $\mathcal{A}_1$ and $\mathcal{A}_2$ and consider *synchronised runs* as the application of a sequence of transition pairs to a configuration pair. Let w be a minimal word that distinguishes $\mathcal{A}_1$ and $\mathcal{A}_2$. Henceforth, we will denote by

$$\Pi = c_0 \tau_0 c_1 \cdots \tau_{L-1} c_L.$$

the run pair of w from the initial configuration pair $c_0 = \langle (p_0, 0, s_0), (q_0, 0, t_0) \rangle$. We denote by $T_\Pi = \tau_0 \cdots \tau_{L-1}$ the sequence of transition pairs of this run pair. Given a run pair Π' of a word w' from c_0, we use $\mathtt{aw}(\Pi')$ to denote the tuple $(f_{\mathcal{A}_1}(w'),\ f_{\mathcal{A}_2}(w'))$.

Lemma 5.7
There is a polynomial $\mathrm{poly}_0 : \mathbb{N} \to \mathbb{N}$ such that if two DWROCAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent, then there exists a witness w such that counter values in the execution of w is less than $\mathrm{poly}_0(|\mathcal{A}_1| + |\mathcal{A}_2|)$.

This is the technically challenging part of the proof, and we will prove this later. Now, we show that the length of w is bounded by a polynomial, assuming the counter values in Π are bounded by $\mathrm{poly}_0(\mathsf{K})$.

Lemma 5.8 (Small model property)
There is a polynomial $\mathrm{poly}_1 : \mathbb{N} \to \mathbb{N}$ such that for any two non-equivalent DWROCAs $\mathcal{A}_1$ and $\mathcal{A}_2$, the length of a minimal witness w is less than or equal to $\mathrm{poly}_1(|\mathcal{A}_1| + |\mathcal{A}_2|)$.

Proof. Let $\mathrm{poly}_0(\mathsf{K})$ be a polynomial in K such that the counter values encountered during the execution of w are bounded by it. Assume for contradiction that $|w|$ is greater than $\mathrm{poly}_1(\mathsf{K}) = 2(\mathsf{K}\mathrm{poly}_0(\mathsf{K}))^2$. Consider the execution $\Pi = c_0\tau_0 c_1 \cdots \tau_{L-1} c_L$ of w. By the pigeon-hole principle, there exist $0 \le i_1 < j_1 < i_2 < j_2 \le L$ such that if the configuration pair $c_{i_1} = \langle (p_1, m_1, s_1), (q_1, n_1, t_1) \rangle$ then $c_{j_1} = \langle (p_1, m_1, s_1'), (q_1, n_1, t_1') \rangle$ and if $c_{i_2} = \langle (p_2, m_2, s_2), (q_2, n_2, t_2) \rangle$ then $c_{j_2} = \langle (p_2, m_2, s_2'), (q_2, n_2, t_2') \rangle$ in Π.

We can deduce that $T_1 = T_{0\cdots i_1-1} T_{j_1\cdots L-1}$, $T_2 = T_{0\cdots i_2-1} T_{j_2\cdots L-1}$ and $T_3 = T_{0\cdots i_1-1} T_{j\cdots i_2-1} T_{j_2\cdots L-1}$ are such that the three runs $T_1(c_0)$, $T_2(c_0)$ and $T_3(c_0)$ are all valid runs that are shorter than Π and end in configuration pairs differing only by their weights. Let $\mathtt{aw}(\Pi) = (s, t)$, $\mathtt{aw}(T_1(c_0)) = (s_1, t_1)$, $\mathtt{aw}(T_2(c_0)) = (s_2, t_2)$, and $\mathtt{aw}(T_3(c_0)) = (s_3, t_3)$. We know from Theorem 5.3 that there exists an $i \in \{1, 2, 3\}$ such that $s_i \ne t_i$ and hence π_i contradicts the minimality of Π. □

We now state the main result of this chapter. We use a small model property stated in Lemma 5.8 to prove this theorem. This property ensures that if two

DWROCAs are not equivalent, then there exists a small word that can distinguish them. The small model property helps us to reduce the equivalence problem of DWROCAs to that of weighted automata by "simulating" the runs of DWROCAs up to polynomial length by two weighted automata.

Theorem 5.9
There is a polynomial time algorithm that decides if two DWROCAs are equivalent or outputs a minimal witness.

Proof. Given a DWROCA $\mathcal{A}$ and $M \in \mathbb{N}$, we define the M-unfolding weighted automata $\mathcal{A}^M$ as a finite state weighted automaton, where the accepting weight of any word whose run does not encounter counter values greater than M in $\mathcal{A}$ is equal in both $\mathcal{A}$ and $\mathcal{A}^M$. Let $\mathcal{A} = (Q, \Sigma, q_0, s_0, \delta_0, \delta_1, \eta_F)$ be a DWROCA and $M \in \mathbb{N}$, we construct its M-unfolding weighted automaton $\mathcal{A}^M = (Q', \Sigma, p_0, s_0, \delta, \xi_F)$ as defined below:

- $Q' = Q \times [0, M]$.
- $p_0 = (q_0, 0)$.
- $\xi_F(q, n) = \eta_F(q)$.
- $\delta((q,n),a) = \begin{cases} ((q', n+o), s) \text{ if } n = 0, \delta_0(q,a) = (q', o, s), & \text{and } n + o \leq M. \\ ((q', n+o), s) \text{ if } n \neq 0, \delta_1(q,a) = (q', o, s), & \text{and } n + o \leq M. \\ \text{undefined} \quad \text{otherwise.} \end{cases}$

It is easy to see that for every word $w \in \Sigma^{\leq M}$, $f_{\mathcal{A}}(w) = f_{\mathcal{A}^M}(w)$. Furthermore $|\mathcal{A}^M| \leq |\mathcal{A}| \times M$. We now consider two DWROCAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and let $M = \text{poly}_1(|\mathcal{A}_1| + |\mathcal{A}_2|)$ where poly_1 is provided by Lemma 5.8. The lemma ensures that the two machines are not equivalent if and only if there is a word in $\Sigma^{\leq M}$ on which they differ.

If two deterministic weighted automata (DWA) are not equivalent, then there is a linear-sized word to distinguish them and the equivalence check can be done in polynomial time (Tzeng, 1992, Lemma 3.4) that returns a minimal word that distinguishes them. Tzeng (1992) shows this for two probabilistic automata. The proof can be extended to weighted automata over fields. Using this, we can now decide the equivalence of $\mathcal{A}_1^M$ and $\mathcal{A}_2^M$ in polynomial time and obtain a minimal word $z \in \Sigma^*$ that distinguishes them, if they are not equivalent. We say that $\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent if and only if $|z| \leq M$. Thus, we have a polynomial time procedure to decide the equivalence of DWROCAs, provided that Lemma 5.8 holds. □

Since deterministic real-time one-counter automata are DWROCAs with weight 1 on its transitions, we get the following corollary.

Corollary 5.10

There is a polynomial time algorithm that decides if two deterministic real-time one-counter automata are equivalent or outputs a minimal witness.

The rest of this section is dedicated to proving Lemma 5.7.

5.3.1 Proof Strategy

A crucial idea here is to partition the set of configurations between those that behave like a weighted automata on short runs and those that do not. In Section 5.2.3, we have defined the notion of underlying uninitialised weighted automaton for a DWROCA. Comparing the configuration of a DWROCA with a configuration of a weighted automaton helps us reduce the problem to that of weighted automata. The technique is adapted from the ideas used in Böhm and Göller (2011) and Böhm et al. (2010), where the non-weighted model is studied. We show that the addition of weights does not change the proof outline presented by Böhm and Göller (2011) and can be used for the weighted model as well with suitable adaptations.

As we need to test the equivalence of configurations from $\mathcal{A}_1$ and $\mathcal{A}_2$, we will consider the disjoint union of $\mathrm{U}(\mathcal{A}_1)$ and $\mathrm{U}(\mathcal{A}_2)$ to have a single automaton to compare their configurations with. We use $\mathrm{U}(\mathcal{A}_1) \cup \mathrm{U}(\mathcal{A}_2)$ to denote this automaton and call it the underlying weighted automata. The set notUWA consists of all configurations of $\mathcal{A}_1$ and $\mathcal{A}_2$ that do not have a K-equivalent configuration in either $\mathrm{U}(\mathcal{A}_1)$ or $\mathrm{U}(\mathcal{A}_2)$ and is defined as follows

$$\mathsf{notUWA} = \bigcup_{i\in\{1,2\}} \bigcap_{j\in\{1,2\}} \mathsf{notEqConfig}(\mathcal{A}_i, \mathrm{U}(\mathcal{A}_j), \mathsf{K}).$$

The set WA, the complement of notUWA, contains all configurations that are K-equivalent to some configuration in the underlying weighted automata. Note that if a configuration $c \in \mathsf{notUWA}$, then $n_c < \mathsf{K}$. For a configuration $c = (q_c, n_c, s_c)$ of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) with $n_c \geq \mathsf{K}$, the configuration (q_c, s_c) of $\mathrm{U}(\mathcal{A}_1)$ (resp. $\mathrm{U}(\mathcal{A}_2)$) itself is K-equivalent to it. This is due to the fact that the counter value cannot reach zero on reading words of length less than or equal to K from a configuration with a counter value greater than K, and the underlying DWA can hence simulate these runs. The distance that will be interesting in our reasoning is the one to the set notUWA. Therefore for simplicity, we denote $\mathrm{dist}(\xi) = \mathrm{dist}(\xi, \mathsf{notUWA})$ for a configuration ξ of $\mathcal{A}_1$ or $\mathcal{A}_2$.

We say that a configuration pair $c = \langle \chi_c, \psi_c \rangle$ is

- surely-equivalent: If $\chi_c {\equiv_{\mathsf{K}}} \psi_c$ and $\mathrm{dist}(\chi_c) = \mathrm{dist}(\psi_c) = \infty$.
- surely-nonequivalent: $\chi_c {\not\equiv_{\mathsf{K}}} \psi_c$ or $\mathrm{dist}(\chi_c) \neq \mathrm{dist}(\psi_c)$.
- unresolved: otherwise, i.e., $\chi_c {\equiv_{\mathsf{K}}} \psi_c$ and $\mathrm{dist}(\chi_c) = \mathrm{dist}(\psi_c) < \infty$.

Let us denote by $\chi_0 = (p_0, 0, s_0)$ and $\psi_0 = (q_0, 0, t_0)$ the initial configurations of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively and $\chi_0 \not\equiv \psi_0$. The categorisation of the configuration pairs into surely-equivalent, surely-nonequivalent and unresolved allows us to solve two easy cases before concentrating on the crux of the argument. It is easy to observe that no configuration pair in Π is surely-equivalent (see Lemma 5.11). Our next observation (see Lemma 5.12) is that if a configuration pair c_j in Π is surely-nonequivalent and has counter values $\{m_j, n_j\}$, then the counter values in the rest of the execution is bounded by a polynomial in m_j, n_j and K. Hence it will remain to prove that if c_j is the first surely-nonequivalent configuration pair in Π, then m_j and n_j are bounded by a polynomial. In order to do this, we prove that any minimal run-pair composed solely of unresolved configuration pairs has all of its counter values bounded by a polynomial. This is the most challenging part of the proof.

$$\overbrace{c_0\tau_0c_1\tau_1c_2\cdots c_{j-1}\tau_{j-1}}^{\substack{\text{unresolved configuration pairs}\\ \text{counters poly-bounded}}}\ \underbrace{c_j = \langle \chi_{c_j}, \psi_{c_j}\rangle}_{\substack{1^{st}\text{ surely-nonequivalent}\\ \text{configuration pair}}}\ \overbrace{\tau_j\cdots\tau_{L-1}c_L}^{\substack{\text{counters}\\ \text{poly-bounded}}}$$

We now solve the cases when we have configuration pairs that are surely-equivalent or surely-nonequivalent. The proof is similar to that of the non-weighted case. The case of unresolved configuration pairs needs to be solved using a different technique and is taken care of in Section 5.3.3.

Lemma 5.11

There is no *surely-equivalent* configuration pair in Π.

Proof. Assume for contradiction that there exists a surely-equivalent configuration pair $c_i = \langle \chi_{c_i}, \psi_{c_i}\rangle$ in Π. Since Π is an execution of a witness $\chi_{c_i} \not\equiv \psi_{c_i}$. Let v be a minimal witness of χ_{c_i} and ψ_{c_i}. Since c_i is a surely-equivalent configuration pair, $\chi_{c_i} \equiv_{\mathsf{K}} \psi_{c_i}$ and thus $|v| > \mathsf{K}$. Therefore, there exists a prefix of v, $u \in \Sigma^{|v|-\mathsf{K}}$, and configurations χ_{c_j} and ψ_{c_j} such that $\langle \chi_{c_i}, \psi_{c_i}\rangle \xrightarrow{u} \langle \chi_{c_j}, \psi_{c_j}\rangle$ and $\chi_{c_j} \not\equiv_{\mathsf{K}} \psi_{c_j}$.

Since v is a minimal witness $\chi_{c_j} \equiv_{\mathsf{K}-1} \psi_{c_j}$. Since c_i is surely-equivalent, $\mathrm{dist}(\chi_{c_i}) = \mathrm{dist}(\psi_{c_i}) = \infty$, χ_{c_j} and ψ_{c_j} are in the set WA. In other words, there exists configurations β and γ in the automaton $\mathrm{U}(\mathcal{A}_1) \cup \mathrm{U}(\mathcal{A}_2)$ such that $\chi_{c_j} \sim_{\mathsf{K}} \beta$ and $\psi_{c_j} \sim_{\mathsf{K}} \gamma$. Since $\chi_{c_j} \equiv_{\mathsf{K}-1} \psi_{c_j}$, it follows that $\beta \sim_{\mathsf{K}-1} \gamma$. From the equivalence result of weighted automata (Tzeng, 1992; Schützenberger, 1961), we know that this is sufficient to prove that the automata with β and γ as initial distributions are equivalent, and thus, in particular, that $\beta \sim_{\mathsf{K}} \gamma$. That allows to deduce $\chi_{c_j} \equiv_{\mathsf{K}} \psi_{c_j}$, which is a contradiction. □

The following lemma bounds the length of a minimal witness from a surely-nonequivalent configuration pair.

Lemma 5.12

Let $c_j = \langle \chi_{c_j}, \psi_{c_j}\rangle$ be the first *surely-nonequivalent* configuration pair in Π, then $L - j \leq \min\{\mathrm{dist}(\chi_{c_j}), \mathrm{dist}(\psi_{c_j})\} + \mathsf{K}$.

Proof. We separately consider the two cases for c_j to be surely-nonequivalent.
Case-1, $\chi_{c_j} \not\equiv_{\mathsf{K}} \psi_{c_j}$: then clearly $L - j \leq \mathsf{K}$, as there is a $w \in \Sigma^{\leq \mathsf{K}}$ that is a witness from c_j.
Case-2, $\mathrm{dist}(\chi_{c_j}) \neq \mathrm{dist}(\psi_{c_j})$: Without loss of generality, we suppose $\mathrm{dist}(\chi_{c_j}) < \mathrm{dist}(\psi_{c_j})$. By definition of dist, there exists a $u \in \Sigma^{\mathrm{dist}(\chi_{c_j})}$ such that $\chi_{c_j} \xrightarrow{u} \chi_{c_{j'}}$ and $\psi_{c_j} \xrightarrow{u} \psi_{c_{j'}}$ where $\chi_{c_{j'}} \in \mathsf{notUWA}$ and $\psi_{c_{j'}} \in \mathsf{WA}$. By definition of WA, there is a configuration β of the automaton $\mathrm{U}(\mathcal{A}_1) \cup \mathrm{U}(\mathcal{A}_2)$ such that $\psi_{c_{j'}} \sim_{\mathsf{K}} \beta$. As $\chi_{c_{j'}} \in \mathsf{notUWA}$, $\chi_{c_{j'}} \not\sim_{\mathsf{K}} \beta$ and thus $\chi_{c_{j'}} \not\equiv_{\mathsf{K}} \psi_{c_{j'}}$. Therefore, there exists a $v \in \Sigma^{\leq \mathsf{K}}$ such that $f(v, \chi_{c_{j'}}) \neq f(v, \psi_{c_{j'}})$ and hence $f(uv, \chi_{c_j}) \neq f(uv, \psi_{c_j})$. As $uv \in \Sigma^{\mathrm{dist}(\chi_{c_j})+\mathsf{K}}$, we get that $\chi_{c_j} \not\equiv_{\mathrm{dist}(\chi_{c_j})+\mathsf{K}} \psi_{c_j}$ and $L - j \leq \mathrm{dist}(\psi_{c_j}) + \mathsf{K}$.

In both the cases, we get that $L - j \leq \min\{\mathrm{dist}(\chi_{c_j}), \mathrm{dist}(\psi_{c_j})\} + \mathsf{K}$. □

From Lemma 7.12, we know that the distance of a configuration is polynomially bounded with respect to its counter value and K. Therefore, Lemma 5.12 tells us that the length of a minimal witness from a surely-nonequivalent configuration pair is polynomially bounded with respect to its counter value and K. A length bound implies a bound on the counter value also, since the counters cannot increase more than the length of a word. The next two sub-sections focus on the remaining case of paths containing only unresolved configuration pairs.

5.3.2 Configuration Space

Each pair of configurations $c = \langle \chi, \psi \rangle$ is mapped to a point in the space $\mathbb{N} \times \mathbb{N} \times (Q \times Q) \times \mathcal{F} \times \mathcal{F}$, henceforth referred to as the *configuration space*, where the first two dimensions represent the two counter values, the third dimension $Q \times Q$ corresponds to the pair of control states, the fourth and fifth dimensions represent the weights. We partition the configuration space into initial space, belt space, and background space. This partition is indexed on two polynomials, $\mathrm{poly}_2 = 516\mathsf{K}^{21}$ and $\mathrm{poly}_3 = 42\mathsf{K}^{14}$.

- initial space: All configuration pairs $\langle (p, m, s), (q, n, t) \rangle$ such that $m, n < \mathrm{poly}_2(\mathsf{K})$.
- belt space: Let $\alpha, \beta \geq 1$ be co-prime. The belt of thickness d and slope $\frac{\alpha}{\beta}$ consists of those configuration pairs $\langle (p, m, s), (q, n, t) \rangle$ that satisfy $|\alpha.m -$

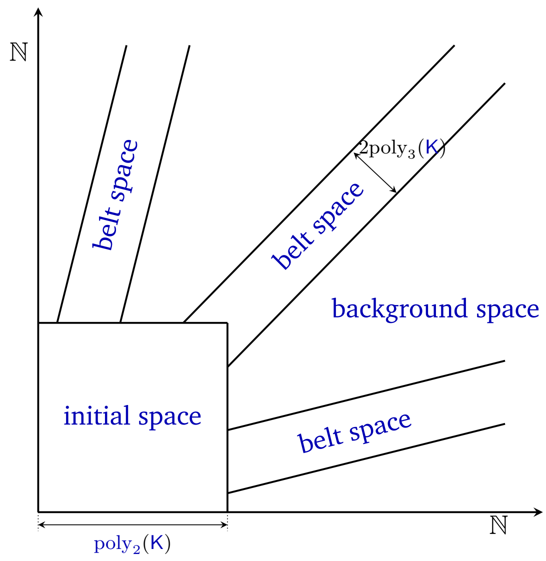


Fig. 5.1. Configuration space.

$\beta.n| \leq d$. The belt space contains all configuration pairs $\langle (p, m, s), (q, n, t) \rangle$ outside the initial space that are inside a belt with thickness $\text{poly}_3(\mathsf{K})$ and slope $\frac{\alpha}{\beta}$, where $\alpha, \beta \in [1, \mathsf{K}^2]$.

- background space: All remaining configuration pairs.

The projection of the configuration space onto the first two dimensions is depicted in Figure 5.1. The polynomials poly_2 and poly_3 are chosen in such a way that all unresolved configuration pairs fall in a belt or the initial space, and all belts are disjoint. This is proved in Lemma 7.17. In fact, these polynomials can be used as the size of the initial space and thickness of belts in the paper by Böhm and Göller (2011) since the properties to be ensured are similar. The precise polynomials are used in the lemma proving these properties.

If the maximum counter value encountered during the execution of the minimal witness is far greater than the size of the initial space, then its length inside the belt space is very long. Since the belts are disjoint outside the initial space, we just need to show that if the length of the run of the witness inside a belt is very long, then we can find a shorter witness by pumping some portion out from the current witness. This is proved in Section 5.3.3 using Lemma 5.14 and Lemma 7.5, that enables us to perform this cut.

The partition of the unresolved configuration pairs thus helps us to show that there is a restriction on the run containing only unresolved configuration pairs in Π. The intuitive idea is that when at least one counter value is very big, the ratios between the counter value pairs on long run-pairs cannot take arbitrarily many different values. If there is a long portion of Π where the ratio between the counter value pairs repeats, we can pump a sub-run out of it and still get a witness contradicting its minimality.

The proof of the following lemma is the same as that of the non-weighted case presented by Böhm and Göller (2011).

Lemma 5.13

If $c = \langle (p_c, m_c, s_c), (q_c, n_c, t_c) \rangle$ is an unresolved configuration pair with $max\{m_c, n_c\} > \mathrm{poly}_2(\mathsf{K})$ then,

1. it lies in a unique belt of thickness $\mathrm{poly}_3(\mathsf{K})$ and slope $\frac{\alpha}{\beta}$, where $\alpha, \beta \in [1, \mathsf{K}^2]$.
2. it cannot take a transition pair to reach a configuration pair c', inside another belt of thickness $\mathrm{poly}_3(\mathsf{K})$ and slope $\frac{\alpha'}{\beta'}$, where $\alpha', \beta' \in [1, \mathsf{K}^2]$.

Proof. To prove Lemma 7.17, we first show that the following claim holds. The proof of this claim is similar to that of the non-weighted case presented by Böhm et al. (2010).

Claim 1. *Let $i \in \{1, 2\}$ and $\chi = (q_\chi, n_\chi, t_\chi)$ be a configuration of $\mathcal{A}_i$. If $\mathrm{dist}(\chi) < \infty$ then, $\mathrm{dist}(\chi) = \frac{a}{b} n_\chi + d$ where $a, b \in [0, |\mathcal{A}_i|]$ and $|d| \le 6|\mathcal{A}_i|^4$.*

Proof. Let us assume that $\mathrm{dist}(\chi) < \infty$. This means that $\chi \to^* \chi'$ with $\chi' \in$ notUWA. Let $k = |\mathcal{A}_i|$. Since $n_{\chi'} < k$, by Lemma 7.12, we know that there is a word $u = u_1 u_2^r u_3$ (with $r \ge 0$) such that that $\chi \xrightarrow{*} \chi'$ where $|u| = \mathrm{dist}(\chi)$, $|u_1 u_3| \le 3k^3$, $|u_2| \le k$ and u_2 is a simple cycle with counter loss $\ell \le k$. Let g be the counter gain/loss of $u_1 u_3$ and $\alpha = \frac{|u_2|}{\ell}$. Since $|u_1 u_3| \le 3k^3$, we have $|g| \le 3k^3$.

$$\begin{aligned} \text{dist}(\chi) &= \frac{n_\chi - n_{\chi'} - g}{\ell}|u_2| + |u_1 u_3| \\ &= \alpha n_\chi - \underbrace{\alpha(n_{\chi'} + g) + |u_1 u_3|}_{d} \end{aligned}$$

Since $1 \leq \alpha \leq k$ it follows that $-6k^4 \leq d \leq 6k^4$. $\square_{Claim:1}$

Let $c = \langle (p, m, s), (q, n, t) \rangle$ be an unresolved configuration pair with $max\{m_c, n_c\} > \text{poly}_2(\mathsf{K})$. Recall that $\text{poly}_2(\mathsf{K}) = 516\mathsf{K}^{21}$ and $\text{poly}_3(\mathsf{K}) = 42\mathsf{K}^{14}$.

1. Let $\chi = (p, m, s)$ and $\psi = (q, n, t)$. Since $\langle \chi, \psi \rangle$ is an unresolved configuration pair we have $\text{dist}(\chi) = \text{dist}(\psi)$. From Claim 1, there exists $a_1, b_1, a_2, b_2 \in [0, \mathsf{K}]$ and $d_1, d_2 \leq 6\mathsf{K}^4$ such that

$$\frac{a_1}{b_1} m + d_1 = \text{dist}(\chi) = \text{dist}(\psi) = \frac{a_2}{b_2} n + d_2$$

Therefore $|\frac{a_1}{b_1} m - \frac{a_2}{b_2} n| \leq |d_2 - d_1| \leq 6\mathsf{K}^4$. This satisfies the belt condition.

2. Let B and B' be two distinct belts with μ being the slope of belt B and μ' the slope of belt B'. Hence $\mu \neq \mu'$. Without loss of generality, let us assume that $\mu' > \mu$. It suffices to show that for all $x > 516\mathsf{K}^{21}$, we have

$$\mu x + 42\mathsf{K}^{14} + 1 < \mu' x - 42\mathsf{K}^{14} - 1$$

We know that $\mu' - \mu \geq \frac{1}{(\mathsf{K})^2}$ and $x > 516\mathsf{K}^{21}$.

$$\begin{aligned} \frac{14\mathsf{K}^6}{\mathsf{K}^2} < (\mu' - \mu)x &\implies \mu x + \frac{14\mathsf{K}^4}{2} < \mu' x - \frac{14\mathsf{K}^4}{2} \\ &\implies \mu x + 42\mathsf{K}^{14} + \mathsf{K}^4 < \mu' x - 42\mathsf{K}^{14} - \mathsf{K}^4 \end{aligned}$$

This concludes the proof. $\square$

By Lemma 7.17, the belt in which an unresolved configuration pair c lies is uniquely determined. It also ensures that the belts are disjoint outside the initial space and that no run composed only of unresolved configuration pairs can go from one belt to another without passing through the initial space.

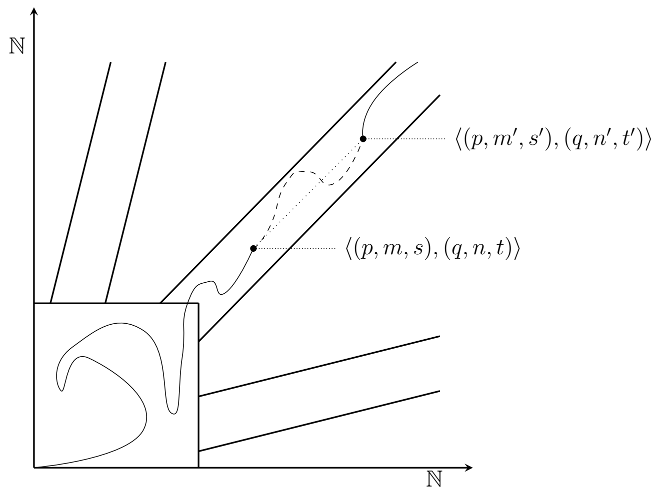


Fig. 5.2. An α-β repetition.

Let $\alpha, \beta \in [1, \mathsf{K}^2]$ be co-prime and $\langle (p, m, s), (q, n, t) \rangle$, $\langle (p', m', s'), (q', n', t') \rangle$ be two configuration pairs. They are α-β *related* if $p = p'$ and $q = q'$ and $\alpha.m - \beta.n = \alpha.m' - \beta.n'$. Roughly speaking, two configuration pairs are α-β related if they have the same state pairs and lie on a line with slope $\frac{\alpha}{\beta}$. An α-β *repetition* is a run-pair $\bar{\pi}_1 = c_i \tau_i c_{i+1} \tau_{i+1} \cdots \tau_{j-1} c_j$ such that the configuration pairs c_i and c_j are α-β related. The projection of an α-β repetition onto the counters is depicted in Figure 7.7.

The following two lemmas help us to show that if there is a witness having a very long sub-run inside a belt, then we can find a shorter witness whose sub-run inside that belt is shorter.

Lemma 5.14 (Cut lemma)

Given a configuration pair c and a sequence of transition pairs T such that all configuration pairs of $T(c)$ are inside a belt with slope $\frac{\alpha}{\beta}$ with $\alpha, \beta \in [1, \mathsf{K}^2]$, and either $\mathtt{ce}(T)|_1 > \mathsf{K}^2 \mathrm{poly}_3(\mathsf{K})$ or $\mathtt{ce}(T)|_2 > \mathsf{K}^2 \mathrm{poly}_3(\mathsf{K})$, then there exist $i < j$ such that:

- $T_{1 \cdots i}(c)$ and $T_{1 \cdots j}(c)$ are α-β related configuration pairs.
- $T_{1 \cdots i} T_{j+1 \cdots |T|}(c)$ is a run inside the same belt.

Proof. Let us assume that $\mathtt{ce}(T)|_1 > \mathsf{K}^2\text{poly}_3(\mathsf{K})$. The case for the second component can be proven analogously. For any $i \in [0, \mathtt{ce}(T)]$, we denote by ℓ_i the index such that $\mathtt{ce}(T_{0\cdots\ell_i}) = i$ and for any $k > \ell_i$, $\mathtt{ce}(T_{0\cdots k}) > i$. We denote $c_i = T_{0\cdots\ell_i}(c)$. As there are at most K^2 many possible state pairs, and $\text{poly}_3(\mathsf{K})$ many possible values for $\alpha m_c - \beta n_c$ for a configuration pair c in a belt. By the pigeon-hole principle, there exist $1 \leq i < j \leq |T|$ such that c_i and c_j are α-β related. As for every $k > \ell_j$, $\mathtt{ce}(T_{0\cdots k}) > \mathtt{ce}(T_{0\cdots\ell_j})$, we have that $\mathtt{ce}(T_{\ell_j+1\cdots k}) \geq 0$. Furthermore, as c_i and c_j have the same control states, $T_{\ell_j+1\cdots|T|}$ can be applied to c_i.

Given $k \in [\ell_j, |T|]$, we call $d_k = T_{\ell_j+1\cdots k}(c_j)$ and $e_k = T_{\ell_j+1\cdots k}(c_i)$. We have

$$\begin{aligned}
\alpha m_{e_k} - \beta n_{e_k} &= \alpha(m_{c_i} + \mathtt{ce}(T_{\ell_j+1\cdots k}))|_1 - \beta(m_{c_i} + \mathtt{ce}(T_{\ell_j+1\cdots k}))|_2 \\
&= \alpha m_{c_i} - \beta m_{c_i} + \alpha(\mathtt{ce}(T_{\ell_j+1\cdots k})|_1) - \beta(\mathtt{ce}(T_{\ell_j+1\cdots k})|_2).
\end{aligned}$$

Also $\alpha m_{e_k} - \beta n_{e_k} = \alpha m_{c_j} - \beta m_{c_j} + \alpha(\mathtt{ce}(T_{\ell_j+1\cdots k})|_1) - \beta(\mathtt{ce}(T_{\ell_j+1\cdots k})|_2) = \alpha m_{d_k} - \beta n_{d_k}$, since c_i and c_j are α-β related. Thus d_k and e_k are also α-β related. As d_k is inside the belt, e_k is also inside the belt. Therefore, $T_{0\cdots\ell_i}T_{\ell_j+1\cdots|T|}(c)$ is a run inside the belt. □

In the following lemma, we show that if there are long runs that stay inside a belt, one can find shorter runs inside the same belt whose last configuration pairs have identical states and counter values (but not weights).

Lemma 5.15 (U-turn lemma)

Let c be a configuration pair, and T a sequence of transition pairs such that:

- $T(c)$ is completely inside some belt with slope $\frac{\alpha}{\beta}$.
- $\mathtt{ce}(T) = 0$.
- $max(\mathtt{ce}(T_{0\cdots k}) \mid 0 \leq k \leq |T|) > (\mathsf{K}^2\text{poly}_3(\mathsf{K}) + 1)^2$.

Then, there exist $i < j < k < \ell$ such that $T' = T_{1\cdots i}T_{j+1\cdots k}T_{\ell+1\cdots|T|}$ is such that $T'(c)$ is a run inside the same belt with $\mathtt{ce}(T') = 0$, and the ending configuration pairs of $T'(c)$ and $T(c)$ only differ by their weights.

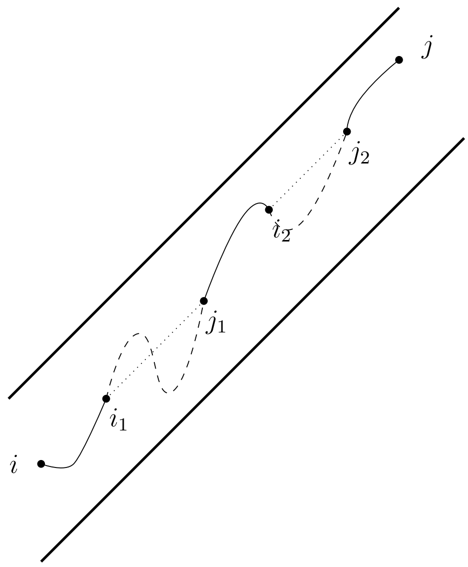


Fig. 5.3. A belt visit ending in a belt.

Proof. We denote $M = max(\texttt{ce}(T_{0\cdots k}) \mid 0 \leq k \leq |T|)$. For any $i \in [0, M]$, we denote by u_i and d_i the indices such that $\texttt{ce}(T_{0\cdots u_i}) = \texttt{ce}(T_{0\cdots d_i}) = i$, and for any $j \in [u_i + 1, d_i - 1]$, $\texttt{ce}(T_{0\cdots j}) > i$. We call $c_i = T_{0\cdots u_i}(c)$ and $c'_i = T_{0\cdots d_i}(c)$.

By the pigeon-hole principle, there exist $i_1 < i_2 \ldots < i_{\mathsf{K}^2\text{poly}_3(\mathsf{K})} \in [0, M]$ such that all configuration pairs c_{i_r}, $r \in [1, \mathsf{K}^2\text{poly}_3(\mathsf{K})]$ are α-β related. Now consider the configuration pairs c'_{i_r}, $r \in [1, \mathsf{K}^2\text{poly}_3(\mathsf{K})]$. From Lemma 5.14, we know that there exist $i < j \in [1, \mathsf{K}^2\text{poly}_3(\mathsf{K})]$ such that the configuration pairs c'_i and c'_j that are α-β related. Note that the configuration pairs c_i and c_j are also α-β related.

Similar to the previous lemma, we get that $T_{u_j+1\cdots d_j}$ is applicable to c_i. Furthermore, $b_i = T_{u_j+1\cdots d_j}(c_i)$ and c'_i only differ by their weights. We can conclude that $T' = T_{0\cdots u_i}T_{u_j+1\cdots d_j}T_{d_i+1\cdots |T|}$ is such that $T'(c)$ is a run inside the same belt, and that $\texttt{ce}(T') = 0$, and the ending states of T and T' are the same. $\square$

5.3.3 Bounding the Counter Values in Unresolved Configuration Pairs

In this section, we look at the run-pair of the minimal witness and show that counter values appearing on portions included in belts and composed only of unresolved configuration pairs can be bounded by some polynomial. There are

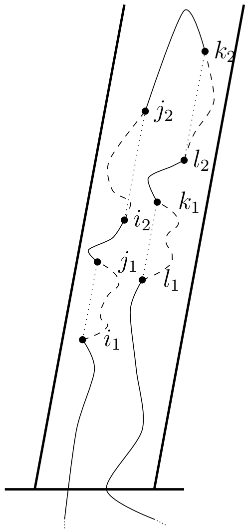


Fig. 5.4. A belt visit returning to the initial space.

two cases to consider here. The first one is where we have a very long belt visit, that does not enter the background space, and the second is the case when it does. We use Theorem 5.3, Lemma 5.14 and Lemma 7.5 to easily show that in the first case, the length of the minimal witness can be bounded and is proved in Lemma 5.16. The second and the most difficult case is when we have a very long belt visit, that enters the background space from the belt space and is considered in Lemma 5.17. In this case, we show that if the minimal witness reaches a surely-nonequivalent configuration pair whose counter values are very large, then we can pump out some portion of the run inside a belt to reach another surely-nonequivalent configuration pair whose counter values are polynomially bounded.

Lemma 5.16 (Belt lemma)
We consider $T = \tau_0 \cdots \tau_{L-1}$ the sequence of transition pairs in Π. Suppose there are $0 < i < k < L-1$ such that $\tau_i \cdots \tau_k(c_i)$ is included in a belt with either $m_{c_i} = m_{c_k}$ or $n_{c_i} = n_{c_k}$ then for every $i \leq r \leq k$, $\mathtt{ce}(\tau_i \cdots \tau_r)|_1 \leq 2((\mathsf{K}^2\mathrm{poly}_3(\mathsf{K}))^2 + 1)$, and $\mathtt{ce}(\tau_i \cdots \tau_r)|_2 \leq 2((\mathsf{K}^2\mathrm{poly}_3(\mathsf{K}))^2 + 1)$.

Proof. Let us assume that there are $0 < i < k < L-1$ such that $\tau_i \cdots \tau_k(c_i)$ is included in a belt with $m_{c_i} = m_{c_k}$. The case where $n_{c_i} = n_{c_k}$ can be proven analogously. We call $R = \tau_i \cdots \tau_k$. Let $\max(\mathtt{ce}(\tau_i \cdots \tau_\ell)|_1) \mid i \leq \ell \leq k) > 2((\mathsf{K}^2\text{poly}_3(\mathsf{K}))^2 + 1)$. By applying Lemma 7.5 twice, we can find $i_1 < j_1 \leq i_2 < j_2 < k_2 < \ell_2 \leq k_1 < \ell_1$ such that $R_1 = R_{0\cdots i_1} R_{j_1+1\cdots k_1} R_{\ell_1+1\cdots|R|}$, $R_2 = R_{0\cdots i_2} R_{j_2+1\cdots k_2} R_{\ell_2+1\cdots|R|}$, and $R_{12} = R_{0\cdots i_1} R_{j_1+1\cdots i_2} R_{j_2+1\cdots k_2} R_{\ell_2+1\cdots k_1} R_{\ell_1+1\cdots|R|}$ are such that $R_1(c_i)$, $R_2(c_i)$ and $R_{12}(c_i)$ are runs included in the same belt as $R(c)$ and the ending configuration pairs only differ by their weights.

We can deduce that $T_1 = T_{0\cdots i-1} R_1 T_{k+1\cdots|T|}$, $T_2 = T_{0\cdots i-1} R_2 T_{k+1\cdots|T|}$ and $T_3 = T_{0\cdots i-1} R_{12} T_{k+1\cdots|T|}$ are such that the three runs $T_1(c_0)$, $T_2(c_0)$ and $T_{12}(c_0)$ are all valid runs that are shorter than $T(c)$ and end in configuration pairs only differing by their weights.

Let $\mathtt{aw}(T(c_0)) = (s, t)$, $\mathtt{aw}(T_1(c_0)) = (s_1, t_1)$, $\mathtt{aw}(T_2(c_0)) = (s_2, t_2)$, and $\mathtt{aw}(T_3(c_0)) = (s_3, t_3)$. From Theorem 5.3, we know that there exists an $i \in \{1, 2, 3\}$ such that $s_i \neq t_i$ and hence $T_i(c_0)$ contradicts the minimality of Π. □

Lemma 5.17

Let c_j be the first surely-nonequivalent configuration pair in Π. Then $max\{m_{c_j}, n_{c_j}\} \leq \text{poly}_2(\mathsf{K}) + \mathsf{K}(\mathsf{K}^2\text{poly}_3(\mathsf{K})) + 1$.

Proof. Assume for contradiction that the execution of a minimal witness w reaches a surely-nonequivalent pair $c_j = \langle (p_{c_j}, m_{c_j}, s_{c_j}), (q_{c_j}, n_{c_j}, t_{c_j}) \rangle$, with $max\{m_{c_j}, n_{c_j}\}$ greater than $\text{poly}_2(\mathsf{K}) + \mathsf{K}(\mathsf{K}^2\text{poly}_3(\mathsf{K})) + 1$. Since c_j is the first surely-nonequivalent configuration pair during the execution of a minimal witness, we know that all the previous configuration pairs are unresolved configuration pairs and lie either in the initial space or belt space. The idea here is to find an α-β repetition that is a factor of the execution inside a belt that can be cut out to obtain a shorter witness (refer Figure 5.3).

Let $T = \tau_0 \cdots \tau_{L-1}$ be the sequence of transition pairs corresponding to a minimal witness from c_0. For all $0 < i \leq L$, let $c_i = \tau_0 \cdots \tau_{i-1}(c_0)$. We use $c_j = \langle \chi_{c_j}, \psi_{c_j} \rangle$, $j \leq L$, to denote the first surely-nonequivalent configuration pair during the execution of w, where $\chi_{c_j} = (p_{c_j}, m_{c_j}, s_{c_j})$ and $\psi_{c_j} = (q_{c_j}, n_{c_j}, t_{c_j})$. Let $T_1 = \tau_0 \cdots \tau_{j-1}$ and $w = v_1 v_2$, where $v_1 = word(T_{0\cdots j-1})$ and $v_2 = word(T_{j\cdots L-1})$. Since c_j is surely-nonequivalent either (1) $\chi_{c_j} \not\equiv_{\mathsf{K}} \psi_{c_j}$ or (2) $\text{dist}(\chi_{c_j}) \neq \text{dist}(\psi_{c_j})$.

Case-1: $\chi_{c_j} \not\equiv_{\mathsf{K}} \psi_{c_j}$.

Since $\chi_{c_j} \not\equiv_{\mathsf{K}} \psi_{c_j}$, we know that $|v_2| \leq \mathsf{K}$. Also, since $\max\{m_j, n_j\}$ is greater than $\mathrm{poly}_2(\mathsf{K}) + 2(\mathsf{K}^2\mathrm{poly}_3(\mathsf{K}) + 1)$, by applying Lemma 5.14 twice, we can find $i < i_1 < j_1 \leq i_2 < j_2 < j$ such that $R(c_i) = T_{i \cdots j}(c_i)$ is a run inside a belt and $R_1 = T_{i \cdots i_1} T_{j_1+1 \cdots j}$, $R_2 = T_{i \cdots i_2} T_{j_2+1 \cdots j}$, and $R_{12} = T_{i \cdots i_1} T_{j_1+1 \cdots i_2} T_{j_2+1 \cdots j}$ are such that $R_1(c_i)$, $R_2(c_i)$ and $R_{12}(c_i)$ are runs inside in the same belt as $R(c_i)$ as shown in Figure 5.3.

We can deduce that $T_1 = T_{0 \cdots i-1} R_1 T_{j+1 \cdots |T|}$, $T_2 = T_{0 \cdots i-1} R_2 T_{j+1 \cdots |T|}$ and $T_3 = T_{0 \cdots i-1} R_{12} T_{j+1 \cdots |T|}$ are such that the three runs $T_1(c_0)$, $T_2(c_0)$ and $T_{12}(c_0)$ are all valid runs that are shorter than $T(c)$ and end in configuration pairs having the same states. Let $\mathtt{aw}(T(c_0)) = (s, t)$, $\mathtt{aw}(T_1(c_0)) = (s_1, t_1)$, $\mathtt{aw}(T_2(c_0)) = (s_2, t_2)$, and $\mathtt{aw}(T_3(c_0)) = (s_3, t_3)$. We know from Theorem 5.3 that there exists an $i \in \{1, 2, 3\}$ such that $s_i \neq t_i$ and hence $T_i(c_0)$ contradicts the minimality of Π.

Case-2: If $\mathrm{dist}(\chi_{c_j}) \neq \mathrm{dist}(\psi_{c_j})$.

We show that if $\max\{m_{c_j}, n_{c_j}\} > \mathrm{poly}_2(\mathsf{K}) + \mathsf{K}(\mathsf{K}^2\mathrm{poly}_3(\mathsf{K})) + 1$, then we can cut out some sub-runs to get a shorter witness contradicting the minimality of w.

Without loss of generality assume $\mathrm{dist}(\chi_{c_j}) < \mathrm{dist}(\psi_{c_j})$ and $\max\{m_{c_j}, n_{c_j}\} > \mathrm{poly}_2(\mathsf{K}) + \mathsf{K}(\mathsf{K}^2\mathrm{poly}_3(\mathsf{K})) + 1$. Since c_j is the first surely-nonequivalent configuration pair in Π, for all $i \in [0, j-1]$, the configuration pairs c_i are either in the initial space or inside a belt. Let k be the smallest index such that for all $i \in [k, j-1]$, the configuration pairs c_i are inside a fixed belt. Since c_k is the first point inside a belt, $max(m_{c_k}, n_{c_k}) = \mathrm{poly}_2(\mathsf{K}) + 1$.

Consider the run $\tau_k \cdots \tau_j(c_k)$. By the pigeon-hole principle, there is a set of $\mathsf{K}+1$ configuration pairs that are α-β-related to each other. Let $d_0, d_1, \ldots, d_{\mathsf{K}}$ denote these configuration pairs such that $m_{d_i} < m_{d_j}$ for $i < j$. Since $\mathrm{dist}(\chi_{c_j}) < \infty$, we know that there exists a word u such that $\chi_{c_j} \xrightarrow{u} (p', m', s') \in \mathsf{notUWA}$ with $m < \mathsf{K}$ and $|u| = \mathrm{dist}(\chi_{c_j})$. Let $\chi_0, \chi_1, \ldots, \chi_{\mathsf{K}}$ denote the configurations, where counter values $m_{d_0}, m_{d_1}, \ldots m_{d_{\mathsf{K}}}$ are encountered for the first time during the run $\chi_{c_j} \xrightarrow{u} (p', m', s')$. By the pigeon-hole principle, there exists $r < l$, such that χ_l and χ_r have the same state. Let $u = u_1 u_2 u_3$ for some $u_1, u_2, u_3 \in \Sigma^*$ such that $\chi_{c_j} \xrightarrow{u_1} \chi_l \xrightarrow{u_2} \chi r \xrightarrow{u_3} (p', m', s')$ and t denote the difference in counter values between χ_l and χr. Since χ_r and χ_l have the same state, for any $s \in \mathcal{F}$, the run

$(p_{c_j}, m_{c_j} - t, s) \xrightarrow{u_1 u_3} (p, m, s'')$ is a valid run for some $s'' \in \mathcal{F}$. Therefore, for all $s \in \mathcal{F}$, $\text{dist}(p_{c_j}, m_{c_j} - t, s) < \infty$.

Note that $t = m_{d_l} - m_{d_r}$, is the same as the difference in counter values between χ_l and χr. Since d_l and d_r are α-β-related configuration pairs, removing the sub-run between them from the run $\tau_k \cdots \tau_j(c_k)$ will take us to the background space point $\langle \chi', \psi' \rangle$, where $\chi' = (p_{c_j}, m_{c_j} - t, s)$, for some $s \in \mathcal{F}$. Since $\langle \chi', \psi' \rangle$ is a point in the background space either $\text{dist}(\chi') = \text{dist}(\psi') = \infty$ or $\text{dist}(\chi') \neq \text{dist}(\psi')$. We already know that for all $s \in \mathcal{F}$, $\text{dist}(p_{c_j}, m_{c_j} - t, s) < \infty$, therefore $\text{dist}(\chi') \neq \text{dist}(\psi')$. Hence, $c_{j'} = \langle \chi', \psi' \rangle$ is a surely-nonequivalent configuration pair with $max\{m_{c_{j'}}, n_{c_{j'}}\} \leq \text{poly}_2(\mathsf{K}) + \mathsf{K}(\mathsf{K}^2\text{poly}_3(\mathsf{K})) + 1$ contradicting our initial assumption. $\square$

Note that the technique used for proving *Case-2* is different from the one used for deterministic real-time one-counter automata by Böhm and Göller (2011). Using a similar technique as mentioned above will help get a better polynomial bound on the maximum counter value encountered during the run of the minimal witness while checking the equivalence of two deterministic real-time one-counter automata.

We finally conclude the proof of our main lemma.

Proof of Lemma 5.7. Let $\Pi = c_0\sigma_0 c_1\sigma_1 \ldots c_L$ is the run of a minimal witness. From Lemma 5.11, we know that there is no surely-equivalent configuration pair in it. Let us assume that $c_j = \langle \chi_{c_j}, \psi_{c_j} \rangle$ is the first surely-nonequivalent configuration pair in Π. Lemma 5.12 ensures that $L - j \leq \min(\text{dist}(\chi_{c_j}), \text{dist}(\psi_{c_j})) + \mathsf{K}$, therefore all counter values appearing at position greater than j are bounded by $\max(n_{\chi_{c_j}}, n_{\psi_{c_j}}) + \min(\text{dist}(\chi_{c_j}, \text{dist}(\psi_{c_j}))) + \mathsf{K}$.

Lemma 5.16 ensures that all counter values appearing before c_j inside belts are bounded by the value $\text{poly}_2(\mathsf{K}) + 2((\mathsf{K}^2\text{poly}_3(\mathsf{K}))^2 + 1)$, and by Lemma 5.17, $\max(n_{\chi_{c_j}}, n_{\psi_{c_j}}) \leq \text{poly}_2(\mathsf{K}) + \mathsf{K}(\mathsf{K}^2\text{poly}_3(\mathsf{K})) + 1$.

Finally, Lemma 7.12 ensures that $\text{dist}(\chi_{c_j})$ and $\text{dist}(\psi_{c_j})$ cannot be more than $(max\{\mathsf{K}, n_{\chi_{c_j}}, m_{\psi_{c_j}}\} + \mathsf{K}^2)\mathsf{K}$. Therefore, we conclude that there exists a polynomial $\text{poly}_0 = \text{poly}_2(\mathsf{K}) + 2(\mathsf{K}^2\text{poly}_3(\mathsf{K}))^2 + 1)$ such that all counter values appearing in Π are bounded by $\text{poly}_0(\mathsf{K})$. $\square$

From Lemma 5.7, we get that the value of poly_0 is $\mathcal{O}(\mathsf{K}^{12})$ and from Lemma 5.8, we get that the value of poly_1 is $\mathcal{O}(\mathsf{K}^{26})$. Hence, if two DWROCAs $\mathcal{A}$ and $\mathcal{B}$ are not equivalent, then there is a word whose length is of $\mathcal{O}(\mathsf{K}^{26})$ such that $f_{\mathcal{A}}(w) \neq f_{\mathcal{B}}(w)$.

5.4 Conclusion

In this chapter, we presented a polynomial time algorithm for checking the equivalence of two DWROCAs that return a word distinguishing them, if it exists. A potential research direction is to remove the "real-time" constraint in DWROCAs. Note that equivalence for deterministic one-counter automata is NL-complete regardless of whether it is real-time. However, the techniques for the non-real-time case differ from those used here. Therefore, it's not guaranteed that those techniques are robust enough to the addition of weights, which makes it an interesting topic for further investigation. Finally, a polynomial time algorithm for equivalence of DWROCAs also paves the way for efficient learning algorithms for DWROCAs.

CHAPTER 6

Real-Time One-Deterministic Counter Automata

We introduce real-time one-deterministic-counter automaton (RODCA) in this chapter. We also show that the equivalence problem for nondeterministic RODCAs is in PSPACE, whereas it is undecidable for nondeterministic OCAs. These are one-counter automaton (OCA) with the property of counter-determinacy, meaning that all paths labelled by a given word starting from the initial configuration have the same counter-effect. RODCAs are a strict extension of VOCAs, which are OCAs where the input alphabet determines the actions on the counter. We also show that the equivalence problem for nondeterministic RODCAs is in PSPACE, whereas it is undecidable for nondeterministic OCAs.

Contents

6.1 Introduction

This chapter investigates a relaxation in the visibly constraint on one-counter automata (OCA): the counter actions are no longer input-driven but are deterministic. We say that a OCA has *counter-determinacy* (see Definition 6.3) if "all paths labelled by a given word, starting from the initial configuration, have the same counter-effect".

We also define a model called real-time one-deterministic-counter automata (RODCA) (see Definition 6.5 for a formal definition). It consists of,

1. Counter: A counter that stays non-negative and allows zero tests.
2. Counter structure: A deterministic finite state machine where the transitions depend only on its current state, the input letter, and whether the counter is zero. The counter structure can increment/decrement the counter by one or leave it unchanged.
3. Finite state machine: A machine whose transitions can be deterministic, nondeterministic or weighted. The transitions depend on its current state, the input letter, and whether the counter value is zero. This machine cannot modify the counter.

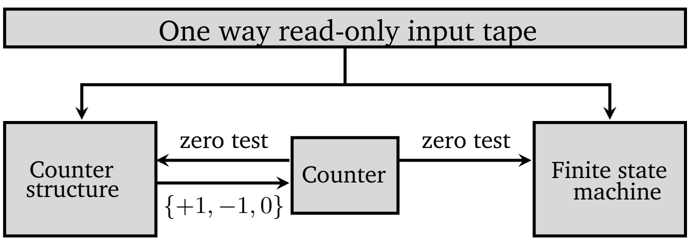


Fig. 6.1. Real-time one-deterministic-counter automata.

If the finite state machine is deterministic (resp. nondeterministic, or weighted), the RODCA will be called deterministic (resp. nondeterministic, or weighted).

In a weighted RODCA, we assume the weights are from a field. The class of DROCAs and the class of VOCAs are special classes of RODCAs. The counter structure and the finite state machine read simultaneously from the input tape.

6.1.1 Related Work

Visibly pushdown automata (VPDA) were introduced by Alur and Madhusudan (2004). They have received much attention as they are a strict subclass of pushdown automata suitable for program analysis. VPDAs enjoy tractable decidable properties, which are undecidable in the general case. The visibly restriction, in essence, is that the stack operations are *input-driven*, i.e., only depend on the letter read. Weighted VPDA is a natural extension to the weighted setting. counter-determinacy can be seen as a relaxation in the visibly constraint on OCAs, as the counter actions are no longer input-driven but are deterministic. The fact that weighted RODCAs is strictly more expressive than weighted VOCAs can be noted from the fact that the functions in Example 6.7 are not recognised by a weighted VOCA. Nowotka and Srba (2007) introduced height-deterministic pushdown automata, where the input string determines the stack height. Weighted RODCAs can be seen as weighted height-deterministic pushdown automata over a single stack alphabet and a bottom-of-stack symbol.

6.2 Preliminaries

6.2.1 Weighted Automata

Definition 6.1 (Weighted Automata)

A weighted automaton (WA) over a semiring $\mathcal{S} = (S, \oplus, \circ, 0_e, 1_e)$ is a tuple $\mathcal{D} = (Q, \Sigma, \boldsymbol{\lambda}, \delta, \boldsymbol{\eta})$

- Q is the finite set of states,
- Σ is the input alphabet,
- $\boldsymbol{\lambda} \in \mathcal{F}^{|Q|}$ is the initial distribution that assigns an initial weight to each state,
- $\delta : \Sigma \to \mathcal{F}^{|Q| \times |Q|}$ is the transition function,and
- $\boldsymbol{\eta} \in \mathcal{F}^{|Q|}$ is the final distribution that assigns a final weight to each state.

For all $a \in \Sigma$, $\delta(a)$ is a $|Q| \times |Q|$ matrix. For $a \in \Sigma$ and $i, j \in [0, |Q| - 1]$, $\delta(a)[i][j] = x$, if and only if there is a transition from state q_i to state q_j on a with weight x. We can extend the function δ to Σ^* in the standard way. $\delta(\varepsilon) = 1$, $\delta(wa) = \delta(w).\delta(a)$ and $\delta(wb) = \delta(w).\delta(b)$, for $w \in \Sigma^*$ and $a, b \in \Sigma$. Here, '.' denotes matrix multiplication. The weight with which the word w is accepted by $\mathcal{D}$ (denoted by $f_{\mathcal{D}}(w)$) is given by $f_{\mathcal{D}}(w) = \boldsymbol{\lambda}.\delta(w).\boldsymbol{\eta}$.

An *uninitialised* WA over a semiring $\mathcal{F} = (S, \oplus, \circ, 0_e, 1_e)$ is a tuple $\mathcal{C} = (Q, \Sigma, \delta, \boldsymbol{\eta})$, where Q is a finite, nonempty set of states, Σ is the input alphabet, $\delta : \Sigma \to \mathcal{F}^{|Q| \times |Q|}$ is the transition function and $\boldsymbol{\eta} \in \mathcal{F}^{|Q|}$ is a function that assigns an output weight to each state. Given an unitialised WA $\mathcal{C} = (Q, \Sigma, \delta, \boldsymbol{\eta})$, and a weight distribution over the states $\boldsymbol{\lambda} \in \mathcal{F}^{|Q|}$, we can get a WA$\mathcal{A} = (Q, \Sigma, \boldsymbol{\lambda}, \delta, \eta_F)$. A configuration of $\mathcal{C}$ is a vector $\mathbf{x} \in \mathcal{F}^{|Q|}$. The set of all configurations of $\mathcal{C}$ is the set $\{\mathbf{x} \mid \mathbf{x} \in \mathcal{F}^{|Q|}\}$.

In the next section, we will define weighted one-counter automata and the notion of counter-determinacy.

6.3 Weighted One-Counter Automata

Definition 6.2 (Weighted OCA)

A *weighted one-counter automaton* $\mathcal{A} = (Q, \Sigma, \boldsymbol{\lambda}, \delta_0, \delta_1, \boldsymbol{\eta})$ is defined over a semiring $\mathcal{S} = (S, \oplus, \circ, 0_e, 1_e)$, where, Q is a non-empty finite set of states, Σ is the finite input alphabet, $\boldsymbol{\lambda} \in \mathcal{S}^{|Q|}$ is the initial distribution that assigns an initial weight to each state, $\delta_0 : Q \times \Sigma \times Q \times \{0, +1\} \to \mathcal{S}$ and $\delta_1 : Q \times \Sigma \times Q \times \{-1, 0, +1\} \to \mathcal{S}$ are the functions that assigns weights to transitions, and $\boldsymbol{\eta} \in \mathcal{S}^{|Q|}$ is the final distribution that assigns an output weight to each state. For $i \in [0, |Q| - 1]$, $\boldsymbol{\lambda}[i]$ indicates the initial weight on state $q_i \in Q$ and $\boldsymbol{\eta}[i]$ indicates the output weight on state $q_i \in Q$.

Let $p, q \in Q, a \in \Sigma, n \in \mathbb{N}, e \in \{-1, 0, +1\}$, and $s \in \mathcal{S}$. We write $(q, n) \hookrightarrow^{a|s} (p, n + e)$ if $\delta_{sign(n)}(q, a, p, e) = s$. Let $w = a_1 a_2 \cdots a_t \in \Sigma^*$ for some $t \in \mathbb{N}$. For a $q_0 \in Q$ and $n_0 \in \mathbb{N}$, we write $(q_0, n_0) \hookrightarrow^{w|s} (q_t, n_t)$ if for all $i \in [1, t]$, there are $q_i \in Q, n_i \in \mathbb{N}$, $s_i \in \mathcal{S}$ such that $(q_{i-1}, n_{i-1}) \hookrightarrow^{a_i|s_i} (q_i, n_i)$ and $s = s_1 \circ s_2 \cdots \circ s_t$. We call $(q_0, n_0) \hookrightarrow^{w|s} (q_t, n_t)$ a *run* of w from the configuration (q_0, n_0). We use $(q_0, n_0) \hookrightarrow^{w} (q_t, n_t)$ to denote the run when we are not concerned about the weights. Given a state $q \in Q$, we use $\boldsymbol{\eta}_q$ to denote the output weight on state q. The accepting weight of the word w along the run $(q_0, n_0) \hookrightarrow^{w|s} (q_t, n_t)$ is $s \circ \boldsymbol{\eta}_{q_t}$. Since the machine is non-deterministic, given a configuration c and a word $w \in \Sigma^*$, there can be multiple runs on w from c. The weight with which a word w is accepted by $\mathcal{A}$ (denoted as $f_{\mathcal{A}}(w)$) is the sum of the accepting weights of w along all its runs in $\mathcal{A}$.

Definition 6.3 (Counter-determinacy)

A weighted OCA with *counter-determinacy* is a weighted one-counter automaton $\mathcal{A} = (Q, \Sigma, \boldsymbol{\lambda}, \delta_0, \delta_1, \boldsymbol{\eta})$ with the following restriction: if $\boldsymbol{\lambda}[i]$ and $\boldsymbol{\lambda}[j]$ are non-zero for some $i, j \in [1, |Q|]$, then for all $w \in \Sigma^*$, if $(q_i, 0) \hookrightarrow^{w|s_1} (p_1, n_1)$ and $(q_j, 0) \hookrightarrow^{w|s_2} (p_2, n_2)$ for some $p_1, p_2 \in Q, n_1, n_2 \in \mathbb{N}$ and $s_1, s_2 \in \mathcal{S}$, then $n_1 = n_2$.

We say that a weighted OCA is *counter-deterministic* if it has the property of counter-determinacy.

Theorem 6.4
Given a weighted OCA $\mathcal{A}$, deciding $\mathcal{A}$ is not counter-deterministic is in NL.

Proof. Let $\mathcal{A} = (Q, \Sigma, \boldsymbol{\lambda}, \delta_0, \delta_1, \boldsymbol{\eta})$ be a weighted OCA. Assume that $\mathcal{A}$ is not counter-deterministic. Consider a minimal length word $z = wa$ for some $w \in \Sigma^*$ and $a \in \Sigma$ such that, there exist $i, j \in [1, |Q|]$ $p_1, p_2 \in Q, n_1, n_2 \in \mathbb{N}$ with $\boldsymbol{\lambda}[i]$, $\boldsymbol{\lambda}[j] \neq 0_e$, $(q_i, 0) \hookrightarrow^w (p_1, n_1) \hookrightarrow^a (p_1', n_1')$, $(q_j, 0) \hookrightarrow^w (p_2, n_2) \hookrightarrow^a (p_2', n_2')$ and $n_1' \neq n_2'$. Consider the runs $(q_i, 0) \hookrightarrow^z (p_1, n_1)$, $(q_j, 0) \hookrightarrow^z (p_2, n_2)$. The counter value reached on reading any prefix of w from $(q_i, 0)$ and $(q_j, 0)$ in these runs are the same. Therefore, the maximum counter value encountered during these runs is less than or equal to $|Q|^4$. The proof is the same as Claim 1 in Theorem 3.5.

Now, we prove that the length of w is at most $|Q|^6$. Assume for contradiction that $|w| > |Q|^6$. Since the maximum counter value encounterd during the runs $(q_i, 0) \hookrightarrow^w (p_1, n_1)$ and $(q_j, 0) \hookrightarrow^w (p_2, n_2)$ is at most $|Q|^4$, there are at most $|Q|^5$ distinct configurations in both these runs. Note that for all prefixes of w, the counter values of configuration reached on both these runs are the same. Therefore, if $|w| > |Q|^6$, then by the pigeon-hole principle, there exist configurations c_1 and c_2 , and words w_1, w_2, w_3 such that $w = w_1 w_2 w_3$, $(q_i, 0) \hookrightarrow^{w_1} c_1 \hookrightarrow^{w_2} c_1 \hookrightarrow w_3(p_1, n_1)$ and $(q_j, 0) \hookrightarrow^{w_1} c_2 \hookrightarrow w_2 c_2 \hookrightarrow w_3(p_2, n_2)$. Therefore, the word $z' = w_1 w_2 a$ is a shorter word than z such that $(q_i, 0) \hookrightarrow^{z'} (p_1', n_1')$ and $(q_j, 0) \hookrightarrow^w (p_2', n_2')$. This contradicts the minimality of z.

A nondeterministic machine can guess z and the runs $(q_i, 0) \hookrightarrow^z (p_1', n_1')$ and $(q_j, 0) \hookrightarrow^z (p_2', n_2')$ on the fly to show that $n_1' \neq n_2'$ using logarithmic space. $\square$

6.4 Weighted Real-Time One-Deterministic-Counter Automata

In this section, we define weighted RODCAs, where the weights are from a semiring $\mathcal{S}$ (possibly infinite). If the weights are from the boolean semiring then we call it deterministic/nondeterministic RODCAs based on how the transition functions are defined. We will now present a definition for weighted RODCAs.

Definition 6.5 (Weighted RODCA)

A *weighted* RODCA, $\mathcal{A}$ over a semiring $\mathcal{S}$ is a tuple $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$ where,

- The tuple $(C, \delta_0, \delta_1, p_0)$ is called the *counter structure* and $(Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta})$ the *finite state machine*,
- C is a non-empty finite set of *counter states*,
- $\delta_0 : C \times \Sigma \to C \times \{0, +1\}$, $\delta_1 : C \times \Sigma \to C \times \{-1, 0, +1\}$ are *counter transitions*,
- $p_0 \in C$ is the initial state for the counter structure,
- Q is a non-empty finite set of states of the finite state machine,
- $\boldsymbol{\lambda} \in \mathcal{S}^{|Q|}$ is the *initial distribution* where the i^{th} component of $\boldsymbol{\lambda}$ indicates the initial weight on state $q_i \in Q$,
- $\Delta : \Sigma \times \{0, 1\} \to \mathcal{S}^{|Q| \times |Q|}$ gives the *transition matrix*. For all $a \in \Sigma$ and $d \in \{0, 1\}$, the component in the i^{th} row and j^{th} column of $\Delta(a, d)$ denote the weight on the transition from state $q_i \in Q$ to state $q_j \in Q$ on reading the symbol a from counter value n with $sign(n) = d$,
- $\boldsymbol{\eta} \in \mathcal{S}^{|Q|}$ is the *final distribution*, where the i^{th} component of $\boldsymbol{\eta}$ indicates the output weight on state $q_i \in Q$, and
- Σ is the input alphabet.

Note that δ_0 and δ_1 are deterministic transition functions. The function Δ is defined from $\Sigma \times \{0, 1\} \to \mathcal{S}^{|Q| \times |Q|}$. For an $a \in \Sigma$, $\Delta(a, 0)$ will give the transition matrix on reading a from a configuration with counter value zero and $\Delta(a, 1)$ will give the transition matrix on reading a from a configuration with positive counter value. The counter structure and the finite state machine run synchronously on any given word. A *configuration* $\mathtt{c}$ of a weighted RODCA is of the form $(\mathbf{x}_{\mathtt{c}}, p_{\mathtt{c}}, n_{\mathtt{c}}) \in \mathcal{S}^{|Q|} \times C \times \mathbb{N}$. We use the notation $\textsc{WeightVector}(\mathtt{c})$ to denote $\mathbf{x}_{\mathtt{c}}$, $\textsc{CounterState}(\mathtt{c})$ to denote $p_{\mathtt{c}}$, and $\textsc{CounterValue}(\mathtt{c})$ to denote $n_{\mathtt{c}}$. The initial configuration is $(\boldsymbol{\lambda}, p_0, 0)$. A transition is a tuple $\tau = (\iota, d, a, \texttt{ce}, \mathbb{A}, \theta)$ where $\iota, \theta \in C$ are counter states, $d \in \{0, 1\}$ denotes the sign of the counter value, $a \in \Sigma$, $\texttt{ce} \in \{-1, 0, 1\}$ is the *counter-effect*, $\mathbb{A} \in \mathcal{S}^{|Q|\times|Q|}$ such that $\Delta(a, d) = \mathbb{A}$, and $\delta_d(\iota, a) = (\theta, \texttt{ce})$. Given a transition $\tau = (\iota, d, a, \texttt{ce}, \mathbb{A}, \theta)$ and a configuration $\texttt{c} = (\mathbf{x}, n, p)$, we denote the application of τ to $\texttt{c}$ as $\tau(\texttt{c}) = (\mathbf{x}\mathbb{A}, \theta, n + \texttt{ce})$ if $p = \iota$ and $d = sign(n)$; $\tau(\texttt{c})$ is undefined otherwise. Note that for a transition τ and configuration (q, n, t) with $n \geq 0$, if $\tau((q, n, t)) = (q', n', t')$, then $n' \geq 0$.

Consider a sequence of transitions $T = \tau_0 \cdots \tau_\ell$ where $\tau_i = (\iota_i, d_i, a_i, \texttt{ce}_i, \mathbb{A}_i, \theta_i)$ for all $i \in [0, \ell]$. We denote by $\texttt{word}(T) = a_0 \cdots a_\ell$ the word labelling it, $\texttt{we}(T) = \mathbb{A}_0 \cdots \mathbb{A}_\ell$ its weight-effect matrix, and $\texttt{ce}(T) = \texttt{ce}_0 + \cdots + \texttt{ce}_\ell$ its counter-effect. For all $0 \leq i < j \leq \ell$, we use $T_{i\cdots j}$ to denote the sequence of transitions $\tau_i \cdots \tau_j$ and $|T|$ to denote its length $\ell + 1$. We call T *floating* if for all $i \in [0, \ell]$, $d_i = 1$ and *non-floating* otherwise.

A *run* π is an alternating sequence of configurations and transitions denoted as $\pi = \texttt{c}_0\tau_0\texttt{c}_1 \cdots \tau_{\ell-1}\texttt{c}_\ell$ such that for every i, $\texttt{c}_{i+1} = \tau_i(\texttt{c}_i)$. The word labelling, length, weight-effect, and counter-effect of the run are those of its underlying sequence of transitions. Given a sequence of transitions $T = \tau_0 \cdots \tau_{\ell-1}$ and a configuration $\texttt{c}$, we denote by $T(\texttt{c})$ the run (if it is defined) $\texttt{c}_0\tau_0\texttt{c}_1 \cdots \tau_{\ell-1}\texttt{c}_\ell$ where $\texttt{c}_0 = \texttt{c}$.

For any word w, there is exactly one run labelled by w starting from a given configuration $\texttt{c}_0$. We denote this run $\pi(w, \texttt{c}_0)$. A run $\pi(w, \texttt{c}_0) = \texttt{c}_0\tau_0\texttt{c}_1 \cdots \tau_{\ell-1}\texttt{c}_\ell$ is also represented as $\texttt{c}_0 \xrightarrow{w} \texttt{c}_\ell$. We write $\texttt{c}_0 \to^* \texttt{c}_\ell$ if there is some word w such that $\texttt{c}_0 \xrightarrow{w} \texttt{c}_\ell$. For a weighted RODCA $\mathcal{A}$, the accepting weight of w is denoted by $f_{\mathcal{A}}(w, \texttt{c}) = \boldsymbol{\lambda}\texttt{we}(\pi(w, \texttt{c}))\boldsymbol{\eta}^\top$, where $\texttt{c}$ is the initial configuration of $\mathcal{A}$. Alternatively, we use $f_{\mathcal{A}}(w)$ to denote $f_{\mathcal{A}}(w, \texttt{c})$. Two weighted RODCAs $\mathcal{A}$ and $\mathcal{B}$ are *equivalent* if for all $w \in \Sigma^*$, $f_{\mathcal{A}}(w, \texttt{c}) = f_{\mathcal{B}}(w, \texttt{d})$ where $\texttt{c}$ and $\texttt{d}$ are the initial configurations of $\mathcal{A}$ and $\mathcal{B}$ respectively. Let $\texttt{c}$ and $\texttt{d}$ be configurations of weighted RODCAs $\mathcal{A}$ and $\mathcal{B}$ respectively. We write $\texttt{c}\equiv_l\texttt{d}$ if for all $w \in \Sigma^{\leq l}$, $f_{\mathcal{A}}(w, \texttt{c}) = f_{\mathcal{B}}(w, \texttt{d})$ otherwise $\texttt{c}\not\equiv_l\texttt{d}$. We write $\texttt{c}\equiv\texttt{d}$ if if for all $l \in \mathbb{N}$, $\texttt{c}\equiv_l\texttt{d}$. Otherwise, we write $\texttt{c}\not\equiv\texttt{d}$.

Consider the weighted RODCA $\mathcal{C}$ recognising the function `prefixAwareDecimal` given in Figure 6.3. Here, $\boldsymbol{\lambda} = [1, 0, 0, 0]$ and $\boldsymbol{\eta} = [0, 0, 0, 1]$. The configuration $\texttt{c}_0 = ([1, 0, 0, 0], p_0, 0)$ is the initial configuration of this automaton. Let

$w = abaaab$. The run of this machine on the word w can be written as:

$$\begin{aligned}\pi(w, \mathsf{c}_0) = &([1,0,0,0], p_0, 0) \xrightarrow{a} ([1,0,0,0], p_0, 1) \xrightarrow{b} ([0,1,0,0], p_1, 0) \xrightarrow{a} \\ &([0,0,1,0], p_2, 0) \xrightarrow{a} ([0,0,1,1], p_2, 1) \xrightarrow{a} ([0,0,1,2], p_2, 2) \xrightarrow{b} \\ &([0,0,1,6], p_2, 1).\end{aligned}$$

The counter-effect of this run is $\mathtt{ce}(\pi(w, \mathsf{c}_0)) = 1$ and the weight-effect matrix is given by

$$\mathtt{we}(\pi(w, \mathsf{c}_0)) = \Delta(a,0)\ \Delta(b,1)\ \Delta(a,0)\ \Delta(a,1)\ \Delta(a,1)\ \Delta(b,1) = \begin{bmatrix} 0 & 0 & 1 & 6 \\ 0 & 0 & 1 & 14 \\ 0 & 0 & 1 & 46 \\ 0 & 0 & 0 & 64 \end{bmatrix}.$$

The accepting weight of the word w is $f_{\mathcal{C}}(w, \mathsf{c}_0) = \boldsymbol{\lambda}\mathtt{we}(\pi(w, \mathsf{c}_0))\boldsymbol{\eta}^\top = 6$.

An *uninitialised weighted* RODCA $\mathcal{D}$ over a semiring $\mathcal{S}$ is a tuple $\mathcal{D} = ((C, \delta_0, \delta_1), (Q, \Delta, \boldsymbol{\eta}), \Sigma)$, where C is a non-empty set of states, $\delta_0 : C \times \Sigma \to C \times \{0, +1\}$, $\delta_1 : C \times \Sigma \to C \times \{-1, 0, +1\}$ are transition functions, Q is a non-empty finite set of states, $\Delta : \Sigma \times \{0,1\} \to \mathcal{S}^{|Q| \times |Q|}$ is a function, $\boldsymbol{\eta} \in \mathcal{S}^{|Q|}$, and Σ is the input alphabet. The set of all configurations $\mathcal{D}$ is the set $\{(\mathbf{x}_c, p_c, n_c) \mid \mathbf{x}_c \in \mathcal{F}^{|Q|}, p_c \in C, n_c \in \mathbb{N}\}$. Given an uninitialised weighted RODCA $\mathcal{C}$ and a configuration $\mathsf{c}_0 = (\mathbf{x}, p, 0)$ of $\mathcal{D}$, we can get a weighted RODCA $\mathcal{A}\langle \mathsf{c}_0 \rangle = ((C, \delta_0, \delta_1, p), (Q, \mathbf{x}, \Delta, \boldsymbol{\eta}), \Sigma)$.

6.4.1 Examples: RODCAs

Example 6.6 (Deterministic/Nondeterministic RODCAs)

The following languages are defined over the alphabet $\Sigma = \{a, b\}$ and are recognised by an OCA with counter-determinacy.

(a) The language `MirrorA` $= \{a^n b a^n \mid n > 0\}$.

(b) The language `MoreA` $= \{w \in (a+b)^* \mid$ number of a's is greater than or equal to number of b's$\}$.

(c) The language `LeadBC` $= \{a^n (b+c)^m b (b+c)^2 \mid m, n \in \mathbb{N}$ and $m > n\}$.

The deterministic RODCAs recognising the languages `MirrorA` and `MoreA` and the nondeterministic RODCA recognising the language `LeadBC` are given in Figure 6.2.

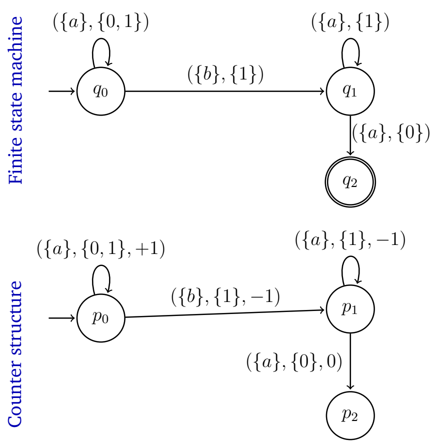
Example 6.7 (Weighted RODCAs)

The following functions are defined over the alphabet $\Sigma = \{a, b\}$. The transition weights of these weighted RODCAs are from the field of rational numbers $\mathbb{Q}$.

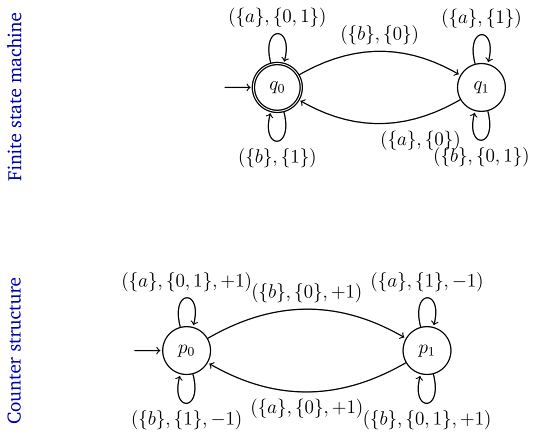
(a) The function `prefixAwareDecimal`: $\Sigma^* \to \mathbb{N}$ is defined as follows: `prefixAwareDecimal`(w) =`decimal`(w_2) if $w = w_1 w_2$, $w_1 \in \{a^n b a^n \mid n > 0\}$, and the number of a's $\geq$ number of b's for any prefix of w_2, and 0 otherwise. Here, `decimal`(w_2) represents the decimal equivalent of w_2 when interpreted as a binary number, where 'a' is treated as a one and 'b' as a zero.

(b) The function `equalPrefixPower`: $\Sigma^* \to \mathbb{N}$ is defined as follows: for all $w \in \Sigma^*$, `equalPrefixPower`$(w) = 2^k$ where k is the number of proper prefixes of w with equal number of a's and b's.

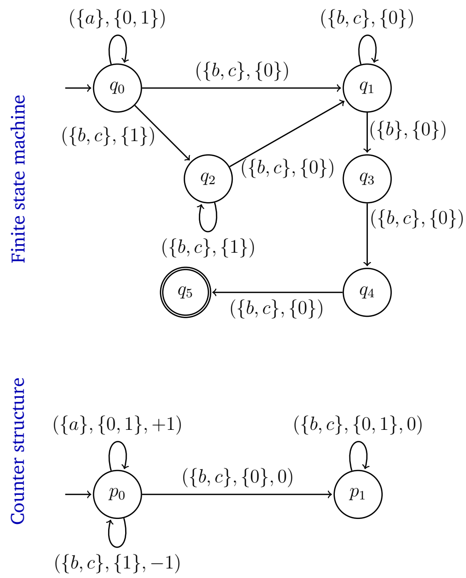
The weighted RODCAs recognising these functions are given in Figure 6.3 and Figure 6.4.



(a) $\texttt{MirrorA} = \{a^n b a^n \mid n > 0\}$.



(b) $\texttt{MoreA} = \{w \in (a+b)^* \mid$ #a's $\geq$ #b's$\}$.



(c) $\texttt{LeadBC} = \{a^n(b+c)^m b(b+c)^2 \mid m,n \in \mathbb{N}$ and $m > n\}$.

Fig. 6.2. The figure gives RODCAs corresponding to examples (a), (b) and (c) given in Example 6.6. Let $A \subseteq \Sigma, R \subseteq \{0,1\}$ are non-empty sets and $d \in \{-1,0,+1\}$. For $i,j \in \mathbb{N}$, if a transition from q_i to q_j is labelled (A,R) and $(a,r) \in A \times R$, then there is a transition from q_i to q_j on reading the symbol a. The current counter value should be 0 if $r=0$ and greater than 0 if $r=1$. Similarly, if a transition from p_i to p_j is labelled (A,R,d), then for $a \in A$ and $r \in R$, there is a transition from p_i to p_j on reading the symbol a that adds d to the current counter value. The current counter value should be 0 if $r=0$ and greater than 0 if $r=1$.

None of these languages are recognised by a visibly pushdown automata.

The functions `prefixAwareDecimal` and `equalPrefixPower`, in Example 6.7, are recognised by weighted OCA with counter-determinacy. Like weighted finite automata, weighted RODCAs recognise functions - every word over a finite alphabet is mapped to a weight. The finite state machine computes the weight associated with the word. Let $A \subseteq \Sigma, R \subseteq \{0, 1\}$ are non-empty sets, $d \in \{-1, 0, +1\}$ and $s \in \mathbb{Q}$. In Figure 6.3 and Figure 6.4, if a transition from p_i to p_j of the counter structure is labelled (A, R, d), then for $a \subseteq A$ and $r \subseteq R$, there is a transition from p_i to p_j on reading the symbol a with counter action d. The current counter value should be 0 if $r = 0$ and greater than 0 if $r = 1$. Similarly, if a transition from q_i to q_j of the finite state machine is labelled (A, R, s) then, for $a \in A$ and $r \in R$ there is a transition from q_i to q_j on reading the symbol a with weight s. In both cases, the current counter value should be 0 if $r = 0$ and greater than 0 if $r = 1$. For the finite state machine, the initial (resp. output) weight is marked using an inward (resp. outward) arrow. The weight of a path is the product of transition weights along that path. The accepting weight of a word is the sum of weights of all the paths from an initial state to an output state labelled by that word.

The reader might feel that a weighted RODCA is equivalent to a cartesian product of a DROCA and a weighted automaton. However, one can note that the functions `prefixAwareDecimal` and `equalPrefixPower` in Example 6.7 are not definable by the cartesian product of DROCA and a weighted automaton. The reason is that the weighted automaton cannot “see” the counter values, so its power is restricted.

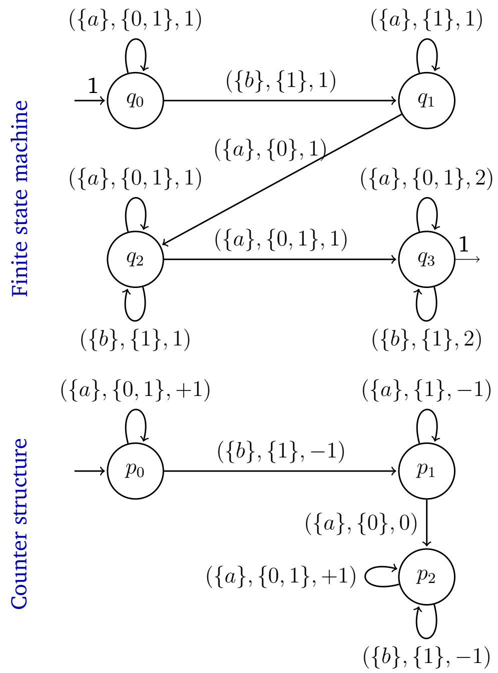


Fig. 6.3. A weighted RODCA recognising the function `prefixAwareDecimal`.

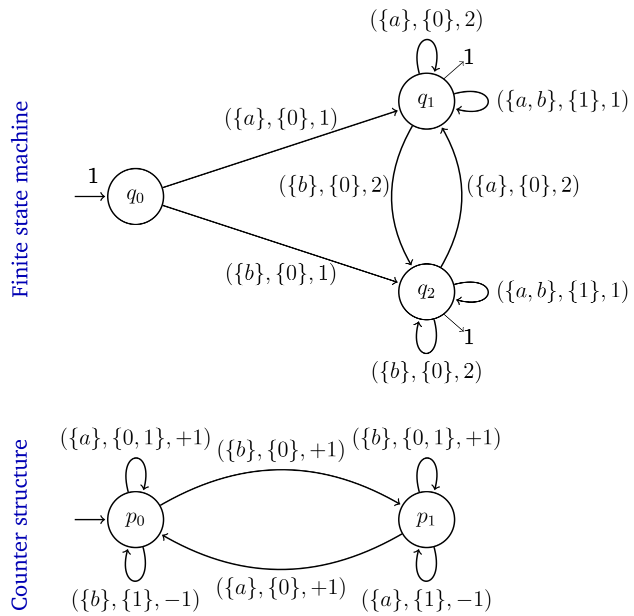


Fig. 6.4. A weighted RODCA recognising the function `equalPrefixPower`.

Given a weighted RODCA $\mathcal{A}$ over the alphabet Σ and a semiring $\mathcal{S}$, we define its M-unfolding weighted automaton $\mathcal{A}^M$ as a finite state weighted automaton that recognises the same function as $\mathcal{A}$ for all runs where the counter value does not exceed M. A formal definition is given in Definition 6.8. We will later reduce the equivalence problem of weighted RODCAs to the equivalence problem of their corresponding M-unfolding weighted automaton, where M is polynomially bounded by the size of the weighted RODCAs.

Definition 6.8 **(M-unfolding weighted automaton)** Let $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$ be a weighted RODCA over the semiring $\mathcal{S}$. For a given $M \in \mathbb{N}$, we define an *M-unfolding weighted automaton* $\mathcal{A}^M$ of $\mathcal{A}$ as follows, $\mathcal{A}^M = (Q', \Sigma, \boldsymbol{\lambda}', \Delta', \boldsymbol{\eta}')$ where,

- $Q' = Q \times C \times [0, M]$ is the finite set of states.
- $\lambda' \in \mathcal{S}^{|Q'|}$ is the initial distribution.

$$\lambda'[i] = \begin{cases} \lambda[i], & \text{if } i < |Q| \\ 0, & \text{otherwise.} \end{cases}$$

- $\Delta' : \Sigma \to \mathcal{S}^{|Q'| \times |Q'|}$ gives the transition matrix.

  For $i, j \in |Q'|$ and $a \in \Sigma$,

$$\Delta'(a)[i][j] = \Delta(a, sign(r))[i \bmod |Q|][j \bmod |Q|]$$

  where $r = \frac{i}{|Q| \times |C|}$, if $\delta_{sign(r)}(p_{\frac{i \bmod (|Q| \times |C|)}{|Q|}}, a) = (p_{\frac{j \bmod (|Q| \times |C|)}{|Q|}}, d)$ and $\frac{j}{|Q| \times |C|} = \frac{i}{|Q| \times |C|} + d$. Otherwise, $\Delta'(a)[i][j]$ is equal to zero.

- $\boldsymbol{\eta}'_F \in \mathcal{S}^{|Q'|}$ is the final distribution.

$$\boldsymbol{\eta}'_F[i] = \boldsymbol{\eta}[i \bmod |Q|]$$

A deterministic/nondeterministic RODCA $\mathcal{A}$ is a weighted RODCA over the boolean semiring $\mathcal{S} = (\{0, 1\}, \vee, \wedge)$. The language recognised by $\mathcal{A}$ is given by $\mathcal{L}(\mathcal{A}) = \{w \mid f_{\mathcal{A}}(w) = 1\}$. We say an RODCA $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$

is a deterministic RODCA if for every sequence of transitions $T = \tau_0 \cdots \tau_{\ell-1}$, the vector $\boldsymbol{\lambda}\mathtt{we}(T)$ contains exactly one 1. We call $\mathcal{A}$ a *nondeterministic* RODCA otherwise. All the above-mentioned notions apply for both deterministic RODCAs and nondeterministic RODCAs.

We observe the following theorem.

Theorem 6.9

1. Given a weighted OCA with counter-determinacy $\mathcal{A}$, a weighted RODCA $\mathcal{A}'$ such that for all $w \in \Sigma^*$ $f_{\mathcal{A}}(w) = f_{\mathcal{A}'}(w)$ can be constructed in polynomial time with respect to $|\mathcal{A}|$.

2. Given a weighted RODCA $\mathcal{A}'$, a weighted OCA with counter-determinacy $\mathcal{A}$ such that for all $w \in \Sigma^*$ $f_{\mathcal{A}'}(w) = f_{\mathcal{A}}(w)$ can be constructed in polynomial time with respect to $|\mathcal{A}'|$.

Proof. First, we prove Point 1 of the Lemma. Let $\mathcal{A} = (Q, \boldsymbol{\lambda}, \delta_0, \delta_1, \boldsymbol{\eta})$ be a weighted OCA with counter-determinacy. For this purpose, we define a function $color : [1, |Q|] \to [1, |Q|]$ as follows: $color(i) = min\{j \mid \forall w \in \Sigma^*, n \in \mathbb{N}$, counter-effect of w from (q_i, n) and (q_j, n) are equal$\}$.

Given a weighted OCA with counter-determinacy with an initial configuration $(\boldsymbol{\lambda}, 0)$, we can find this coloring function in polynomial time. First, we look at the smallest $i \in [1, |Q|]$ such that $\boldsymbol{\lambda}[i] \neq 0$. For all $j \in [0, |Q|]$, where $\boldsymbol{\lambda}[j] \neq 0$, we write $color(j) = i$. We initialise an integer $depth = 1$ and look at the configurations reachable from $(\boldsymbol{\lambda}, 0)$ by reading words of length $depth$. Let $(\mathbf{x}, c)$ for some $c \in \mathbb{N}$ be such a configuration. If there exists a $j \in [1, |Q|]$ with $\mathbf{x}[j] \neq 0$ and $color(j) = i$ for some $i \in [1, |Q|]$, then for all $k \in [0, |Q|]$, where $\mathbf{x}[k] \neq 0$, we write $color(k) = i$. If for all $j \in [1, |Q|]$ with $\mathbf{x}[j] \neq 0$, $color(j)$ is not defined, then we look at the smallest $i \in [1, |Q|]$ such that $\mathbf{x}[i] \neq 0$. For all $j \in [0, |Q|]$, where $\mathbf{x}[j] \neq 0$, we write $color(j) = i$. We increment $depth$ by one and repeat this process until $color(i)$ is defined for all $i \in [1, |Q|]$. This terminates after polynomial steps as all reachable states from the initial configuration can be reached by reading a polynomial length word.

Let $(\mathbf{x}, n)$ be a configuration reachable from the initial configuration of a weighted OCA$\mathcal{A}$ with counter-determinacy.

Claim 1. *If* $\mathbf{x}[i] \neq 0$ *and* $\mathbf{x}[j] \neq 0$*, then* $color(i) = color(j)$*.*

Proof. This follows from the notion of counter-determinacy. If $\mathbf{x}[i] \neq 0$ and $\mathbf{x}[j] \neq 0$ and $color(i) \neq color(j)$, there there exists a $w \in \Sigma^*$ such that $(q_i, n) \hookrightarrow^{w|s_1} (p_1, n_1)$ and $(q_j, n) \hookrightarrow^{w|s_2} (p_2, n_2)$ for some $p_1, p_2 \in Q, s_1, s_2 \in \mathcal{S}$ and $n_1, n_2 \in \mathbb{N}$ such that $n_1 \neq n_2$. This contradicts the fact that the machine is counter-deterministic. $\square_{Claim:1}$

Therefore, for all $w \in \Sigma^*$, if $(q_i, n) \hookrightarrow^{w|s_1} (p_1, n_1)$ and $(q_j, n) \hookrightarrow^{w|s_2} (p_2, n_1)$ for some $q, p_1, p_2 \in Q, n, n_1 \in \mathbb{N}$ and $s_1, s_2 \in \mathcal{S}$, then $color(p_1) = color(p_2)$. The colors are analogous to the counter states in the syntactic definition. The transition from one color to another depends solely on the current input symbol and whether the current counter value is zero or non-zero. Hence, in a weighted RODCA, the counter transitions are determined by a deterministic one-counter automata where the states represent these colors and transitions represent the transition from one color to another. Now, we formally define the equivalent weighted RODCA $((C, \delta'_0, \delta'_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$ as follows:

- $C = \{j \mid color(i) = j, i \in [1, |Q|]\}$ is the set of counter states.
- $\delta'_0 : C \times \Sigma \times \{0\} \to C \times \{0, +1\}$ and $\delta'_1 : C \times \Sigma \times \{1\} \to C \times \{-1, 0, +1\}$ are the deterministic counter transitions. For all $q \in |Q|, a \in \Sigma$ we define $\delta'_1(q, a)$ and $\delta'_0(q, a)$ as:

  $$\delta'_1(q, a) = (p, d) \text{ if } (q, 1) \xhookrightarrow{a|s} (p, 1 + d) \text{ and } \delta'_0(q, a) = (p, d) \text{ if } (q, 0) \xhookrightarrow{a|s} (p, d)$$

  for some $p \in Q, s \in \mathcal{S}$, and $d \in \{-1, 0, +1\}$.
- Let $i \in [1, |Q|]$ such that $\lambda[i] \neq 0$. $p_0 = j$, is the start state for counter transition, where $j = color(i)$.
- $\Delta : \Sigma \times \{0, 1\} \to \mathcal{S}^{|Q| \times |Q|}$ gives the transition matrix for all $a \in \Sigma$ and $d \in \{0, 1\}$. For $i, j \in [1, |Q|]$, $\Delta(a, d)[i][j] = s$, if $\delta_d(q_i, a, q_j, e) = s$ for $q_i, q_j \in Q, s \in \mathcal{S}$ and $e \in \{-1, 0, +1\}$.

Hence, we can construct an equivalent weighted RODCA from a given weighted OCA with counter-determinacy in polynomial time. Proving Point 2 is straightforward. $\square$

6.5 Deterministic and Nondeterministic RODCAs

In this section, we consider deterministic RODCAs and nondeterministic RODCAs. These are weighted RODCAs over the boolean semiring. On top of equivalence and learning, we consider the following problems of deterministic/nondeterministic RODCAs.

- Regularity: given an RODCA $\mathcal{A}$, check whether the language recognised by $\mathcal{A}$ is regular.
- Covering: given two uninitialised RODCAs $\mathcal{A}$ and $\mathcal{B}$, we say $\mathcal{A}$ *covers* $\mathcal{B}$ if for all initial configurations $\mathtt{c}_0$ of $\mathcal{A}$ there exists an initial configuration $\mathtt{d}_0$ of $\mathcal{B}$ such that $\mathcal{A}\langle\mathtt{c}_0\rangle$ and $\mathcal{B}\langle\mathtt{d}_0\rangle$ are equivalent.
- Coverable equivalence: two RODCAs $\mathcal{A}$ and $\mathcal{B}$ are said to be coverable equivalent if $\mathcal{A}$ *covers* $\mathcal{B}$ and $\mathcal{B}$ *covers* $\mathcal{A}$.
- Reachability: given an RODCA $\mathcal{A}$, a state q of the finite state machine and a counter state p and a counter value n, the reachability problem asks whether there exists a word that takes you from the initial configuration of $\mathcal{A}$ to a configuration (q, p, n).
- Coverability: given an RODCA $\mathcal{A}$, a state q of the finite state machine and a counter state p, the coverability problem asks whether there exists a word that takes you from the initial configuration of $\mathcal{A}$ to a configuration (q, p, n) for some $n \in \mathbb{N}$.

By definition, deterministic RODCAs have at most one unique path for any fixed word. Therefore, they are deterministic real-time OCAs with counter-determinacy. It is also easy to observe that DROCAs are deterministic RODCAs. It follows that deterministic RODCAs and DROCAs are expressively equivalent. Hence, we get the following theorem.

Proposition 6.10

There is a polynomial time translation between deterministic RODCAs and DROCAs.

The equivalence and regularity of DROCAs was shown to be in P by Böhm and Göller (2011). From Section 3.1, we know that the reachability and coverability of DROCAs is in P. Given two uninitialised RODCAs $\mathcal{A}$ and $\mathcal{B}$, to check whether $\mathcal{A}$ *covers* $\mathcal{B}$, it suffices to iteratively make every state of $\mathcal{A}$ as the initial state and check whether there exists an initial state $\mathcal{B}$ to which it is equivalent. Since the equivalence of DROCAs is in P, this check can be done in polynomial time with respect to the number of states of the DROCAs. Therefore, Proposition 6.10 immediately gives us the following results for deterministic RODCAs.

Theorem 6.11

Equivalence, regularity, reachability, coverability, covering and coverable equivalence of deterministic RODCAs are in P.

From Chapter 4, we also get the following theorem.

Theorem 6.12

Deterministic RODCAs can be learned using polynomial queries using a SAT solver.

We observe that the relationship between deterministic RODCAs and nondeterministic RODCAs is similar to that between deterministic and nondeterministic finite automata. Similar to nondeterministic finite automata, nondeterministic RODCAs can be determinised by a subset construction of the states of the finite state machine. However, this results in an exponential blow-up. In Example 6.6, the deterministic RODCA that recognises the language $\mathcal{L}_3$ has to check whether every b encountered after reading the word $a^n(b+c)^{n+1}$ is at the k^{th} position from the end. This will require at least $\mathcal{O}(2^k)$ states. On the other hand, there is a nondeterministic RODCA with $\mathcal{O}(k)$ states recognising the same language. Like finite automata, nondeterministic RODCAs are "succinct" than deterministic RODCAs.

For every language recognised by a nondeterministic RODCA, there is a deterministic RODCA of at most exponential size that recognises it. The idea is a simple subset construction (see Theorem 6.13).

Theorem 6.13 (Determinisation of nondeterministic RODCA)
Given a nondeterministic RODCA, a polynomial space machine can output an equivalent deterministic RODCA of exponential size.

Proof. Let $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$ be a nondeterministic RODCA. Given a vector $\mathbf{x} \in \mathcal{S}^k$ for some $k \in \mathbb{N}$, we define the function IsDet: $\mathcal{S}^k \to \{true, false\}$ as follows:

$$\text{IsDet}(\mathbf{x}) = \begin{cases} \text{true, if } \exists i < k \text{ s.t } \mathbf{x}[i] = 1 \text{ and } \forall j \neq i, \mathbf{x}[i] = 0 \\ \text{false, otherwise.} \end{cases}$$

Given a transition matrix $\mathbb{A}$ corresponding to the states Q, we define its determinisation $\det(\mathbb{A})$ as follows. There are rows and columns corresponding to each set in 2^Q. For any $q_i \in Q$, let $\mathcal{M}(q_i, \mathbb{A}) = \{q_j \mid \mathbb{A}[i][j] = 1\}$ be the set of all states in the row of q_i whose entries are 1. With the notation that $\det(\mathbb{A})[s][s']$ corresponds to the entry of the cell corresponding to the sets $s, s' \in 2^Q$, we let $\det(\mathbb{A})[s][s'] = 1$ if and only if $s' = \bigcup_{q \in s} \mathcal{M}(q_i, \mathbb{A})$. We claim that $\mathcal{A}_{\text{det}} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta', \boldsymbol{\eta}'), \Sigma)$, with $\boldsymbol{\eta}'$ such that for any $S \in 2^Q$,$\boldsymbol{\eta}'[S] = \bigvee_{s \in S} \boldsymbol{\eta}[s]$ and for all $a \in \Sigma$ and $d \in \{0, 1\}$, $\Delta'(a, d) = \det(\Delta(a, d))$ is such that it is deterministic and $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\text{det}})$.

For this, for any sequence of operations $T = \tau_0 \cdots \tau_{\ell-1}$, let $\mathbf{v}_T, \mathbf{v}'_T$ be the vectors corresponding to $\boldsymbol{\lambda}\mathtt{we}(T)$ in $\mathcal{A}$ and $\mathcal{A}_{\text{det}}$ respectively. Then we have $\text{IsDet}(\mathbf{v}'_T) = 1$ and for any $S \in 2^Q$, $\mathbf{v}'_T[S] = 1$ if and only if for all $q_i \in S$, $\mathbf{v}_T[i] = 1$. $\square$

The idea in proving the above theorem is a simple subset construction. The above result and the fact that equivalence of deterministic RODCAs is in NL gives us the upper bound in the following theorem. The lower bound follows from the equivalence of NFAs (Stockmeyer and Meyer, 1973).

Theorem 6.14
The equivalence problem for nondeterministic RODCAs is PSPACE-complete.

The equivalence of nondeterministic OCA is undecidable (Ibarra, 1979). Our theorem shows that undecidability is due to nondeterminism in the component that modifies the counter.

We also get the following results as a corollary of Theorem 6.11 and Theorem 6.13.

Theorem 6.15
Regularity, covering, coverable equivalence and learning of nondeterministic RODCAs are in PSPACE.

The complexity of the reachability and coverability problems remains the same as that of deterministic RODCAs.

Theorem 6.16
Reachability and coverability of nondeterministic RODCAs are in P.

Unlike deterministic RODCAs and nondeterministic RODCAs, the results on weighted RODCAs over fields are not very straightforward. The next chapter is dedicated to proving our results on weighted RODCAs over fields.

6.6 Conclusion

In this chapter, we introduced a new model called one-deterministic-counter automata. The model "separates" the machine into two components, (1) counter structure – that can modify the counter, and (2) finite state machine – that can access the counter. This separation of the "writing" and "reading" parts gives some natural advantages to the model. These are one-counter automata where the operations are counter-deterministic. We considered weighted RODCAs over the boolean semiring and showed that they are equivalent to DROCAs. We showed that the equivalence of nondeterministic RODCAs is in PSPACE, in contrast to that of nondeterministic OCA, which is undecidable.

CHAPTER 7

Equivalence of Weighted RODCAs Over Fields

In this chapter, we look at weighted RODCAs over *fields*. We present a novel problem called the co-VS (complement to a vector space) reachability problem for weighted RODCAs over fields, which seeks to determine if there exists a run from a given configuration of a weighted RODCA to another configuration whose weight vector lies outside a given vector space. We establish two significant properties of witnesses for co-VS reachability: they satisfy a pseudo-pumping lemma, and the length lexicographically minimal witness has a special form. It follows that the co-VS reachability problem is in P (resp. NP), when the input counter values are specified in unary (resp. binary).

These reachability problems help us to show that the equivalence problem of weighted RODCAs over fields is in P by adapting the equivalence proof of DROCAs by Böhm and Göller (2011). This is a step towards resolving the open question of the equivalence problem of weighted OCAs over fields. Next, we demonstrate that the regularity problem, the problem of checking whether an input weighted RODCA over a field is equivalent to some weighted automaton, is in P. Finally, we look at the covering problem, the problem of checking whether one uninitialised weighted RODCA covers another. Specifically, an uninitialised weighted RODCA $\mathcal{A}$ is said to cover another uninitialised weighted RODCA $\mathcal{B}$ if, for every initial configuration of $\mathcal{B}$, there exists an initial configuration of $\mathcal{A}$ that makes them equivalent.

Contents

7.1 Introduction

7.1.1 Motivation

Probabilistic pushdown automaton (pPDA) has been studied for the analysis of stochastic programs with recursion (Kucera et al., 2006; Olmedo et al., 2016). They are equivalent to recursive Markov chains (Brázdil et al., 2005; Kucera, 2005). pPDAs are also a generalisation of stochastic context-free grammars (Abney et al., 1999) used in natural language processing and many variants of one-dimensional random walks (Brázdil et al., 2013).

The decidability of equivalence of probabilistic pushdown automata is a long-standing open problem (Forejt et al., 2014). The problem is inter-reducible to multiplicity equivalence of context-free grammars. In fact, the decidability is only known for some special subclasses of pPDAs. It is known that the equivalence problem for pPDAs is in PSPACE if the alphabet contains only one letter and is at least as hard as polynomial identity testing (Forejt et al., 2014). There is a randomised polynomial time algorithm that determines the non-equivalence of two visibly pPDAs over the pushdown alphabet triple $(\Sigma_{call}, \Sigma_{ret}, \Sigma_{int})$ where both machines perform push, pop, and no-action on the stack over the symbols in Σ_{call}, Σ_{ret}, and Σ_{int} respectively (Kiefer et al., 2013). There is a polynomial-time reduction from polynomial identity testing to this problem. Hence, it is highly unlikely that the problem is in P.

Since the equivalence problem for pPDAs is unknown, the natural question to ask is the equivalence problem for probabilistic one-counter automata. However, this problem is also unresolved. In this paper, we identify a subclass of probabilistic OCAs (probabilistic RODCAs are also a superclass of visibly probabilistic OCAs) for which the equivalence problem is decidable. In particular, we show that the problem is in P. Note that our results are slightly more general since we consider weighted RODCAs where weights are from a field.

7.1.2 Our Contributions on Weighted RODCAs (Weights from a Field)

The chapter's primary focus is on the equivalence problem for weighted RODCAs where the weights are from a field (possibly infinite).

We first introduce a novel reachability problem on weighted RODCAs, called the complement to a vector space (co-VS) reachability problem. The co-VS reachability problem (see Section 7.2) takes a weighted RODCA, an initial configuration, a vector space, a final counter state, and a final counter value as input. It asks, starting from the initial configuration, whether it is possible to reach a configuration with the final counter state, final counter value, and weight distribution over the states that is not in the given vector space. We consider the cases where the input counter values are specified in unary and binary.

Let us call a word a *witness* if the run of the word 'reaches' a configuration desired by the reachability problem. We identify two interesting properties of witnesses.

1. pseudo-pumping lemma (Lemma 7.5): If the run of a witness encounters a 'large' counter value, then it can be pumped down (resp. pumped-up) to get a run where the maximum counter value encountered is smaller (resp. larger). However, the lemma is distinct from a traditional pumping lemma, where the same subword can be pumped down (or pumped up) multiple times while maintaining reachability. In the case of weighted RODCAs, we only claim that a subword can be pumped once while maintaining reachability. However, repeatedly pumping the same subword might not give a reachability witness. It follows from the pseudo-pumping lemma that the co-VS reachability problem is in P (Theorem 7.11).

2. special-word lemma (Lemma 7.13): The length lexicographically smallest witness is of the form $uy_1^{r_1}vy_2^{r_2}w$ where u, v, w, y_1 and y_2 are 'small' words and $r_1, r_2 \in \mathbb{N}$. The length of the word uy_1vy_2w is bounded by a polynomial in the number of states of the weighted RODCA, whereas r_1 and r_2 also depend on the counter values of the initial and final configurations.

Comparing the above properties with that of deterministic one-counter automata will be interesting. In a DROCA, the reachability problem is equivalent to asking whether there is a path to a final state (rather than a weight distribution over states) and a counter value from an initial state and counter value. Let z be an arbitrary 'long' witness. Consider the run on z of the DROCA. By the pigeonhole principle (see Valiant and Paterson (1975)), there will be words u, y_1, v, y_2, and w such that $z = uy_1vy_2w$, and y_1 (and similarly y_2) starts and ends in the same state and the effect of y_1 on the counter is minus of the effect of y_2 on the counter. In short, y_1 and y_2 form loops with inverse counter-effects and can be pumped simultaneously. Therefore, for all $r \in \mathbb{N}$, the word $uy_1^rvy_2^rw$ is a witness. One can view this as a pumping lemma for DROCAs (see Ogden's lemma for pushdown automata (Ogden, 1968)). Such a property does not hold in the case of weighted RODCAs.

The proofs of Lemma 7.5 and Lemma 7.13 use linear algebra and combinatorics on words and are distinct from those employed for DROCAs. We also introduce a similar problem called co-VS coverability (see Section 7.2). The two properties of the witness and co-VS coverability are crucial along with the ideas developed by Böhm and Göller (2011); Böhm et al. (2010, 2014) and Valiant and Paterson (1975) in solving the equivalence problem. The complete proof is provided in Section 7.3.

Theorem 7.1
There is a polynomial time algorithm that decides if two weighted RODCAs (weights from a field) are equivalent and outputs a word that distinguishes them otherwise.

Finally, we consider the regularity problem - the problem of deciding whether a weighted RODCA is equivalent to some weighted automaton. We show that the regularity problem of weighted RODCAs (weights from a field) is in P. The proof technique is adapted from the ideas developed by Böhm et al. (2014) in the context of DROCAs. The crucial idea in proving regularity is to check for the existence of infinitely many equivalence classes. The pseudo-pumping lemma (particularly pumping-up) is used in proving this. A detailed proof can be found in Section 7.4.

7.1.3 Related Work

Extensive studies have been conducted on weighted automata with weights from semirings. Tzeng (1992) (also see Schützenberger (1961)) gave a polynomial time algorithm to decide the equivalence of two probabilistic automata. The result has been extended to weighted automata with weights over a field. On the other hand, the problem is undecidable if the weights are over the semiring $(\mathbb{N}, \min, +)$ (Krob, 1994). Unlike the extensive literature on weighted automata, the study on weighted versions of pushdown or one-counter machines is limited (Forejt et al., 2012; Hromkovic and Schnitger, 2010; Kucera et al., 2006).

Moving on to the non-weighted models, the equivalence problem for nondeterministic pushdown automata is known to be undecidable. On the other hand, from the seminal result by Sénizergues (1997), we know that the equivalence problem for deterministic pushdown automata is decidable. It was later proved to be primitive recursive (Stirling, 2002). The language equivalence of synchronised real-time height-deterministic pushdown automata is in EXPTIME (Nowotka and Srba, 2007). The equivalence problem for deterministic one-counter automata (with and without ε transitions), similar to that of deterministic finite automata, is NL-complete (Böhm et al., 2013).

7.2 Reachability Problems

In this section, we introduce the co-VS reachability and co-VS coverability problems for weighted RODCAs over a field $\mathcal{F}$. We fix a weighted RODCA $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$. We use $\mathcal{V} \subseteq \mathcal{F}^{|Q|}$ to denote a vector space and $\overline{\mathcal{V}}$ its complement. Let $S \subseteq C$ be a subset of the set of counter states, $X \subseteq \mathbb{N}$ a set of counter values, and $w \in \Sigma^*$. The notation $\mathtt{c} \xrightarrow{w} \overline{\mathcal{V}} \times S \times X$ denotes the run $\mathtt{c} \xrightarrow{w} \mathtt{d}$ where $\mathtt{d} \in \overline{\mathcal{V}} \times S \times X$ if it exists. We use $\mathtt{c} \xrightarrow{*} \overline{\mathcal{V}} \times S \times X$ to denote that there exists a word $u \in \Sigma^*$ such that $\mathtt{c} \xrightarrow{u} \overline{\mathcal{V}} \times S \times X$.

CO-VS REACHABILITY PROBLEM
INPUT: a weighted RODCA $\mathcal{A}$, an initial configuration $\mathtt{c}$, a vector space $\mathcal{V}$, a set of counter states S, and a counter value m.
OUTPUT: *Yes*, if there exists a run $\mathtt{c} \xrightarrow{*} \overline{\mathcal{V}} \times S \times \{m\}$ in $\mathcal{A}$. *No*, otherwise.

CO-VS COVERABILITY PROBLEM
INPUT: a weighted RODCA $\mathcal{A}$, an initial configuration $\mathtt{c}$, a vector space $\mathcal{V}$, and a set of counter states S.
OUTPUT: *Yes*, if there exists a run $\mathtt{c} \xrightarrow{*} \overline{\mathcal{V}} \times S \times \mathbb{N}$ in $\mathcal{A}$. *No*, otherwise.

Unlike the *co-VS reachability* problem, the final configuration's counter value is not considered part of the input for *co-VS coverability* problem. The number of possible vectors in the given vector space can be infinite, and therefore, it is impossible to list them all. However, for $\mathcal{V} \subseteq \mathcal{F}^{|Q|}$, there is a basis set with $|Q|$ vectors in it, and all vectors in $\mathcal{V} \subseteq \mathcal{F}^{|Q|}$ can be written as a linear combination of these basis vectors. Also, any linear combination of the basis vectors is in $\mathcal{V} \subseteq \mathcal{F}^{|Q|}$. We assume that the vector space $\mathcal{V} \subseteq \mathcal{F}^{|Q|}$ is provided by giving a basis. We call $z \in \Sigma^*$ a reachability witness of $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$ if $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times X$. Furthermore, z is called a minimal reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$ if for all $u \in \Sigma^*$ with $\mathsf{c} \xrightarrow{u} \overline{\mathcal{V}} \times S \times X$, $|u| \geq |z|$.

First, we look at the particular case of co-VS reachability problem for weighted automata. Given a weighted automaton $\mathcal{D} = (Q, \Sigma, \boldsymbol{\lambda}, \delta, \boldsymbol{\eta})$ over a field $\mathcal{F}$, with k states, and a vector space $\mathcal{U} \subseteq \mathcal{F}^k$, the co-VS reachability problem asks whether there exists a word w such that $\boldsymbol{\lambda}\delta(w) \in \overline{\mathcal{U}}$. The minimal reachability witness is the minimal word w such that $\boldsymbol{\lambda}\delta(w) \in \overline{\mathcal{U}}$.

The idea of equivalence checking of weighted automata goes back to the seminal paper by Schützenberger (1961). Tzeng (1992) provided a polynomial time algorithm for the equivalence of two probabilistic automata. The same algorithm can be modified to solve the co-VS reachability problem of weighted automata. However, we give the algorithm for the sake of completeness.

Algorithm 5: co-VS reachability of weighted automata.

input : A weighted automaton $\mathcal{D} = (Q, \Sigma, \boldsymbol{\lambda}, \delta, \boldsymbol{\eta})$ and a vector space $\mathcal{U} \subseteq \mathcal{F}^k$.

output: Yes and a word $w \in \Sigma^{\leq |\mathcal{D}|}$, if there exists $w \in \Sigma^*$ such that $\boldsymbol{\lambda}\delta(w) \in \overline{\mathcal{U}}$. No, otherwise.

```
Initialize B = {λ}, i = 0, queue = [(ε, λ)].
repeat
    (w, x) = queue.dequeue().
    if (x ∈ Ū) then
        return yes and w.
    end
    foreach a ∈ Σ do
        if (xδ(a) not in span of B) then
            B = B ∪ {x.δ(a)}.
            queue.enqueue((wa, x.δ(a))).
        end
    end
until queue not empty;
return no.
```

Theorem 7.2
There is a polynomial time algorithm that decides the co-VS reachability problem for weighted automata and outputs a minimal reachability witness if it exists.

Proof. Let $\mathcal{D} = (Q, \Sigma, \boldsymbol{\lambda}, \delta, \boldsymbol{\eta})$ a weighted automaton over a field $\mathcal{F}$ and w be a word such that $\boldsymbol{\lambda}\delta(w) \in \overline{\mathcal{U}}$. Consider Algorithm 5. From Lemma 2.2 Point 1, we get that there can be at most $|\mathcal{D}|$ elements in the set B. Therefore, at most $|\mathcal{D}|$ entries are added to $queue$ during the execution of the algorithm. Hence, the algorithm terminates in polynomial time with respect to $|\mathcal{D}|$.

Now, we show the correctness of Algorithm 5. If the algorithm returned *no*, then for all $\mathbf{x} \in B$, $\mathbf{x} \in \mathcal{V}$ and all $a \in \Sigma$, $\mathbf{x}.\delta(a)$ is in the span of B. Assume for contradiction that there exists $w \in \Sigma^*$ such that $\boldsymbol{\lambda}\delta(w) \in \overline{\mathcal{U}}$. We can represent $\boldsymbol{\lambda}\delta(w)$ as a linear combination of elements in B. However, we know that for all $\mathbf{x} \in B$, $\mathbf{x} \in \mathcal{U}$. Therefore, their linear combinations must also be in $\mathcal{U}$. Since $\boldsymbol{\lambda}\delta(w)$ is a linear combination of elements in B, we get that $\boldsymbol{\lambda}\delta(w) \in \mathcal{U}$, which is a contradiction. Therefore, if the algorithm returns *no*, then for all $w \in \Sigma^*$, $\boldsymbol{\lambda}\delta(w) \in \mathcal{U}$. If the algorithm returns *yes* and a word $w \in \Sigma^*$ then $\boldsymbol{\lambda}\delta(w) \in \overline{\mathcal{U}}$. Since at most $|\mathcal{D}|$ entries are added to $queue$ during Algorithm 5, we get that $|w| \leq |\mathcal{D}|$. ☐

In the upcoming subsection, we give some interesting properties of minimal witnesses. In Section 7.2.2, we provide a pseudo-pumping lemma which helps us show that co-VS reachability and co-VS coverability are in P if the counter values are given in unary notation. Finally, in Section 7.2.3.1, we demonstrate that the length lexicographically minimal reachability witness has a canonical form. In the following subsections, $\mathcal{V}$ denotes a vector space, c a configuration, S a subset of counter states, and $X \subseteq \mathbb{N}$. We also denote by $K = |Q| \cdot |C|$, where C is the set of counter states, and Q is the set of states of the finite state machine.

7.2.1 Minimal Witness and Its Properties

The following observation helps in breaking down the reachability problem into sub-problems. If $z \in \Sigma^*$ is a minimal reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, X)$, then for every z_1, z_2 such that $z = z_1 z_2$, there is a vector space $\mathcal{U}$ such that z_1 is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{U}}, \{p\}, \{n\})$ where p is the counter state and n is the counter value reached after reading z_1 from $\mathtt{c}$.

Proposition 7.3

Let $\mathcal{A}$ be a weighted RODCA. Consider arbitrary $z, z_1, z_2 \in \Sigma^*$ such that $z = z_1 z_2$. Let $\mathtt{d} = (\mathbf{x}_\mathtt{d}, p_\mathtt{d}, n_\mathtt{d})$ and $\mathtt{e} = (\mathbf{x}_\mathtt{e}, p_\mathtt{e}, n_\mathtt{e})$ be configurations of $\mathcal{A}$ such that $\mathtt{c} \xrightarrow{z_1} \mathtt{d} \xrightarrow{z_2} \mathtt{e}$ in $\mathcal{A}$ and $\mathbb{A} \in \mathcal{F}^{|Q| \times |Q|}$ be such that $\mathbf{x}_\mathtt{d} \mathbb{A} = \mathbf{x}_\mathtt{e}$. If z is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, X)$ in $\mathcal{A}$, then z_1 is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{U}}, \{p_\mathtt{d}\}, \{n_\mathtt{d}\})$ in $\mathcal{A}$, where $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{|Q|} \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}$.

Proof. Let $z \in \Sigma^*$ be a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, X)$ in a weighted RODCA $\mathcal{A}$, $\mathtt{d} = (\mathbf{x}_\mathtt{d}, p_\mathtt{d}, n_\mathtt{d})$ and $\mathtt{e} = (\mathbf{x}_\mathtt{e}, p_\mathtt{e}, n_\mathtt{e})$ be configurations of $\mathcal{A}$ such that $\mathtt{c} \xrightarrow{z_1} \mathtt{d} \xrightarrow{z_2} \mathtt{e}$ where $z_1, z_2 \in \Sigma^*$ with $z = z_1 z_2$ and $\mathbb{A} \in \mathcal{F}^{|Q| \times |Q|}$ be such that $\mathbf{x}_\mathtt{d} \mathbb{A} = \mathbf{x}_\mathtt{e}$. Let $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{|Q|} \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}$. Assume for contradiction that there exists $z_1' \in \Sigma^*$ smaller than z_1 and $\mathtt{c} \xrightarrow{z_1'} \mathtt{f}$ for some configuration $\mathtt{f} \in \overline{\mathcal{U}} \times \{p_\mathtt{d}\} \times \{n_\mathtt{d}\}$ of $\mathcal{A}$. Note that for all $\mathbf{y} \in \overline{\mathcal{U}}$, the vector $\mathbf{y}\mathbb{A} \in \overline{\mathcal{V}}$. Since the configurations $\mathtt{f}$ and $\mathtt{d}$ have the same counter state and counter value, $\mathtt{c} \xrightarrow{z_1'} \mathtt{f} \xrightarrow{z_2} \overline{\mathcal{V}} \times \{p_\mathtt{e}\} \times \{n_\mathtt{e}\}$ is a run in $\mathcal{A}$ and the word $z_1' z_2$ contradicts the minimality of z. $\square$

We aim to show that the length of a minimal witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, X)$ is polynomially bounded. The following lemma shows that if the counter values are polynomially bounded during the run of a minimal witness, then its length is also polynomially bounded.

Lemma 7.4

Let $\mathcal{A}$ be a weighted RODCA and $z \in \Sigma^*$ be a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, X)$ in $\mathcal{A}$. If the number of distinct counter values encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times X$ in $\mathcal{A}$ is t, then $|z| \leq t \cdot K$.

Proof. Let $\mathtt{c} = \mathtt{c}_1$ and $T(\mathtt{c}_1) = \mathtt{c}_1 \tau_1 \mathtt{c}_2 \cdots \tau_{h-1} \mathtt{c}_h$ be the run on word z from $\mathtt{c}_1$ and T the corresponding sequence of transitions of the weighted RODCA $\mathcal{A}$. Let t be the number of distinct counter values encountered during this run. Now assume for contradiction that $h > |Q| \cdot |C| \cdot t$, then by the pigeonhole principle, there are $|Q| + 1$ many configurations $\mathsf{c}_{i_0}, \mathsf{c}_{i_1}, \ldots, \mathsf{c}_{i_{|Q|}}$ with the same counter state and counter value during this run. Given a configuration c, let $\mathbf{x}_\mathsf{c}$ denote $\textsc{WeightVector}(\mathsf{c})$. Let $\mathbb{A}_j$ denote the matrix such that $\mathbf{x}_{\mathsf{c}_{i_j}} \mathbb{A}_j = \mathbf{x}_{\mathsf{c}_h}$ for all $j \in [0, |Q|]$. Using linear algebra, we get that there exists $r \le |Q|$, and $t \in [0, r-1]$ such that $\mathbf{x}_{\mathsf{c}_{i_t}} \mathbb{A}_r \in \overline{\mathcal{V}}$. Consider the sequence of transitions $T' = \tau_{1 \cdots i_t} \tau_{r \cdots \ell-1}$ and $v = \texttt{word}(T')$. The run $\pi(v, \mathsf{c}_1) = T'(\mathsf{c}_1)$ is a run in $\mathcal{A}$ since configurations c_t and c_r have the same counter state and counter value. This is a shorter run than $\pi(z, \mathsf{c}_1)$ and $\mathsf{c}_1 \xrightarrow{v} \overline{\mathcal{V}} \times S \times X$ in $\mathcal{A}$. This contradicts the minimality of z. $\square$

It now suffices to show that the counter values encountered during the run of a minimal reachability witness are polynomially bounded.

7.2.2 Pseudo-Pumping Lemma

The pseudo-pumping lemma is a valuable tool in our analysis, allowing us to pump up or down a sufficiently long word while maintaining the reachability conditions.

Lemma 7.5 (Pseudo-pumping lemma)

Let c be a configuration of a weighted RODCA $\mathcal{A}$, $m, R \in \mathbb{N}$, be such that $\textsc{CounterValue}(\mathsf{c}) = m$ and $z \in \Sigma^*$ be such that $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run in $\mathcal{A}$, and the maximum counter value encountered during this run is $m + R$. If $R > K^2$, then there exists $z_{sub}, z_{sup} \in \Sigma^*$ such that the following hold:

1. there exist $x, y, u, v, w \in \Sigma^*$ such that $z = xyuvw, z_{sub} = xuw$, $\mathsf{c} \xrightarrow{z_{sub}} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run in $\mathcal{A}$, and the counter values encountered during this run are less than $m + R$, and
2. there exist $x, y, u, v, w \in \Sigma^*$ such that $z = xyuvw, z_{sup} = xy^2uv^2w$, $\mathsf{c} \xrightarrow{z_{sup}} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run in $\mathcal{A}$, and the maximum counter value encountered in this run exceeds $m + R$.

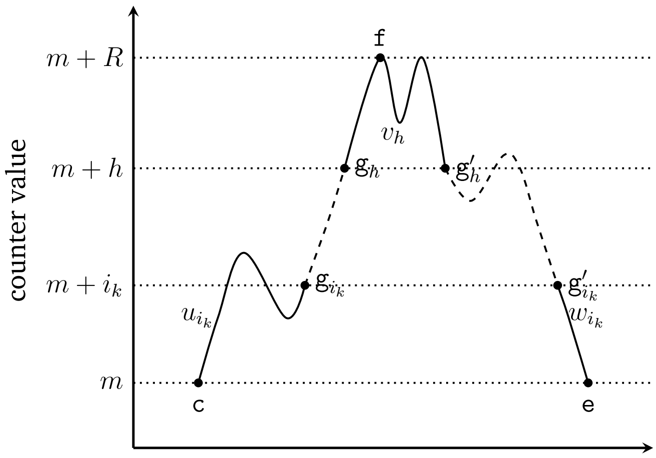


Fig. 7.1. The figure shows the floating run from a configuration c with $\textsc{CounterValue}(\mathsf{c}) = m$ to a configuration $\mathsf{e} = (\mathbf{x}, p, m)$ such that $\mathbf{x} \in \overline{\mathcal{U}}$. configurations g_{i_k} and g_h (resp. g'_{i_k} and g'_h) are where the counter values $m + i_k$ and $m + h$ are encountered for the last (resp. first) time before (resp. after) reaching $m + R$. Also, $\textsc{CounterState}(\mathsf{g}_{i_k}) = \textsc{CounterState}(\mathsf{g}_h)$ and $\textsc{CounterState}(\mathsf{g}'_h) = \textsc{CounterState}(\mathsf{g}'_{i_k})$. The dashed line denotes the part of the run that can be removed to get a shorter witness for $(\mathsf{c}, \overline{\mathcal{U}}, \{p\}, \{n\})$.

Proof. Let $z \in \Sigma^*$ be a witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$ in the weighted RODCA $\mathcal{A}$ and $\mathsf{e} \in \overline{\mathcal{V}} \times S \times \{m\}$ be such that $\mathsf{c} \xrightarrow{z} \mathsf{e}$ is a floating run in $\mathcal{A}$, and the maximum counter value encountered in this run be $m + R$ where $R > K^2$. Let $\textsc{CounterValue}(\mathsf{c}) = m$. There exist $z_1, z_2 \in \Sigma^*$ and configuration f of $\mathcal{A}$ such that $z = z_1 z_2$ and $\mathsf{c} \xrightarrow{z_1} \mathsf{f} \xrightarrow{z_2} \mathsf{e}$, where $\textsc{CounterValue}(\mathsf{f}) = m + R$ (see Figure 7.1).

Let $\mathsf{c}_1 = \mathsf{c}$ and $\pi = \mathsf{c}_1 \tau_1 \mathsf{c}_2 \cdots \tau_{\ell-1} \mathsf{c}_\ell$ denote the run on word z from the configuration c_1 and $T = \tau_1 \tau_2 \cdots \tau_{\ell-1}$ the sequence of transitions of π. For any $i \in [0, R]$, we denote by l_i and d_i the indices such that a configuration with counter value $m + i$ is encountered for the last (resp. first) time before (resp. after) reaching counter value $m + R$ in π. That is, $\textsc{CounterValue}(\mathsf{c}_{l_i}) = \textsc{CounterValue}(\mathsf{c}_{d_i}) = m + i$, and for any j where $l_i < j < d_i$, $\textsc{CounterValue}(\mathsf{c}_j) > m + i$. To simplify the notation, we denote by $\mathsf{g}_i = \mathsf{c}_{l_i}$ and $\mathsf{g}'_i = \mathsf{c}_{d_i}$.

Consider the pairs of configurations $(\mathsf{g}_1, \mathsf{g}'_1), (\mathsf{g}_2, \mathsf{g}'_2), \ldots, (\mathsf{g}_R, \mathsf{g}'_R)$. Since $R > (|Q| \cdot |C|)^2$, by the pigeonhole principle, there exist two counter states p, q, and a set of indices $I \subseteq [0, R]$ where $|I| = |Q|^2 + 1$ such that for all $h \in I$, $\textsc{CounterState}(\mathsf{g}_h) = p$ and $\textsc{CounterState}(\mathsf{g}'_h) = q$. For all $j \in I$, let $u_j, v_j, w_j \in$

Σ^* be such that $\mathtt{c}_1 \xrightarrow{u_j} \mathtt{g}_j \xrightarrow{v_j} \mathtt{g}'_j \xrightarrow{w_j} \mathtt{e}$. We use the following shorthand for any configuration $\mathtt{g}$: $\mathbf{x}_{\mathtt{g}} = \text{WEIGHTVECTOR}(\mathtt{g})$. For all $j \in I$, let matrix $\mathbb{A}_j$ and $\mathbb{B}_j$ be such that $\mathbf{x}_{\mathtt{g}'_j} = \mathbf{x}_{\mathtt{g}_j}\mathbb{A}_j$ and $\mathbf{x}_{\mathtt{e}} = \mathbf{x}_{\mathtt{g}'_j}\mathbb{B}_j$. Since $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{V}}$, for all $j \in I$, $\mathbf{x}_{\mathtt{g}_j}\mathbb{A}_j\mathbb{B}_j \in \overline{\mathcal{V}}$. Let $r = |Q|^2 + 1$, and $i_1 < i_2 < \cdots < i_r$ be the indices in I. We prove Point 1 and Point 2 of the Lemma separately.

1. Consider the sequence of matrices $\mathbb{A}_{i_r}, \mathbb{A}_{i_{r-1}}, \ldots, \mathbb{A}_{i_1}$. From Lemma 2.3 Point 1, we get that there exists $k \in [1, r]$ such that $\mathbb{A}_{i_k}$ is a linear combination of $\mathbb{A}_{i_r}, \ldots, \mathbb{A}_{i_{k+1}}$. Now, from Lemma 2.3 Point 2, there exists $h \in \{i_r, \ldots, i_{k+1}\}$ such that $\mathbf{x}_{\mathtt{g}_{i_k}}\mathbb{A}_h\mathbb{B}_{i_k} \in \overline{\mathcal{V}}$.

Let $z_{sub} = u_{i_k}v_hw_{i_k}$. It is easy to observe that z_{sub} is a subword of z, as mentioned in the lemma. To conclude the proof, it now suffices to show that z_{sub} is a witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ and the counter values encountered during the run $\mathtt{c} \xrightarrow{z_{sub}} \mathtt{h}$ are less than $m + R$. Consider the floating run $\mathtt{g}_h \xrightarrow{v_h} \mathtt{g}'_h$. From the choice of $\mathtt{g}_h$ and $\mathtt{g}'_h$ we know that $\text{COUNTERVALUE}(\mathtt{g}_h) = \text{COUNTERVALUE}(\mathtt{g}'_h) = m + h$ and for all j where $l_h < j < d_h$, $\text{COUNTERVALUE}(\mathtt{c}_j) > m + h$. Since $\text{COUNTERSTATE}(\mathtt{g}_h) = \text{COUNTERSTATE}(\mathtt{g}_{i_k})$, $\pi(v_h, \mathtt{g}_{i_k})$ is also a floating run $\mathtt{g}_{i_k} \xrightarrow{v_h} \mathtt{d}$ such that $\text{COUNTERSTATE}(\mathtt{g}'_h) = \text{COUNTERSTATE}(\mathtt{d})$, $\text{COUNTERVALUE}(\mathtt{g}_{i_k}) = \text{COUNTERVALUE}(\mathtt{d}) = m + i_k < m + h$, and the minimum and maximum counter values encountered in the run is $m + i_k$ and $m + R - (h - i_k)$ respectively (see Figure 7.1). Furthermore, $\mathbf{x}_d = \mathbf{x}_{\mathtt{g}_{i_k}}\mathbb{A}_h$. Since $\text{COUNTERSTATE}(\mathtt{g}'_{i_k}) = \text{COUNTERSTATE}(\mathtt{g}'_h)$, we get that $\text{COUNTERSTATE}(\mathtt{g}'_{i_k}) = \text{COUNTERSTATE}(\mathtt{d})$. Since $\text{COUNTERVALUE}(\mathtt{g}'_{i_k}) = \text{COUNTERVALUE}(\mathtt{g}_{i_k})$, we have $\text{COUNTERVALUE}(\mathtt{g}'_{i_k}) = \text{COUNTERVALUE}(\mathtt{d})$. Therefore, $\pi(w_{i_k}, \mathtt{d})$ is the run $\mathtt{d} \xrightarrow{w_{i_k}} \mathtt{h}$ where $\mathbf{x}_{\mathtt{h}} = \mathbf{x}_{\mathtt{d}}\mathbb{B}_{i_k}$ and hence $\mathbf{x}_{\mathtt{h}} = \mathbf{x}_{\mathtt{g}_{i_k}}\mathbb{A}_h\mathbb{B}_{i_k} \in \overline{\mathcal{V}}$. This concludes that z_{sub} is a reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ and satisfies the properties mentioned in the lemma.

2. Consider the sequence of matrices: $\mathbb{A}_{i_1}, \mathbb{A}_{i_2}, \ldots, \mathbb{A}_{i_r}$. Note that the matrices are ordered in reverse compared to the ordering in the previous case. From Lemma 2.3 Point 1, there exists $k \in [1, r]$ such that $\mathbb{A}_{i_k}$ is a linear combination of $\mathbb{A}_{i_1}, \ldots, \mathbb{A}_{i_{k-1}}$. Now from Lemma 2.3 Point 2, there exists an $h \in \{i_1, \ldots, i_{k-1}\}$ such that $\mathbf{x}_{\mathtt{g}_{i_k}}\mathbb{A}_h\mathbb{B}_{i_k} \in \overline{\mathcal{V}}$. Let $z_{sup} = u_{i_k}v_hw_{i_k}$. It is easy to observe that z_{sup} is a superword of z as mentioned in the lemma. To conclude the proof, it suffices to show that z_{sup} is a witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ and the counter values encountered during the run $\mathtt{c} \xrightarrow{z_{sup}} \mathtt{h}$ is greater than $m + R$.

Consider the floating run $\mathtt{g}_h \xrightarrow{v_h} \mathtt{g}'_h$. From the choice of $\mathtt{g}_h$ and $\mathtt{g}'_h$ we know that $\textsc{CounterValue}(\mathtt{g}_h) = \textsc{CounterValue}(\mathtt{g}'_h) = m + h$ and for all j where $l_h < j < d_h$, $\textsc{CounterValue}(\mathtt{c}_j) > m + h$.

Since $\textsc{CounterState}(\mathtt{g}_h) = \textsc{CounterState}(\mathtt{g}_{i_k})$, $\pi(\mathtt{g}_{i_k}, v_h)$ is also a floating run $\mathtt{g}_{i_k} \xrightarrow{v_h} \mathtt{d}$ such that $\textsc{CounterValue}(\mathtt{g}_{i_k}) = \textsc{CounterValue}(\mathtt{d}) = m + i_k > m+h$, $\textsc{CounterState}(\mathtt{g}'_h) = \textsc{CounterState}(\mathtt{d})$, and the minimum and maximum counter values encountered in the run is $m + i_k$ and $m + R + (i_k - h)$ respectively. Furthermore, $\mathbf{x}_d = \mathbf{x}_{\mathtt{g}_{i_k}} \mathbb{A}_h$. Since $\textsc{CounterState}(\mathtt{g}'_{i_k}) = \textsc{CounterState}(\mathtt{g}'_h)$, $\textsc{CounterState}(\mathtt{g}'_{i_k}) = \textsc{CounterState}(\mathtt{d})$. Moreover, since $\textsc{CounterValue}(\mathtt{g}'_{i_k}) = \textsc{CounterValue}(\mathtt{g}_{i_k})$, we have $\textsc{CounterValue}(\mathtt{g}'_{i_k}) = \textsc{CounterValue}(\mathtt{d})$. Therefore $\pi(\mathtt{d}, w_{i_k})$ is the run $\mathtt{d} \xrightarrow{w_{i_k}} \mathtt{h}$ where $\mathbf{x}_\mathtt{h} = \mathbf{x}_\mathtt{d} \mathbb{B}_{i_k}$ and hence $\mathbf{x}_\mathtt{h} = \mathbf{x}_{\mathtt{g}_{i_k}} \mathbb{A}_h \mathbb{B}_{i_k} \in \overline{\mathcal{V}}$. This concludes that z_{sup} is a reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ and satisfies the properties mentioned in the lemma. □

It is important to note that we do not end up in the same configuration while pumping up/down, but we ensure that we reach a configuration with the same counter state, counter value, and whose weight vector is in the complement of the given vector space. A similar lemma can be obtained that gives a bound on the minimal counter value encountered during the run of a minimal reachability witness if its run is a floating run.

Lemma 7.6

Let $\mathtt{c}$ be a configuration of a weighted RODCA $\mathcal{A}$. Let $m, R \in \mathbb{N}$, be such that $\textsc{CounterValue}(\mathtt{c}) = m$ and $z \in \Sigma^*$ be such that $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run in $\mathcal{A}$, and the minimum counter value encountered during this run is $m - R$. If $R > K^2$, then there exists $z_{sub} \in \Sigma^*$ and $x, y, u, v, w \in \Sigma^*$ such that $z = xyuvw$, $z_{sub} = xuw$, $\mathtt{c} \xrightarrow{z_{sub}} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run in $\mathcal{A}$, and the counter values encountered during this run are greater than $m - R$.

Proof. The proof follows from arguments symmetric to those used in proving Lemma 7.5, Point 1. □

Now, we prove that for any run (it need not necessarily be a floating run) of a minimal reachability witness z for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$, the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is bounded by a polynomial in the

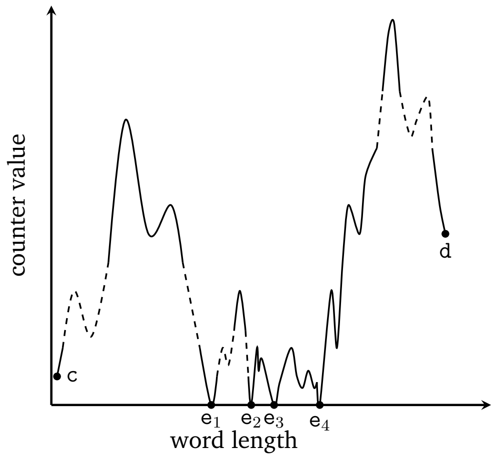


Fig. 7.2. The figure shows a run from configuration $\mathtt{c}$ to $\mathtt{d} = (\mathbf{x}_\mathtt{d}, p_\mathtt{d}, n_\mathtt{d})$ such that $\mathbf{x}_\mathtt{d} \in \overline{\mathcal{V}}$. configurations $\mathtt{e}_1, \mathtt{e}_2, \mathtt{e}_3, \mathtt{e}_4$ are where the counter value zero is encountered during the run. The dashed lines denote the parts that can be removed to obtain a shorter witness for $(\mathtt{c}, \overline{\mathcal{V}}, \{p_\mathtt{d}\}, \{n_\mathtt{d}\})$.

number of states of the machine, and the initial and final counter values. This can be achieved by iteratively applying Lemma 7.5 on the run of the minimal reachability witness (refer Figure 7.2) and using Proposition 7.3 and Lemma 7.4.

Corollary 7.7

If $z \in \Sigma^*$ is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$, then

1. the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is less than $max(\text{CounterValue}(\mathtt{c}), m) + K^2$, and

2. $|z| \leq K^3 + max(\text{CounterValue}(\mathtt{c}), m) \cdot K$.

Proof. Let $z \in \Sigma^*$ be a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$, where $\mathtt{c}$ is a configuration with counter value n.

1. Consider the run of word z from $\mathtt{c}$. Let $\mathtt{d} \in \overline{\mathcal{V}} \times S \times \{m\}$ such that $\mathtt{c} \xrightarrow{z} \mathtt{d}$. Assume for contradiction that the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is greater than $max(n, m) + (|Q| \cdot |C|)^2$. Let $\mathtt{e}_1, \mathtt{e}_2, \cdots, \mathtt{e}_t$ be all the configurations in this run such that their counter values are zero. There exists words $u_1, u_2, \cdots, u_{t+1} \in \Sigma^*$ such that $z = u_1 u_2 \cdots u_{t+1}$ and $\mathtt{c} \xrightarrow{u_1} \mathtt{e}_1 \xrightarrow{u_2} \mathtt{e}_2 \xrightarrow{u_3} \cdots \xrightarrow{u_t} \mathtt{e}_t \xrightarrow{u_{t+1}} \mathtt{d}$. Note that $\mathtt{c} \xrightarrow{u_1} \mathtt{e}_1$, $\mathtt{e}_t \xrightarrow{u_{t+1}} \mathtt{d}$ and $\mathtt{e}_i \xrightarrow{u_{i+1}} \mathtt{e}_{i+1}$ for all $i \in [1, t-1]$

are floating runs (refer Figure 7.2).

We show that the counter values are bounded during these floating runs. First, we consider the floating run $\mathsf{c} \xrightarrow{u_1} \mathsf{e}_1$. Given a configuration c, we use $\mathbf{x}_\mathsf{c}$ to denote $\textsc{WeightVector}(\mathsf{c})$. Let $\mathbb{A} \in \mathcal{F}^{|Q|\times|Q|}$ be such that $\mathbf{x}_\mathsf{d} = \mathbf{x}_{\mathsf{e}_1}\mathbb{A}$. The set $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{|Q|} \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}$ is a vector space and hence the vector $\mathbf{x}_{\mathsf{e}_1} \in \overline{\mathcal{U}}$. From Proposition 7.3, we know that u_1 is a minimal reachability witness for $(\mathsf{c}, \overline{\mathcal{U}}, \{p_{\mathsf{e}_1}\}, \{0\})$ and therefore by Lemma 7.5 we know that the maximum counter value encountered during the run $\pi(u_1, \mathsf{c})$ is less than $n + (|Q| \cdot |C|)^2$.

Similarly for the floating run $\mathsf{e}_t \xrightarrow{u_{t+1}} \mathsf{d}$, the maximum counter value is bounded by $m + (|Q| \cdot |C|)^2$. Now consider the floating runs $\mathsf{e}_i \xrightarrow{u_{i+1}} \mathsf{e}_{i+1}$ for all $i \in [1, t-1]$. Again, by applying Lemma 7.5, we get that the maximum counter value encountered during these sub-runs is less than $(|Q| \cdot |C|)^2$. Therefore, the maximum counter value encountered during the run $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is less than $max(n, m) + (|Q| \cdot |C|)^2$.

2. From the previous point, we know that the maximum counter value encountered during the run $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is less than $max(n, m) + (|Q| \cdot |C|)^2$. Therefore, there are at most $max(n, m) + (|Q| \cdot |C|)^2$ many distinct counter values encountered during this run. Now from Lemma 7.4 we get that $|z| \leq (|Q| \cdot |C|) \cdot (max(n, m) + (|Q| \cdot |C|)^2)$. □

The following lemma (depicted in Figure 7.3) helps us show that the length of a minimal witness for co-VS coverability is polynomially bounded in the number of states.

Lemma 7.8 (Cut lemma)

Let $z \in \Sigma^*$ be a reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, where c is a configuration with $\textsc{CounterValue}(\mathsf{c}) = n$ for some $n \in \mathbb{N}$, and $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run for some $m \in \mathbb{N}$. If $m - n > K$, then there exists $z_{sub} \in \Sigma^*$ such that z_{sub} is a subword of z, $\mathsf{c} \xrightarrow{z_{sub}} \overline{\mathcal{V}} \times S \times \{m'\}$ is a floating run and $m' - n < m - n$.

Proof. Let $z \in \Sigma^*$ be a reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \mathbb{N})$ and $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run. Let n be the counter value of configuration c and $m > n + |Q| \cdot |C|$.

Let $\mathsf{c}_1 = \mathsf{c}$ and $\pi(z, \mathsf{c}_1) = \mathsf{c}_1\tau_1\mathsf{c}_2\cdots\tau_{\ell-1}\mathsf{c}_\ell$ be such that configuration c_ℓ has counter value m. Consider the sequence of transitions $T = \tau_0\tau_1\cdots\tau_{\ell-1}$ in $\pi(z, \mathsf{c}_1)$.

Since there are only $|C|$ counter states, by the pigeonhole principle, there exists a strictly increasing sequence $I = 0 < i_0 < i_1 < \cdots < i_{|Q|} \leq \ell$ such that for all $j, j' \in I$, (1) if $\text{COUNTERSTATE}(\mathsf{c}_j) = \text{COUNTERSTATE}(\mathsf{c}_{j'})$, and (2) if $j < j'$, then for all $d \in [j+1, j'-1]$, $\text{COUNTERVALUE}(\mathsf{c}_j) < \text{COUNTERVALUE}(\mathsf{c}_d) < \text{COUNTERVALUE}(\mathsf{c}_{j'})$ and $\text{COUNTERVALUE}(\mathsf{c}_j) < \text{COUNTERVALUE}(\mathsf{c}_{j'})$. Given a configuration c, let $\mathbf{x}_\mathsf{c}$ denote $\text{WEIGHTVECTOR}(\mathsf{c})$. Consider the set of configurations $\mathsf{c}_{i_0}, \mathsf{c}_{i_1}, \ldots, \mathsf{c}_{i_{|Q|}}$. For any $j \in [0, |Q|]$, let $\mathbb{A}_j$ denote the matrix such that $\mathbf{x}_{\mathsf{c}_{i_j}}\mathbb{A}_j = \mathbf{x}_{\mathsf{c}_\ell}$. Since $\mathbf{x}_{\mathsf{c}_{i_d}}\mathbb{A}_d \in \overline{\mathcal{V}}$ for all $d \in [0, |Q|]$, using linear algebra, we get that there exists $l, k \in [0, |Q|]$ with $l < k$ such that $\mathbf{x}_{\mathsf{c}_{i_l}}\mathbb{A}_k \in \overline{\mathcal{V}}$. Consider a configuration $\mathsf{e} = (\mathbf{x}, p, n)$. If $\pi(u, \mathsf{e})$ is a floating run with the minimal counter value encountered during the run $\pi(u, \mathsf{e})$ greater than 0, then for all $m \in \mathbb{N}$ and $\mathbf{y} \in \mathcal{F}^{|Q|}$, $\pi(u, (\mathbf{y}, p, m))$ is a run. Consider the sequence of transitions $T' = \tau_{i_k\cdots\ell-1}$ and let $u = \texttt{word}(T')$. Because of (2), the minimal counter value encountered during the run $\pi(u, \mathsf{c}_{i_k})$ is greater than 0. Therefore the run $T''(\mathsf{c}_1)$ where $T'' = \tau_{1\cdots i_l-1}\tau_{i_k\cdots\ell-1}$ is a run shorter than $\pi(z, \mathsf{c}_1)$ with smaller counter-effect. $\square$

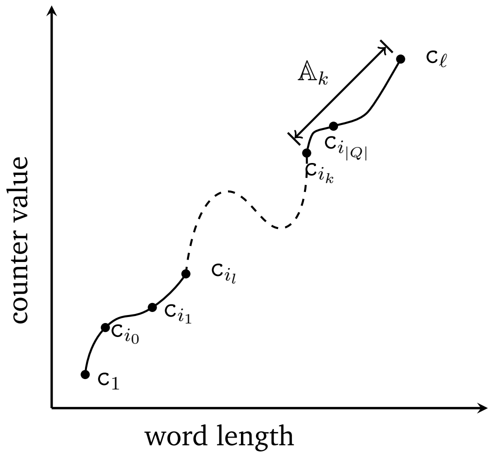


Fig. 7.3. The figure shows a run from configuration c_1 to $\mathsf{c}_\ell = (\mathbf{x}_{\mathsf{c}_\ell}, p_{\mathsf{c}_\ell}, n_{\mathsf{c}_\ell})$ such that $\mathbf{x}_{\mathsf{c}_\ell} \in \overline{\mathcal{V}}$. The configurations c_{i_l} and c_{i_k} are where the counter values $n_{\mathsf{c}_{i_l}}$ and $n_{\mathsf{c}_{i_k}}$ are encountered for the last time. Also the configurations c_{i_l} and c_{i_k} have the same counter state. The dashed line is the part that can be removed to get a shorter reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, \{p_{\mathsf{c}_\ell}\}, \mathbb{N})$.

Now, we prove that the co-VS reachability and co-VS coverability problems of weighted RODCAs are in P when the input counter values are specified in unary notation, by demonstrating a small model property. We have already established using Lemma 7.5, Corollary 7.7, and Lemma 7.8 that the maximum and minimum counter values encountered during the run of the minimal witness do not exceed some polynomial bound. This, in turn, implies a polynomial bound on the length of the witness by Lemma 7.4. As a result, we get the following theorem.

Now we show that for any run (need not be floating) of a minimal reachability witness z for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \mathbb{N}$ is polynomially bounded in the number of states of the machine and the initial counter value.

Corollary 7.9

If $z \in \Sigma^*$ is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, where $\mathtt{c}$ is a configuration with counter value n, then the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \mathbb{N}$ is less than $max(n, |Q| \cdot |C|) + (|Q| \cdot |C|)^2$.

Proof. Let $z \in \Sigma^*$ be a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$, where $\mathtt{c}$ is a configuration with counter value n. Consider the run of word z from $\mathtt{c}$. Let $\mathtt{d} \in \overline{\mathcal{V}} \times S \times \mathbb{N}$ such that $\mathtt{c} \xrightarrow{z} \mathtt{d}$. If $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is a floating run, then by Lemma 7.8 the maximum counter value encountered during this run will be less than $n + |Q| \cdot |C|$. Now if $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is not a floating run, then there exists $u_1, u_2 \in \Sigma^*$ such that $z = u_1 u_2$ and $\mathtt{c} \xrightarrow{u_1} \mathtt{e} \xrightarrow{u_2} \mathtt{d}$ where, counter value of configuration $\mathtt{e}$ is zero and $\mathtt{e} \xrightarrow{u_2} \mathtt{d}$ is a floating run.

Given a configuration $\mathtt{c}$, let $\mathbf{x}_\mathtt{c}$ denote $\textsc{WeightVector}(\mathtt{c})$. Let $\mathbb{A} \in \mathcal{F}^{|Q| \times |Q|}$ be such that $\mathbf{x}_\mathtt{d} = \mathbf{x}_\mathtt{e}\mathbb{A}$. The set $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{|Q|} \mid \mathbf{y}\mathbb{A} \in \mathcal{V}\}$ is a vector space and hence the vector $\mathbf{x}_\mathtt{e} \in \overline{\mathcal{U}}$. Note that for all $\mathbf{y} \in \overline{\mathcal{U}}$, the vector $\mathbf{y}\mathbb{A} \in \overline{\mathcal{V}}$. From Proposition 7.3, we know that u_1 is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{U}}, \{p_\mathtt{e}\}, \{0\})$, where $p_\mathtt{e}$ is the counter state of configuration $\mathtt{e}$, and therefore by Corollary 7.7, we know that the maximum counter value encountered during the run $\pi(u_1, \mathtt{c})$ is less than $n + (|Q| \cdot |C|)^2$. Now since $\mathtt{e} \xrightarrow{u_2} \mathtt{d}$ is a floating run and u_2 is the minimal such word, from Lemma 7.8, we get that the counter value of configuration $\mathtt{d}$ is less than or equal to $|Q| \cdot |C|$, and by Lemma 7.5, we know that the maximum counter value encountered during this run is less than $|Q| \cdot |C| + (|Q| \cdot |C|)^2$. Therefore, we get that the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \mathtt{d}$ is less than $max(n, |Q| \cdot |C|) + (|Q| \cdot |C|)^2$. $\square$

Our next objective is to show that the counter values are polynomially bounded during the run of a minimal witness for coverability. The problem is similar to co-VS reachability, except that now we are not given a final counter value. A crucial ingredient in proving this is Lemma 7.8 which will help us in proving that if the run of a minimal reachability witness z for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$ is a floating run, then the number of distinct counter values encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \mathbb{N}$ is polynomially bounded in the number of states of the machine and the initial counter value. Using this and the ideas presented earlier for co-VS reachability, we can prove the existence of a polynomial length witness for the co-VS coverability problem.

Corollary 7.10

Let $\mathtt{c}$ be a configuration with counter value n. If z is a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$ then $|z| \leq (|Q| \cdot |C|) \cdot (max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2)$.

Proof. Let $z \in \Sigma^*$ be a minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \mathbb{N})$. From Corollary 7.9, we know that the maximum counter value encountered during the run $\mathtt{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \mathbb{N}$ is less than $max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2$. Therefore, there are at most $max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2$ many distinct counter values encountered during this run. Now from Lemma 7.4 we get that $|z| \leq (|Q| \cdot |C|) \cdot (max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2)$. $\square$

Now, we prove that the co-VS reachability and co-VS coverability problems of weighted RODCAs are in P by demonstrating a small model property. We have already established using Lemma 7.5, Corollary 7.7, and Lemma 7.8 that the maximum and minimum counter values encountered during the run of the minimal witness do not exceed some polynomial bound. This, in turn, implies a polynomial bound on the length of the witness by Lemma 7.4. As a result, we get the following theorem.

Theorem 7.11

The co-VS reachability and co-VS coverability problems for weighted RODCAs can be decided in polynomial time when the counter values are given in unary notation.

Proof. Assume we are given a weighted RODCA $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$, initial configuration $\mathsf{c} = (\mathbf{x}, p, n)$, vector space $\mathcal{V}$, set of counter states S and counter value m as inputs for the co-VS reachability problem. Let $t = max(n, m) + (|Q| \cdot |C|)^2$. To solve this reachability problem, we first consider the t-unfolding weighted automaton $\mathcal{A}^t = ((C', \delta', p'_0), (Q', \boldsymbol{\lambda}', \Delta', \boldsymbol{\eta}'_F), \Sigma)$ of $\mathcal{A}$. From Corollary 7.7, we know that the maximum counter value encountered during the run of the minimal reachability witness z for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$ is less than t. We define a vector space $\mathcal{U} \subseteq \mathcal{F}^{|Q'|}$ as follows: A vector $\mathbf{z} \in \mathcal{F}^{|Q'|}$ is in $\mathcal{U}$ if there exists $\mathbf{y} \in \mathcal{V}$ such that for all $i \in [0, |Q| - 1]$, $\mathbf{z}[|Q| \cdot m + i] = \mathbf{y}[i]$ and for all $m' \neq m$ and $i \in [0, |Q| - 1]$, $\mathbf{z}[|Q| \cdot m' + i] = 0$.

Given a configuration $\mathsf{c} = (\mathbf{x}, p, n)$ of a weighted RODCA, we define the vector $\mathbf{z}_\mathsf{c} \in \mathcal{F}^{|Q'|}$.

$$\mathbf{z}_\mathsf{c}[i] = \begin{cases} \mathbf{x}[i \bmod |Q|], & \text{if } \frac{i}{|Q|} = n \\ 0, & \text{otherwise} \end{cases}$$

Now, consider the configuration $\bar{\mathsf{c}} = (\mathbf{z}_\mathsf{c}, (p, n))$ of $\mathcal{A}^t$ and check whether $\bar{\mathsf{c}} \xrightarrow{*} \overline{\mathcal{U}} \times S \times \{0\}$. This is a co-VS reachability problem of weighted automata. Using Theorem 7.2, this can be solved in polynomial time.

Let $t' = max(n, (|Q| \cdot |C|)) + (|Q| \cdot |C|)^2$. For solving co-VS coverability problem when a weighted RODCA $\mathcal{A}$ with an initial configuration $\mathsf{c} = (\mathbf{z}, p, n)$, a vector space $\mathcal{V}$ and a set of counter states S are given as inputs, we consider the t'-unfolding weighted automaton $\mathcal{A}^{t'} = ((C', \delta', p'_0), (Q', \boldsymbol{\lambda}', \Delta', \boldsymbol{\eta}'_F), \Sigma)$ of $\mathcal{A}$. From Corollary 7.9, we know that the maximum counter value encountered during the run of a minimal reachability witness z for $(\mathsf{c}, \overline{\mathcal{V}}, S, \mathbb{N})$ is less than t'. We define a vector space $\mathcal{U} \subseteq \mathcal{F}^{|Q'|}$ as follows: A vector $\mathbf{x} \in \mathcal{F}^{|Q'|}$ is in $\mathcal{U}$ if there exists $\mathbf{y} \in \mathcal{V}$ and $m \in \mathbb{N}$ such that for all $i \in [0, |Q| - 1]$, $\mathbf{x}[|Q| \cdot m + i] = \mathbf{y}[i]$ and for all $m' \neq m$ and $i \in [0, |Q| - 1]$, $\mathbf{x}[|Q| \cdot m' + i] = 0$. Given a configuration $\mathsf{c} = (\mathbf{x}, p, n)$ of a weighted RODCA, we define the vector $\mathbf{z}_{\mathsf{c}} \in \mathcal{F}^{|Q'|}$.

$$\mathbf{z}_{\mathsf{c}}[i] = \begin{cases} \mathbf{x}[i \bmod |Q|], \text{ if } \frac{i}{|Q|} = n \\ 0, \text{ otherwise} \end{cases}$$

Now, consider the configuration $\bar{\mathsf{c}} = (\mathbf{z}_{\mathsf{c}}, (p, n))$ of $\mathcal{A}^{t'}$ and check whether $\bar{\mathsf{c}} \xrightarrow{*} \overline{\mathcal{U}} \times S \times \{0\}$. This is a co-VS reachability problem of a weighted automaton. From Theorem 7.2, we know that this can be solved in polynomial time. □

7.2.3 Binary co-VS Reachability and Coverability

In this section, we consider the case where the counter values to the co-VS reachability and co-VS coverability are specified in binary notation. Theorem 7.11 can still be applied to get an algorithm whose running time is polynomial in the input counter values. But, since the counter values are represented in binary, their values can be exponentially large compared to their size. Therefore, we only get an exponential time algorithm for reachability from Theorem 7.11. This section shows that co-VS reachability can be tested in NP even if the counter values are specified in binary notation. The technically challenging part of the proof is proved in Lemma 7.12 and Lemma 7.13.

7.2.3.1 Length Lexicographically Minimal Witness

This section will show that the lexicographically minimal witness has a distinct structure. We assume a total order on the symbols in Σ. Given two words $u, v \in \Sigma^*$, we say that u precedes v in the *length lexicographical ordering* if $|u| < |v|$ or if $|u| = |v|$ and there exists an $i \in [0, |u| - 1]$ such that $u[0, i-1] = v[0, i-1]$ and $u[i]$ precedes $v[i]$ in the total ordering assumed on Σ. A word $z \in \Sigma^*$ is called the *length lexicographically minimal* reachability witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, if $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ and for all $u \in \Sigma^* \setminus \{z\}$ with $\mathsf{c} \xrightarrow{u} \overline{\mathcal{V}} \times S \times \{m\}$, z precedes u in the length lexicographical ordering. We show that the length lexicographically minimal reachability witness z for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$ has a canonical form. First, we prove this for floating runs.

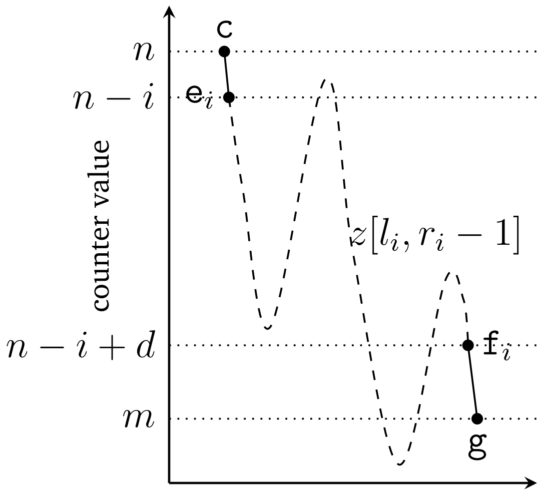


Fig. 7.4. The figure shows the floating run from a configuration c with $\text{COUNTERVALUE}(\mathsf{c}) = n$ to a configuration $\mathsf{g} = (\mathbf{x}, p, m)$ such that $\mathbf{x} \in \overline{\mathcal{V}}$. The points $\mathtt{e}_i$ and $\mathtt{f}_i$ denote the configurations where the counter values $n-i$ and $n-i+d$ are encountered for the first (resp. last) time during this run. The dashed line represents the part of the run due to factor $z[l_i, r_i - 1]$ and has a counter-effect d.

Lemma 7.12

There exist polynomials $p_1 : \mathbb{N} \to \mathbb{N}$, and $p_2 : \mathbb{N}^2 \to \mathbb{N}$ such that, if $z \in \Sigma^*$ is the length lexicographically minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$ in the weighted RODCA $\mathcal{A}$ and $\mathsf{c} \xrightarrow{z} \overline{\mathcal{V}} \times S \times \{m\}$ is a floating run in $\mathcal{A}$, then there exist $u, y, w \in \Sigma^*$ and $r \in \mathbb{N}$ such that $z = uy^r w$ and the following are true:

1. $|uyw| \leq p_1(K)$, and
2. $r \leq p_2(K, |\text{COUNTERVALUE}(\mathsf{c}) - m|)$.

Proof. Let z be the length lexicographically minimal witness for $(\mathsf{c}, \overline{\mathcal{V}}, S, \{m\})$, and $\mathsf{g} \in \overline{\mathcal{V}} \times S \times \{m\}$ such that $\mathsf{c} \xrightarrow{z} \mathsf{g}$ is a floating run. Let n be such that $\text{COUNTERVALUE}(c) = n$. We consider the case $n > m$. The case where $m \geq n$ is analogous. Let $t = n - m$ and $p_1(K) = 12K^3$.

Claim 1. $|z| \leq 2K^3 + t \cdot K$.

Proof. From Point 1 of Lemma 7.5, it follows that the maximum counter value during the run $\mathsf{c} \xrightarrow{z} \mathsf{g}$ is less than $n + K^2$. From Lemma 7.6, we get that the minimum counter value during the run is greater than $m - K^2$. Hence, there are at most $t + 2K^2$ distinct counter values during the run. From Lemma 7.4 it follows that $|z| \leq 2K^3 + t \cdot K$. $\square_{Claim:1}$

If $t \leq K^2$, then from Claim 1, we get that $|z| \leq 3K^3$, and the lemma is trivially true. Let us assume $t > K^2$ and let $d = K^2 - t$. Note that d is a negative number. Let $\mathtt{c}_1 = \mathtt{c}$ and $\pi(z, \mathtt{c}_1) = \mathtt{c}_1\tau_1\mathtt{c}_2 \cdots \tau_{\ell-1}\mathtt{c}_\ell$ denote the run on word z from $\mathtt{c}$. For any $i \in [0, K^2]$, we denote by l_i the index such that the counter value $n - i$ is encountered for the first time, and r_i the index such that the counter value $n - i + d$ is encountered for the last time in $\pi(z, \mathtt{c}_1)$ (see Figure 7.4). i.e., for all $i \in [0, K^2]$, $m - \mathtt{ce}(T_{1\cdots l_i-1}) = n - i$ and for all $j < l_i - 1$, $m - \mathtt{ce}(T_{1\cdots j}) > n - i$. Similarly, for all $i \in [0, K^2]$, $m - \mathtt{ce}(T_{1\cdots r_i-1}) = n - i$ and for all $j > r_i - 1$, $m - \mathtt{ce}(T_{1\cdots j}) < n - i$. Let $I = \{(l_i, r_i)\}_{i\in[0,K^2]}$ be the set of these pairs of indices, and let $W = \{z[l, r-1] \mid (l, r) \in I\}$ be the set of corresponding factors. Note that $|I| > K^2$. We argue that these factors $z[l_i, r_i - 1]$ for $i \in [0, K^2]$ cannot be pairwise distinct.

Claim 2. $|W| \leq K^2$.

Proof. Assume for contradiction that $|W| > (|Q| \cdot |C|)^2$. Since the number of counter states is $|C|$, by the pigeonhole principle there exists $Y \subseteq I$ with $|Y| = |Q|^2 + 1$ such that for all $(l, r), (l', r') \in Y$, configurations $\mathtt{c}_l$ and $\mathtt{c}_{l'}$ have the same counter state, configurations $\mathtt{c}_r$ and $\mathtt{c}_{r'}$ have the same counter state, and $z[l, r-1] \neq z[l', r'-1]$. We say $(l, r) < (l', r')$ if $z[l, r-1]$ precedes $z[l', r'-1]$ in the length lexicographical order. Therefore, the elements in Y have an ordering as follows: $(l_0, r_0) < (l_1, r_1) < \cdots < (l_{|Q|^2}, r_{|Q|^2})$. For any configuration $\mathtt{h}$, let $\mathbf{x}_\mathtt{h} = \textsc{WeightVector}(\mathtt{h})$. For all $i \in [0, |Q|^2]$, let $u_i = z[1, l_i - 1], x_i = z[l_i, r_i - 1], w_i = z[r_i, \ell - 1]$, configurations $\mathtt{e}_i, \mathtt{f}_i$ be such that $\mathtt{c} \xrightarrow{u_i} \mathtt{e}_i \xrightarrow{x_i} \mathtt{f}_i \xrightarrow{w_i} \mathtt{g}$ and matrices $\mathbb{A}_i, \mathbb{M}_i, \mathbb{B}_i$ be such that $\mathbf{x}_{\mathtt{e}_i} = \mathbf{x}_\mathtt{c}\mathbb{A}_i$, $\mathbf{x}_{\mathtt{f}_i} = \mathbf{x}_{\mathtt{e}_i}\mathbb{M}_i$, $\mathbf{x}_\mathtt{g} = \mathbf{x}_{\mathtt{f}_i}\mathbb{B}_i$.

We know that for all $k \in [0, |Q|^2]$, $\mathbf{x}_\mathtt{c}\mathbb{A}_k\mathbb{M}_k\mathbb{B}_k \in \overline{\mathcal{V}}$. Consider the sequence of matrices $\mathbb{M}_0, \mathbb{M}_1, \cdots, \mathbb{M}_{|Q|^2}$. Since there can be at most $|Q|^2$ independent matrices, we get that there exists $i \in [0, |Q|^2]$ such that $\mathbb{M}_i$ is a linear combination of $\mathbb{M}_0, \ldots, \mathbb{M}_{i-1}$. Hence, we get that there exists a j where $j < i$ such that $\mathbf{x}_\mathtt{c}\mathbb{A}_i\mathbb{M}_j\mathbb{B}_i \in \overline{\mathcal{V}}$. Since $x_j = z[l_j, r_j - 1]$ precedes $x_i = z[l_i, r_i - 1]$, the word $u_i x_j w_i$

precedes z in the length lexicographical ordering. Therefore the run $\pi(u_i x_j w_i, \mathsf{c})$ contradicts the length lexicographical minimality of z. $\square_{Claim:2}$

Since $|W| \leq K^2$ and $|I| > K^2$, there exists $i, j \in [0, K^2]$, with $i < j$ and $x \in \Sigma^*$ such that $(l_i, r_i) \in I, (l_j, r_j) \in I$ and $x = z[l_i, r_i - 1] = z[l_j, r_j - 1]$ (see Figure 7.5). Let $u_1, w_1, u_2, w_2 \in \Sigma^*$ such that $z = u_1 x w_1 = u_2 x w_2$. Since $u_1 \neq u_2$, either u_1 is a prefix of u_2 or u_2 a prefix of u_1. Without loss of generality, let us assume u_1 is a prefix of u_2. Therefore, there exists $v \in \Sigma^*$ such that $u_2 = u_1 v$. Let e be a configuration such that $\mathsf{c} \xrightarrow{u_1} \mathsf{e}$.

Claim 3. $|u_1|, |v|, |w_1| \leq 3K^3$.

Proof. Consider the set I. For any $i, j \in [0, K^2]$, the difference between the counter values of configurations c_{l_i} and c_{l_j} and the difference between the counter values of the configurations c_{r_j} and c_{r_i} is at most $K^2 + 1$. Therefore the counter-effect of u_2, w_2, and v can be at most K^2. Since $\pi(v, \mathsf{e})$ is a floating run from Claim 1, we get that $|v| \leq 3K^3$. By similar arguments, the counter-effect of u_1 and w_1 can be at most K^2, and again by Claim 1, we get that their lengths are at most $3K^3$. $\square_{Claim:3}$

Claim 4. *There exist $v' \in \Sigma^*$ and $r \in [0, K^3 + t \cdot K]$ such that $x = v^r v'$ with $|v'| \leq |v|$.*

Proof. Let $r \in \mathbb{N}$ be the largest number such that x is of the form $v^r v'$ for some $v' \in \Sigma^*$ (see Figure 7.5). We know that $z = u_2 x w_2$ and $u_2 = u_1 v$. Therefore, $z = u_1 v x w_2 = u_1 v v^r v' w_2 = u_1 v^r v v' w_2$. Furthermore, $z = u_1 x w_1 = u_1 v^r v' w_1$. Now since $u_1 v^r v v' w_2 = u_1 v^r v' w_1$, we get that $v v' w_2 = v' w_1$. Hence, if $|v'| \geq |v|$, then v is a prefix of v'. This is a contradiction since r was chosen to be the largest number such that x is of the form $v^r v'$.

To show the bound on the value r, we observe the following. We know that the counter-effect of the run $\pi(x, \mathsf{e})$ is d. Therefore from Claim 1, we get that $|x| \leq 2K^3 + |d| \cdot K$. Hence, $r \leq 2K^3 + |d| \cdot K$. $\square_{Claim:4}$

From Claim 4 and Claim 3, we get that $|u_1 v v' w_1| \leq 12K^3$ and $z = u_1 v^r v' w_1$ for some $r \in [0, 2K^3 + |d| \cdot K)]$. $\square$

We now establish that the length lexicographically minimal witness z (whose run need not be floating) for a co-VS reachability problem has the form $u y_1^{r_1} v y_2^{r_2} w$.

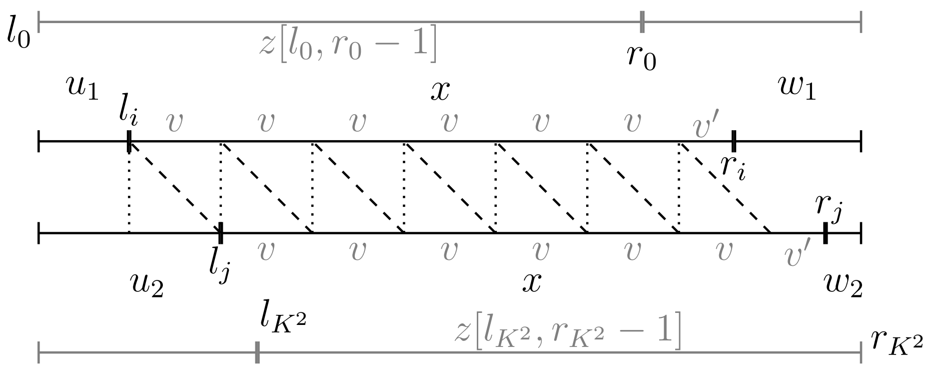


Fig. 7.5. The figure shows the factorisation of a word $z = u_1 x w_1 = u_2 x w_2$, where $x = z[l_i, r_i - 1] = z[l_j, r_j - 1]$, and $u_1 \neq u_2$. The factor v is a prefix of x such that $u_2 = u_1 v$. The word z can be written as $u_1 v^i v' w_2$ for some $i \in \mathbb{N}$ and v' prefix of v. For $k \in [0, K^2]$, l_k is the index such that the counter value $n - k$ is encountered for the first time and r_k the index such that the counter value $n - k + d$ is encountered for the last time during the run $\mathtt{c} \xrightarrow{z} \mathtt{g}$.

Here, lengths of the words u, y_1, y_2, v, and w are polynomially bounded in the number of states, and r_1 and r_2 are polynomial values dependent on the number of states and the input counter values.

Lemma 7.13 (Special-word lemma)

If $z \in \Sigma^*$ is the length lexicographically minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ in the weighted RODCA $\mathcal{A}$, then there exists $u, y_1, v_1, v_2, v_3, y_2, w \in \Sigma^*$ and $r_1, r_2 \in \mathbb{N}$ such that $z = u y_1^{r_1} v y_2^{r_2} w$ and the following are true:

1. $|u y_1 v y_2 w|$ is polynomially bounded in $|\mathcal{A}|$.

2. r_1 and r_2 are polynomially bounded in $|\mathcal{A}|$, m, and $\textsc{CounterValue}(\mathtt{c})$.

Proof. Let $z \in \Sigma^*$ be the length lexicographically minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$, where $\mathtt{c}$ is a configuration with counter value n. Consider the run of word z from $\mathtt{c}$. Let $\mathtt{d} \in \overline{\mathcal{V}} \times S \times \{m\}$ such that $\mathtt{c} \xrightarrow{z} \mathtt{d}$. Let $\mathtt{c} = \mathtt{c}_1$ and $T(\mathtt{c}_1) = \mathtt{c}_1 \tau_1 \mathtt{c}_2 \cdots \tau_{\ell-1} \mathtt{c}_\ell$ denote the run on word z from the configuration $\mathtt{c}_1$ and T the corresponding sequence of transitions. Let $\mathtt{e}_1$ be the first configuration with counter value zero and $\mathtt{e}_2$ be the last configuration with counter value zero during this run. Let $z_1, z_2, z_3 \in \Sigma^*$ be such that $\mathtt{c} \xrightarrow{z_1} \mathtt{e}_1 \xrightarrow{z_2} \mathtt{e}_2 \xrightarrow{z_3} \mathtt{c}_\ell$ and $z = z_1 z_2 z_3$. Observe that $\mathtt{c} \xrightarrow{z_1} \mathtt{e}_1$ and $\mathtt{e}_2 \xrightarrow{z_3} \mathtt{c}_\ell$ are floating runs.

From Lemma 7.12, we know that there exists $u_1, u_3, v_1, v_3, y_1, y_3 \in \Sigma^*$ and $r_1, r_3 \in \mathbb{N}$ such that $z_1 = u_1 y_1^{r_1} v_1$, $z_3 = u_3 y_3^{r_3} v_3$, $|u_1 y_1 v_1 u_3 y_3 v_2| \leq 2 \cdot p_1(|Q| \cdot |C|)$, $r_1 \leq p_2(|Q| \cdot |C|, n)$, and $r_3 \leq p_2(|Q| \cdot |C|, m)$. Also, from Corollary 7.7 we get that $|z_2| \leq (|Q| \cdot |C|)^3$. □

From Lemma 7.13, we get that the length lexicographically minimal reachability witness z for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$ is of the form $u y_1^{r_1} v y_2^{r_2} w$, where the length of the words u, y_1, y_2, v and w are polynomially bounded in K, and r_1, r_2 are polynomially bounded by in K and the input counter values. A nondeterministic machine guesses the words u, y_1, y_2, v, and w and verify reachability in polynomial time. We give a formal proof below.

Theorem 7.14

Binary co-VS reachability and co-VS coverability problems are in NP.

Proof. Let us first look at the binary co-VS reachability problem. Let $z \in \Sigma^*$ be the length lexicographically minimal reachability witness for $(\mathtt{c}, \overline{\mathcal{V}}, S, \{m\})$. Consider the run of the word z from $\mathtt{c}$. Let $\mathtt{d} \in \overline{\mathcal{V}} \times S \times \{m\}$ such that $\mathtt{c} \xrightarrow{z} \mathtt{d}$. Let $\mathtt{c} = \mathtt{c}_1$ and $T(\mathtt{c}_1) = \mathtt{c}_1 \tau_1 \mathtt{c}_2 \cdots \tau_{\ell-1} \mathtt{c}_\ell$ denote the run on word z from the configuration $\mathtt{c}_1$ and T the corresponding sequence of transitions. Let $\mathtt{e}_1$ be the first configuration with counter value zero and $\mathtt{e}_2$ be the last configuration with counter value zero during this run. Let $z_1, z_2, z_3 \in \Sigma^*$ be such that $z = z_1 z_2 z_3$ and $\mathtt{c} \xrightarrow{z_1} \mathtt{e}_1 \xrightarrow{z_2} \mathtt{e}_2 \xrightarrow{z_3} \mathtt{c}_\ell$. Observe that $\mathtt{c} \xrightarrow{z_1} \mathtt{e}_1$ and $\mathtt{e}_2 \xrightarrow{z_3} \mathtt{c}_\ell$ are floating runs.

From Lemma 7.12, we get the existence of polynomials $p_1 : \mathbb{N} \to \mathbb{N}$, and $p_2 : \mathbb{N}^2 \to \mathbb{N}$ such that, there exist $u_1, u_3, v_1, v_3, y_1, y_3 \in \Sigma^*$ and $r_1, r_3 \in \mathbb{N}$ satisfying the following conditions: $z_1 = u_1 y_1^{r_1} v_1$, $z_3 = u_3 y_3^{r_3} v_3$, $|u_1 y_1 v_1|, |u_3 y_3 v_3| \leq p_1(K)$, $r_1 \leq p_2(K, \textsc{CounterValue}(\mathtt{c}))$ and $r_2 \leq p_2(K, m)$. Also, from Corollary 7.7 we get that $|z_2| \leq K^6$.

Our NP algorithm starts by guessing the words $u_1, y_1, v_1, z_2, u_3, y_3, v_3$, the values r_1, r_2, and the configurations $\mathtt{e}_1$ and $\mathtt{e}_2$. We first show how to verify if $\mathtt{c} \xrightarrow{u_1 y_1^{r_1} v_1} \mathtt{e}_1$. The algorithm computes configuration $\mathtt{f}_0$ such that $\mathtt{c} \xrightarrow{u_1} \mathtt{f}_0$. Now it constructs the matrix $\mathbb{M}_{y_1}$ and computes the configuration $\mathtt{f}_1$ such that $\mathtt{f}_0 \xrightarrow{y_1} \mathtt{f}_1$ and $\textsc{WeightVector}(\mathtt{f}_1) = \textsc{WeightVector}(\mathtt{f}_0)\mathbb{M}_{y_1}$. From linear algebra we know that $(\mathbb{M}_{y_1})^{r_1}$ can be computed by repeated powering in time polynomial in $\log(r_1)$ and K. Let $\mathtt{f}_{r_1}$ be a configuration such that $\mathtt{f}_0 \xrightarrow{y^{r_1}} \mathtt{f}_{r_1}$. From Lemma 7.12, we know that $\textsc{CounterState}(\mathtt{f}_0) = \textsc{CounterState}(\mathtt{f}_{r_1})$ and $\textsc{CounterValue}(\mathtt{f}_{r_1}) = \textsc{CounterValue}(\mathtt{f}_0) - r_1 \cdot (\textsc{CounterValue}(\mathtt{f}_0) - \textsc{CounterValue}(\mathtt{f}_1))$. Since we know that $\textsc{WeightVector}(\mathtt{f}_{r_1}) = \textsc{WeightVector}(\mathtt{f}_0)(\mathbb{M}_{y_1})^{r_1}$, we can construct the configuration $\mathtt{f}_{r_1}$ in polynomial time. We now verify in polynomial time whether $\mathtt{f}_{r_1} \xrightarrow{v_1} \mathtt{e}_1$ or not. We can verify if $\mathtt{e}_2 \xrightarrow{u_3 y_3^{r_3} v_3} \mathtt{d}$ in a similar manner. The algorithm can also check whether $\mathtt{e}_1 \xrightarrow{z_2} \mathtt{e}_2$ in polynomial time since $|z_2| \leq K^6$. It finally checks whether $\mathtt{d} \in \overline{\mathcal{V}} \times S \times \{m\}$. Hence the binary co-VS reachability problem is decidable in NP.

As for the binary co-VS coverability problem, either the run of a minimal witness is a floating run or is not. In the former case where the run is non-floating, from Lemma 7.8, we know that the difference between the final and initial counter values is at most K^2. In the latter case where the run is non-floating, by Lemma 7.8, we get that the final value is at most K^2. In both the cases, the algorithm guesses the final counter value, and the problem is reduced to the binary co-VS reachability problem, which is in NP. Hence the binary co-VS coverability problem is decidable in NP. □

7.3 Equivalence

In this section, we present a polynomial time algorithm to decide the equivalence of two weighted RODCAs whose weights come from a fixed field.

> EQUIVALENCE PROBLEM
> INPUT: Two weighted RODCAs $\mathcal{A}$ and $\mathcal{B}$ over fields.
> OUTPUT: *Yes*, if $\mathcal{A}$ and $\mathcal{B}$ are equivalent. *No*, otherwise.

The techniques developed in the previous section, in conjunction with those presented by Valiant and Paterson (1975) and Böhm and Göller (2011) for DROCAs, give us the algorithm. The idea here is to prove that the maximum counter value encountered during the run of a minimal witness is polynomially bounded. We use this to reduce the equivalence problem to that of weighted automata.

In the remainder of this section, we fix two non-equivalent weighted RODCAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over an alphabet Σ and a field $\mathcal{F}$. For $i \in \{1, 2\}$,

$$\mathcal{A}_i = ((C_i, \delta_{0_i}, \delta_{1_i}, p_{0_i}), (Q_i, \boldsymbol{\lambda}_i, \Delta_i, \boldsymbol{\eta}_i), \Sigma).$$

Without loss of generality assume $\mathsf{K} = |C_1| = |Q_1| = |C_2| = |Q_2|$. We will reason on the synchronised runs on pairs of configurations. Given two weighted RODCAs, $\mathcal{A}_1$ and $\mathcal{A}_2$ and $i \in \mathbb{N}$, we denote a *configuration pair* as $\mathtt{h}_i = \langle \mathtt{c}_i, \mathtt{d}_i \rangle$ where $\mathtt{c}_i$ is a configuration of $\mathcal{A}_1$ and $\mathtt{d}_i$ is a configuration of $\mathcal{A}_2$. We similarly consider *transition pairs* of $\mathcal{A}_1$ and $\mathcal{A}_2$, and consider *synchronised runs* as the application of a sequence of transition pairs to a configuration pair. A word is called a *witness*, if it is accepted with different weights by $\mathcal{A}_1$ and $\mathcal{A}_2$. We fix a minimal word z, also called a *minimal witness*, that is accepted with different weights in $\mathcal{A}_1$ and $\mathcal{A}_2$ and use ℓ to denote $|z|$. Henceforth we will denote by

$$\Pi = \mathtt{h}_0 \tau_0 \mathtt{h}_1 \cdots \tau_{\ell-1} \mathtt{h}_\ell$$

the synchronisation of runs over z in $\mathcal{A}_1$ and $\mathcal{A}_2$ from their initial configurations, where $\mathtt{h}_i$ are pairs of configurations and τ_i are pairs of transitions. We denote by $T = \tau_0 \cdots \tau_{\ell-1}$ the sequence of transition pairs of this run pair. The main idea to prove Theorem 7.1 is to show that the length of z is polynomially bounded in the size of the two weighted RODCAs.

Lemma 7.15

There is a polynomial $\mathrm{poly}_0 : \mathbb{N} \to \mathbb{N}$ such that if two weighted RODCAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent, then there exists a witness z such that the counter values encountered during Π are less than $\mathrm{poly}_0(\mathsf{K})$.

We use Lemma 7.15 to show that the length of a minimal witness z is bounded by a polynomial $\mathrm{poly}_1(\mathsf{K}) = 2\mathsf{K}^5 \mathrm{poly}_0(\mathsf{K})$.

Lemma 7.16
There is a polynomial $\text{poly}_1 : \mathbb{N} \to \mathbb{N}$ such that if two weighted RODCAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent, then there exists a witness z such that $|z|$ is less than or equal to $\text{poly}_1(\mathsf{K})$.

Proof. Assume for contradiction that the length of a minimal witness z is greater than $\text{poly}_1(\mathsf{K})$. From Lemma 7.15, we know that the counter values encountered during the run Π in less than $\text{poly}_0(\mathsf{K})$. Since $|z| > \text{poly}_1(\mathsf{K})$, by the pigeonhole principle, we get that there exist indices $0 \leq i_0 < i_2 < \cdots < i_{2\mathsf{K}} \leq \ell$ such that for all configuration pairs $\mathtt{h}_{i_j}, j \in [1, 2\mathsf{K}]$, $\mathtt{c}_{i_j}$ and $\mathtt{c}_{i_{j-1}}$ have the same counter value and counter state and $\mathtt{d}_{i_j}$ and $\mathtt{d}_{i_{j-1}}$ have the same counter state and counter value.

For all $j \in [0, 2\mathsf{K}]$ we define the vector $\mathbf{x}_j \in \mathcal{F}^{2\mathsf{K}}$ such that $\mathbf{x}_j[r] = \mathbf{x}_{\mathtt{c}_{i_j}}[r]$, if $r < \mathsf{K}$ and $\mathbf{x}_{\mathtt{d}_{i_j}}[r - \mathsf{K}]$, otherwise. We also define the vector $\boldsymbol{\eta} \in \mathcal{F}^{2\mathsf{K}}$ such that $\boldsymbol{\eta}[r] = \boldsymbol{\eta}_1[r]$, if $r < \mathsf{K}$ and $\boldsymbol{\eta}_2[r - \mathsf{K}]$, otherwise. For all $j \in [0, 2\mathsf{K}]$, let $\mathbb{A}_j$ denote the matrix such that $\mathbf{x}_j \mathbb{A}_j = \mathbf{x}_\ell$. Since z is a minimal witness, we know that for all $j \in [0, 2\mathsf{K}]$, $\mathbf{x}_j \mathbb{A}_j \boldsymbol{\eta}^\top \neq 0$. From Lemma 2.3, we get that there exists $r, r' \in [0, 2\mathsf{K}]$, with $r' < r$ such that $\mathbf{x}_{r'} \mathbb{A}_r \boldsymbol{\eta}^\top \neq 0$. The sequence of transitions $\tau_{i_r+1} \cdots \tau_\ell$ can be taken from $\mathtt{h}_{i'_r}$ since the counter values and counter states are the same for both configurations. Consider the sequence of transitions $T' = \tau_0 \cdots \tau_{i'_r} \tau_{i_r+1} \cdots \tau_\ell$ and let $w = \mathtt{word}(T')$. The word w is a shorter witness than z and contradicts its minimality. $\square$

Thus we can reduce the equivalence problem of weighted RODCAs over fields to that of weighted automata over fields (which is in P (Tzeng, 1992)) by "simulating" the runs of weighted RODCAs $\mathcal{A}_1$ and $\mathcal{A}_2$ up to length $\text{poly}_1(\mathsf{K})$ by two weighted automata. The naive algorithm will only give us a PSPACE procedure, but there is a polynomial time procedure to do this, and the proof is given below.

Proof of Theorem 7.1. We consider the two weighted RODCAs $\mathcal{A}_1$ and $\mathcal{A}_2$. From Lemma 7.16, we know that the length of the minimal witness z is less than $\text{poly}_1(\mathsf{K})$. Let $M = \text{poly}_1(\mathsf{K})$. We construct the M-unfolding weighted automata $\mathcal{A}_1^M$ and $\mathcal{A}_2^M$ as described in Definition 6.8. Let $\mathcal{A}_1^M = (Q, \Sigma, \boldsymbol{\lambda}, \delta, \boldsymbol{\eta})$ and $\mathcal{A}_2^M = (Q', \Sigma, \boldsymbol{\lambda}', \delta', \boldsymbol{\eta}')$. It follows that, $\mathcal{A}_1$ is not equivalent to $\mathcal{A}_2$ if and only if there exists a word $w \in \Sigma^{\leq M}$ such that $f_{\mathcal{A}_1^M}(w) \neq f_{\mathcal{A}_2^M}(w)$. We conclude the proof by showing that this can be reduced to a co-VS reachability problem of a weighted automaton.

Consider the weighted automata $\mathcal{B}_2^M = (Q', \Sigma, \boldsymbol{\lambda}'', \delta', \boldsymbol{\eta}')$, where $\boldsymbol{\lambda}'' = -1 \times \boldsymbol{\lambda}'$. Let $\mathcal{D}$ be the weighted automata obtained by taking the disjoint union of $\mathcal{A}_1^M$ and $\mathcal{B}_2^M$. Now since $f_{\mathcal{D}}(u) = f_{\mathcal{A}_1^M}(u) - f_{\mathcal{A}_2^M}(u)$ for all $u \in \Sigma^*$, if $f_{\mathcal{D}}(w) \neq 0$ for some $w \in \Sigma^*$, then $f_{\mathcal{A}_1^M}(w) \neq f_{\mathcal{A}_1^M}(w)$. Let $\boldsymbol{\eta}_F$ denote the output distribution of $\mathcal{D}$. From Theorem 7.2, there is an algorithm (see Algorithm 5) to check if $f_{\mathcal{D}}(u) \neq 0$ for some $u \in \Sigma^*$ when provided with the input $\mathcal{D}$ and the vector space $\{\mathbf{x} \mid \mathbf{x}.\boldsymbol{\eta}_F = 0\}$ that runs in polynomial time with respect to $|\mathcal{D}|$. $\square$

The rest of this section is dedicated to proving Lemma 7.15.

7.3.1 Configuration Space

Each pair of configuration $\mathtt{h} = \langle \mathtt{c}, \mathtt{d} \rangle$ is mapped to a point in the space $\mathbb{N} \times \mathbb{N} \times (C_1 \times C_2) \times \mathcal{F}^{\mathsf{K}} \times \mathcal{F}^{\mathsf{K}}$, henceforth referred to as the configuration space. Here, the first two dimensions represent the two counter values, the third dimension $C_1 \times C_2$ corresponds to the pair of counter states, and the remaining dimensions represent the weight vector. We partition the configuration space into three: initial space, belt space, and background space. The size of the initial space and thickness and number of belts will be polynomially bounded in K. These partitions are indexed on two carefully chosen polynomials $\mathrm{poly}_2(\mathsf{K}) = 516\mathsf{K}^{21}$ and $\mathrm{poly}_3(\mathsf{K}) = 42\mathsf{K}^{14}$, so that all belts are disjoint outside the initial space. The precise polynomials are required in the proofs of Lemma 7.17 and Lemma 7.23. We use some properties of these partitions to show that the length of a minimal witness is bounded. The projection of the configuration space onto the first two dimensions is depicted in Figure 7.6. The figure is identical to Figure 5.1 in Chapter 5, except that the polynomials involved are different. However, we include it here for clarity. Given a configuration $\mathtt{c}$, we use $n_{\mathtt{c}}$ to denote $\textsc{CounterValue}(\mathtt{c})$.

- initial space: All configuration pairs $\langle \mathtt{c}, \mathtt{d} \rangle$ such that $n_{\mathtt{c}}, n_{\mathtt{d}} < \mathrm{poly}_2(\mathsf{K})$.
- belt space: Let $\alpha, \beta \in [1, 3\mathsf{K}^7]$ be co-prime. A belt of slope $\frac{\alpha}{\beta}$ consists of those configuration pairs $\langle \mathtt{c}, \mathtt{d} \rangle$ outside the initial space that satisfies

$|\alpha.n_{\mathtt{c}} - \beta.n_{\mathtt{d}}| \leq \mathrm{poly}_3(\mathsf{K})$. The belt space contains all configuration pairs $\langle \mathtt{c}, \mathtt{d} \rangle$ that are inside belts with slope $\frac{\alpha}{\beta}$.

- background space: All remaining configuration pairs.

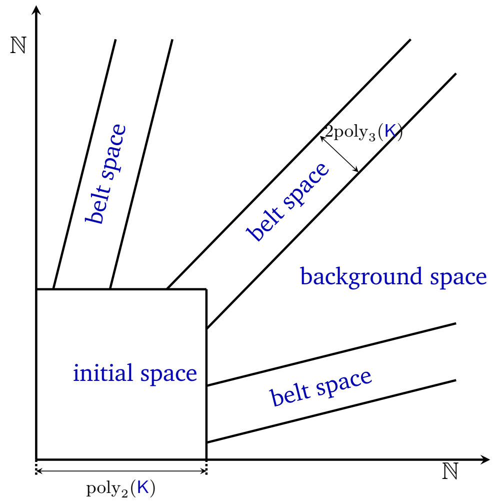


Fig. 7.6. Projection of configuration space.

The proof of the following lemma is similar to that of the non-weighted case presented by Böhm and Göller (2011). It shows that all belts are disjoint outside the initial space.

Lemma 7.17
If $\langle \mathtt{c}, \mathtt{d} \rangle$ and $\langle \mathtt{e}, \mathtt{f} \rangle$ are configuration pairs inside two distinct belts and lie outside the initial space, then there is no $a \in \Sigma$ such that $\langle \mathtt{c}, \mathtt{d} \rangle \xrightarrow{a} \langle \mathtt{e}, \mathtt{f} \rangle$.

Proof. Recall $\mathrm{poly}_2(\mathsf{K}) = 516\mathsf{K}^{21}$ and $\mathrm{poly}_3(\mathsf{K}) = 42\mathsf{K}^{14}$. Let B and B' be two distinct belts with μ being the slope of the belt B and μ' the slope of the belt B'. Hence $\mu \neq \mu'$. Without loss of generality, let us assume that $\mu' > \mu$. It suffices to show that for all $x > \mathrm{poly}_2(\mathsf{K})$, we have

$$\mu x + \mathrm{poly}_3(\mathsf{K}) + 1 < \mu' x - \mathrm{poly}_3(\mathsf{K}) - 1.$$

We know that $\mu' - \mu \geq \frac{1}{3\mathsf{K}^7}$ and $x > 516\mathsf{K}^{21}$.

$$\text{Therefore, } \frac{516\mathsf{K}^{21}}{6\mathsf{K}^7} < (\mu' - \mu) \cdot x.$$

$$\implies \mu x + \frac{86\mathsf{K}^{14}}{2} < \mu' x - \frac{86\mathsf{K}^{14}}{2}$$
$$\implies \mu x + 42\mathsf{K}^{14} + \mathsf{K}^{14} < \mu' x - 42\mathsf{K}^{14} - \mathsf{K}^{14}$$
$$\implies \mu x + 42\mathsf{K}^{14} + 1 < \mu' x - 42\mathsf{K}^{14} - 1.$$

□

Lemma 7.17 ensures that the belts are disjoint outside the initial space and that no run can go from one belt to another without passing through the initial space or background space. To prove Lemma 7.15, there are two cases to consider: either there is no background space point in Π, or there is a background space point in Π.

7.3.2 Case 1: Minimal Witness Does Not Enter the Background Space

Since there is no background space point in Π, all the points in Π are either in the initial or belt space. The counter values of configuration pairs inside the initial space are bounded by $\text{poly}_2(\mathsf{K})$. Now, we look at the sub-run of Π inside the belt space. If a sub-run of Π enters and exits a belt from the initial space or if Π ends inside a belt, then we show that the counter values encountered during that belt visit are polynomially bounded. This is shown by reducing to co-VS reachability of a weighted RODCA. For this proof, it is crucial that the belts are disjoint.

Let $\Pi_b = \mathtt{h}_i \tau_i \mathtt{h}_{i+1} \cdots \tau_{j-1} \mathtt{h}_j$ be a sub-run of the run of z inside a belt with slope $\frac{\alpha}{\beta}$ for $\alpha, \beta \in [1, 3\mathsf{K}^7]$. Given a belt with slope $\frac{\alpha}{\beta}$, let $L = \{\alpha n - \beta n' = d \mid |d| \leq 42\mathsf{K}^{14}\}$ be a set of lines with slope $\frac{\alpha}{\beta}$ inside the given belt. Similar to the technique mentioned by Böhm et al. (2013), each configuration pair $\mathtt{h}_r = ((\mathbf{x}_{\mathtt{c}_r}, p_{\mathtt{c}_r}, n_{\mathtt{c}_r}), (\mathbf{x}_{\mathtt{d}_r}, p_{\mathtt{d}_r}, n_{\mathtt{d}_r}))$, where $r \in [i, j]$ can alternatively be presented as $((\mathbf{x}_{\mathtt{c}_r}, \mathbf{x}_{\mathtt{d}_r}), p_{\mathtt{c}_r}, p_{\mathtt{d}_r}, l_r)$ where $l_r \in L$ denotes a line that contains the point $(n_{\mathtt{c}_r}, n_{\mathtt{d}_r})$. Note that $|L| = 2\text{poly}_3(\mathsf{K})$. The run Π_b is similar to the run of a weighted RODCA $\mathcal{D}_{\frac{\alpha}{\beta}}$ that has the tuple $(p_{\mathtt{c}_r}, p_{\mathtt{d}_r}, l_r)$ as the state of the finite state machine and $\mathbf{x}_r \in \mathcal{F}^{2\mathsf{K}}$ as its weight vector where $\mathbf{x}_r[i] = \mathbf{x}_{\mathtt{c}_r}[i]$, if $i < \mathsf{K}$ and $\mathbf{x}_r[i] = \mathbf{x}_{\mathtt{d}_r}[i - \mathsf{K}]$, otherwise. A formal definition of the weighted RODCA $\mathcal{D}_{\frac{\alpha}{\beta}}$ is given in Definition 7.18.

Definition 7.18 (Belt Machine)

Let $\mathcal{A}_i = ((C_i, \delta_{0_i}, \delta_{1_i}, p_{0_i}), (Q_i, \boldsymbol{\lambda}_i, \Delta_i, \boldsymbol{\eta}_i), \Sigma)$ for $i \in \{1, 2\}$, be two weighted RODCAs, $\alpha, \beta \in [1, 3\mathsf{K}^7]$, and $L = \{\alpha n - \beta n' = d \mid |d| \leq 42\mathsf{K}^{14}\}$ be a set of lines with slope $\frac{\alpha}{\beta}$. We define the weighted RODCA $\mathcal{D}_{\frac{\alpha}{\beta}} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}), \Sigma)$, where the initial state p_0 and the initial distribution $\boldsymbol{\lambda}$ are arbitrarily chosen.

- $C = C_1 \times C_2 \times L$ is a non-empty finite set of states.
- $\delta_1 : C \times \Sigma \to C \times \{-1, 0, +1\}$ is the deterministic counter transition. Let $p_1, q_1 \in C_1, p_2, q_2 \in C_2$, $a \in \Sigma$ and $d_1, d_2 \in \{-1, 0, +1\}$. Let $l_1, l_2 \in L$ and $m_1, m_2 \in \mathbb{N}$, such that the point (m_1, m_2) lies on the line l_1. $\delta_1((p_1, p_2, l_1), a) = ((q_1, q_2, l_2), d_1)$, if $\delta_{1_1}(p_1, a) = (q_1, d_1)$ and $\delta_{1_2}(p_2, a) = (q_2, d_2)$ and the point $(m_1 + d_1, m_2 + d_2)$ lies on the line l_2. It is undefined otherwise. The function $\delta_0 : C \times \Sigma \to C \times \{0, +1\}$ is arbitrarily chosen.
- $Q = Q_1 \cup Q_2$ is a non-empty finite set of states of the finite state machine.
- $\Delta : \Sigma \times \{0, 1\} \to \mathcal{F}^{2\mathsf{K} \times 2\mathsf{K}}$ gives the transition matrix for all $a \in \Sigma$ and $d \in \{0, 1\}$. For $l \in L, m \in \mathbb{N}$ and $a \in \Sigma$,

$$\Delta(l, a)[i][j] = \begin{cases} \Delta_1(a, 1)[i][j], \text{if } i, j < \mathsf{K} \\ \Delta_2(a, 1)[i - \mathsf{K}][j - \mathsf{K}], \text{if } i, j > \mathsf{K} \\ 0, \text{ otherwise} \end{cases}$$

- $\boldsymbol{\eta} \in \mathcal{F}^{2\mathsf{K}}$ is the final distribution. $\boldsymbol{\eta}[i]$ is equal to $\boldsymbol{\eta}_1[i]$, if $i < \mathsf{K}$ and is equal to $\boldsymbol{\eta}_2[i - \mathsf{K}]$, otherwise.

The sub-run Π_b inside a belt with slope $\frac{\alpha}{\beta}$ can now be seen as a floating run of the weighted RODCA $\mathcal{D}_{\frac{\alpha}{\beta}}$. We use co-VS reachability/co-VS coverability to show that if a sub-run of this weighted RODCA $\mathcal{D}_{\frac{\alpha}{\beta}}$ reaches counter values higher than some polynomial in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$, then there exists a shorter witness (contradicting minimality). We achieve this by applying pseudo-pumping lemma (Lemma 7.5) and cut lemma (Lemma 7.8) on this sub-run. Hence, we get that the pair of runs of the minimal witness cannot reach counter values higher than some polynomial bound if it does not enter the background space. If the run Π ends inside a belt, then $\Pi_b = \mathtt{h}_i \tau_i \cdots \tau_{\ell-1} \mathtt{h}_\ell$. In this case, we show that the difference between the counter values of the first and last configuration pairs is smaller than a polynomial in K.

Lemma 7.19
There is a polynomial $poly_4 : \mathbb{N} \to \mathbb{N}$, such that if $\Pi_b = \mathtt{h}_i \tau_i \cdots \tau_{\ell-1} \mathtt{h}_\ell$ lies inside a belt with slope $\frac{\alpha}{\beta}$ for $\alpha, \beta \in [1, 3\mathsf{K}^7]$, where $\mathtt{h}_r = \langle \mathtt{c}_r, \mathtt{d}_r \rangle$, for $r \in [i, \ell]$, then $|\textsc{CounterValue}(\mathtt{c}_\ell) - \textsc{CounterValue}(\mathtt{c}_i)| \leq poly_4(\mathsf{K})$ and $|\textsc{CounterValue}(\mathtt{d}_\ell) - \textsc{CounterValue}(\mathtt{d}_i)| \leq poly_4(\mathsf{K})$.

Proof. Let $\Pi_b = \mathtt{h}_i \tau_i \mathtt{h}_{i+1} \cdots \tau_{\ell-1} \mathtt{h}_\ell$ be a sub-run of the run of a minimal witness inside a belt with slop $\frac{\alpha}{\beta}$ and ends in it, where $\mathtt{h}_r = \langle (\mathbf{x}_{\mathtt{c}_r}, p_{\mathtt{c}_r}, n_{\mathtt{c}_r}), (\mathbf{x}_{\mathtt{d}_r}, p_{\mathtt{d}_r}, n_{\mathtt{d}_r}) \rangle$, for $r \in [i, \ell]$. As mentioned in Definition 7.18, we consider this as the run of the weighted RODCA $\mathcal{D}_{\frac{\alpha}{\beta}}$. Since it is the run of a witness, $\mathbf{x}_j \boldsymbol{\eta}^\top \neq 0$. Consider the vector space $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{2\mathsf{K}} \mid \mathbf{y}\boldsymbol{\eta}^\top = 0\}$. Our problem now reduces to the co-VS coverability problem in machine $\mathcal{D}_{\frac{\alpha}{\beta}}$ and asks whether $(\mathbf{x}_i, (p_{\mathtt{c}_i}, p_{\mathtt{d}_i}, l_i), n_{\mathtt{c}_i}) \xrightarrow{*} \overline{\mathcal{U}} \times \{(p_{\mathtt{c}_\ell}, p_{\mathtt{d}_\ell}, l_\ell)\} \times \mathbb{N}$ in $\mathcal{D}_{\frac{\alpha}{\beta}}$. From Corollary 7.7, we know that the length of a minimal reachability witness for $((\mathbf{x}_i, (p_{\mathtt{c}_i}, p_{\mathtt{d}_i}, l_i), n_{\mathtt{c}_i}), \overline{\mathcal{U}}, (p_{\mathtt{c}_\ell}, p_{\mathtt{d}_\ell}, l_\ell), \mathbb{N})$ is polynomially bounded in $n_{\mathtt{c}_i}$ and K. Hence proved. □

In the following lemma, we show that if $\Pi_b = \mathtt{h}_i \tau_i \mathtt{h}_{i+1} \cdots \tau_{j-1} \mathtt{h}_j$ is a sub-run of Π inside a belt and either $\textsc{CounterValue}(\mathtt{c}_i) = \textsc{CounterValue}(\mathtt{c}_j)$ or $\textsc{CounterValue}(\mathtt{d}_i) = \textsc{CounterValue}(\mathtt{d}_j)$, where $\mathtt{h}_r = \langle \mathtt{c}_r, \mathtt{d}_r \rangle$, for $r \in [i, j]$, then the counter values in Π_b cannot increase more than a polynomial in K from $\textsc{CounterValue}(\mathtt{c}_i)$ and $\textsc{CounterValue}(\mathtt{d}_i)$.

Lemma 7.20

There is a polynomial $poly_5 : \mathbb{N} \to \mathbb{N}$ such that, if $\Pi_b = \mathtt{h}_i \tau_i \mathtt{h}_{i+1} \cdots \tau_{j-1} \mathtt{h}_j$ is a run inside a belt with slope $\frac{\alpha}{\beta}$ for $\alpha, \beta \in [1, 3\mathsf{K}^7]$, with $\textsc{CounterValue}(\mathtt{c}_i) = \textsc{CounterValue}(\mathtt{c}_j)$ or $\textsc{CounterValue}(\mathtt{d}_i) = \textsc{CounterValue}(\mathtt{d}_j)$, where $\mathtt{h}_r = \langle \mathtt{c}_r, \mathtt{d}_r \rangle$, for $r \in [i, j]$, then the counter-effect of any sub-run of Π_b is less than or equal to $poly_5(K)$.

Proof. Let $\Pi_b = \mathtt{h}_i \tau_i \mathtt{h}_{i+1} \cdots \tau_{j-1} \mathtt{h}_j$ be a sub-run of the run of a minimal witness inside a belt with slope $\frac{\alpha}{\beta}$ such that $n_{\mathtt{c}_i} = n_{\mathtt{c}_j}$ and for all $r \in [i, j]$, $\mathtt{h}_r = \langle (\mathbf{x}_{\mathtt{c}_r}, p_{\mathtt{c}_r}, n_{\mathtt{c}_r}), (\mathbf{x}_{\mathtt{d}_r}, p_{\mathtt{d}_r}, n_{\mathtt{d}_r}) \rangle$. We consider this as the run of the weighted RODCA $\mathcal{D}_{\frac{\alpha}{\beta}}$ as mentioned in Definition 7.18. Since it is the run of a witness, we know that there exists $\mathbb{A} \in \mathcal{F}^{2\mathsf{K} \times 2\mathsf{K}}$ such that $\mathbf{x}_j \mathbb{A} \boldsymbol{\eta}^\top \neq 0$. Consider the vector space $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{2\mathsf{K}} \mid \mathbf{y} \mathbb{A} \boldsymbol{\eta}^\top = 0\}$.

Our problem now reduces to the co-VS reachability problem in machine $\mathcal{D}_{\frac{\alpha}{\beta}}$ and asks whether $(\mathbf{x}_i, (p_{\mathtt{c}_i}, p_{\mathtt{d}_i}, l_i), n_{\mathtt{c}_i}) \xrightarrow{*} \overline{\mathcal{U}} \times \{(p_{\mathtt{c}_j}, p_{\mathtt{d}_j}, l_j)\} \times \{n_{\mathtt{c}_i}\}$ in $\mathcal{D}_{\frac{\alpha}{\beta}}$. From Corollary 7.7, we get that the minimal reachability witness for $((\mathbf{x}_i, (p_{\mathtt{c}_i}, p_{\mathtt{d}_i}, l_i), n_{\mathtt{c}_i}), \overline{\mathcal{U}}, (p_{\mathtt{c}_j}, p_{\mathtt{d}_j}, l_j), \{n_{\mathtt{c}_i}\})$ will have its length bounded by a polynomial in $n_{\mathtt{c}_i}$ and K. Hence proved. $\square$

Hence, we get that the pair of runs of the minimal witness cannot reach counter values higher than some polynomial bound if it does not enter the background space. Now we look at the case where the run enters the background space.

7.3.3 Case 2: Minimal Witness Enters the Background Space

We now consider the case where the witness ultimately enters the background space. Using co-VS reachability, we prove that the counter values encountered during Π till the first background space point are polynomially bounded. We also show that the length of the remaining run is polynomially bounded in the number of states of the machines.

To that end, we need the notion of *underlying uninitialised weighted automaton*. Floating runs of $\mathcal{A}$ are isomorphic to runs of this weighted automaton $\mathrm{U}(\mathcal{A})$.

Definition 7.21

For $l \in \{1,2\}$, the underlying uninitialised weighted automaton of $\mathcal{A}$ is the uninitialised weighted automaton $\mathrm{U}(\mathcal{A}_l) = (Q'_l, \Delta'_l, \boldsymbol{\eta}'_l)$, where $Q'_l = C_l \times Q_l$ and $\boldsymbol{\eta}'_l \in \mathcal{F}^{\mathsf{K}^2}$ is the final distribution. For $i < \mathsf{K}^2, \boldsymbol{\eta}'_l[i] = \eta_l[i \bmod \mathsf{K}]$. The transition matrix is given by $\Delta'_l : \Sigma \to \mathcal{F}^{\mathsf{K}^2 \times \mathsf{K}^2}$. Let $a \in \Sigma$, $d \in \{-1, 0, +1\}, i, j < \mathsf{K}^2$,

$$\Delta'_l(a)[i][j] = \begin{cases} \Delta_l(p_{\frac{i}{\mathsf{K}}}, a, 1)[i \bmod \mathsf{K}][j \bmod \mathsf{K}], & \text{if } \delta_{l_1}(p_{\frac{i}{\mathsf{K}}}, a) = (p_{\frac{j}{\mathsf{K}}}, d) \\ 0 \text{ otherwise} \end{cases}$$

Given $k \in \mathbb{N}$, a configuration c of a weighted RODCA $\mathcal{A}$ is said to be k-equivalent to a configuration $\bar{\mathsf{c}}$ of an uninitialised weighted automata $\mathcal{B}$, denoted $\mathsf{c} \sim_k \bar{\mathsf{c}}$, if for all $w \in \Sigma^{\leq k}$, $f_{\mathcal{A}}(w, \mathsf{c}) = f_{\mathcal{B}}(w, \bar{\mathsf{c}})$. We say that c is not k-equivalent to $\bar{\mathsf{c}}$ otherwise and denote this as $\mathsf{c} \not\sim_k \bar{\mathsf{c}}$.

As we need to test the equivalence of configurations from $\mathcal{A}_1$ and $\mathcal{A}_2$, we consider the uninitialised weighted automata $\mathcal{B}$, which is a disjoint union of $\mathrm{U}(\mathcal{A}_1)$ and $\mathrm{U}(\mathcal{A}_2)$. This gives us a single automaton with which we can compare the configurations of $\mathcal{A}_1$ and $\mathcal{A}_2$. Let $i \in \{1,2\}$ and c be a configuration of $\mathcal{A}_i$. For all $p \in C_i$ and $m < |\mathcal{B}|$, we define the sets $\mathcal{W}_i^{p,m}$. The set $\mathcal{W}_i^{p,m}$ contains vectors $\mathbf{x} \in \mathcal{F}^{\mathsf{K}}$ such that the configuration $(\mathbf{x}, p, m)$ is $|\mathcal{B}|$-equivalent to some configuration of $\mathcal{B}$. The set $\overline{\mathcal{W}_i^{p,m}}$ is the set $\mathcal{F}^{\mathsf{K}} \setminus \mathcal{W}_i^{p,m}$. Formally,

$$\mathcal{W}_i^{p,m} = \{\mathbf{x} \in \mathcal{F}^{\mathsf{K}} | \exists \bar{\mathsf{c}} \in \mathcal{F}^{|\mathcal{B}|}, \mathsf{c} = (\mathbf{x}, p, m) \sim_{|\mathcal{B}|} \bar{\mathsf{c}}\}$$

Lemma 7.22

For any $i \in \{1,2\}$, $p \in C_i$ and $m < |\mathcal{B}|$, the set $\mathcal{W}_i^{p,m}$ is a vector space.

Proof. To prove this, it suffices to show that it is closed under vector addition and scalar multiplication. We fix a set $\mathcal{W}_i^{p,m}$. First, we prove that it is closed under scalar multiplication. For any vector $\mathbf{z}_1 \in \mathcal{W}_i^{p,m}$, we know that there exists a configuration $\mathsf{c} = (\mathbf{z}_1, p, m)$ and $\bar{\mathsf{c}} \in \mathcal{F}^{|\mathcal{B}|}$ such that $\mathsf{c} \sim_{|\mathcal{B}|} \bar{\mathsf{c}}$. Now, for any scalar $r \in \mathcal{F}$, the configuration $(r.\mathbf{z}_1, p, m) \sim_{|\mathcal{B}|2} r \cdot \bar{\mathsf{c}}$. Therefore $r \cdot \mathbf{z}_1 \in \mathcal{W}_i^{p,m}$. Now, we show that it is closed under vector addition. Let $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{W}_i^{p,m}$ be two vectors. Therefore, there exists configurations $\mathsf{c}_1 = (\mathbf{z}_1, p, m)$, $\mathsf{c}_2 = (\mathbf{z}_2, p, m)$, $\bar{\mathsf{c}}_1 \in \mathcal{F}^{|\mathcal{B}|}$

and $\bar{\mathtt{c}}_2 \in \mathcal{F}^{|\mathcal{B}|}$, such that $\mathtt{c}_1 \sim_{|\mathcal{B}|} \bar{\mathtt{c}}_1$ and $\mathtt{c}_2 \sim_{|\mathcal{B}|} \bar{\mathtt{c}}_2$. Consider the configuration $\mathtt{c}_3 = (\mathbf{z}_1 + \mathbf{z}_2, p, m)$, $\mathtt{c}_3 \sim_{|\mathcal{B}|} \bar{\mathtt{c}}_1 + \bar{\mathtt{c}}_2$. Therefore, $\mathbf{z}_1 + \mathbf{z}_2 \in \mathcal{W}_i^{p,m}$. □

The distance of a configuration $\mathtt{c}$ of $\mathcal{A}_i$ (denoted as $\mathrm{dist}_{\mathcal{A}_i}(\mathtt{c})$) is the length of a minimal word that takes you from $\mathtt{c}$ to a configuration $(\mathbf{x}, p, m)$ for some $m < |\mathcal{B}|$ and $p \in C_i$ such that $\mathbf{x} \in \overline{\mathcal{W}_i^{p,m}}$. We define $\mathrm{dist}_{\mathcal{A}_i}(\mathtt{c})$ as:

$$\min\{|w| \mid \mathtt{c} \xrightarrow{w} (\mathbf{x}, p, m)\ \exists p \in C_i, m < |\mathcal{B}|, \mathbf{x} \in \overline{\mathcal{W}_i^{p,m}}\}$$

The notion of distance play a key role in determining which parts of the run of a witness can be pumped out if it is not minimal. Given two configurations $\mathtt{c}$, $\mathtt{d}$ of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively, if $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d})$, then $\mathtt{c} \not\equiv \mathtt{d}$. By special word lemma (Lemma 7.12), the length lexicographically minimal reachability witness has a special form. This is used to show that if a configuration $\mathtt{c}$ of a weighted RODCA $\mathcal{A}$ has finite distance, then $\mathrm{dist}_{\mathcal{A}}(\mathtt{c}) = \frac{a}{b}\textsc{CounterValue}(\mathtt{c}) + t$, where $a, b, t \in \mathbb{N}$ and are polynomially bounded in $|\mathcal{A}|$. This helps us in proving that configuration pairs outside the initial space having equal distance lie inside a belt.

Lemma 7.23

Let $\mathtt{c} = (\mathbf{x}_\mathtt{c}, p_\mathtt{c}, n_\mathtt{c})$ be a configuration of a weighted RODCA $\mathcal{A}$. If $\mathrm{dist}_{\mathcal{A}}(\mathtt{c}) < \infty$ then, $\mathrm{dist}_{\mathcal{A}}(\mathtt{c}) = \frac{a}{b} n_\mathtt{c} + t$ where $a, b \in [0, 3\mathsf{K}^7]$ and $|t| < 42\mathsf{K}^{14}$.

Proof. Without loss of generality, let us consider the weighted RODCA $\mathcal{A}_1$ and a configuration $\mathtt{c} = (\mathbf{x}_\mathtt{c}, p_\mathtt{c}, n_\mathtt{c})$ of $\mathcal{A}_1$. Let us assume that $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) < \infty$. This means that $\mathtt{c} \to^* \mathtt{d}$, for some configuration $\mathtt{d} = (\mathbf{x}_\mathtt{d}, p, m)$ with $\mathbf{x}_\mathtt{d} \in \overline{\mathcal{W}_1^{p,m}}$ for some $p \in C_1$ and $m < 2\mathsf{K}^2$. Since $\textsc{CounterValue}(\mathtt{d}) = m$, by Lemma 7.12, we know that there is a word $u = u_1 u_2^r u_3$ (with $r \geq 0$) such that that $\mathtt{c} \xrightarrow{u} \mathtt{d}$ where $|u| = \mathrm{dist}_{\mathcal{A}_1}(\mathtt{c})$, $|u_1 u_3| \leq 9\mathsf{K}^7$, $|u_2| \leq 3\mathsf{K}^7$ and u_2 has a negative counter-effect ℓ. Let g be the combined counter-effect of u_1, u_3 and $\alpha = \frac{|u_2|}{\ell}$. Since $|u_1 u_3| \leq 9\mathsf{K}^7$, we have $|g| \leq 9\mathsf{K}^7$.

$$\begin{aligned} \mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) &= \frac{n_\mathtt{c} - m - g}{\ell}|u_2| + |u_1 u_3| \\ &= \alpha n_\mathtt{c} - \underbrace{\alpha(m+g) + |u_1 u_3|}_{t} \end{aligned}$$

Since $1 \leq \alpha \leq 3\mathsf{K}^7$ it follows that $-42\mathsf{K}^{14} < t < 42\mathsf{K}^{14}$. Hence proved. □

Therefore, the background space points either have unequal or infinite distances.

Lemma 7.24

For any configuration pair $\langle \mathtt{c}, \mathtt{d} \rangle$, in the background space, either $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d})$ or $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}) = \infty$.

Proof. Assume for contradiction that there is a configuration pair $\langle \mathtt{c}, \mathtt{d} \rangle$, in the background space with $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}) < \infty$. Let $n_{\mathtt{c}} = \textsc{CounterValue}(\mathtt{c})$ and $n_{\mathtt{d}} = \textsc{CounterValue}(\mathtt{d})$. Since $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d})$. From Lemma 7.23, there exists $a_1, b_1, a_2, b_2 \in [0, 3\mathsf{K}^7]$ and $d_1, d_2 < 42\mathsf{K}^{14}$ such that

$$\frac{a_1}{b_1} n_{\mathtt{c}} + d_1 = \mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}) = \mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}) = \frac{a_2}{b_2} n_{\mathtt{d}} + d_2$$

Therefore $|\frac{a_1}{b_1} n_{\mathtt{c}} - \frac{a_2}{b_2} n_{\mathtt{d}}| \leq |d_2 - d_1| < 42\mathsf{K}^{14}$. This satisfies the belt condition and is a configuration pair in the belt space. This contradicts our initial assumptions. $\square$

Similar to the work by Böhm and Göller (2011), we can show that the length of the run Π in the background space is polynomially bounded in K, and the counter values of the first background point in Π.

Lemma 7.25

If $\mathtt{h}_j = \langle \mathtt{c}_j, \mathtt{d}_j \rangle$ is the first configuration pair in the background space during Π, then $\ell - j$ is bounded by a polynomial in $\textsc{CounterValue}(\mathtt{c}_j)$, $\textsc{CounterValue}(\mathtt{d}_j)$, and K.

Proof. Let $\mathtt{h}_j = \langle \mathtt{c}_j, \mathtt{d}_j \rangle$ be the first configuration pair in the background space during the run Π, then from Lemma 7.24, either $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j) = \infty$ or $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j)$. We separately consider the two cases.

Case-1: $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j) = \infty$. In this case, we prove that the remaining length of the witness from $\langle \mathtt{c}_j, \mathtt{d}_j \rangle$ is bounded by $2\mathsf{K}^2$. Assume for contradiction that this is not the case and $\mathtt{c}_j \sim_{2\mathsf{K}^2} \mathtt{d}_j$ but $\mathtt{c}_j \not\equiv \mathtt{d}_j$. Let $v \in \Sigma^{>2\mathsf{K}^2}$ be the word which distinguishes $\mathtt{c}$ and $\mathtt{d}$. Therefore, there exists a prefix of v, $u \in \Sigma^{|v|-2\mathsf{K}^2}$, and $i = \ell - 2\mathsf{K}^2$ such that $\langle \mathtt{c}_j, \mathtt{d}_j \rangle \xrightarrow{u} \langle \mathtt{c}_i, \mathtt{d}_i \rangle$ and $\mathtt{c}_i \not\equiv_{2\mathsf{K}^2} \mathtt{d}_i$.

Since v is a minimal witness $\mathtt{c}_i \equiv_{2\mathsf{K}^2-1} \mathtt{d}_i$ and $\mathtt{c}_i \not\equiv_{2\mathsf{K}^2} \mathtt{d}_i$. Since $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) = \mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j) = \infty$, there exists configurations $\bar{\mathtt{c}}_i$ and $\bar{\mathtt{d}}_i$ in the underlying automaton $\mathcal{B}$ such that $\mathtt{c}_i{\sim}_{2\mathsf{K}^2}\bar{\mathtt{c}}_i$ and $\mathtt{d}_i{\sim}_{2\mathsf{K}^2}\bar{\mathtt{d}}_i$. Since $\mathtt{c}_i{\equiv}_{2\mathsf{K}^2-1}\mathtt{d}_i$, it follows that $\bar{\mathtt{c}}_i{\sim}_{2\mathsf{K}^2-1}\bar{\mathtt{d}}_i$. From the equivalence result of weighted automata, we know that if two configurations of a weighted automata with k states are non-equivalent, then there is a word of length less than k which distinguishes them. Therefore, this is sufficient to prove that the underlying weighted automaton with $\bar{\mathtt{c}}_i$ and $\bar{\mathtt{d}}_i$ as initial distributions are equivalent, and thus $\bar{\mathtt{c}}_i{\sim}_{2\mathsf{K}^2}\bar{\mathtt{d}}_i$. This allows us to deduce that $\mathtt{c}_i{\equiv}_{2\mathsf{K}^2}\mathtt{d}_i$, which is a contradiction. Therefore, the remaining length of the witness from $\langle \mathtt{c}_j, \mathtt{d}_j\rangle$ is bounded by $2\mathsf{K}^2$.

Case-2: $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) \neq \mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j)$. Without loss of generality, we suppose that $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) > \mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j)$. By definition of $\mathrm{dist}_{\mathcal{A}_1}$, there exists $u \in \Sigma^{\mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j)}$, $i > j$ and a configuration $\bar{\mathtt{c}}$ of the underlying automaton $\mathcal{B}$ such that $\mathtt{c}_j \xrightarrow{u} \mathtt{c}_i$, $\mathtt{d}_j \xrightarrow{u} \mathtt{d}_i$, $\mathtt{c}_i{\sim}_{2\mathsf{K}^2}\bar{\mathtt{c}}_i$ and $\mathtt{d}_i{\not\sim}_{2\mathsf{K}^2}\bar{\mathtt{c}}_i$. Therefore $\mathtt{c}_i{\not\equiv}_{2\mathsf{K}^2}\mathtt{d}_i$. By definition, there exists $v \in \Sigma^{\leq 2\mathsf{K}^2}$ such that $f_{\mathcal{A}_1}(v, \mathtt{c}_i) \neq f_{\mathcal{A}_2}(v, \mathtt{d}_i)$ and hence $f_{\mathcal{A}_1}(uv, \mathtt{c}_j) \neq f_{\mathcal{A}_2}(uv, \mathtt{d}_j)$. As $uv \in \Sigma^{\mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j)+2\mathsf{K}^2}$, we get that $\mathtt{c}_j{\not\equiv}_{\mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j)+2\mathsf{K}^2}\mathtt{d}_j$. Therefore, there is $w \in \Sigma^{\leq \min\{\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j), \mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j)\}+2\mathsf{K}^2}$ that distinguishes $\mathtt{c}_j$ and $\mathtt{d}_j$. □

Let $\alpha, \beta \in [1, 3\mathsf{K}^7]$ be co-prime. We say that two given configuration pairs $\langle(\mathbf{x}_\mathtt{c}, p_\mathtt{c}, n_\mathtt{c}), (\mathbf{x}_\mathtt{d}, p_\mathtt{d}, n_\mathtt{d})\rangle$ and $\langle(\mathbf{x}_\mathtt{e}, p_\mathtt{e}, n_\mathtt{e}), (\mathbf{x}_\mathtt{f}, p_\mathtt{f}, n_\mathtt{f})\rangle$ are α-β *related* if $p_\mathtt{c} = p_\mathtt{e}$, $p_\mathtt{d} = p_\mathtt{f}$ and $\alpha \cdot n_\mathtt{c} - \beta \cdot n_\mathtt{d} = \alpha \cdot n_\mathtt{e} - \beta \cdot n_\mathtt{f}$. Roughly speaking, two configuration pairs are α-β related if they have the same state pairs and lie on a line with slope $\frac{\alpha}{\beta}$. An α-β *repetition* is a run $\bar{\pi}_1 = c_i \tau_i c_{i+1} \tau_{i+1} \cdots \tau_{j-1} c_j$ that lies inside a belt with slope $\frac{\alpha}{\beta}$ such that c_i and c_j are α-β related.

The following lemma bounds the counter values of the first configuration pair in the background space, if it exists, during the run Π.

Lemma 7.26

If $\mathtt{h}_j = \langle \mathtt{c}_j, \mathtt{d}_j\rangle$ is the first background point in Π then, COUNTERVALUE$(\mathtt{c}_j)$ and COUNTERVALUE$(\mathtt{d}_j)$ are less than $\mathsf{K}^5 \cdot 42\mathsf{K}^{14}$.

Proof. Let $\mathtt{h}_j = \langle \mathtt{c}_j, \mathtt{d}_j\rangle$ be the first point in the background space during the run Π. Given any configuration $\mathtt{c}$, let $n_\mathtt{c}$ denote COUNTERVALUE$(\mathtt{c})$. Assume for contradiction that $n_{\mathtt{c}_j}$ is greater than $\mathsf{K}^5 \cdot 42\mathsf{K}^{14}$. Let $\Pi = \mathtt{h}_0\tau_0 \cdots \mathtt{h}_{j-1}\tau_{j-1}\mathtt{h}_j \cdots \mathtt{h}_\ell$ be a run of a minimal witness. Since $\mathtt{h}_j$ is the first point in the background space in this run and $n_{\mathtt{c}_j} > \mathsf{K}^5 \cdot 42\mathsf{K}^{14}$, there exists $0 < i < j$ such that the sub-run

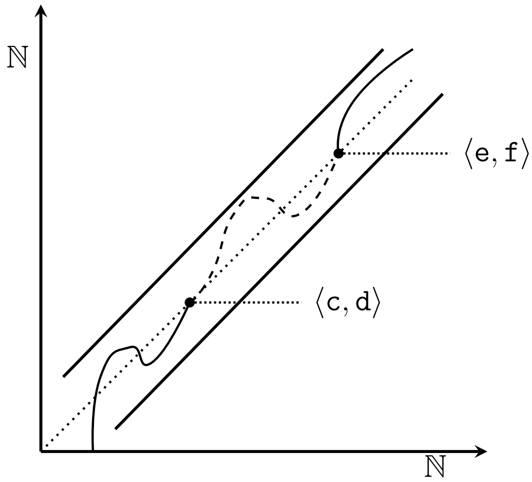


Fig. 7.7. The figure depicts an α-β repetition inside a belt with slope $\frac{\alpha}{\beta}$. The configuration pairs $\langle \mathtt{c}, \mathtt{d} \rangle$ and $\langle \mathtt{e}, \mathtt{f} \rangle$ are α-β-related. i.e., they lie on a line with slope $\frac{\alpha}{\beta}$. Note that $\textsc{CounterState}(\mathtt{c}) = \textsc{CounterState}(\mathtt{e})$ and $\textsc{CounterState}(\mathtt{d}) = \textsc{CounterState}(\mathtt{f})$ if $\langle \mathtt{c}, \mathtt{d} \rangle$ and $\langle \mathtt{e}, \mathtt{f} \rangle$ are α-β-related.

$\Pi_b = \mathtt{h}_i \tau_i \mathtt{h}_{i+1} \cdots \tau_{j-2} \mathtt{h}_{j-1}$ lies inside a belt B with slope $\frac{\alpha}{\beta}$ for some $\alpha, \beta \in [1, 3\mathsf{K}^7]$. Since we are looking at the run of a minimal witness, from Lemma 7.24 either $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j)$ or $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j) = \infty$. We separately consider the two cases.

Case-1: $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) \neq \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j)$. Without loss of generality, let us assume $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) < \mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_j)$. Therefore there exists $t \in \mathbb{N}$ with $j < t \leq \ell$ and configuration pair $\mathtt{h}_t$ such that $\mathtt{h}_t = \langle (\mathbf{x}_{\mathtt{c}_t}, p, m), (\mathbf{x}_{\mathtt{d}_t}, p_{\mathtt{d}_t}, n_{\mathtt{d}_t}) \rangle$, $m < 2\mathsf{K}^2$ and $\mathbf{x}_{\mathtt{c}_t} \in \overline{\mathcal{W}_1^{p,m}}$. We show that we can pump some portion out from Π_b to reach a configuration in the background space with unequal distance and smaller counter values.

Since $n_{\mathtt{c}_j} > \mathsf{K}^5 \cdot 42\mathsf{K}^{14}$, by the pigeonhole principle, there exists indices $i_0 < i_1 < i_2 < \cdots, i_{\mathsf{K}^2} < i'_0 < i'_1 < i'_2, \cdots, < i'_{\mathsf{K}^2}$ such that for all $r \in [1, \mathsf{K}^2]$, (1) $\mathtt{h}_{i_{r-1}}$ and $\mathtt{h}_{i_r}$ are α-β related and lie in belt B, (2) $n_{\mathtt{c}_{i_{r-1}}} < n_{\mathtt{c}_{i_r}} = n_{\mathtt{c}_{i'_r}}$, (3) $p_{\mathtt{c}_{i'_r}} = p_{\mathtt{c}_{i'_{r-1}}}$, (4) for all $t \in \mathbb{N}$ with $i_r < t < j$, $n_{\mathtt{c}_t} > n_{\mathtt{c}_{i_r}}$, and (5) for all $t \in \mathbb{N}$ with $j < t < i'_r$, $n_{\mathtt{c}_t} < n_{\mathtt{c}_{i'_r}}$.

Given any configuration $\mathtt{c}$ let $\mathbf{x}_\mathtt{c}$ denote $\textsc{WeightVector}(\mathtt{c})$. For $r \in [0, \mathsf{K}^2]$ let $\mathbb{A}_r \in \mathcal{F}^{\mathsf{K} \times \mathsf{K}}$ denote the matrix such that $\mathbf{x}_{\mathtt{c}_{i_r}} \mathbb{A}_r = \mathbf{x}_{\mathtt{c}_{i'_r}}$ and $\mathbb{B}_r \in \mathcal{F}^{\mathsf{K} \times \mathsf{K}}$ denote the matrix such that $\mathbf{x}_{\mathtt{c}_{i'_r}} \mathbb{B}_r = \mathbf{x}_{\mathtt{c}_t} \in \overline{\mathcal{W}_1^{p,m}}$. Therefore for all $r \in [0, \mathsf{K}^2]$, we have $\mathbf{x}_{\mathtt{c}_{i_r}} \mathbb{A}_r \mathbb{B}_r \in \overline{\mathcal{W}_1^{p,m}}$. From Lemma 2.3, we have that there exists $s, r \in [0, \mathsf{K}^2]$

with $s < r$ such that $\mathbf{x}_{\mathtt{c}_{i_s}}\mathbb{A}_r\mathbb{B}_s \in \overline{\mathcal{W}_1^{p,m}}$. Consider the sequence of transitions $T' = \tau_0, \cdots, \tau_{i_s-1}\tau_{i_r}, \cdots, \tau_{j-1}$ and let $w = \mathtt{word}(T')$. Let $\mathtt{h}_{j'}$ be the configuration such that $\mathtt{h}_0 \xrightarrow{w} \mathtt{h}_{j'}$. Since we have removed an α-β repetitions inside the belt, the configuration pair $\mathtt{h}_{j'}$ is a point in the background space (see Figure 7.8). Moreover, $n_{\mathtt{c}_{j'}} < n_{\mathtt{c}_j}$ and $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_{j'}) < \infty$. Since it is a point in the background space, from Lemma 7.24, we get that $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_{j'}) \neq \mathrm{dist}_{\mathcal{A}_1}(\mathtt{d}_{j'})$. Therefore, there is a shorter path to a configuration in background space with smaller counter values and unequal distance. This is a contradiction.

Case-2: $\mathrm{dist}_{\mathcal{A}_1}(\mathtt{c}_j) = \mathrm{dist}_{\mathcal{A}_2}(\mathtt{d}_j) = \infty$. In this case we know that $\mathtt{c}_j \not\equiv_{2\mathsf{K}^2} \mathtt{d}_j$. $\mathtt{c}_j \not\equiv_{|\mathcal{B}|} \mathtt{d}_j$.

Consider the sub-run Π_b. Since it is a run inside a belt, we can consider this as the run of the weighted RODCA $\mathcal{D}$. Since $n_{\mathtt{c}_j} > \mathsf{K}^4 \cdot 42\mathsf{K}^{14}$, by the pigeon-hole principle, there exists indices $i_0, i_1, i_2, \cdots, i_{\mathsf{K}^2}$ such that for all $r \in [1, \mathsf{K}^2]$, $\mathtt{h}_{i_{r-1}}$ and $\mathtt{h}_{i_r}$ are α-β related with $n_{\mathtt{c}_{i_{r-1}}} < n_{\mathtt{c}_{i_r}}$ and for all $t \in \mathbb{N}$ with $i_r < t < j$, $n_{\mathtt{c}_t} > n_{\mathtt{c}_{i_r}}$.

Since it is the run of a minimal witness, we know that there exists $\mathbb{A} \in \mathcal{F}^{2\mathsf{K}\times 2\mathsf{K}}$ such that $\mathbf{x}_{j-1}\mathbb{A}\eta_F^\top \neq 0$. Consider the vector space $\mathcal{U} = \{\mathbf{y} \in \mathcal{F}^{2\mathsf{K}} \mid \mathbf{y}\mathbb{A}\eta_F^\top = 0\}$. For $r \in [0, \mathsf{K}^2]$, let $\mathbb{A}_r$ denote the matrices such that $\mathbf{x}_{i_r}\mathbb{A}_r = \mathbf{x}_j \in \mathcal{U}$. Since $\mathbf{x}_{i_r}\mathbb{A}_r \in \overline{\mathcal{U}}$ for all $r \in [0, \mathsf{K}^2]$, from Lemma 2.3, we get that there exists $r' \in [0, r-1]$ such that $\mathbf{x}_{\mathtt{c}_{i'_r}}\mathbb{A}_r \in \overline{\mathcal{V}}$. The sequence of transitions $\tau_{i_r+1}\cdots\tau_\ell$ can be taken from $\mathtt{h}_{i'_r}$ since the counter values always stay positive. Consider the sequence of transitions $T' = \tau_0 \cdots \tau_{i'_r}\tau_{i_r+1}\cdots\tau_\ell$ and let $w = \mathtt{word}(T')$. The word w is a shorter witness than z and contradicts its minimality. □

Finally, we prove that the counter values encountered during the run Π are polynomially bounded in K using above lemmas.

Proof of Lemma 7.15. Consider the run Π. From Lemma 7.19, Lemma 7.20 and Lemma 7.26, we get that the counter values of configuration pairs inside the belt space during this run in polynomially bounded in K. Therefore, if it exists, the first background point in Π has polynomially bounded counter values. From Lemma 7.25, the length of Π after the first background point is polynomially bounded in K. Since the counter values of configuration pairs inside the initial space are also bounded by a polynomial in K, the maximum counter value in Π is polynomially bounded in K. □

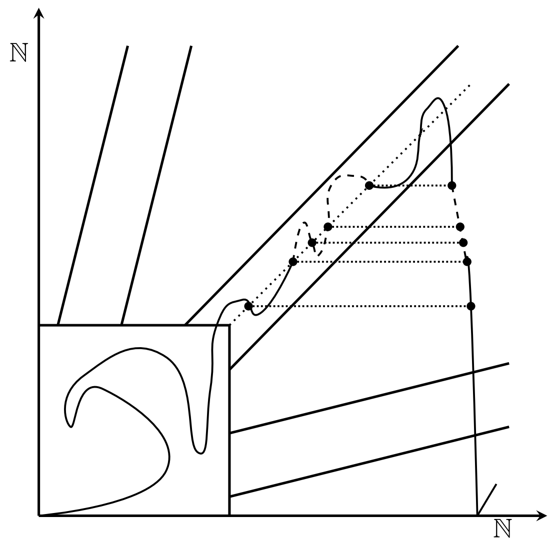


Fig. 7.8. The figure shows the run of a word that enters the background space from the belt such that the counter values of the first configuration pair in the background space exceed a polynomial bound. Some α-β repetitions inside the belt can be removed to show the existence of a shorter witness.

7.4 Regularity of ODCAs is in P

We say that a weighted RODCA $\mathcal{A}$ is regular if there is a weighted automaton $\mathcal{B}$ that is equivalent to it. We look at the *regularity* problem - the problem of deciding whether a weighted RODCA over fields is regular.

REGULARITY PROBLEM
INPUT: a weighted RODCA $\mathcal{A}$ over a field $\mathcal{F}$.
OUTPUT: *Yes*, if there exists a weighted automaton $\mathcal{D}$ over $\mathcal{F}$ such that $\mathcal{A}$ and $\mathcal{D}$ are equivalent. *No*, otherwise.

We fix a weighted RODCA $\mathcal{A} = ((C, \delta_0, \delta_1, p_0), (Q, \boldsymbol{\lambda}, \Delta, \boldsymbol{\eta}))$ and use N to denote $|C| \cdot |Q|$. The proof technique is adapted from the ideas developed by Böhm et al. (2014) in the context of DROCAs. The crucial idea in proving regularity is to check for the existence of infinitely many equivalence classes. The proof relies on the notion of distance of configurations. Distance of a configuration is the length of a minimal word to be read to reach a configuration that does not have an N equivalent configuration in the underlying uninitialised weighted automaton. The challenge is to find infinitely many configurations reachable from the initial configuration so that no two of them have the same distance.

We use $\mathtt{c}$ to denote some configuration of $\mathcal{A}$ and $\bar{\mathtt{c}}$ to denote some configuration of $\mathrm{U}(\mathcal{A})$. For a $p \in C$ and $m \in \mathbb{N}$, we define $\mathcal{W}^{p,m} = \{\mathbf{x} \in \mathcal{F}^{|Q|} \mid \exists \bar{\mathtt{c}} \in \mathcal{F}^{\mathsf{N}}, \mathtt{c} = (\mathbf{x}, p, m) \sim_{\mathsf{N}} \bar{\mathtt{c}}\}$. The set $\overline{\mathcal{W}}^{p,m}$ is $\mathcal{F}^{|Q|} \setminus \mathcal{W}^{p,m}$. The distance of a configuration $\mathtt{c}$ (denoted by $\mathrm{dist}(\mathtt{c})$) is

$$\mathrm{dist}(\mathtt{c}) = \min\{|w| \mid \mathtt{c} \xrightarrow{w} (\mathbf{x}, p, m) \text{ for some } p \in C, m < \mathsf{N}, \text{ and } \mathbf{x} \in \overline{\mathcal{W}}^{p,m}\} .$$

The following lemma shows when $\mathcal{A}$ is not regular. Given any configuration $\mathtt{c}$, we use $\mathbf{x}_{\mathtt{c}}$ to denote $\textsc{WeightVector}(\mathtt{c})$, $p_{\mathtt{c}}$ to denote $\textsc{CounterState}(\mathtt{c})$ and $n_{\mathtt{c}}$ to denote $\textsc{CounterValue}(\mathtt{c})$.

Lemma 7.27

Let $\mathtt{c}$ be the initial configuration of an weighted RODCA $\mathcal{A}$. The following are equivalent.

1. $\mathcal{A}$ is not regular.
2. for all $t \in \mathbb{N}$, there exists configurations $\mathtt{d}, \mathtt{e}$ s.t. $n_{\mathtt{e}} < \mathsf{N}, \mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \mathtt{e}$, $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$ and $t < \mathrm{dist}(\mathtt{d}) < \infty$.
3. there exists configurations $\mathtt{d}, \mathtt{e}$ and a run $\mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \mathtt{e}$ s.t. $\mathsf{N}^2 + \mathsf{N} \leq n_{\mathtt{d}} \leq 2\mathsf{N}^2 + \mathsf{N}$, $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$ with $n_{\mathtt{e}} < \mathsf{N}$.

Proof. $3 \to 2$: Consider an arbitrary $q \in C$, $m < \mathsf{N}$ and vector space $\mathcal{V} = \mathcal{W}^{q,m}$. Let us assume for contradiction the complement of Point 2. That is, there exists a $t \in \mathbb{N}$ such that for all configurations $\mathtt{d}'$ where $\mathtt{c} \xrightarrow{*} \mathtt{d}' \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$, $\mathrm{dist}(\mathtt{d}') \leq t$. Note that for all $\mathtt{d}'$ where $n_{\mathtt{d}'} > \mathsf{N}$, $\mathrm{dist}(\mathtt{d}') \geq n_{\mathtt{d}'} - \mathsf{N}$. Hence there exists an $M \in \mathbb{N}$ such that for all $\mathtt{d}'$ where $\mathtt{c} \xrightarrow{*} \mathtt{d}' \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$, $n_{\mathtt{d}'} \leq M$.

Consider a configuration $\mathtt{d}$ where $n_{\mathtt{d}} > \mathsf{N}^2 + \mathsf{N}$ and a run $\mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$. Point 3 shows the existence of such a run. For contradiction, it suffices to show there exists a $\mathtt{d}'$ such that $\mathtt{c} \xrightarrow{*} \mathtt{d}' \xrightarrow{*} \overline{\mathcal{V}} \times \{q\} \times \{m\}$ and $n_{\mathtt{d}'} > n_{\mathtt{d}}$. We get this from Lemma 7.5 Point 2, since $n_{\mathtt{c}} = 0$ and $n_{\mathtt{d}} > \mathsf{N}^2 + \mathsf{N}$.

$2 \to 1$: Assume for contradiction that for all $t \in \mathbb{N}$, there exists configurations $\mathtt{d}, \mathtt{e}$ such that $\mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \mathtt{e}$, $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}, n_{\mathtt{e}} < \mathsf{N}$ and $t < \mathrm{dist}(\mathtt{d}) < \infty$ but $\mathcal{A}$ is regular. Let $\mathcal{B}$ be the weighted automaton equivalent to $\mathcal{A}$. We use $|\mathcal{B}|$ to represent the number of states of $\mathcal{B}$.

Let $t_1, t_2, \ldots t_{|\mathcal{B}|+1} \in \mathbb{N}$ such that for $i \in [1, |\mathcal{B}|]$, $t_i < t_{i+1}$, and $\mathtt{d}_{t_i}$ be such that $\mathtt{c} \xrightarrow{*} \mathtt{d}_{t_i} \xrightarrow{*} (\mathbf{x}_i, p_{\mathtt{e}}, n_{\mathtt{e}})$, $\mathbf{x}_i \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$ and $t_i < \mathrm{dist}(\mathtt{d}_{t_i}) < t_{i+1} < \infty$. Clearly $\mathtt{d}_{t_i} \not\equiv \mathtt{d}_{t_j}$ for $i \neq j$ and corresponds to two different states of $\mathcal{B}$. Since we can find more than $|\mathcal{B}|$ pairwise non-equivalent configurations, it contradicts the assumption that $\mathcal{B}$ is equivalent to $\mathcal{A}$.

$1 \to 3$: We prove the contrapositive of the statement. Let us assume that there are no configurations $\mathtt{d}, \mathtt{e}$ and a run $\mathtt{c} \xrightarrow{*} \mathtt{d} \xrightarrow{*} \mathtt{e}$ such that $\mathsf{N}^2 + \mathsf{N} \leq n_{\mathtt{d}} \leq 2\mathsf{N}^2 + \mathsf{N}$, $\mathbf{x}_{\mathtt{e}} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$ with $n_{\mathtt{e}} < \mathsf{N}$. This implies that there does not exists a configuration $\mathtt{d}'$ such that $n_{\mathtt{d}'} > 2\mathsf{N}^2$, $\mathtt{c} \xrightarrow{*} \mathtt{d}' \xrightarrow{*} (\mathbf{y}, p_{\mathtt{e}}, n_{\mathtt{e}})$ for some $\mathbf{y} \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$. Assume for contradiction that there is such a run, then there exists a portion in this run that can be "pumped down" to get a run $\mathtt{c} \xrightarrow{*} \mathtt{d}'' \xrightarrow{*} (\mathbf{y}', p_{\mathtt{e}}, n_{\mathtt{e}})$ for some configuration $\mathtt{d}''$ such that $\mathsf{N}^2 + \mathsf{N} \leq n_{\mathtt{d}''} \leq 2\mathsf{N}^2 + \mathsf{N}$ and $\mathbf{y}' \in \overline{\mathcal{W}}^{p_{\mathtt{e}}, n_{\mathtt{e}}}$. This is a contradiction. Therefore all runs starting from a configuration with a counter value greater than or equal to $\mathsf{N}^2 + \mathsf{N}$ "looks" similar to a run on a weighted automaton. This allows us to simulate the runs of $\mathcal{A}$ using a weighted automaton. $\square$

Theorem 7.28
The regularity problem of weighted RODCAs (weights from a field) is in P.

Proof. Let $\mathcal{A}$ be a weighted RODCA and $\mathtt{c}_0$ be its initial configuration. Lemma 7.27 shows that if $\mathcal{A}$ is not regular, then there are words $u, v \in \Sigma^*$ and configurations $\mathtt{d}, \mathtt{e}$ such that there is a run of the form $\mathtt{c}_0 \xrightarrow{u} \mathtt{d} \xrightarrow{v} \mathtt{e}$ such that $\mathsf{N}^2 + \mathsf{N} \leq \textsc{CounterValue}(\mathtt{d}) \leq 2\mathsf{N}^2 + \mathsf{N}$, $\textsc{WeightVector}(\mathtt{e}) \in \overline{\mathcal{W}}^{\textsc{CounterState}(\mathtt{e}), \textsc{CounterValue}(\mathtt{e})}$ with $\textsc{CounterValue}(\mathtt{e}) < \mathsf{N}$. The existence of such words u and v can be decided in polynomial time since the minimal length of such a path if it exists, is polynomially bounded in the number of states of the weighted RODCA by Corollary 7.7. This concludes the proof. $\square$

7.5 Covering

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two uninitialised weighted RODCAs. We say $\mathcal{A}_2$ covers $\mathcal{A}_1$ if for all initial configurations $\mathtt{c}_0$ of $\mathcal{A}_1$ there exists an initial configuration $\mathtt{d}_0$ of $\mathcal{A}_2$ such that $\mathcal{A}_1\langle\mathtt{c}_0\rangle$ and $\mathcal{A}_2\langle\mathtt{d}_0\rangle$ are equivalent.

> COVERING PROBLEM
> INPUT: Two weighted RODCAs $\mathcal{A}$ and $\mathcal{B}$ over a field $\mathcal{F}$.
> OUTPUT: *Yes*, if $\mathcal{A}$ covers $\mathcal{B}$. *No*, otherwise.

We say $\mathcal{A}_1$ and $\mathcal{A}_2$ are coverable equivalent if $\mathcal{A}_1$ covers $\mathcal{A}_2$, and $\mathcal{A}_2$ covers $\mathcal{A}_1$. We show that the covering and coverable equivalence problems for uninitialised weighted RODCAs are decidable in polynomial time. The proof relies on the algorithm to check the equivalence of two weighted RODCAs.

Theorem 7.29
Covering and coverable equivalence problems of uninitialised weighted RODCAs over fields are in P.

Proof. We fix two uninitialised weighted RODCAs $\mathcal{A}_1 = ((C_1, \delta_1), (Q_1, \Delta_1, \boldsymbol{\eta}_1))$ and $\mathcal{A}_2 = ((C_2, \delta_2), (Q_2, \Delta_2, \boldsymbol{\eta}_2))$ for this section. Without loss of generality, assume $\mathsf{K} = |C_1| = |Q_1| = |C_2| = |Q_2|$. For $i \in [1, \mathsf{K}]$ we define the vector $\mathbf{e}_i \in \mathcal{F}^{\mathsf{K}}$ as follows:

$$\mathbf{e}_i[j] = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

For $j \in [1, \mathsf{K}]$, $q \in C_1$, we use $\mathtt{h}_{j,q}$ to denote the configuration $(\mathbf{e}_j, q, 0)$ of $\mathcal{A}_1$ and for $i \in [1, \mathsf{K}]$, $p \in C_2$, we use $\mathtt{g}_{i,p}$ to denote the configuration $(\mathbf{e}_i, p, 0)$ of $\mathcal{A}_2$.

Claim 1. *There is a polynomial time algorithm to decide whether $\mathcal{A}_2$ covers $\mathcal{A}_1\langle\mathtt{h}_{j,q}\rangle$ for any $j \in [1, \mathsf{K}]$ and $q \in C_1$.*

Proof. First, we check, in polynomial time (equivalence with a zero machine), whether $\mathcal{A}_1\langle\mathtt{h}_{j,q}\rangle$ accepts all words with weight $0 \in \mathcal{F}$. If that is the case, then $\mathcal{A}_1\langle\mathtt{h}_{j,q}\rangle$ and $\mathcal{A}_2\langle\mathtt{g}_0\rangle$ are equivalent for the configuration $\mathtt{g}_0 = (\{0\}^{\mathsf{K}}, p, 0)$, for any $p \in C_2$. Otherwise, there is some word w_0 accepted by $\mathcal{A}_1\langle\mathtt{h}_{j,q}\rangle$ with non-zero weight s (returned by the previous equivalence check). Without loss of generality, we consider the smallest one, whose size is polynomial in K.

We pick a $p \in C_2$ and check whether there exists an initial distribution from counter state p that makes the two machines equivalent. Assume that such an initial distribution exists, and for all $i \in [1, \mathsf{K}]$, let α_i denote the initial weight on state $q_i \in Q_2$. We use $\boldsymbol{\alpha}$ to denote the resultant initial distribution. We initialise an empty set B to store a system of linear equations.

The following steps will be repeated at most K times to check the existence of an initial distribution with initial state $p \in C_2$. Let w be the counter-example returned by the equivalence query in the previous step. For all $i \in [1, \mathsf{K}]$, we compute $f_{\mathcal{A}_2\langle \mathtt{g}_{i,p}\rangle}(w)$. We add the linear equation $\sum_{i=1}^{\mathsf{K}} \alpha_i \cdot f_{\mathcal{A}_2\langle \mathtt{g}_{i,p}\rangle}(w) = f_{\mathcal{A}_1\langle \mathtt{h}_{j,q}\rangle}(w)$ to B and compute values for α_i, $i \in [1, \mathsf{K}]$, such that it satisfies the system of linear equations in B. We check whether $\mathcal{A}_1\langle \mathtt{h}_{j,q}\rangle$ and $\mathcal{A}_2\langle(\alpha, p, 0)\rangle$ are equivalent or not. If they are not equivalent, we get a new counter-example that distinguishes them. Now, we repeat the procedure to compute a new initial distribution.

Note that the above procedure is executed at most K times to find an initial distribution if it exists. We can find only K many linearly independent linear equations in K variables. Suppose the above procedure fails to find an initial distribution for which the machines are equivalent. In that case, there is an initial distribution of $\mathcal{A}_1$ with initial counter state q, for which $\mathcal{A}_2$ with initial counter state p does not have an equivalent initial distribution. We now pick a different counter state of C_2 and repeat the process until we find a $p \in C_2$ for which the algorithm finds an equivalent initial distribution. If for all $p \in C_2$, the algorithm returns false, then $\mathcal{A}_2$ does not cover $\mathcal{A}_1\langle \mathtt{h}_{j,q}\rangle$. $\square_{Claim:1}$

Now, we show the existence of a polynomial time procedure to check whether $\mathcal{A}_2$ covers $\mathcal{A}_1$. For all $j \in [1, \mathsf{K}]$, we check whether there exists an initial state $p \in C_2$ such that $\mathcal{A}_2$ with initial counter state p has an initial distribution that makes it equivalent to $\mathcal{A}_1\langle \mathtt{h}_{j,q}\rangle$ using Claim 1. If we fail to find such a state in C_2, then we return false. We repeat this procedure for all $q \in C_1$. If for all $q \in C_1$ there exists a $p \in C_2$ such that $\mathcal{A}_2$ with initial counter state p has an initial distribution that makes it equivalent to $\mathcal{A}_1\langle \mathtt{h}_{j,q}\rangle$ for all $j \in [1, K]$, then we say that $\mathcal{A}_2$ covers $\mathcal{A}_1$ otherwise we say that $\mathcal{A}_2$ does not cover $\mathcal{A}_1$. Let us see why this is true. For simplifying the arguments, we fix a $q \in C_1$. Assume that for all $j \in [1, \mathsf{K}]$, there exits $p \in C_2$ such that $\mathcal{A}_1\langle \mathtt{h}_{j,q} \rangle$ is equivalent to the configuration $(\mathbf{x}_{j,q}, p, 0)$ for some $\mathbf{x}_{j,q} \in \mathcal{F}^{\mathsf{K}}$. Now, any initial configuration $(\boldsymbol{\lambda}, q, 0)$ of $\mathcal{A}_1$ is equivalent to the configuration $(\sum_{j=1}^{\mathsf{K}} \boldsymbol{\lambda}[j] \cdot \mathbf{x}_{j,q}, p, 0)$ of $\mathcal{A}_2$.

The coverable equivalence problem can now be solved by checking whether $\mathcal{A}_1$ covers $\mathcal{A}_2$ and $\mathcal{A}_2$ covers $\mathcal{A}_1$, which can be done in time polynomial in K. $\square$

7.6 Conclusion

In this chapter, we showed that the equivalence problem for weighted RODCAs is in P if the weights are from a field. Note that the equivalence of weighted OCAs (weights from a field) is still an open problem. We also looked at the co-VS reachability and co-VS coverability problems for weighted RODCAs over fields, and showed that they are in P. We also looked at the regularity, coverability and coverable equivalence problems and showed that they are all in P for weighted RODCAs over fields. In future work, it will be interesting to look at other models where the stack operations are deterministic. For example, a natural extension is to consider *stack-deterministic* pushdown automata - where a deterministic machine updates the stack. We also leave open the question of learning of weighted RODCAs over fields.

Part IV

Conclusions

CHAPTER 8

Summary and Future Work

8.1 Summary of Contributions

My research journey has been driven by a deep interest in automata theory, with a particular focus on understanding the learning and equivalence of one-counter systems.

8.1.1 Learning

In Chapter 4, we designed a new learning algorithm for deterministic real-time one-counter automaton (DROCA) with the help of a SAT solver. Traditional approaches to learning DROCAs often involve learning an initial portion of an infinite behaviour graph. The aim is to identify a repetitive structure in the graph that defines the overall behaviour of a DROCA. However, this repetitive structure often requires constructing an exponentially large graph, leading to an exponential runtime and number of queries. Our SAT-based method reduces the number of queries to a polynomial number, significantly improving the efficiency of learning algorithms for one-counter automata. Additionally, it learns a minimal counter-synchronous DROCA recognising the language, making it more feasible for practical applications. From our implementation and experiments, we observed that our method outperforms the existing techniques in learning one-counter automata.

8.1.2 Equivalence

In Chapter 6, we introduced a model called weighted one-deterministic counter automaton (RODCA). These are weighed one-counter automata with counter-determinacy, meaning that all paths labelled by a given word, starting from the initial configuration, reach the same counter value. In Chapter 7, we proved that the equivalence, regularity and covering problems for this model are in P, whereas the problem is not known to be decidable for weighted one-counter automata in general. Our positive result on weighted RODCAs forms a foundation for developing their learning algorithms, creating new possibilities for efficient verification tools. In Chapter 5, we studied deterministic weighted real-time one-counter automata, a subclass of weighted RODCAs and proved that the equivalence of this model can be checked faster than that of weighted RODCAs.

Even though the equivalence of DROCAs is in P, it cannot be used in practice because of its high computational costs. Given two DROCAs with number of states less than some integer K, the equivalence check takes $\mathcal{O}(K^{26})$ time. To overcome this, we introduced the concept of counter-synchronous DROCAs, which allows for a faster equivalence check that runs in $\mathcal{O}(\alpha(K^5)K^5)$ time. Using the techniques from Section 7.2, we discovered an even faster $\mathcal{O}(\alpha(K^3)K^3)$ algorithm for checking the equivalence of two visibly one-counter automata.

These results motivate further research on equivalence and learning of complex automata models.

8.2 Future Directions

We aim to extend our findings on one-counter automata to visibly pushdown automata and develop efficient learning algorithms using SAT solvers. We also plan to improve algorithms for learning one-counter automaton (OCA), ensuring they work well in theory and practice. Another key direction is the development of algorithms for weighted OCAs, with an initial focus on deterministic weighted OCAs.

The equivalence problem for probabilistic push-down and one-counter systems are not known to be solvable algorithmically. However, our work on weighted one-counter models with counter-determinacy opens the door to potential solutions. In particular, *stack-deterministic* pushdown systems may provide a practical approach to overcoming these obstacles. Stack determinism restricts the stack configuration reached on reading a given word from the initial configuration to be the same on all paths. This would offer significant benefits for analysing real-world software systems, enabling verification tools to handle more complex automata models with realistic constraints. As a first step, we will explore equivalence and learning of real-time stack-deterministic pushdown automata and then extend this to real-time weighted stack-deterministic pushdown automata. Additionally, we plan to explore active learning methods for automata with additional resources, such as timing constraints, which are vital for the analysis of real-time systems. This will contribute significantly to automata learning, enabling more effective, practical tools for verifying and analysing complex software systems.

Specifically, my future research goals include the following:

1. **Exploring practical learning algorithms for visibly pushdown automata:** Build upon the recently developed ideas in polynomial query learning algorithm for one-counter automata using SAT solvers, to learn visibly pushdown automata, which are more complex and widely applicable in software verification.

2. **Find better learning algorithms for OCAs:** Design algorithms for learning one-counter systems that work well in theory and practice. This includes learning algorithms for OCA with ε-transitions and faster learning algorithms for VOCAs.

3. **Investigating learning algorithms for weighted one-counter systems:** Building on our recent results and our ongoing work on learning DROCAs, we aim to find learning algorithms for DWROCAs as an initial model. We plan to extend this to weighted RODCAs, leveraging known equivalence checks.

4. **Equivalence checking and learning for weighted real-time stack-deterministic pushdown automaton:** An essential aspect of advancing pushdown automata learning is solving the equivalence problem for real-time stack

deterministic pushdown automaton, followed by the development of active learning algorithms for these systems. The equivalence of deterministic pushdown automata is known to be decidable (Sénizergues, 1997). Proving the decidability of equivalence of real-time stack-deterministic pushdown automata will push the boundary of known decidable equivalence results. Building on the equivalence of real-time stack-deterministic pushdown automata and results on weighted RODCAs, we plan to explore learning algorithms for real-time weighted stack-deterministic pushdown automata with weights from fields. The first step in this direction would be to look at weighted deterministic real-time pushdown automata.

5. **Active and passive learning of automata with resources:** Finally, we aim to explore both active and passive learning approaches for automata that include additional resources, such as timing constraints, to create models capable of handling analysis of complex systems.

References

S. P. Abney, D. A. McAllester, and F. Pereira. Relating probabilistic grammars and automata. In R. Dale and K. W. Church, editors, *27th Annual Meeting of the Association for Computational Linguistics, University of Maryland, College Park, Maryland, USA, 20-26 June 1999*, pages 542–549. ACL, 1999. doi: 10.3115/1034678.1034759. URL https://doi.org/10.3115/1034678.1034759.

M. Almeida, N. Moreira, and R. Reis. Testing the equivalence of regular languages. *J. Autom. Lang. Comb.*, 15(1/2):7–25, 2010. doi: 10.25596/JALC-2010-007. URL https://doi.org/10.25596/jalc-2010-007.

R. Alur and P. Madhusudan. Visibly pushdown languages. In L. Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211. ACM, 2004. doi: 10.1145/1007352.1007390. URL https://doi.org/10.1145/1007352.1007390.

D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987. doi: 10.1016/0890-5401(87)90052-6. URL https://doi.org/10.1016/0890-5401(87)90052-6.

V. Bárány, C. Löding, and O. Serre. Regularity problems for visibly pushdown languages. In B. Durand and W. Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 420–431. Springer, 2006. doi: 10.1007/11672142_34. URL https://doi.org/10.1007/11672142_34.

P. Berman and R. Roos. Learning one-counter languages in polynomial time. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 61–67, 1987. doi: 10.1109/SFCS.1987.36.

S. Böhm and S. Göller. Language equivalence of deterministic real-time one-counter automata is nl-complete. In F. Murlak and P. Sankowski, editors, *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2011. doi: 10.1007/978-3-642-22993-0_20. URL https://doi.org/10.1007/978-3-642-22993-0_20.

S. Böhm, S. Göller, and P. Jancar. Bisimilarity of one-counter processes is pspace-complete. In P. Gastin and F. Laroussinie, editors, *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2010. doi: 10.1007/978-3-642-15375-4_13. URL https://doi.org/10.1007/978-3-642-15375-4_13.

S. Böhm, S. Göller, and P. Jancar. Equivalence of deterministic one-counter automata is nl-complete. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 131–140. ACM, 2013. doi: 10.1145/2488608.2488626. URL https://doi.org/10.1145/2488608.2488626.

S. Böhm, S. Göller, and P. Jancar. Bisimulation equivalence and regularity for real-time one-counter automata. *J. Comput. Syst. Sci.*, 80(4):720–743, 2014. doi: 10.1016/J.JCSS.2013.11.003. URL https://doi.org/10.1016/j.jcss.2013.11.003.

T. Brázdil, A. Kucera, and O. Strazovský. On the decidability of temporal properties of probabilistic pushdown automata. In V. Diekert and B. Durand, editors, *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, volume 3404 of *Lecture Notes in Computer Science*, pages 145–157. Springer,

2005. doi: 10.1007/978-3-540-31856-9_12. URL https://doi.org/10.1007/978-3-540-31856-9_12.

T. Brázdil, J. Esparza, S. Kiefer, and A. Kucera. Analyzing probabilistic pushdown automata. *Formal Methods Syst. Des.*, 43(2):124–163, 2013. doi: 10.1007/S10703-012-0166-0. URL https://doi.org/10.1007/s10703-012-0166-0.

V. Bruyère, G. A. Pérez, and G. Staquet. Learning realtime one-counter automata. In D. Fisman and G. Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 244–262. Springer, 2022. doi: 10.1007/978-3-030-99524-9_13. URL https://doi.org/10.1007/978-3-030-99524-9_13.

D. Chistikov, W. Czerwinski, P. Hofman, M. Pilipczuk, and M. Wehar. Shortest paths in one-counter systems. *Log. Methods Comput. Sci.*, 15(1), 2019. doi: 10.23638/LMCS-15(1:19)2019. URL https://doi.org/10.23638/LMCS-15(1:19)2019.

D. Dell'Erba, Y. Li, and S. Schewe. Dfaminer: Mining minimal separating dfas from labelled samples. In A. Platzer, K. Y. Rozier, M. Pradella, and M. Rossi, editors, *Formal Methods - 26th International Symposium, FM 2024, Milan, Italy, September 9-13, 2024, Proceedings, Part II*, volume 14934 of *Lecture Notes in Computer Science*, pages 48–66. Springer, 2024. doi: 10.1007/978-3-031-71177-0_4. URL https://doi.org/10.1007/978-3-031-71177-0_4.

A. F. Fahmy and R. S. Roos. Efficient learning of real time one-counter automata. In K. P. Jantke, T. Shinohara, and T. Zeugmann, editors, *Algorithmic Learning Theory, 6th International Conference, ALT '95, Fukuoka, Japan, October 18-20, 1995, Proceedings*, volume 997 of *Lecture Notes in Computer Science*, pages 25–40. Springer, 1995. doi: 10.1007/3-540-60454-5_26. URL https://doi.org/10.1007/3-540-60454-5_26.

V. Forejt, P. Jancar, S. Kiefer, and J. Worrell. Bisimilarity of probabilistic pushdown automata. In D. D'Souza, T. Kavitha, and J. Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 448–460. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. doi: 10.4230/LIPICS.FSTTCS.2012.448. URL https://doi.org/10.4230/LIPIcs.FSTTCS.2012.448.

V. Forejt, P. Jancar, S. Kiefer, and J. Worrell. Language equivalence of probabilistic pushdown automata. *Inf. Comput.*, 237:1–11, 2014. doi: 10.1016/J.IC.2014.04.003. URL https://doi.org/10.1016/j.ic.2014.04.003.

O. Gauwin, A. Muscholl, and M. Raskin. Minimization of visibly pushdown automata is np-complete. *Log. Methods Comput. Sci.*, 16(1), 2020. doi: 10.23638/LMCS-16(1:14)2020. URL https://doi.org/10.23638/LMCS-16(1:14)2020.

E. M. Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978. doi: 10.1016/S0019-9958(78)90562-4. URL https://doi.org/10.1016/S0019-9958(78)90562-4.

M. Heule and S. Verwer. Exact DFA identification using SAT solvers. In J. M. Sempere and P. García, editors, *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings*, volume 6339 of *Lecture Notes in Computer Science*, pages 66–79. Springer, 2010. doi: 10.1007/978-3-642-15488-1_7. URL https://doi.org/10.1007/978-3-642-15488-1_7.

J. E. Hopcroft and R. M. Karp. A linear algorithm for testing equivalence of finite automata. *Technical Report, University of California*, pages 71–114, 1971.

J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979. ISBN 0-201-02988-X.

J. Hromkovic and G. Schnitger. On probabilistic pushdown automata. *Inf. Comput.*, 208(8):982–995, 2010. doi: 10.1016/J.IC.2009.11.001. URL https://doi.org/10.1016/j.ic.2009.11.001.

O. H. Ibarra. Restricted one-counter machines with undecidable universe problems. *Mathematical systems theory*, 13(1):181–186, 1979. URL https://doi.org/10.1007/BF01744294.

M. Isberner. *Foundations of active automata learning: an algorithmic perspective*. PhD thesis, Technical University Dortmund, Germany, 2015. URL https://hdl.handle.net/2003/34282.

M. Isberner, F. Howar, and B. Steffen. The TTT algorithm: A redundancy-free approach to active automata learning. In B. Bonakdarpour and S. A. Smolka, editors, *Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings*, volume 8734 of *Lecture Notes in Computer Science*, pages 307–322. Springer, 2014. doi: 10.1007/978-3-319-11164-3_26. URL https://doi.org/10.1007/978-3-319-11164-3_26.

S. Kiefer, A. S. Murawski, J. Ouaknine, B. Wachter, and J. Worrell. On the complexity of equivalence and minimisation for q-weighted automata. *Log. Methods Comput. Sci.*, 9(1), 2013. doi: 10.2168/LMCS-9(1:8)2013. URL https://doi.org/10.2168/LMCS-9(1:8)2013.

D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Int. J. Algebra Comput.*, 4(3):405–426, 1994. doi: 10.1142/S0218196794000063. URL https://doi.org/10.1142/S0218196794000063.

A. Kucera. Methods for quantitative analysis of probabilistic pushdown automata. *Electronic Notes in Theoretical Computer Science*, 149(1):3–15, 2005. doi: 10.1016/J.ENTCS.2005.11.013. URL https://doi.org/10.1016/j.entcs.2005.11.013.

A. Kucera, J. Esparza, and R. Mayr. Model checking probabilistic pushdown automata. *Log. Methods Comput. Sci.*, 2(1), 2006. doi: 10.2168/LMCS-2(1:2)2006. URL https://doi.org/10.2168/LMCS-2(1:2)2006.

M. Leucker and D. Neider. Learning minimal deterministic automata from inexperienced teachers. In T. Margaria and B. Steffen, editors, *Leveraging Applications* *of Formal Methods, Verification and Validation. Technologies for Mastering Change - 5th International Symposium, ISoLA 2012, Heraklion, Crete, Greece, October 15-18, 2012, Proceedings, Part I*, volume 7609 of *Lecture Notes in Computer Science*, pages 524–538. Springer, 2012. doi: 10.1007/978-3-642-34026-0_39. URL https://doi.org/10.1007/978-3-642-34026-0_39.

P. Mathew, V. Penelle, P. Saivasan, and A. V. Sreejith. Weighted one-deterministic-counter automata. In P. Bouyer and S. Srinivasan, editors, *43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2023, December 18-20, 2023, IIIT Hyderabad, Telangana, India*, volume 284 of *LIPIcs*, pages 39:1–39:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPICS.FSTTCS.2023.39. URL https://doi.org/10.4230/LIPIcs.FSTTCS.2023.39.

P. Mathew, V. Penelle, P. Saivasan, and A. V. Sreejith. Equivalence of deterministic weighted real-time one-counter automata. In *the proceedings of 11th Indian Conference on Logic and its Applications (ICLA 2025) (to appear)*, 2025a. URL https://arxiv.org/abs/2411.03066.

P. Mathew, V. Penelle, and A. V. Sreejith. Learning real-time one-counter automata using polynomially many queries. In *the proceedings of the 31st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2025) (to appear)*, 2025b. URL https://arxiv.org/abs/2411.08815.

P. Mathew, V. Penelle, and A. V. Sreejith. Learning deterministic one-counter automata in polynomial time. In *the proceedings of the 40th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2025) (to appear)*, 2025c. URL https://arxiv.org/abs/2503.04525.

J. Michaliszyn and J. Otop. Learning deterministic visibly pushdown automata under accessible stack. In S. Szeider, R. Ganian, and A. Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, volume 241 of *LIPIcs*, pages 74:1–74:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPICS.MFCS.2022.74. URL https://doi.org/10.4230/LIPIcs.MFCS.2022.74.

D. Neider. Computing minimal separating DFAs and regular invariants using SAT and SMT solvers. In S. Chakraborty and M. Mukund, editors, *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, volume 7561 of *Lecture Notes in Computer Science*, pages 354–369. Springer, 2012. doi: 10.1007/978-3-642-33386-6_28. URL https://doi.org/10.1007/978-3-642-33386-6_28.

D. Neider and C. Löding. Learning visibly one-counter automata in polynomial time. *Technical Report, RWTH Aachen*, AIB-2010-02, 2010.

D. Nowotka and J. Srba. Height-deterministic pushdown automata. In L. Kucera and A. Kucera, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 125–134. Springer, 2007. doi: 10.1007/978-3-540-74456-6_13. URL https://doi.org/10.1007/978-3-540-74456-6_13.

W. F. Ogden. A helpful result for proving inherent ambiguity. *Math. Syst. Theory*, 2(3):191–194, 1968. doi: 10.1007/BF01694004. URL https://doi.org/10.1007/BF01694004.

F. Olmedo, B. L. Kaminski, J. Katoen, and C. Matheja. Reasoning about recursive probabilistic programs. In M. Grohe, E. Koskinen, and N. Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 672–681. ACM, 2016. doi: 10.1145/2933575.2935317. URL https://doi.org/10.1145/2933575.2935317.

L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *J. ACM*, 40(1):95–142, 1993. doi: 10.1145/138027.138042. URL https://doi.org/10.1145/138027.138042.

M. P. Schützenberger. On the definition of a family of automata. *Inf. Control.*, 4(2-3):245–270, 1961. doi: 10.1016/S0019-9958(61)80020-X. URL https://doi.org/10.1016/S0019-9958(61)80020-X.

G. Sénizergues. The equivalence problem for deterministic pushdown automata is decidable. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 671–681. Springer, 1997. doi: 10.1007/3-540-63165-8_221. URL https://doi.org/10.1007/3-540-63165-8_221.

M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997. ISBN 978-0-534-94728-6.

G. Staquet. *Active Learning of Automata with Resources*. PhD thesis, Université de Mons & Universiteit Antwerpen, Belgium, 2024. URL https://hdl.handle.net/10067/2079640151162165141.

C. Stirling. Deciding DPDA equivalence is primitive recursive. In P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. J. Eidenbenz, and R. Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 821–832. Springer, 2002. doi: 10.1007/3-540-45465-9_70. URL https://doi.org/10.1007/3-540-45465-9_70.

L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In A. V. Aho, A. Borodin, R. L. Constable, R. W. Floyd, M. A. Harrison, R. M. Karp, and H. R. Strong, editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9. ACM, 1973. doi: 10.1145/800125.804029. URL https://doi.org/10.1145/800125.804029.

G. Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006. URL http://www.amazon.com/Linear-Algebra-Its-Applications-Edition/dp/0030105676.

W. Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM J. Comput.*, 21(2):216–227, 1992. doi: 10.1137/0221017. URL https://doi.org/10.1137/0221017.

F. W. Vaandrager, B. Garhewal, J. Rot, and T. Wißmann. A new approach for active automata learning based on apartness. In D. Fisman and G. Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 223–243. Springer, 2022. doi: 10.1007/978-3-030-99524-9_12. URL https://doi.org/10.1007/978-3-030-99524-9_12.

L. G. Valiant and M. Paterson. Deterministic one-counter automata. *J. Comput. Syst. Sci.*, 10(3):340–350, 1975. doi: 10.1016/S0022-0000(75)80005-5. URL https://doi.org/10.1016/S0022-0000(75)80005-5.

M. Vazquez-Chanlatte, V. Lee, and A. Shah. DFA-identify, Aug. 2021. URL https://github.com/mvcisback/dfa-identify.

D. Wojtczak. *Recursive probabilistic models : efficient analysis and implementation*. PhD thesis, University of Edinburgh, UK, 2009. URL https://hdl.handle.net/1842/3217.

Abbreviations

OCA	One-Counter Automaton
DOCA	Deterministic One-Counter Automaton
DROCA	Deterministic Real-Time One-Counter Automaton
VOCA	Visibly One-Counter Automaton
WA	Weighted Automaton
DWA	Deterministic Weighted Automaton
DWROCA	Deterministic Weighted Real-Time One-Counter Automaton
RODCA	Real-Time One-Deterministic Counter Automaton

Index