On the Complexity of Verifying Differential Privacy

by

David Purser

Thesis

for the degree of

Doctor of Philosophy in Computer Science

The University of Warwick

Department of Computer Science

June 2020

Contents

Contents i

List of Figures iv

Acknowledgments v

Declarations vi

Abstract vii

Acronyms viii

Symbols ix

Chapter 1 Introduction 1

1.1 Contributions 6

Chapter 2 Background 9

2.1 Differential privacy 9

2.1.1 Laplacian and exponential mechanisms 10

2.1.2 Composition 11

2.1.3 The relation to quantitative information flow 12

2.2 Programming language based verification 12

2.3 Relational logic based verification 14

2.4 Verification on probabilistic transition systems 17

2.4.1 Bisimilarity pseudometrics for ε-differential privacy . . . 21

2.4.2 Other techniques 23

Chapter 3 Common Definitions 25

3.1 Basic notation 25

3.1.1 Set operations 25

3.1.2 Representations of numbers 25

3.1.3 Approximation and computation of numbers 26

3.2 Models of automata and Markov chains 27

3.2.1 Weighted automata 28

3.2.2 Finite word labelled Markov chains 28

3.2.3 Infinite word labelled Markov chains 30

3.2.4 Transition labelled vs state labelled LMCs 31
3.2.5 Relation between models 33
3.2.6 Comparison of labelled Markov chains 33
3.3 Differential privacy in LMCs 34

Chapter 4 Symmetric Bisimilarity Distances for δ 36
4.1 Preliminaries 37
4.2 Skewed bisimilarity distance, bd_α 39
4.3 Proving Theorem 4.11 42
4.4 Skewed Kantorovich distances 53
4.5 Computing bd_α 57
4.6 Examples 61
4.7 Conclusion 65

Chapter 5 Asymmetric Distances for δ 66
5.1 Defining lv_α to capture δ 67
5.2 lv_α is not computable 69
5.3 Approximation of lv_α 70
5.4 A least fixed point bound ld_α 74
5.4.1 Comparison with bd_α from Chapter 4 77
5.4.2 Computing ld_α 78
5.5 A greatest fixed point bound lgd_α 80
5.5.1 A unique fixed point? 86
5.6 Examples 86
5.7 Future work: extended models 90
5.8 Conclusion 93

Chapter 6 Distances for ε 94
6.1 The big-O problem 95
6.1.1 The relation to differential privacy 97
6.1.2 The big-Θ problem 97
6.2 Big-O, threshold and approximation problems are undecidable . 98
6.3 The relation to the VALUE-1 problem 104
6.4 The language containment condition 110
6.4.1 Unambiguous weighted automata 111
6.5 The big-O problem is **coNP**-complete 112
6.5.1 Preliminaries 113
6.5.2 Eventual inclusion 117
6.5.3 The big-O problem is in **coNP** 119
6.5.4 TV-BOUNDED is **coNP**-hard for unary LMCs 123
6.6 Decidability for weighted automata with bounded languages . . 128
6.6.1 Detector automata 129

6.6.2 The plus-letter-bounded case 132
6.6.3 The letter-bounded case 139
6.6.4 The bounded case 140
6.7 Bisimilarity distances for ε 142
6.7.1 Dual form and simplification 142
6.7.2 Computing $bd_{\ln}$ 145
6.7.3 A direct approach $bd_{\otimes}$ 145
6.7.4 Computing $bd_{\otimes}$ 148
6.7.5 Looking for a unique fixed point 148
6.8 Conclusion 151

Chapter 7 Verifying Differential Privacy in Circuits 152
7.1 Preliminaries 153
7.1.1 Randomised circuits 153
7.1.2 Differential privacy in randomised circuits 154
7.1.3 Problems of deciding and approximating DP 155
7.1.4 The class $\mathbf{coNP}^{\#\mathbf{P}}$ 155
7.2 The complexity of deciding pure differential privacy 156
7.2.1 Decide-ε-DP $\in \mathbf{coNP}^{\#\mathbf{P}}$: 156
7.2.2 $\mathbf{coNP}^{\#\mathbf{P}}$-hardness of Decide-$\varepsilon$-DP 157
7.3 The complexity of deciding approximate differential privacy . . 161
7.3.1 Direct proof that Decide-ε, δ-DP is $\mathbf{coNP}^{\#\mathbf{P}}$-hard . . 162
7.3.2 Decide-ε, δ-DP $\in \mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ 166
7.4 Inapproximability of the privacy parameters ε and δ 167
7.5 Conclusion 170

Chapter 8 Conclusions and Future Work 171
8.1 Open problems and future work 173

Bibliography 175

List of Figures

2.1 States are language equivalent but not bisimilar. 19

3.1 Partial order of automata. 32

4.1 States s_0 and s_1 are not bisimilar, but $tv_{1.5}(s_0, s_1) = 0$. 38
4.2 Logical formulation of least pre-fixed point bd_α. 58
4.3 **NP** formula for BD$_\alpha$-Threshold. 59
4.4 Simulation of dining cryptographers protocol 62
4.5 Markov chain for dining cryptographers. 62
4.6 Alternative Markov chain for dining cryptographers. 63
4.7 Plot of δ against α for the model in Figure 4.6. 63
4.8 Simple Markov chain. 64

5.1 Partial order of distances. 67
5.2 Markov chain in the reduction from $tv(q, q')$ to $lv_\alpha(s, s')$. 69
5.3 **NP** formula for LD$_\alpha$-Threshold. 79
5.4 PIN checker example. 88
5.5 Approximations of δ given ε for Figure 5.4. 88
5.6 Randomised response example. 89

6.1 Reductions between big-O and big-Θ. 98
6.2 Reduction from emptiness of probabilistic automata to big-O. . 101
6.3 Reduction from the big-O problem to Value-1. 107
6.4 Language equivalent but not big-O. 111
6.5 Different rates for different phases. 116
6.6 Reduction from NFA universality to the big-O problem. 124
6.7 Example unary labelled Markov chain. 127
6.8 Same relative orderings, but different big-O status. 129
6.9 Logical formulation for BD$_{\ln}$Threshold. 146
6.10 Logical formulation for BD$_\otimes$Threshold. 149

7.1 Example randomised circuit. 154

Acknowledgments

Firstly, I would like to thank my supervisors **Andrzej Murawski**, **Dmitry Chistikov** and **Graham Cormode** who have provided excellent supervision, guidance and encouragement throughout my research.

I would also like to thank my external collaborators, **Petr Jančar, Marco Gaboardi, Kobbi Nissim** and **Stefan Kiefer**, who have helped make my research better—as have the many more people at Warwick and in the community who have taken the time to discuss my work. Thank you to the thesis examiners **Marcin Jurdzinski** and **Franck van Breugel** for their positive and constructive feedback.

Those I have shared the experience as part of the CDT have made my time at Warwick thoroughly enjoyable, in particular: **Alex Dixon, Caroline Player, Corinne Muir, Greg Watson, Helen McKay, Ian Tu, James Van Hinsbergh, John Rahilly, Katherine Ascott, Liam Steadman, Matthew Bradbury, Melissa Kenny, Richard Kirk, Vikki Houlden, Zakiyya Adam** and **Zhenyu Li**. Thank you also to the administrators who have made everything run smoothly, particularly **Yvonne Colmer** and **Katie Martin**.

Finally, thank you to **my parents** who have supported me throughout this PhD.

Sponsorships and Grants

The work in this thesis was supported by funding form the UK Engineering and Physical Sciences Research Council (EP/L016400/1), the EPSRC Centre for Doctoral Training in Urban Science.

Declarations

Parts of this work in this thesis is from joint collaborations and much of it is published, or being prepared for publication, in related conferences.

The following articles published by the author form parts of this thesis:

[CMP18] Bisimilarity Distances for Approximate Differential Privacy. Dmitry Chistikov, Andrzej S. Murawski, and David Purser. Appeared at ATVA 2018.
This work forms the majority of Chapter 4.

[CMP19] Asymmetric Distances for Approximate Differential Privacy. Dmitry Chistikov, Andrzej S. Murawski, and David Purser. Appeared at CONCUR 2019.
This work forms the majority of Chapter 5.

[GNP20] The Complexity of Verifying Loop-Free Programs as Differentially Private. Marco Gaboardi, Kobbi Nissim, and David Purser. To appear at ICALP 2020.
This work forms the majority of Chapter 7.

Parts of this thesis are currently under submission and review for potential future publication:

[CKMP20] The Big-O Problem for Labelled Markov Chains and Weighted Automata. Dmitry Chistikov, Stefan Kiefer, Andrzej S Murawski, and David Purser.
This work forms the majority of Chapter 6.

The following research was performed in collaborations during the development of this thesis, but does not form part of the thesis:

[JP19] Structural liveness of Petri nets is ExpSpace-hard and decidable. Petr Jančar and David Purser. In Acta Inf 56.6 (2019).

[Gup+18] Twitter Usage Across Industry: A Spatiotemporal Analysis. Neha Gupta, Henry Crosby, David Purser, Stephen A. Jarvis, and Weisi Guo. Appeared at BigDataService 2018.

Abstract

This thesis contributes to the understanding of the computational complexity of verifying differential privacy. The problem is considered in two constrained, but expressive, models; namely labelled Markov chains and randomised circuits.

In the setting of *labelled Markov chains* (LMC) it is shown that most relevant decision problems are undecidable when considered directly and exactly. Given an LMC, and an ε, consider the problem of finding the least value of δ such that the chain is (ε, δ)-differentially private. Finding this value of δ can be expressed as a variant of the total variation distance. Whilst finding the exact value is not possible, it can be approximated, with a complexity between $\#\mathbf{P}$ and **PSPACE**. Instead, bisimilarity distances are studied as over-estimate of δ, which can be computed in polynomial time assuming access to an **NP** oracle and a slightly weaker distance can be computed in polynomial time.

One may also wish to estimate the minimal value of ε such that the LMC is ε-differentially private. The question of whether such an ε even exists is studied through the big-O problem. That is, does there exist a constant C such that the probability of each word in one system is at most C times the probability in the other machine. However in general this problem is undecidable but can be decided on unary chains (and is **coNP**-complete). On chains with bounded language (that is, when there exists $w_1, \ldots, w_m \in \Sigma^*$ such that all words are of the form $w_1^* \ldots w_m^*$) the problem is decidable subject to Schanuel's conjecture by invoking the first order theory of the reals with exponential function. The minimal such constant C corresponds exactly to $\exp(\varepsilon)$ and approximating this value is not possible, even when the value is known to exist. A bisimilarity distance to over-estimate $\exp(\varepsilon)$ can be computed in **PSPACE**.

In the setting of *randomised circuits*, the complexity of verifying pure differential privacy is fully captured as $\mathbf{coNP}^{\#\mathbf{P}}$-complete; formalising the intuition that differential privacy is universal quantification followed by a condition on probabilities. However verifying approximate differential privacy is between $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$, and $\mathbf{coNP}^{\#\mathbf{P}}$-complete when the number of output bits is small (poly-logarithmic) relative to the total size of the circuit. Further, each parameter cannot be approximated given the other in polynomial time (assuming $\mathbf{P} \neq \mathbf{NP}$).

Acronyms

DFA	Deterministic Finite Automaton
DP	Differential Privacy
LMC	Labelled Markov Chain
LP	Linear Programming
LRA	Linear Real Arithmetic
NFA	Non-deterministic Finite Automaton
PA	Probabilistic Automata
w.l.o.g.	Without loss of generality
w.r.t.	With respect to

Symbols

$\mathbb{N}, \mathbb{Q}, \mathbb{R}$	The sets of natural (including zero), rational and real numbers
ε	Usually denotes the main parameter for differential privacy
δ	Usually denotes the second parameter for differential privacy
α	Usually denotes e^ε
γ	Usually denotes the accuracy of approximation
Δ_α	The skewed distance
tv	The total variation distance
μ, ν	Usually denotes a generic distribution or measure
μ_s	Usually denotes the transition distribution from state s
ν_s	Usually denotes the measure on traces from state s
$\mathcal{M}$	Usually denotes a labelled Markov chain
$\mathcal{W}$	Usually denotes a weighted automaton
$\mathcal{A}$	Usually denotes a probabilistic automaton
$\mathcal{N}$	Usually denotes a non-deterministic finite automaton
Σ	Usually denotes a finite alphabet
w, u	Usually denotes a word
a	Usually denotes an alphabet character
Q	Usually denotes a finite set of states
s, q	Usually denotes a state in Q
$\mathcal{F}$	Usually denotes a σ-algebra, usually on Σ^ω
φ	Usually denotes a strongly connected component
ψ	Usually denotes a randomised circuit
ϕ	Usually denotes a Boolean formula
$\boldsymbol{x}, \boldsymbol{y}$	Usually denotes an assignment to the inputs of a Boolean formula or circuit
$\boldsymbol{r}$	Usually denotes an assignment to the probabilistically chosen bits
$\boldsymbol{o}$	Usually denotes an assignment to the output of a circuit
$\square$	Denotes the end of a main proof
$\blacksquare$	Denotes the end of a proof of claim, usually inside another proof
$\blacktriangleleft$	Denotes the end of a definition, remark or example

Chapter 1

Introduction

There is a great deal of data continually collected; much of this data is of a personal nature and an individual is likely to want their data to remain private. However there is benefit to be had by analysing such data; one could consider medical records which could be used to understand disease patterns in epidemiology. However each individual record is extremely personal, sensitive and could have real and lasting consequences should it be released. The natural approach is to take great care to "sanitise" the data for release, so that the data can be analysed in such a way that it cannot be linked back to its original subject. However this can go wrong, incorrect assumptions can be made, the data can be re-linked and privacy lost.

Differential privacy, introduced by Dwork, McSherry, Nissim, and Smith [DMNS06], is a definition of privacy typically formulated in the context of data analysis. It was originally formulated as a technique to conduct randomised computations over *statistical* databases, usually by returning noisy results in a mechanism to preserve the privacy of the subjects of the database entries. Differential privacy captures the intuition that information specific to an individual is protected if every single user's input has a bounded influence on the computation's outcome distribution. Unlike many privacy techniques such as data anonymisation, it is robust to attacks for which the attacker has additional information and post-processing so that the results of differentially private computations can be analysed further with no additional loss to privacy to build more interesting analyses. This bounded influence is specified by two parameters, usually denoted by ε and δ. Intuitively, these parameters set an upper-bound on privacy loss, where the parameter ε limits the loss and the parameter δ limits the probability with which the loss may exceed ε.

Differential privacy is currently making significant strides towards being used in large scale real-world applications. Prominent examples include the use of differentially private computations by the US Census' OnTheMap project [US 19; Mac+08], applications by companies such as Google and Apple [EPK14; Pap+17; App; Dif17], and the US Census' plan to deploy differentially private releases in the upcoming 2020 Decennial Census [Abo18].

There have been numerous high-profile failures of attempts to anonymise data by supposedly stripping out identifying information. Despite assurances that personal identifiers had been removed by then Governor Weld of Massachusetts, the release of hospital insurance records led to the identification of a hospital visit by the governor [Bar12]; this attack relied on the uniqueness of a certain combinations of demographic attributes (such as age, gender, zip code, etc.). Another such attack was performed by "re-linking" a Netflix viewing database with publicly available IMDb reviews, thus revealing what else these reviewers had watched in the "private" Netflix data [NS06]. AOL released the search history of 600,000 users, although they had replaced the AOL username with a random number, one could determine who they were by what they searched for [Arr06]. In another, the destination locations of celebrities' taxi journeys were discovered by matching paparazzi photos (with known start time/location) to a list of journeys [Tro14]. Attempts to use more sophisticated methods to maintain the privacy of the data subjects, such as the addition of noise, but short of differential privacy, have also been shown to be vulnerable to attack [Gad+19].

A large body of literature has generated many algorithms intended to release data in a differentially private way. A standard collection of algorithms enables the release of counting queries, histograms, (noisy) maximums, proportional selection and more. These are presented in various forms, and composed together in various ways to produce new and more useful algorithms. Composition allows the construction of complex differentially private mechanisms from simpler ones, using the latter as building blocks. This, however, comes with a degradation in the privacy guarantee, and a collection of *composition theorems* provide a calculus for bounding the privacy loss of a differentially private analysis as a function of the privacy loss of each of its components. While the existence of such a calculus is a major benefit of differential privacy, it leads in practice to conservative estimates, and an exact calculation of privacy loss parameters is generally out of reach.

However, not all mechanisms rely upon this compositionality, there are a range of more sophisticated mechanisms, which can answer more specialised queries; and have analysis independent of this compositionality. The sparse vector algorithm [Dwo+09] is such an example; for which a much tighter analysis can be provided than a standard compositionality approach may derive.

There are three main models of differential privacy. The original *curator model* requires all of the data to be held by a trusted party, and for this party to conduct queries on the data, adding noise in a suitable way and then release the data publicly. The second *local model* applies when each data subject randomises their data before handing it over, so that the party collecting the data can essentially do what they like with the data. This local model requires significantly more noise to be added to each data point to obtain the same level of privacy, but can assume a lower level of trust. This model has been gaining traction amongst the large technology companies, who have started to employ such methods in their products and devices as it is usually simple to implement. Increasingly gaining traction, as an intermediate model, is the *shuffle model* [Che+19; Erl+19], which allows each data subject to add some noise locally, and then a trusted party will shuffle data so that it is not clear who sent each message; after which the data can be released in full. This third model requires less noise, but also less trust than the curator model as the shuffler only needs to do this relatively easy permutation and then can delete the data, rather than trusting the curator indefinitely.

More often than not, algorithms and their implementations are analysed "on paper" to show that they provide differential privacy. This analysis—a proof that the outcome distribution of the algorithm is stable under the change in any single individual's information—is often intricate and may contain errors (see [LSL17] for a discussion about several wrong versions of the sparse vector algorithm that appeared in the literature). Moreover, even if it is actually differentially private, an algorithm may be incorrectly implemented when used in practice, e.g. due to coding errors, or because the analysis makes assumptions which do not hold in finite computers, such as the ability to sample from continuous distributions (see [Mir12] for a discussion about privacy attacks on naive implementations of continuous distributions). Verification tools may help validate, given the code of an implementation, that it would indeed provide the privacy guarantees it is intended to provide.

Extensive work has occurred in the computer-assisted or automated verification of differential privacy. Early work includes, PINQ [McS09] and Airavat [Roy+10] which are systems that keep track of the privacy budgets (ε and δ) using trusted privacy primitives in SQL-like and MapReduce-like paradigms respectively. In other work, programming languages were developed, that use the type system to keep track of the sensitivity and ensure the correct level of noise is added [RP10; BKOB12; DAn+13; BGHP16]. Another line of work uses proof assistants to help prove that an algorithm is differentially private [Bar+16b]; although much of this work is not automated, recent work has gone in this direction [AH18; ZK17]. Recent works have focused on developing techniques for finding violations of differential privacy [Din+18; Bic+18; GM18], although these can be seen as forms of testing, rather than verification.

However, despite the many verification efforts that have targeted differential privacy, based on automated or interactive techniques, little is known about the complexity of some of the basic problems in this area. The aim of this thesis is to complement these verification efforts with a greater understanding of the feasibility of *automated* verification of differential privacy, and clarify the complexity of some of these problems.

It is understood that verifying differential privacy is a *too complex* task to be approached through a brute force procedure. Clearly by Rice's theorem it is undecidable in full generality, although if the language is sufficiently expressive it is already undecidable on models with a single input and single output [Bar+20]. Hence, tools that have been developed for this task [BGHP16; FJ14; ZK17; AH18] focus typically on "soundness", rather than "completeness", that is, they work only on a subclass of programs for which they are optimised, or focus on interactive verification where the support of a human verifier is required to simplify the verification steps. Nevertheless, the clear understanding of what "too complex" means, in this setting, is still missing; in targeting constrained models this thesis helps clarify this. To this end, this thesis will conduct complexity analysis on the computational complexity of verification on *labelled Markov chains* and *randomised circuits* as abstractions of programs.

The computational complexity approach focuses on classifying problems according to the resources required to solve the problem, regardless of the algorithm used to solve the problem. Problems are usually shown to be in the class by exhibiting an algorithm using the resources allowed by the class. Problems are complete for some class when any other problem in the class reduces to it. This indicates that both require the same level of resource, up to some factor, and thus a breakthrough for any one of the problem would entail a breakthrough for all of them. There are many problems which are classified, it is well known that SAT is **NP**-complete for example, and its generalisation to arbitrary quantifier alternation is **PSPACE**-complete; yet there are many more that are not yet known to be complete for any class. Whilst there is a large taxonomy of classes, there are many examples where it is not known whether they are the same or different. The most well known such question is whether $\mathbf{P} = \mathbf{NP}$, but it is not even known whether **P** is different from **PSPACE**. This thesis will focus on understanding how the verification of differential privacy fits in the wider picture of computational problems.

Despite its initial definition in the context of private query answering on statistical databases, differential privacy is a widely studied notion of privacy for various models of computation. It is not limited to statistical queries, and its principles can be adapted to a range of scenarios in which something should be kept hidden from a (possibly malicious) observer of the outcomes of the program, system or device. For example, consider a protocol which should not, by virtue of its sequence of actions, reveal (much) information about the information it is exchanging to those observing it. Differential privacy allows one to quantify the privacy loss between two scenarios one would like to be indistinguishable.

From a verification perspective, a natural question is how to analyse systems with respect to (ε, δ)-differential privacy. This thesis contributed to the understanding of the computational complexity of verifying differential privacy. Since such decision problems would be undecidable for a full programming language, the problem is considered in two constrained, but expressive, models, namely labelled Markov chains (LMCs) and randomised circuits. Many of the traditional verification techniques are not amenable to complexity analysis as they are either not automated, or incomplete (not guaranteed to terminate or find a solution).

LMCs are abstractions of autonomous systems with probabilistic behaviour with partial observability. States of an LMC $\mathcal{M}$ can be thought of as generating probability distributions on sets of traces, and these traces are taken to correspond to observable events. An LMC will be (ε, δ)-differentially private between two states (configurations) s and s' if, the distributions on traces from these states are sufficiently close. From this perspective, this thesis will build upon total variation based [Kie18; CK14] and bisimulation based techniques [TKD11; CGPX14], for which there is scope for studying the complexity and furthering the development of such distances.

Randomised circuits are an abstraction of straight line programs, that is, programs without branches or loops, operating at the level of individual bits and Boolean operations. Such models have a finite state space and thus almost all questions are likely to be decidable, if necessary by total exhaustion of the possible inputs, outputs and probabilistic behaviour. However, these models can admit a more concise representation of the state space. This thesis studies the tractability of verifying such models as differentially private and the extent to which such total exhaustion is, in the worst case, necessary.

1.1 Contributions

Verification of labelled Markov chains: Markov chains are a standard formalism for studying probabilistic programs, protocols and control flows. A probabilistic program can be converted to a labelled Markov chain by expanding its control state; and in general possibly resulting in an infinite system. However for a constrained class of programs the resulting Markov chain is finite. *Labelled* Markov chains are considered here, covering programs which provide output throughout their computation.

The problem of automatically verifying both pure and approximate differential privacy in *labelled Markov chains* is studied and the decidability and complexity of the relevant problems is classified. That is, given a labelled Markov chain, one would like to find the relevant parameters ε and/or δ which the labelled Markov chain minimally achieves so that one can assess whether it attains an acceptable level of privacy. These parameters can be captured by distances between states. This thesis addresses the extent to which these distances can be computed within the framework of labelled Markov chains, and where direct consideration is not feasible or possible, proposes new distances to overestimate the parameters and studies the complexity of computing these new distances.

Symmetric Distances for δ: The relevant distance to compute δ in a labelled Markov chain is considered from two perspectives. Firstly a symmetric variant is considered, for which a particular distance forms a sound upper bound on δ. This distance is based on bisimilarity pseudometrics, which are a way to quantify the behavioural differences between states of labelled Markov chains. This particular variant of the bisimilarity distance is shown to be always rational, the associated threshold problem is in **NP**, and the distance can be computed exactly with polynomially many calls to an **NP** oracle.

Asymmetric Distances for δ: The distance is improved by separating the distance into an asymmetric variant, in the sense the corresponding bisimilarity distance produces a better approximation, which is simpler to express and can also be computed in polynomial time with an **NP** oracle. A further distance which can be computed in polynomial time is also presented.

Considering the distance of direct interest for δ, it is shown that the relevant threshold problem is undecidable; thus entailing that it cannot be computed exactly. However, it is shown that approximating the distance is #**P**-hard and in **PSPACE**, matching the complexity results for the more specialised, but well-known, total variation distance.

The full lattice of relevant distances for δ can be seen in Figure 5.1 on page 67.

Distances for ε: Towards studying ε, the big-O problem for labelled Markov chains is considered; this condition specifies there exists a constant C, such that probability of every word from one state must be no greater than C times the corresponding probability from the other state.

This minimal such constant C corresponds exactly with computing the minimal value of $\exp(\varepsilon)$ such that ε-differential privacy is satisfied. This can also be seen through the lens of a ratio variant of the total variation distance to capture $\exp(\varepsilon)$. Deciding the big-O problem will turn out to be undecidable, corresponding to deciding whether ε is bounded. Whilst the exact answer for δ is not computable exactly, it can be approximated. However, unlike for δ, the distance for ε (or the optimal constant of the big-O problem) cannot be approximated.

These undecidability results rely on reasonably general chains, leaving the question open for more specialised machines; to this end it is shown that the big-O problem is **coNP**-complete for unary chains and for chains with bounded language the problem is decidable subject to a well-known conjecture.

A bisimilarity distance to overestimate ε exists [CGPX14], and it is shown that by slight modification a bisimilarity distance to overestimate $\exp(\varepsilon)$ can be approximated in **PSPACE**.

Verification of randomised circuits: Circuits are another standard formalism of programs. Here, a probabilistic variant is considered, which in addition to having access to the input bits of the program also have access to a further set of bits which are determined randomly. Directly, circuits cover the class of straight line programs, although more generally algorithms with a bound on the length of their computation can be unrolled to become straight line programs.

The problem of automatically verifying both pure and approximate differential privacy in *randomised circuits* is studied and the complexity of the relevant problems is classified.

This gives rise to the problem of checking whether such circuits are differentially private. Two variants of the problem are considered, the decision problem of checking, given ε or (ε, δ), whether the circuit meets the given level of differential privacy; and the problem of approximating the minimal ε or δ. The complexity of these questions are considered.

Verifying pure differential privacy: It is shown that determining whether a randomised circuit is ε-differentially private is $\mathbf{coNP}^{\#\mathbf{P}}$-complete. Hardness is shown by the complement to the problem E-Maj-Sat [LGM98], which is complete for $\mathbf{NP}^{\#\mathbf{P}}$ [CDM17]. In the complementary problem, All-Min-Sat, given a formula ϕ over $n+m$ variables the task is to determine if for all allocations $\boldsymbol{x} \in \{0,1\}^n$, $\phi(\boldsymbol{x}, \boldsymbol{y})$ evaluates to true on no more than $\frac{1}{2}$ of allocations to $\boldsymbol{y} \in \{0,1\}^m$.

Verifying approximate differential privacy: For the case where $\delta > 0$, it is shown that determining whether a randomised circuit is (ε, δ)-differentially private is $\mathbf{coNP}^{\#\mathbf{P}}$-complete when the number of output bits is small (poly-logarithmic) relative to the total size of the circuit and otherwise between $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$.

Approximating the parameters ε and δ: Efficient approximation algorithms exist for optimal composition [MV16], and one might expect the existence of polynomial time algorithms to approximate ε or δ. It is shown this is **NP**-hard and **coNP**-hard, and therefore an efficient algorithm does not exist (unless $\mathbf{P} = \mathbf{NP}$).

Chapter 2

Background

2.1 Differential privacy

Technically, differential privacy is based on measuring differences between probability distributions. The following definition sets out the formal definition in the context of statistical databases.

Definition 2.1. Let $M : \mathcal{D} \to \mathit{Dist}(\mathcal{E})$ be a randomised mechanism, where $\mathcal{D}$ is the set of possible inputs (datasets) and $\mathcal{E}$ is the set of possible outcomes. Then M is (ε, δ)-differentially private if for any two neighbouring databases $D_1, D_2 \in \mathcal{D}$ and for any output subsets (event $E \subseteq \mathcal{E}$) it is the case that:

$$\mathbb{P}[M(D_1) \in E] \leq e^{\varepsilon} \cdot \mathbb{P}[M(D_2) \in E] + \delta. \qquad \blacktriangleleft$$

The notion that a database is neighbouring means the database differs in one record. This usually means one record could be removed from one of the databases to obtain the other; although in some works it can mean one record is altered or replaced by another record. By composition properties of differential privacy (see Section 2.1.2), the privacy parameters change by a factor of two between the two definitions, by considering replacement as one removal and one subsequent addition.

Differential privacy ensures that a small perturbation of the input leads to only a small perturbation in the output, so that observing the output makes it difficult to determine whether a particular piece of information was present in the input.

The above formulation is often called *approximate differential privacy.* For $\delta = 0$, one talks about (pure) *ε-differential privacy.* Note that then the above definition boils down to measuring the ratio between the probabilities of possible outcomes and can be simplified to quantification over any output $r \in \mathcal{E}$: $\mathbb{P}[M(D_1) = r] \leq e^{\varepsilon} \cdot \mathbb{P}[M(D_2) = r]$. It is often the case that one cannot expect to achieve pure *ε-differential privacy*, for which the relaxed approximate differential privacy is then used [Mei18].

The parameters ε and δ are used to control the level of leakage which is acceptable. There is much debate as to what is an appropriate level of these parameters [Hsu+14], however that is not the topic of this thesis. Intuitively, one could interpret δ as an indicator of the extent to which ε-differential privacy holds for the given states; (ε, δ)-differential privacy as "ε-differential privacy with probability at least $1 - \delta$" [Vad17]. This, perhaps, remains a useful way to think of the relative importance of ε and δ, but it is not strictly correct. Indeed this "probabilistic differential privacy" (ε-differential privacy with probability $1 - \delta$) is considerably weaker than approximate differential privacy [Mei18].

There are great deal of extensions to differential privacy, aimed at overcoming various shortcomings of pure and approximate definitions of privacy. However, this thesis focuses on these two "standard" definitions which are the most used in the literature. Examples of these extended definitions are Renyi differential privacy [Mir17] and concentrated differential privacy [DR16].

2.1.1 Laplacian and exponential mechanisms

There are many differential privacy techniques and algorithms, two fundamental mechanisms can be used in simple cases: the Laplacian mechanism in continuous space and the exponential mechanism in discrete space.

The *Laplacian mechanism* M is defined by taking $M(D_1) = f(D_1) + N$ where $f : D \to \mathbb{R}$ is the function to compute the true answer, and N is random noise drawn from the Laplace distribution with mean 0 and variance $\frac{S}{\varepsilon}$. Here S is the sensitivity of f, the largest difference to the true answer which can be caused by a change to a single record. In a counting query the sensitivity is 1, a record's presence can contribute at most one by being present or not. In a summation query the sensitivity is the largest absolute value of the items to be added.

Theorem 2.2. *[DMNS06] The Laplacian mechanism is ε-differentially private.*

Whilst the Laplacian obscures continuous valued outputs, it is not appropriate when the output is from a discrete set; in these cases the *exponential mechanism* can be used. This uses a score function $s : \mathcal{D} \times \mathcal{E} \to \mathbb{R}$, where $\mathcal{D}$ is the dataset and $\mathcal{E}$ is the set of possible outcomes. The score function should return the suitability of that output to that dataset. The exponential mechanism returns by choosing a response $r \in \mathcal{E}$ with probability proportional to $\exp\left(\frac{\varepsilon}{2S} s(D, r)\right)$, where S is the analogue of sensitivity: $S = \max\{s(D_1, r) - s(D_2, r)\}$, where the maximum ranges over neighbouring $D_1, D_2 \in \mathcal{D}$. This results in an ε-differentially private mechanism.

Computational Issues When implementing these methods on finite computers it is not possible to truly sample from the continuous distributions, thus it is necessary to make some rounding assumptions due to representation, which leads to a slightly different mechanism for which the initial analysis may not apply.

Further, sampling the exponential mechanism is computationally difficult. It can also be problematic because one must ensure every possible answer has non-zero probability, and so all possible outputs must be known [SY13]. As a result of these issues, when implemented in practice it is sometimes the case that simplifications have to be made. For example, in practice the distribution may be sampled from a Markov chain, such that if the Markov chain is fully mixed it will satisfy ε-differential privacy, but otherwise may satisfy (ε', δ')-differential privacy for some ε', δ'.

2.1.2 Composition

A fundamental concept of differential privacy is composition. This allows the composition of two differentially private mechanisms to be the sum of their differential privacy parameters. That is, given mechanisms $M_1 : D \to \mathbb{R}, M_2 : D \times \mathbb{R} \to \mathbb{R}$ which are $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$-differentially private respectively then we have $M_2(D, M_1(D))$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differentially private.

In the case where $\delta_1 = \delta_2 = 0$, the composition is optimal, however this basic composition fact is an overestimate of the parameters for $\delta_1, \delta_2 > 0$. Thus advanced composition theorems have been considered [DRV10; MV16], for example if M is composed from $M_1, \ldots, M_k$, each (ε, δ)-differentially private and $k < \frac{1}{\varepsilon^2}$, then M is $(O(\sqrt{k \log(\frac{1}{\delta'})}) \cdot \varepsilon, k\delta + \delta')$-differentially private for all $\delta, \delta' > 0$.

Parallel composition allows the combination of results from independent subsets of the database to be applied, taking only the maximum of the privacy losses of the components. Suppose M_1 is ε_1 differentially private and M_2 is ε_2 differentially private then, on dataset D, any function $g(M_1(D \setminus A), M_2(A))$ is $\max(\varepsilon_1, \varepsilon_2)$-differentially private.

Using a combination of the Laplacian and exponential mechanisms, with composition theorems it is possible to build up larger differentially private mechanisms, knowing that each application of a query can be used by adding the parameters. These theorems form the basis of many more advanced differentially private algorithms.

Complexity of Composition

Murtagh and Vadhan [MV16] showed that finding the optimal values for the privacy parameters when composing different algorithms in a black-box way is #**P**-complete. That is, consider the composition of k differentially private algorithms with privacy parameters $(\varepsilon_1, \delta_1), \dots, (\varepsilon_k, \delta_k)$. The resulting program is $(\varepsilon_g, \delta_g)$-differentially private for a multitude of possible $(\varepsilon_g, \delta_g)$ pairs. Murtagh and Vadhan showed that determining the minimal ε_g given δ_g is #**P**-complete [MV16]. Despite the theoretically poor complexity of finding the exact answer, it is possible to approximate the optimal value efficiently.

However, the resulting answer is not necessarily optimal; it could be the initial analysis was not tight for each sub-mechanism and then it is possible that a better answer is possible. Further some mechanisms (e.g. in the case of the sparse vector technique [Dwo+09]), due to the specific make up of their composition, can have a better privacy guarantee than a naïve composition argument of this form would provide. Thus it is desirable to consider general purpose verification systems which can handle more complex mechanisms.

Relation to this thesis This thesis will, in Chapter 7, consider the problem of computing the optimal ε or δ in a white-box setting, in particular where the mechanisms are specified as circuits. This generalises the work of [MV16] by allowing the composed mechanisms by to be specified as circuits.

2.1.3 The relation to quantitative information flow

Differential privacy has similarities with quantitative probabilistic information flow [AACP11], which is an entropy-based theory measuring how secure a program is. Checking that a program does not have probabilistic information flow is equivalent to checking that a program is 0-differentially private. For loop free Boolean programs with probabilistic choice, this problem is **coNP**-complete [YT10]. Comparing the quantitative information flow of two programs on inputs coming from the uniform distribution is #**P**-hard [YT10]. However, when quantifying over all distributions the question is **coNP**-complete [YT10]. Checking whether the quantitative information flow of a program is less than a threshold has been shown to be **PP**-hard [YT11] and in **PSPACE** for loop-free Boolean programs and to be **PSPACE**-complete for Boolean programs with loops [CKV14b].

2.2 Programming language based verification

The purpose of a verification routine to verify differential privacy is to ensure that a mechanism attains its claimed level of privacy. A naïve "verification algorithm" can check the compositions of several Laplacian mechanisms by appropriately tracking the parameters of each application. Naturally, various levels of sophistication may be used to keep track of the level of privacy and give tighter guarantees.

A common verification technique involves using the programming code to keep track of more information than simply the commands. A common example of this is to specify "types", which may be useful as a verification procedure but not always strictly necessary for the execution. Another example is annotations in Java. Similar techniques can be used to keep track of relevant parameters for privacy; either through an extended type system directly or through further annotations. This can then keep better track of the level of privacy attained, for instance one may track the sensitivity of each function call.

Airavat [Roy+10], PINQ [McS09], ProPer [ESS15], Fuzz [RP10] and DFuzz [Gab+13] are all programming language techniques, with built in language features to ensure differential privacy.

PINQ [McS09] is one of the first attempts to create a system which promises to maintain differential privacy. It uses a SQL like query language to define queries, but extends the SQL programming language with additional query keywords such as NoisySum (instead of sum) and NoisyCount (instead of count), which in addition to performing the standard sum or count also applies additional noise from the Laplacian distribution. By supporting sequential and parallel composition, more advanced queries can be built. Whilst an interesting early example, it is a reasonably basic system with limited expressive power, for example it does not cover approximate differential privacy ($\delta > 0$). Recently, DPella, a new Haskell implementation for PINQ-like queries has been introduced [VRG20].

On a similar vein Airavat [Roy+10] is another environment to keep track of the privacy budget and decide whether a given query is suitable. It uses a MapReduce framework, with built-in primitives which apply the correct level of noise to the output of each computation. It is mainly built on compositional techniques from differential privacy and requires that any new components or more complex algorithms are checked and trusted before being incorporated into the system.

Both PINQ and Airavat maintain an environment to automatically decide whether to run a query, which can be vulnerable to security attacks. Haeberlen, Pierce and Narayan [HPN11] show that these systems are vulnerable to timing attacks by observing how soon the system decided not to run the query. Another possibility is an attack on the privacy budget, by watching the effect of a query on the privacy budget, or even the decision to let the query run at all. One of their proposals to overcome the problems is to use default values, which are used when the true value is not allowed to be released. This covers up the fact that the true value was not returned and avoids terminating early.

Fuzz and DFuzz extend the type system by introducing privacy parameters into the types, this allows the algorithms to be statically checked, as part of the type check during compiling, before running. The program is then deemed to satisfy differential privacy if a successful type check is accomplished. DFuzz extends this further by allowing dependent types, so that these privacy parameters can depend on the inputs and program state, such as the loop counter, or number of queries. Both systems can be used to verify a differentially private version of the k-means clustering algorithm. However, in Fuzz the number of iterations of the k-Means algorithm would need to be specified up front, whereas in DFuzz this could be a parameter, which would then effect the amount of noise added to maintain the privacy guarantee.

All of these techniques require additional work by the program writer to ensure that the additional annotations/types are correctly inserted to enable a successful verification. These programming language based techniques do not go much beyond the composition theorem, and more complex differential privacy proofs may need more powerful tools. Relational logic techniques can help with this.

2.3 Relational logic based verification

A long line of research [BGB09; BKOB12; BO13; BKOB13; Bar+14; Bar+15b; Bar+15a; Bar+16a; BGHP16; Bar+16b] studies the relational verification of differential privacy, using techniques to relate program memories in such a way that guarantees privacy. These tools are generally less automated than other techniques, but may have more expressive power; pushing the boundaries of machine checked proofs rather than machine generated proofs. In particular, significant level of annotations are required, for which the author must have a full understanding of the algorithm's privacy proof. Following a series of papers and tools it was determined that the work had a very close relationship with couplings, a notion of probability that allows two distributions to be related.

Of particular interest is the introduction of a skewed distance Δ_α, with $\alpha = e^\varepsilon$, which will feature heavily throughout this thesis.

Definition 2.3 (Skewed Distance [BKOB12]). For $\alpha \geq 1$, let $\Delta_\alpha : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be defined by $\Delta_\alpha(x, y) = \max\{x - \alpha y,\ y - \alpha x,\ 0\}$. ◀

This distance can be used to reformulate differential privacy: a mechanism

M is (ε, δ)-differentially private, if for every D_1, D_2 neighbouring and every possible output E we have $\Delta_{e^\varepsilon}(\mathbb{P}[M(D_1) = E], \mathbb{P}[M(D_2) = E]) \leq \delta$.

The starting point is to use Hoare logic [Hoa69] as a formalism to prove the correctness of computer programs using pre-conditions and post-conditions to reason about each statement of the program. This can be extended with probabilistic reasoning and relational logic to prove properties between two programs, or for the case of differential privacy, two runs of the same program.

Initial work in this line starts with CertiCrypt, a relational verification tool for cryptographic protocols [BGB09]. This uses pRHL, probabilistic relational Hoare logic, which uses judgements of the form $\vdash G_1 \sim G_2 : \Psi \implies \Phi$; representing that executions of programs G_1 and G_2 are related when the pre-condition Ψ is satisfied and will also satisfy post-condition Φ. This is combined with suitable proof rules, such as for composition, if-then-else statements, etc. to build up larger programs or protocols to reason about.

Barthe, Köpf, Olmedo and Zanella-Béguelin [BKOB12; BKOB13] then extended the pRHL logic to approximate probabilistic Hoare logic (apRHL), which supports approximate differential privacy. Here the judgements are annotated with differential privacy parameters $\alpha = e^\varepsilon$ and δ such as $\vdash \mu_1 \sim_R^{(\alpha,\delta)} \mu_2 : \Psi \implies \Phi$, where μ_1, μ_2 are distributions over outputs. This requires that for the relation $R \in A \times B$, there exists an appropriate "approximate" coupling which corresponds to, or lifts, the relation R. This coupling μ must relate μ_1, μ_2 using the appropriate distance controlled by differential privacy.

A coupling of a pair of distributions is a probability distribution over a pair of sets (a joint distribution) whose marginals on each side are the same as the original distributions. Formally $\mu \in D(A \times B)$ is a coupling of $\mu_1 \in D(A)$ and $\mu_2 \in D(B)$ if $\sum_{b \in B} \mu(a, b) = \mu_1(a)$ for every $a \in A$ and $\sum_{a \in A} \mu(a, b) = \mu_2(b)$ for every $b \in B$. Couplings can be used to relate two runs of probabilistic programs. This means that both the first and second programs act exactly as they were defined, since the relevant part of the distribution acts exactly as before. However, now reasoning can be performed on just one program, over the joint distribution [Bar+15a]. This notion must be generalised for approximate differential privacy, whereas in a standard coupling one ensures that the distributions behave the same, in an approximate lifting there is a bound on the distance. One such notion requires two witnesses μ_L and μ_R, such that given $\mu_1 \in \mathit{Dist}(A)$ and $\mu_2 \in \mathit{Dist}(B)$ there exists a coupling if there are μ_L and μ_R in $\mathit{Dist}(A \times B)$ such that $\mu_1(a) = \sum_{b \in B} \mu_L(a, b)$ for all $a \in A$ and $\mu_2(b) = \sum_{a \in A} \mu_R(a, b)$ for all $b \in B$ with $\sup_{E \subseteq A \times B} \Delta_\alpha(\mu_L(E), \mu_R(E)) \leq \delta$.

The logic apRHL is included in the tool EasyCrypt, which can be used to verify differential privacy proofs. The tool relies upon a series of proof rules—most of these are routine, such as sequential composition and if statements. The most complicated rules deal with while loops, which may need to cope with different invariant for different parts of the procedure (before and after the one different record), or cope with running one fewer or greater times than the other run (since the database size may vary by exactly one record).

These techniques are used in a tool called HOARe^2 using higher order relational refinement types [Bar+15b; Bar+14] and implemented using SMT-solvers. The tool combines the techniques from apRHL and DFuzz and completely covers all examples that are provable with DFuzz. Examples in this work show the differential privacy of the *dual query release* mechanism and the *private counter* mechanism.

The *sparse vector mechanism* is a particularly interesting algorithm for which there was a long running attempt to formally verify as differentially private. This is because it does not use purely compositional reasoning about differential privacy; that is, it allows more queries to be answered than a naïve analysis of the privacy budget would suggest it pays for. It does this by refusing to answer queries whose answer falls below a threshold and answering a limited number which fall above the threshold. The privacy budget is only used on the queries falling above the threshold, despite information being learnt about the ones below the threshold. Both the threshold and query answers have noise added to allow the information release. Many incorrect versions of the algorithm have been presented in literature; a common fault is to use the same noise when deciding whether to release, as the noise in the release, yet in reality the noise should be sampled again.

In 2016, using the couplings techniques of [Bar+15a], the correctness of the *sparse vector mechanism* was verified [Bar+16b], a task that had previously not been possible under existing techniques. They furthered this work by verifying the *between thresholds* algorithm, a generalisation of the *sparse vector* mechanism [Bar+16a] as well as the *exponential* mechanism and the *above threshold* mechanism.

Automation The primary drawback with each of these methods is the lack of automation. Significant effort is required to come up with the relevant relations, couplings and annotations to the program. To that end, Albarghouthi and Hsu [AH18] provided techniques to synthesise proofs in the framework of couplings, automating some parts of the process by using SMT solvers to generate the couplings required and was successfully applied to the *noisy max* and the *sparse vector* mechanisms. On a related line, Smith and Albarghouthi [SA19] try to automatically synthesise queries which satisfy differential privacy by design and within the query budget.

The tool LightDP [ZK17] provides another custom language for showing differential privacy, with an aim at requiring less annotation than other relational methods; whilst being powerful enough to work with techniques that use more than simple compositional theorems (as is the case of PINQ, Fuzz etc.). They also claim it is more powerful than bisimulation based techniques and finds the lowest cost differential privacy proof amongst all proofs by invoking SMT solvers. However, Barthe et al. [Bar+16a] claims that the system is in effect also using couplings but limited to simple bijection based couplings, thus limiting the expressiveness. The method remains far from a plug and play system, and relies upon programmers to carefully annotate the code with invariants.

Relation to this thesis This thesis complements this line of research by taking an algorithmic verification-centred approach; that is, a fully automated approach, albeit on a more formalised representation of the programmes (namely labelled Markov chains and circuits) rather than working at the level of program code directly. The distances defined in the relational verification work, in particular Definition 2.3, will be heavily be relied upon. There will be further links to couplings, of a slightly different form, used in the development of distances for approximate differential privacy.

2.4 Verification on probabilistic transition systems

Tschantz, Kaynar and Datta [TKD11] first studied differential privacy using a notion similar to bisimulation, which was extended to a more general class of bisimulation relations by Xu, Chatzikokolakis and Lin [XCL14]. Both consider only ε-differential privacy and are approached from a proof technique perspective, thus do not examine how these could be computed. Chatzikokolakis, Gebler, Palamidessi and Xu [CGPX14] have advocated the development of Kantorovich pseudometrics, instantiated with any metric distance function (rather than absolute value) in the context of differential privacy. Like the earlier bisimulation based distances of Tschantz et al. and Xu et al. [TKD11; XCL14] the pseudometric of Chatzikokolakis et al. [CGPX14] was presented as proof technique, rather than automated methods for computation; and hence these papers did not discuss the complexity of calculating these relations and pseudometrics. Moreover, it was left open whether it was possible to extend the pseudometric techniques to (ε, δ)-differential privacy [Xu15]; a question which is answered by this thesis.

Early relational techniques The first paper to appear to use bisimulation for differential privacy, described as a non-interference property, is that of Tschantz et al. [TKD11], however they do not describe it as bisimulation. Instead they define a parameterised relation R_ε, where for each ε two states of an automaton are related if all computation from that point on only use ε of their privacy budget. Starting at R_ε each time some privacy is lost the ε parameter decreases, the goal is to verify this cannot decrease below R_0, to satisfy ε-differential privacy. Each transition of the automaton may therefore use some of the privacy budget. This is somewhat similar to the notion of probabilistic bisimulation, which is formalised in later work of Xu et al. [XCL14].

Tschantz's [TKD11] goal is not to verify that an individual mechanism is differential private, for example the Laplacian mechanism, but to consider if an online algorithm obtains a certain level of privacy by invoking black box ε-differentially private mechanisms. For example is the overall system $n \cdot \varepsilon$-differentially private for some n. The automaton operates by taking inputs, which could consist of either new data or a query, and returning an output for each query asked. Their example shows that the *Truncated Geometric mechanism* is differentially private. A problem with this approach forces each use of the privacy budget to increase by the same amount, losing accuracy. Methods are provided to verify that a proposed bisimulation relation is valid and thus satisfies ε-differential non-interference. However, it does not provide a method to find such a relation automatically—it is instead considered a proof technique, for which the person constructing the proof must come up with the relation. The work has been extended by Xu [XCL14] to amortised bisimulations, where the privacy parameter can be incremented or decremented at each stage, allowing greater expressivity.

Bisimulations Bisimulations, introduced by Park and Milner [Mil89], are a common technique in the verification literature to ensure a program is consistent with its specification. Bisimulations verify that two programs behave the same, by allowing transitions only with the same actions (formal definition to follow). In differential privacy the idea is to show that two programs (actually neighbouring runs of the same program) are *nearly* the same and relaxations of bisimulation techniques can be used to show this.

Definition 2.4. A relation R is a *bisimulation relation* if and only if $(s, s') \in R$ implies that for all transitions $s \xrightarrow{x} q$ there exists $s' \xrightarrow{x} q'$ such that $(q, q') \in R$ and for all transitions $s' \xrightarrow{x} q'$ there exists $s \xrightarrow{x} q$ such that $(q, q') \in R$. ◀

Two states s, s' are said to be *bisimilar*, denoted $s \sim s'$, if there exists a bisimulation relation R such that $(s, s') \in R$. Usually the largest bisimulation relation is considered as $\sim = \bigcup \{R : R \text{ is a bisimulation relation}\}$, for which any bisimilar pair must be in.

A more intuitive definition of bisimulation proceeds as a two player game, with an attacker and defender. At each round the attacker can pick a transition in either machine, and the defender must match it by picking a transition with the same label in the other machine. The next round of play continues from the new states of the two machines, where the attacker can then again pick from either machine. The attacker wins by making a transition which the defender cannot match. The defender wins by reaching a state in which the attacker has no transitions to make, or infinitely matching the attacker's moves. The system is bisimilar if the defender always has a winning strategy.

Bisimulation is a stronger notion than language equivalence, that is, two states s, s' are capable of producing the same words. All bisimilar pairs of states are language equivalent, but language equivalent pairs of states need not be bisimilar. One can consider bisimilarity to be a behavioural equivalence, where as language equivalent pairs may be able to produce the same language but require different behaviours to produce the same words. Bisimilarity is sometimes used in place of language equivalence, particularly when there are efficient techniques to decide bisimilarity but language equivalence is intractable. In deterministic systems (e.g. a deterministic finite automaton), the two notions coincide, as there is exactly one transition available for each action, and thus one behaviour. Figure 2.1 demonstrates a non-deterministic finite automaton that is language equivalent but not bisimilar.

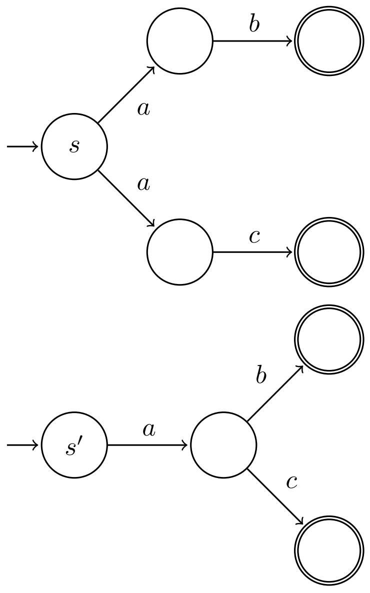


Figure 2.1: States s, s' are language equivalent but not bisimilar. The choice of b, c as the second action is brought forward in the first system, so that an attacker may move in the second machine choosing the action that is disabled in the first.

Probabilistic Bisimulation Probabilistic bisimulations generalise the standard deterministic notion by additionally requiring that the probabilities match when transitions are taken.

Definition 2.5. [LS89] A *probabilistic bisimulation* is an equivalence relation R such that for $(s,t) \in R$ and each equivalence class E of R it is the case that $\mathbb{P}[s \xrightarrow{a} E] = \mathbb{P}[t \xrightarrow{a} E]$, where $\mathbb{P}[s \xrightarrow{a} E]$ is the probability of transitioning to a state in class E from state s with action a. ◀

Probabilistic bisimulations are also closed under union and hence there exists a largest one. Two states are called *bisimilar*, written $s \sim s'$, if the pair (s, s') belongs to some probabilistic bisimulation, or equivalently the largest probabilistic bisimulation.

There is considerable literature on the computation of probabilistic bisimulations, culminating in Chen, van Breugel and Worrell [CBW12] showing that deciding probabilistic bisimulation on Markov chains is **P**-complete.

Behavioural Pseudometrics Research into behavioural pseudometrics has a long history going back to Giacalone, Jou and Smolka [GJS90]. Bisimulation pseudometrics based on the Kantorovich distance were started by Desharnais, Jagadeesan, Gupta and Panangaden [DJGP02; DGJP04], as a metric analogue of classic probabilistic bisimulation [LS91]. The original motivation was to overcome the problem that bisimilarity is too sensitive to minor changes in probabilities. Such robustness is highly desirable, because probabilistic systems arising in practice may often be based on approximate probability values, extracted or learnt from real world data. Such distances are pseudometrics have the property that $d(s,t) = 0$ if and only if $s \sim t$ [BW14].

The standard bisimilarity distance has been the subject of intense efforts to find techniques to compute it [Bre17]. van Breugel, Sharma and Worrell [BSW07; BSW08] identified that it could be approximated using the existential fragment of the first order theory of the real numbers. That is, by reducing to deciding the truth status of a sentence of first order logic with existential quantifiers and logical combinations of inequalities of arithmetic expressions of real variables.

On probabilistic automata the bisimilarity distance can be computed in the complexity class **PPAD** [BW14]. In the quest to pin down the complexity for labelled Markov chains Chen, van Breugel and Worrell [CBW12] finally showed that the distance is **P**-hard and could be computed in polynomial time using the ellipsoid method. However the algorithm is not suited to implementation and there are ongoing efforts to improve the speed of realistic algorithms using iterative methods and speed-ups [Bac+19; TB18; TB17; BBLM17; TB16;

BBLM13].

The bisimilarity pseudometric can be used as an upper bound on the total variation distance [CBW12] defined as $tv(\nu, \nu') = \sup_{E \subseteq \Sigma^*} \nu(E) - \nu'(E)$. Recall, one can think of bisimilarity as defining behavioural equivalence, a stronger condition than language equivalence. This lifts to quantitative measures, where total variation is an analogue of language equivalence and the stronger bisimilarity distance quantifies behavioural similarity. Since language equivalence is more likely to consider states equal than behavioural equivalence, total variation is never a larger value than the bisimilarity distance.

2.4.1 Bisimilarity pseudometrics for ε-differential privacy

When studying Markov chains, the value of ε (to achieve ε-differential privacy) can be captured by a variant of the total variation

$$tv_{\ln}(\nu, \nu') = \sup_{E \in \Sigma^*} \max \left\{ \ln(\nu(E)) - \ln(\nu'(E)), \ln(\nu'(E)) - \ln(\nu(E)) \right\},$$

where ν, ν' are the measures on traces produced from two neighbouring states. This can be seen as a generalisation of the total variation distance, with the absolute value function replaced with the distance $d_{\ln}(x, x') = |\ln(x) - \ln(x')|$. Chatzikokolakis et al. [CGPX14] generalised bisimulation distances to arbitrary metrics, in the sense that the absolute value function is replaced by any metric. A *metric* between objects in X is defined as follows:

Definition 2.6. A metric $d : X \times X \to \mathbb{R}$ satisfies the following conditions:

- (zero between self) for all x $d(x, x) = 0$,
- (non-negative) for all x, x' $d(x, x') \geq 0$,
- (indistinguishable at zero) for all x, x' $d(x, x') = 0 \implies x = x'$,
- (symmetric) for all x, x' $d(x, x') = d(x', x)$,
- (triangle inequality) for all x, x', x'' $d(x, x'') \leq d(x, x') + d(x', x'')$. ◀

A *pseudometric* is a distance which satisfies all of the above, except indistinguishable at zero; meaning that two different objects can have distance zero.

One may wish to compare the difference between two measures on X; to do this one uses the distance between the ground objects in X and lifts this to a distance on measures. One such lifting is the Kantorovich lifting

$$K(d)(\mu, \mu') = \sup_{\substack{f : X \to [0,1] \\ |f(x) - f(x')| \leq d(x, x')}} \left| \int_X f(\mu) d(\mu) - \int_X f(\mu') d(\mu') \right|.$$

The standard bisimilarity distance uses the Kantorovich lifting and is defined as a distance $m : S \times S \to [0,1]$ as the least fixed point of the function $F : [0,1]^{S\times S} \to [0,1]^{S\times S}$ defined as:

$$F(m)(s,t) = \max_{a\in A}\left\{ \sup_{s\xrightarrow{a}\mu} \inf_{t\xrightarrow{a}\nu} K(m)(\mu,\nu), \sup_{t\xrightarrow{a}\nu} \inf_{s\xrightarrow{a}\mu} K(m)(\mu,\nu) \right\}.$$

Here S refers to the set of states, A refers to the set of actions and $s \xrightarrow{a} \mu$ refers to all of the transitions labelled with action a, defining a distribution μ over the next state.

In the classical Kantorovich lifting, the absolute value function is used in two positions; first to quantify the difference in expectation and secondly to quantify the difference in any two positions as a Lipschitz condition. Chatzikokolakis et al. [CGPX14] replaces both applications of this absolute value function with any metric distance function $d_V : S \times S \to \mathbb{R}$. This can then define a new bisimilarity distance as the least fixed point bm_V of

$$F_V(m)(s,t) = \max_{a\in A}\left\{ \sup_{s\xrightarrow{a}\mu} \inf_{t\xrightarrow{a}\nu} K_V(m)(\mu,\nu), \sup_{t\xrightarrow{a}\nu} \inf_{s\xrightarrow{a}\mu} K_V(m)(\mu,\nu) \right\}$$

on the Kantorovich distance

$$K_V(m)(\mu,\mu') = \sup_{\substack{f:X\to Y \\ d_V(f(x),f(x'))\leq m(x,x')}} d_V\left(\int_X f(\mu)d(\mu), \int_X f(\mu')d(\mu')\right).$$

They then go on to show that this bisimilarity distance bm_V is a bound on the analogous total variation distance $\sup_{E\subseteq\Sigma^*} d_V(\nu(E),\nu'(E))$ [CGPX14]. In particular, applying this to the distance $d_{\ln}(x,x') = |\ln(x) - \ln(x')|$ gives that the corresponding bisimilarity distance $bd_{\ln}$ is an upper bound on $tv_{\ln}$ and thus the privacy parameter ε. This then needs to be considered over all pairs of states representing neighbouring databases.

The distances considered are summarised in Table 2.1.

Name	Metric Distance	Total Variation	Bisimilarity Distance	Functor	Kantorovich
Standard	$\lvert\ldots\rvert$	tv	bm	F	K
Generic	d_V	tv_V	bm_V	F_V	K_V
Multiplicative Variant	$d_{\ln}$	$tv_{\ln}$	$bm_{\ln}$	$F_{\ln}$	$K_{\ln}$

Table 2.1: Table of total variation distances and bisimilarity distances induced by different metrics.

Relation to this thesis There exist linear programming techniques to compute the multiplicative variant of the Kantorovich function [CGPX14], but there the complexity of computing $bm_{\ln}$ is not known. This, along with the direct consideration of $tv_{\ln}$, are open questions addressed in Chapter 6.

In their line of work, significant emphasis is placed on the resulting distance $bm_{\ln}$ being a pseudometric [CGPX14; Xu15]. Further, the zero points are stressed to correspond with bisimilarity; that is $s \sim s' \iff bd_{\ln}(s, s') = 0$. It was hypothesised that by using Δ_α (of Definition 2.3) the distance could be extended to approximate differential privacy to capture δ [Xu15]. However this was not developed further, and it should be noted that Δ_α is not a metric and so does not fit in the framework applicable to all metrics. This is a further compelling direction which this thesis will develop in Chapter 4 and Chapter 5. In this work the distances will not be pseudometrics, as they will not satisfy the triangle inequality, and in Chapter 5 will not satisfy symmetry either.

2.4.2 Other techniques

Fully automated methods Very recently Barthe, Chadha, Jagannath, Sistla and Viswanathan [Bar+20; Rav19] show that, in general, differential privacy is, unsurprisingly, undecidable. However their proof shows that this is the case even for programs with a single input and a single output. Further, they show that differential privacy can be decided for the programming language DiPWhile in an automated way. This language is a restricted class of programs supporting while operations and the Laplacian, discrete Laplacian and Exponential mechanisms, with a syntactic restriction ensuring each loop is bounded. Such programs are then converted to parameterised discrete time Markov chains; which allow some of the transitions to be parameterised so that they depend on the value of the privacy parameter ε. A tool DiPC is provided which can verify such programs and demonstrated on several compelling examples such as *noisy max*, *randomised response* and the *sparse vector technique*. The tool also provides a counter example when it finds a violation to differential privacy.

Property Testing In the line of property testing, Gilbert and McMillan [GM18] consider the problem of testing whether a system satisfies differential privacy in a “black-box” setting. That is when access to the source code is not provided and the application can only be understood by trying various combinations and observing their outcomes. They provide bounds on the number of queries that need to be tested to determine whether the system is differentially private. This work supports four versions of differential privacy; pure and approximate-differential privacy and their randomised variation (ε-differentially private with probability γ or (ε, δ)-differentially private with probability γ). The work suggests that the bounds obtained are infeasible for “usual” choices of parameter.

Model checking Liu, Wang and Zhang [LWZ18] show another technique by defining a temporal logic dpCTL*, operating on Markov chains and Markov decision processes. The system allows the specification of temporal properties for which paths satisfying these temporal properties must satisfy (ε, δ)-differential privacy; in the sense the probability of observing a path with this property must be similar to the probability of a path with this property from a neighbouring state.

The full definition requires quantification over all possible events, however this work requires each event that should enjoy privacy to be specified as a temporal path formula. This limits both the expressiveness (as not all events are expressible), and the completeness (since only the specified events are verified).

Markov chains can be verified against a dpCTL* formula with the same complexity as CTL* which is well-known to be **PSPACE**-complete [BK08, Theorem 6.89]. However, dpCTL* is undecidable for Markov decision processes. The techniques of Liu et al. [LWZ18] are demonstrated on a range of differential privacy mechanisms including randomised response, the truncated geometric mechanism, sub-sampling majority and the above threshold mechanism.

Relation to this thesis Since the general problem is undecidable, it is required that any (terminating) decision procedure have some limitation; the system will not be complete in some sense. In the case of Liu et al. [LWZ18] this is quantifying over all events, since this would entail a general purpose technique for the verification of differential privacy, something which Barthe et al. [Bar+20], and the work in this thesis, indicates is undecidable in the context of Markov chains. The work of this thesis will complement the work of Liu et al. [LWZ18] by being able to quantify over all events and the work of Barthe et al. [Bar+20] by not restricting the class of Markov chains, however accuracy will be lost. That is, Chapters 4 and 5 will develop methods to confirm (ε, δ')-differential privacy for some $\delta' > \delta$, where δ is the minimal satisfying (ε, δ)-differentially privacy.

Chapter 3

Common Definitions

This chapter introduces definitions required across several chapters; more specialised definitions are delayed until their use.

3.1 Basic notation

3.1.1 Set operations

Given sets $X_1, \dots, X_n$, their Cartesian product is

$$X_1 \times \cdots \times X_n = \{(x_1, \dots, x_n) \mid x_i \in X_i \text{ for } i \in \{1, \dots, n\}\} .$$

As a shorthand X^i is used to mean the Cartesian product of X with itself i times. For example $\{0,1\}^2 = \{(0,0),(0,1),(1,0),(1,1)\}$. If X is a set of symbols then X^* is the set of all finite sequences of symbols from X, X^+ all finite sequences of length at least one, and X^ω all infinite sequences.

When X is finite, the cardinality of a set X is denoted by $|X|$, that is the number of elements in the set. For example $\left|\{0,1\}^2\right| = 4$. X^Y denotes $X^{|Y|}$ with elements of $x \in X^Y$ indexed by elements of Y, rather than $\{1, \dots, |Y|\}$; or used as a function $f : Y \to X$. Elements of $X^{Y \times Y}$ are viewed as matrices, or functions $f : Y \times Y \to X$.

A vector is stochastic if its elements are non-negative and sum to 1, then given a finite set X, let $\mathit{Dist}(X)$ be the set of all stochastic vectors in $[0,1]^X$. A matrix is stochastic if every row, viewed as a vector, is stochastic.

The symmetric difference of a set is denoted by $X \ominus X'$, that is the elements in either X or X', but not both. Let $\mathcal{P}(X)$ denote the power set of X, that is the set of all subsets of X.

3.1.2 Representations of numbers

The set of integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$, is denoted by $\mathbb{Z}$. The natural numbers are the non-negative integers $\{0, 1, 2, \dots\}$, denoted by $\mathbb{N}$.

Integers are usually assumed to be given in binary, or where explicitly stated may be in unary.

A *number given in binary* means a number of the form $\frac{x}{2^y}$ for x, y integers, where x is given in binary and y is indicated by the position of the '.'. For example $1.15625_{10} = 1.00101_2 = \frac{100101_2}{2^5} = \frac{37}{2^5}$.

A *number given as a rational* means a number of the form $\frac{x}{y}$ where x, y are given as binary integers. Such numbers are denoted by $\mathbb{Q}$.

An *algebraic number* is a complex number that is a root of a non-zero polynomial in one variable with rational coefficients. Such numbers are assumed to be represented by the polynomial and a sufficiently close rational representation in both the real and complex space, such that no other root of the polynomial could be confused with it. Formally, an algebraic number z can be represented by a tuple $(p_z, a, b, r) \in \mathbb{Q}[x] \times \mathbb{Q}^3$. Here p_z is a polynomial with rational coefficients over x and a, b, r form an approximation to disambiguate between all other roots: more precisely z is the only root of $p_z(x)$ with $|z - (a + bi)| \leq r$. This representation admits standard operations (addition, multiplication, absolute value, (in)equality testing) in polynomial time [BPR05; Coh13; Pan96] (see e.g. [OW14, Section 3] for a succinct summary).

3.1.3 Approximation and computation of numbers

In this thesis, an *approximation problem* means, given an error $\gamma > 0$, find any value $\hat{x}$ such that $|\hat{x} - x| \leq \gamma$, where x is the target value; that is, the error is additive. One can also consider approximation problems where the error is a multiplicative, that is, to find any value $\hat{x}$ where $\frac{1}{\gamma} \leq \frac{\hat{x}}{x} \leq \gamma$; where this notion is intended it will be stated explicitly.

A number, typically the solution of a problem, is *approximable* if there exists an approximation algorithm to find it. Note that a number which is only *approximable* does not admit inequality testing. For example, if the approximation results in $0.0000\ldots$, how long should a procedure wait before deciding whether the value is or is not strictly positive?

The notion of approximation here requires a sufficiently close representation with no uncertainty in the closeness of the estimate. This is in contrast to the following notion, given two error parameters $\gamma, \lambda > 0$ which should find any $\hat{x}$ satisfying $|\hat{x} - x| \leq \gamma$ with probability at least $1 - \lambda$, and can be arbitrarily bad with probability at most λ. This notion of approximation is *not* considered in this thesis.

A number being *computable* refers to there being a terminating procedure to find a finite, "sensible" representation of the *exact* value. Indeed the problem definition and input could be used to refer to the *exact* value, but this would not admit several key properties one would expect from a representation; namely equality testing, inequality testing and the ability to write out an arbitrary number of digits. One would also expect to be able to perform basic arithmetic, such as addition, subtraction, multiplication, (possibly division), and result in another number of the same representation. Whilst other representations would fit these conditions, in this thesis, when a number is claimed to be *computable* this will mean that one can find a representation of the exact number in binary, as a rational or as an algebraic number. When it is claimed to be *not computable*, one of these properties will be missing (usually inequality testing).

Some authors, notably Turing [Tur37], use *computable* to refer only to approximation i.e. an arbitrary number of digits can be found by the procedure on request. However this does not (necessarily) admit (in)equality testing, and this thesis considers this to be only *approximable*.

Hence the solution to a problem can be *computable* (and thus also approximable), *approximable* (but not necessarily computable), or *inapproximable*.

3.2 Models of automata and Markov chains

One of the main models of study in this thesis are labelled Markov chains. Labelled Markov chains come in many different varieties, they can operate over finite or infinite words and they can be labelled on the state or on the transition. This thesis will present different results on different variants of the model. At their core, labelled Markov chains assign measure to words; by generating words from their states, transitioning from one state to the next in a Markovian manner, that is the choice of the next state depends only on the current state, and emitting a character at each step.

Labelled Markov chains can be seen as a restriction on weighted automata; so that the weightings behave in a probabilistic manner. In this section the definition of weighted automata will be introduced, enabling the definition of labelled Markov chains in the context of finite words on labelled transitions. For the treatment of infinite words, the heavier machinery of measure theory is required and introduced next. Finally the treatment of labelled states and labelled transitions will be shown to be unimportant as such systems can be effectively translated in polynomial time; thus not affecting the complexity of decision questions. Hence each problem to be addressed in the thesis will use the most convenient formulation.

3.2.1 Weighted automata

Definition 3.1 (Weighted Automata). A weighted automaton $\mathcal{W}$ over the $(\mathcal{S}, \oplus, \otimes)$ semi-ring is tuple $\langle Q, \Sigma, M, F\rangle$ where Q is a finite set of states, Σ is a finite alphabet, $M : \Sigma \to \mathcal{S}^{Q\times Q}$ is a transition weighting function and $F \subseteq Q$ is a set of final states. $\mathcal{W}$ defines a weighting function $\nu_s : \Sigma^* \to \mathcal{S}$ for every state $s \in Q$, where for a word $w = a_1 \dots a_n \in \Sigma^*$:

$$\nu_s(a_1 \dots a_n) = \bigoplus_{t\in F} (M(a_1) \otimes M(a_2) \otimes \cdots \otimes M(a_n))(s, t)$$

and $(A \otimes B)(i, j) = \bigoplus_{k\in Q} A(i, k) \otimes B(k, j)$ is matrix multiplication over $(\mathcal{S}, \oplus, \otimes)$. ◀

In this thesis only *non-negative* $(\mathbb{Q}, +, \times)$ *weighted automata* are considered, i.e. for all $q, q' \in Q, a \in \Sigma$: $M(a)(q, q') \geq 0$. In this case, $A \times B$ is the standard matrix multiplication. All transition probabilities are assumed to be rational (recall these are represented as a pair of binary integers). For a machine $\mathcal{W}$, *size*$(\mathcal{W})$ is the number of bits required to represent each component, including the bit size of the weights or probabilities.

Note that the definition does not specify an initial state, rather defines the weighting function ν_s from every state. Transitions will typically be described by $q \xrightarrow[a]{p} q'$ to mean $M(a)(q, q') = p$, with the assumption that any unspecified transition has weight zero.

Without loss of generality, a weighted automaton can have a single final state. If the final state is not unique, one can introduce a new unique final state t s.t. $M(a)(q, t) = \sum_{q'\in F} M(a)(q, q')$ for all $q \in Q$,$a \in \Sigma$.

A *unary weighted automaton* is a *non-negative* $(\mathbb{Q}, +, \times)$ *weighted automaton* with $|\Sigma| = 1$. Without loss of generality it is assumed that $\Sigma = \{a\}$ and Σ is omitted from the description. Further the transitions are described with a single matrix $A = M(a)$ and $q \xrightarrow{p} q'$ is written to mean $A(q, q') = p$. Then the weight of the word a^n from state s is $\nu_s(a^n) = A^n_{s,t}$ where t is the unique final state. Here, and throughout the thesis, when $A^n_{s,t}$ is written it means $(A^n)(s, t)$ and not $(A(s, t))^n$.

3.2.2 Finite word labelled Markov chains

Labelled Markov chains be described using non-negative $(\mathbb{Q}, +, \times)$ weighted automata with additional properties such that the transition weights rather become probabilities.

Definition 3.2. A *labelled Markov chain* (LMC) $\mathcal{M} = \langle Q, \Sigma, M, F\rangle$ is non-negative $(\mathbb{Q}, +, \times)$ weighted automaton such that $\sum_{q'\in Q}\sum_{a\in\Sigma} M(a)(q,q') = 1$ for all $q \in Q$. ◀

In this definition it is not possible to assume a unique final state, as the construction above would destroy the stochasticity constraint. However the construction can be performed and the chain interpreted only as a weighted automaton.

A Markov chain can be seen as a generator of words, where the configuration moves from one state to the next by observing the probability on the last state. Using $\nu_s(w)$ inherited from weighted automata one can refer to the probabilities of words. If all of the states are accepting, then one can refer to the probability of observing a word w. If some are accepting, then one can refer to the probability of being in an accepting state whilst observing a word w. In this general definition the Markov chain continues operating after visiting an accepting state; however one can also consider the case that the Markov chain "announces" the end of the word, and one can only refer to the probability of "ended" words. This is achieved by the Markov chain moving to a designated final state.

Definition 3.3. A *word terminating labelled Markov chain* is a non-negative $(\mathbb{Q}, +, \times)$ weighted automaton with the restriction that:

- $\sum_{q'\in Q}\sum_{a\in\Sigma} M(a)(q,q') = 1$ for all $q \in Q \setminus F$, and
- $M(a)(q,q') = 0$ for all $q \in F, q' \in Q, a \in \Sigma$ (i.e. final states have no outgoing transitions).

In addition to the measure on finite words $\nu_s(w)$, the measure can be extended to measure sets of finite words ($E \subseteq \Sigma^*$) so that $\nu_s(E) = \sum_{w\in E}\nu_s(w)$. ◀

Any word terminating labelled Markov chain has the property that the measure is a *sub-distribution* over words, i.e. $\sum_{w\in\Sigma^*}\nu_s(w) \le 1$. Note that for these models one can again assume, without loss of generality, that there is a unique final state. A *word terminating labelled Markov chain* can be embedded into a *labelled Markov chain*; the missing probability at sink states can be used by transitioning to a new sink state which is not accepting and loops on itself with probability 1 on any character (so cannot reach any accepting state).

In the same way as for weighted automata, one can define a *unary Markov chain* or *unary word terminating labelled Markov chain* with $|\Sigma| = 1$, where the transition probabilities are described by a single matrix, say A. Note that a *unary Markov chain* is syntactically equivalent to a *unary probabilistic* *automaton* (probabilistic automata are defined next), or what one may simply call a *Markov chain*.

Probabilistic Automata

Weighted automata can also be used to define probabilistic automata. Probabilistic automata are similar to labelled Markov chains, except that $M(a)$ is stochastic for every a, rather than $\sum_{a\in\Sigma} M(a)$ being stochastic.

Definition 3.4. A *probabilistic automaton* $\mathcal{A} = \langle Q, \Sigma, M, F\rangle$ is a *non-negative* $(\mathbb{Q}, +, \times)$ *weighted automaton*, with the restriction that $\sum_{q'\in Q} M(a)(q, q') = 1$ for all $q \in Q$ and $a \in \Sigma$. ◀

When a probabilistic automaton is combined with a dedicated starting state $q_s \in Q$, then $\mathbb{P}_{\mathcal{A}}(w)$ is written to denote the weighting function $\nu_{q_s}(w)$, or where the choice of probabilistic automaton is clear, simply $\mathbb{P}(w)$ is written.

3.2.3 Infinite word labelled Markov chains

In the previous subsections labelled Markov chains were introduced as weighted automata on finite words. Such a formulation does not extend in the context of infinite words (over non-singleton alphabets), as it is not appropriate to ask the probability of a single infinite word; every individual word may have measure zero, hence it is necessary to consider what is meant by "measurable" sets.

Infinite word labelled Markov chains are introduced using labelled states as this is the formulation convenient for Chapters 4 and 5, however in the next section it will become clear that the distinction is unimportant.

Definition 3.5. An *infinite word labelled Markov chain* (LMC) $\mathcal{M}$ is a tuple $\langle Q, \Sigma, \mu, \ell\rangle$, where Q is a finite set of states, Σ is a finite alphabet, $\mu : Q \to \mathit{Dist}(Q)$ is the transition function and $\ell : Q \to \Sigma$ is the labelling function. For a state $s \in Q$, μ_s is written for $\mu(s)$. ◀

The relevant probability spaces are specified next using standard measure theory [Bil86; BK08]. First the general definitions of measures are introduced and these are then used to associate the relevant probabilities associated with sets of infinite sequences of labels generated by a labelled Markov chain.

Given a set X, a measure is used to associate a value to its subsets. To do this a measurable space $(X, \mathcal{F})$ is used, where $\mathcal{F}$ is a σ-algebra, that is, the measurable subsets of X. A set $\mathcal{F}$ is a *σ-algebra* on set X if it includes X, is closed under complement and closed under countable unions. This is an extension of an *algebra*, which also requires X and closure under complement, but only need to be closed under finite unions (instead of countable unions). Then a function $\mu : \mathcal{F} \to [0, \infty]$ is a measure if $\nu(\emptyset) = 0$ and the for any countable collection of pairwise disjoint sets $E_1, E_2, \dots \in \mathcal{F}$ we have $\nu(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \nu(E_i)$. A probability measure is a measure with $\nu(X) = 1$.

Definition 3.6. A subset $C \subseteq \Sigma^\omega$ is a *cylinder set* if there exists $u \in \Sigma^*$ such that C consists of all infinite sequences from Σ^ω whose prefix is u. Then C_u is written to refer to this cylinder C. ◀

Cylinder sets play a prominent role in measure theory in that their finite unions can be used as a generating family (an algebra) for the set $\mathcal{F}_\Sigma$ of measurable subsets of Σ^ω (the cylindrical σ-algebra). Where clear from context, Σ is omitted in the subscript of $\mathcal{F}$. It is important here that any measure ν on $(\Sigma^\omega, \mathcal{F}_\Sigma)$ is uniquely determined by its values on cylinder sets (see [Bil86, Chapter 1, Section 2] or [BK08, Section 10.1]). Definitions 3.7 and 3.8 describe how to assign a measure ν_s on $(\Sigma^\omega, \mathcal{F}_\Sigma)$ to an arbitrary state of an LMC $\mathcal{M}$.

Definition 3.7. Given $\mathcal{M} = \langle Q, \Sigma, \mu, \ell \rangle$, let $\mu^+ : Q^+ \to [0, 1]$ and $\ell^+ : Q^+ \to \Sigma^+$ be the natural extensions of the functions μ and ℓ to Q^+, i.e. $\mu^+(s_0 \cdots s_k) = \prod_{i=0}^{k-1} \mu_{s_i}(s_{i+1})$ and $\ell^+(s_0 \cdots s_k) = \ell(s_0) \cdots \ell(s_k)$, where $k \geq 0$ and $s_i \in Q$ $(0 \leq i \leq k)$. Note that, for any $s \in Q$, we have $\mu^+(s) = 1$, by assuming the product over the empty set is 1. Given $s \in Q$, let $\mathit{Paths}_s(\mathcal{M})$ be the subset of Q^+ consisting of all sequences that start with s. ◀

Definition 3.8. Let $\mathcal{M} = \langle Q, \Sigma, \mu, \ell \rangle$ and $s \in Q$. Define $\nu_s : \mathcal{F}_\Sigma \to [0, 1]$ to be the *unique measure* on $(\Sigma^\omega, \mathcal{F}_\Sigma)$ such that for any cylinder C_u we have $\nu_s(C_u) = \sum \mu^+(p)$ where the summation is over all $p \in \mathit{Paths}_s(\mathcal{M})$ such that $\ell^+(p) = u$. ◀

3.2.4 Transition labelled vs state labelled LMCs

The presentation of finite word machines involves the use of labelled transitions (where one considers the label to be emitted as the transition is taken) like in [CK14; Kie18] and the presentation of infinite word machines uses labelled states (where one considers the label to be emitted on arrival to the state) like in [Bre17; CBW12; BBLM13; TB16]. However there is a polynomial translation between the two notations, such that the most convenient for the result at hand can be used.

In particular, consider a transition labelled $\mathcal{M}$ of the form $\langle Q, \Sigma, M, F \rangle$ with transition described by $q \xrightarrow[b]{p} q'$. One can construct an equivalent state labelled Markov chain $\mathcal{M}'$. For each state and each label, add new state (q, a) labelled with a, such that, when $q \xrightarrow[b]{p} q'$, let $\mu_{(q,a)}((q', b)) = p$ for every $a \in \Sigma$.

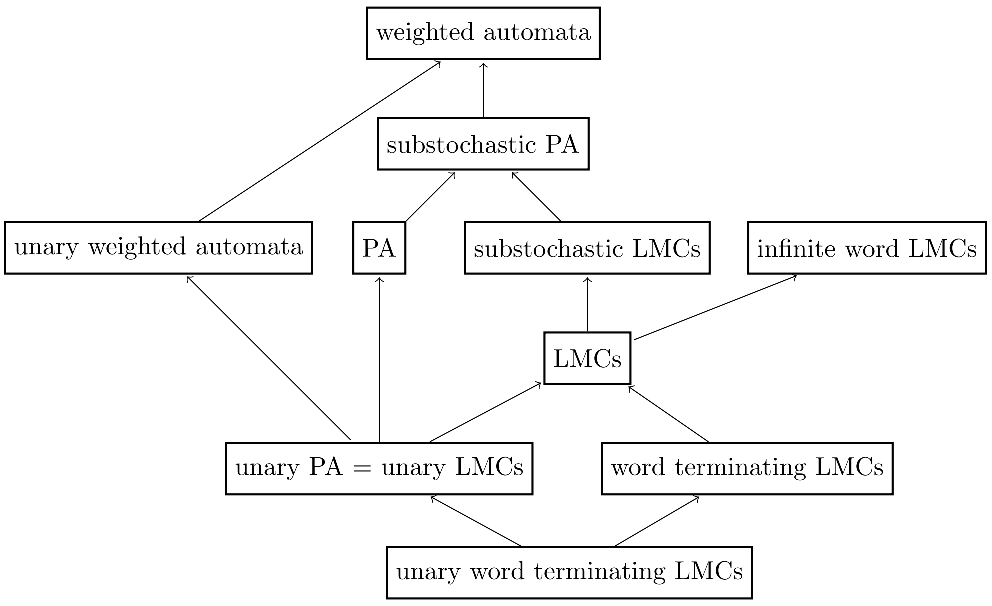


Figure 3.1: Partial order of automata, such that $A \to B$ if A can be defined as a restriction of B. The abbreviations PA denote probabilistic automata and LMC denote labelled Markov chains. A substochastic PA requires that $\sum_{q' \in Q} M(a)(q, q') \leq 1$ for every $q \in Q$ and $a \in \Sigma$ and a substochastic LMC requires that $\sum_{a \in \Sigma, q' \in Q} M(a)(q, q') \leq 1$ for every $q \in Q$.

Technically, this delays reading of the first character until the second state is visited. To account for this, introduce an additional character, say $\vdash$, so that $\nu_s(C_w) = \nu'_{(s,\vdash)}(C_{\vdash w})$, where ν and ν' refer to the measures associated with $\mathcal{M}$ and $\mathcal{M}'$ respectively (Definition 3.8).

The reverse construction from state labelled to transition labelled is even simpler: for a state q labelled with a, create a transitions of the form $q \xrightarrow[a]{p} q'$ where $\mu_q(q') = p$.

Finite word LMCs as infinite word LMCs

Finite word labelled Markov chains can also be represented by infinite word Markov chains. To embed a word terminating finite word Markov chain in an infinite word Markov chain, the end of the word can be simulated by an additional character, say \$ such that, for $q_F \in F$, $\mu_q(q) = 1$ and $\ell(q) = \$$, so that the only trace that can be observed from q_F is $\$^\omega$. Then, for a word $w \in \Sigma^*$, the word $w\$\$\$\ldots$ can be studied instead, corresponding to the cylinder $C_{w\$}$. In the translated infinite word model, the event C_u corresponds to the set of traces $\{w \in \Sigma^* \mid u \text{ is a prefix of } w\}$ in the original finite word model.

3.2.5 Relation between models

There is a relation between many of the models introduced, some can be embedded into another and some are defined directly as a restricted variant of another. The relevant relations between models are summarised in Figure 3.1. When the hardness of a problem is shown using word terminating labelled Markov chains it also applies to labelled Markov chains, infinite word Markov chains and weighted automata. When computability results are shown on weighted automata, they apply to labelled Markov chains and probabilistic automata and when a computability result is carried out in the infinite word setting it also applies to finite word chains.

Where possible, the arguments in this thesis are carried out in the strongest setting; that is, using word terminating labelled Markov chains for hardness and for computability results using infinite word Markov chains (particularly in Chapters 4 and 5) or weighted automata (particularly in Chapter 6). However, some results may apply only to the more specialised models.

3.2.6 Comparison of labelled Markov chains

The aim of the next chapters will be to compare the states of labelled Markov chains from the point of view of differential privacy. Any two states s, s' can be viewed as indistinguishable if they induce identical measures, i.e. $\nu_s(E) = \nu_{s'}(E)$ for every $E \in \mathcal{F}$; this is decidable in polynomial time [Sch61; Tze92; Kie+13]. More generally, the difference between them can be quantified using the *total variation distance*, see e.g. [GS02].

Definition 3.9 (Total Variation Distance). Let ν, ν' be measures on $(X, \mathcal{F})$, then:

$$tv(\nu, \nu') = \sup_{E \in \mathcal{F}} \left|\nu(E) - \nu'(E)\right|. \qquad \blacktriangleleft$$

Remark. The total variation distance is equivalent without using the absolute value, i.e.

$$tv(\nu, \nu') = \sup_{E \in \mathcal{F}} \nu(E) - \nu'(E). \qquad \blacktriangleleft$$

Given a Markov chain with states Q and measure ν then for $s, s' \in Q$, $tv(s, s')$ is written to refer to $tv(\nu_s, \nu_{s'})$. Ensuring such pairs of measures $(\nu_s, \nu_{s'})$ are "similar" is essential for privacy, so that it is difficult to observe which of the states was the originating position. To measure probabilities relevant to differential privacy, a more general variant of the above distance will be studied, which is introduced in Chapters 4 and 5.

$tv(s, s')$ turns out surprisingly difficult to compute: it is undecidable whether the distance is strictly greater than a given threshold, and the non-strict variant of the problem ("greater or equal") is not known to be decidable [Kie18].

Definition 3.10 (TV-THRESHOLD).

INPUT	An LMC $\mathcal{M}$, states $s, s' \in Q$ and a threshold $\theta \in [0,1] \cap \mathbb{Q}$
OUTPUT	is $tv(s, s') \leq \theta$? ◀

Theorem 3.11. *[Kie18, Theorem 3]* TV-THRESHOLD *is undecidable.*

Remark. TV-THRESHOLD is decidable for $\theta = 1$, the answer is always YES. Note also that for $\theta = 0$ it asks if the states have the same measures; recall this is decidable in polynomial time. ◀

The presentation of chains here considers the weight (or probability) of a word from an *initial state* rather than an *initial distribution* over states; leading to the presentation of distances between states rather than distributions. The problems are equivalent because the given initial distribution can be embedded into the labelled Markov chain. For this, new initial states are added, whose only operation is to simulate the initial distributions on some designated character.

3.3 Differential privacy in LMCs

Differential privacy in the context of *labelled Markov chains* requires that for two related states there only ever be a small change in output probabilities, and therefore discerning the two is difficult, which maintains the privacy of the states. In the typical database scenario, one would relate database states that differ by exactly one entry. In the context of labelled Markov chains the difference is between states of a machine, for which it is desired to be indiscernible as to which was the start state, assuming that the states are hidden and the traces are observable.

Below the definition of differential privacy is recast formally in the setting of labelled Markov chains.

Reformulation 3.12. Let $\mathcal{M} = \langle Q, \Sigma, \mu, \ell \rangle$ be a labelled Markov chain and let $R \subseteq Q \times Q$ be a symmetric relation. Given $\varepsilon \geq 0$ and $\delta \in [0,1]$, then $\mathcal{M}$ is (ε, δ)-differentially private with respect to R if, for any $s, s' \in Q$ such that $(s, s') \in R$, we have

$$\nu_s(E) \leq e^{\varepsilon} \cdot \nu_{s'}(E) + \delta$$

for any measurable set $E \in \mathcal{F}$. ◀

Remark. Note that each state $s \in Q$ can be viewed as defining a random variable X_s with outcomes from Σ^ω such that $\mathbb{P}[X_s \in E] = \nu_s(E)$. Then the above can be rewritten as $\mathbb{P}[X_s \in E] \leq e^\varepsilon\, \mathbb{P}[X_{s'} \in E] + \delta$, which matches the standard definition of differential privacy (Definition 2.1), where one would consider $X_s, X_{s'}$ neighbouring in some natural sense. ◀

What it means for two states to be related, as specified by R, is to a large extent domain-specific. In general, R makes it possible to spell out which states should not appear too different and, consequently, should enjoy a quantitative amount of privacy.

When $\varepsilon = 0$, the smallest value of δ between two states s, s' is captured exactly by the total variation distance $tv(s, s')$.

Chapter 4

Symmetric Bisimilarity Distances for δ

This chapter considers the verification of (ε, δ)-differential privacy, also referred to as *approximate differential privacy*, and designs a version of bisimilarity distance which will constitute a sound upper bound on δ, thus providing a reliable measure of privacy.

While the exact differences relevant to (ε, δ)-differential privacy are not computable in the framework of labelled Markov chains, this chapter presents a computable bisimilarity distance that yields a sound technique for measuring δ, the parameter that quantifies deviation from pure differential privacy. This bisimilarity distance is always rational, the associated threshold problem is in **NP**, and the distance can be computed exactly with polynomially many calls to an **NP** oracle.

The explicit problem is defined as follows: given an LMC $\mathcal{M}$, states s and s', and a value of ε, determine the smallest δ such that s and s' satisfy (ε, δ)-differential privacy. Unfortunately, the smallest of such δ is not computable, which motivates the search for upper bounds.

In the spirit of generalised bisimilarity pseudometrics [CGPX14], the distance, denoted bd_α, is based on the Kantorovich-style lifting of distance between states to distance between distributions. However, because the underpinning distances turn out not to be metrics, the setting does not quite fit into the standard picture, which presents a technical challenge. How the proposed distance may be computed is discussed, using techniques from linear programming, linear real arithmetic, and computational logic. The first result is that the distance always takes on rational values of polynomial size with respect to the size of the LMC and the bit size of the probability values associated with transitions (Theorem 4.29).

This is then used to show that the associated threshold problem ("is bd_α upper-bounded by a given threshold value for two given states?") is in **NP** (Theorem 4.31). Note that the distance can be approximated to arbitrary precision by solving polynomially many instances of the threshold problem.

Finally, it is shown that the distance can be computed exactly in polynomial time, given an **NP** oracle (Theorem 4.32). This places it in the search version of **NP** and leaves the possibility of polynomial-time computation open. In the subsequent chapter, the distance will be improved upon.

4.1 Preliminaries

To rephrase the inequality underpinning differential privacy in a more succinct form, it will be convenient to work with the *skewed distance* Δ_α, first introduced by Barthe et al. [BKOB12] in the context of Hoare logics and (ε, δ)-differential privacy. As a fundamental part of this chapter, Definition 2.3 is recalled here:

Definition 4.1 (Skewed Distance). For $\alpha \geq 1$, let $\Delta_\alpha : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be defined by $\Delta_\alpha(x, y) = \max\{x - \alpha y,\ y - \alpha x,\ 0\}$. ◀

Remark. It is easy to see that Δ_α is anti-monotone with respect to α. In particular, because $\alpha \geq 1$, we have $\Delta_\alpha(x, y) \leq \Delta_1(x, y) = |x - y|$. Observe that $\Delta_2(9, 3) = 9 - 2 \times 3 = 3$, $\Delta_2(9, 6) = 0$ and $\Delta_2(6, 3) = 0$. Recall the definition of a metric and pseudometric defined in Definition 2.6. Note that $\Delta_2(x, y) = 0$ need not imply $x = y$, so Δ_2 is not a metric. Note also that the triangle inequality may fail: $\Delta_2(9, 3) > \Delta_2(9, 6) + \Delta_2(6, 3)$, so Δ_2 is not a pseudometric. This complicates the technical development, because Δ_α does not fall into the framework of Chatzikokolakis et al. [CGPX14] directly. ◀

A skewed variant of the total variation distance can be used to capture differential privacy, which will be shown in Reformulation 4.3. This skewed total variation distance called tv_α, uses Δ_α, rather than the absolute value function, for which the standard total variation distance (tv) is a special case ($\alpha = 1$).

Definition 4.2 (Skewed Total Variation Distance). Let $\alpha \geq 1$. Given two measures ν, ν' on $(\Sigma^\omega, \mathcal{F})$, let

$$tv_\alpha(\nu, \nu') = \sup_{E \in \mathcal{F}} \Delta_\alpha(\nu(E), \nu'(E)).$$ ◀

Following the convention for tv, $tv_\alpha(s, s')$ will stand for $tv_\alpha(\nu_s, \nu_{s'})$. Definitions 3.12, 4.1, and 4.2 leads to the following reformulation:

Reformulation 4.3. $\mathcal{M}$ is (ε, δ)-differentially private wrt R if and only if, for all $s, s' \in Q$ such that $(s, s') \in R$, we have $tv_\alpha(s, s') \leq \delta$, where $\alpha = e^\varepsilon$. ◀

Some values of tv_α are readily known. Recall the definition of a probabilistic bisimulation defined in Reformulation 4.4, the distance between any bisimilar

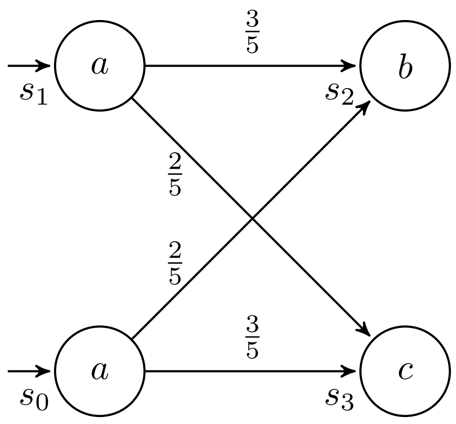


Figure 4.1: States s_0 and s_1 are not bisimilar, but $tv_{1.5}(s_0, s_1) = 0$.

states turns out to be zero. The definition is reformulated here in the context of state labelled chains:

Reformulation 4.4. A probabilistic bisimulation on an LMC $\mathcal{M} = \langle Q, \Sigma, \mu, \ell \rangle$ is an equivalence relation $R \subseteq Q \times Q$ such that if $(s, s') \in R$ then $\ell(s) = \ell(s')$ and for all equivalence classes X of R, $\sum_{u \in X} \mu(s)(u) = \sum_{u \in X} \mu(s')(u)$, i.e. related states have the same label and probability of transitioning into any given equivalence class. ◀

It follows from [CBW12, Proposition 9, Lemma 10], that for bisimilar s, s', we have $tv_1(s, s') = 0$. Then $tv_\alpha(s, s') \leq tv_1(s, s')$ entails the following:

Lemma 4.5. *If $s \sim s'$ then $tv_\alpha(s, s') = 0$, for all $\alpha \geq 1$.*

Note however that the converse does not hold, as shown in the following example.

Example 4.6. In the LMC shown in Figure 4.1, states s_0 and s_1 are *not* bisimilar. To see this, observe first that s_2 must be the only state in its equivalence class with respect $\sim$, because other states have different labels. Now note that the probabilities of reaching s_2 from s_0 and s_1 respectively are different (0.4 vs 0.6).

However, for $\alpha = 1.5$, we have $tv_\alpha(s_0, s_1) = 0$, because $\Delta_\alpha(0.6, 0.4) = \max(0.6 - 1.5 \cdot 0.4, 0.4 - 1.5 \cdot 0.6, 0) = 0$. ◀

In an "acyclic" system, tv_α can be calculated by exhaustive search in exponential time. It is convenient to note that in the case of finite words, as will be the case from an acyclic systems, we have $\sup_{E \subseteq \Sigma^*} \nu_s(E) - \alpha \nu_{s'}(E) = \sum_{w \in \Sigma^*} \max(\nu_s(w) - \alpha \nu_{s'}(w), 0)$. Hence, because there are finitely many words, in fact exponentially many, this sum and $\sum_{w \in \Sigma^*} \max(\nu_{s'}(w) - \alpha \nu_s(w), 0)$ can be computed. However, in general, tv_α is not computable, because, in particular, tv_1 is not computable (Theorem 3.11).

Thus, in the next section another distance will be introduced and studied, called bd_α. It will turn out possible to compute it and it will provide a sound method for bounding δ for $(\ln(\alpha), \delta)$-differential privacy. The main result is Theorem 4.32: the new distance can be calculated in polynomial time, assuming an **NP** oracle. Pragmatically, this means that this new distance can be computed efficiently, assuming access to an appropriate satisfiability or theory solver.

4.2 Skewed bisimilarity distance, bd_α

The distance bd_α will be defined in the spirit of bisimilarity distances [DJGP02; DGJP04; CBW12; CGPX14] through a fixed point definition based on a variation of the Kantorovich lifting. Its shape is motivated by first considering how one would go about calculating tv_α recursively.

Note that between two states with different labels, it is possible to immediately conclude that the distance should be one. That is, if $\ell(s) \neq \ell(s')$ then $\nu_s(C_{\ell(s)}) = 1$, $\nu_{s'}(C_{\ell(s)}) = 0$, therefore $tv_\alpha(s, s') = 1$.

So, assume $\ell(s) = \ell(s')$. Given $E \subseteq \Sigma^\omega$ and $a \in \Sigma$, let $E_a = \{w \in \Sigma^\omega \mid aw \in E\}$. Then:

$$\begin{aligned} tv_\alpha(\nu_s, \nu_{s'}) &= \sup_{E \in \mathcal{F}} \Delta_\alpha\left(\nu_s(E), \nu_{s'}(E)\right) \\ &= \sup_{E_{\ell(s)} \in \mathcal{F}} \Delta_\alpha \left(\sum_{u \in Q} \mu_s(u)\, \nu_u(E_{\ell(s)}), \sum_{u \in Q} \mu_{s'}(u)\, \nu_u(E_{\ell(s)}) \right). \end{aligned}$$

Define $f : Q \to [0,1]$ by $f(u) = \nu_u(E_{\ell(s)})$, then $tv_\alpha(\nu_s, \nu_{s'})$ can be rewritten as

$$\sup_{E_{\ell(s)} \in \mathcal{F}} \Delta_\alpha \left(\sum_{u \in Q} \mu_s(u)\, f(u), \sum_{u \in Q} \mu_{s'}(u)\, f(u) \right).$$

Given exact knowledge of f, it would be possible to compute tv_α. But from the definition of tv_α, it is required that $\Delta_\alpha(f(v), f(v')) \leq tv_\alpha(v, v')$ for any $v, v' \in Q$. Consequently, the following inequality holds.

$$tv_\alpha(s, s') \leq \sup_{\substack{f : Q \to [0,1] \\ \forall v, v' \in Q \Delta_\alpha(f(v), f(v')) \leq tv_\alpha(v, v')}} \Delta_\alpha \left(\sum_{u \in Q} \mu_s(u) f(u), \sum_{u \in Q} \mu_{s'}(u) f(u) \right)$$

The expression on the right is an instance of the Kantorovich lifting [Kan42; DD09], which uses ("lifts") the distance tv_α between states s, s' to define a distance between the distributions $\mu_s, \mu_{s'}$ associated with the states. Now recall the standard definition of the Kantorovich distance between distributions.

Definition 4.7 (Kantorovich lifting [Kan42]). Given $\mu, \mu' \in \mathit{Dist}(X)$ and a pseudometric $m : X \times X \to [0,1]$, the *Kantorovich distance* between μ and μ' is defined to be

$$K(m)(\mu,\mu') = \sup_{\substack{f:X\to[0,1] \\ \forall v,v'\in X\, |f(v)-f(v')|\le m(v,v')}} \left| \int f d\mu - \int f d\mu' \right|. \qquad ◀$$

Remark. Here, and throughout the thesis, the integral is sometimes specified without limits, in which it means the integral over the whole space, i.e. $\int_X f d\mu$, which indicates the expectation of f on the distribution μ. In the discrete case, e.g. $\mu \in \mathit{Dist}(Q)$, we have $\int f d\mu = \sum_{u\in Q} f(u)\,\mu(u)$.

The notation $\hat{f}(\mu)$ will be used for this integral $\int f d\mu$ (over the whole space). In particular this will be used when $\int f d\mu$ can be treated as an object with little need for its meaning; when the integral needs to be manipulated it will be translated back to the integral form. ◀

For the purposes of this chapter, a new variant of the Kantoroivch lifting is considered here, where the absolute value function $|\ldots|$ is replaced by Δ_α.

Definition 4.8 (Skewed Kantorovich). Given μ, μ' measures on $(X, \mathcal{F})$ and a symmetric distance $d : X \times X \to [0,1]$, the *skewed Kantorovich distance* between μ and μ' is defined to be

$$K_\alpha(d)(\mu,\mu') = \sup_{\substack{f:X\to[0,1] \\ \forall x,x'\in X\, \Delta_\alpha(f(x),f(x'))\le d(x,x')}} \Delta_\alpha\left(\int f d\mu, \int f d\mu' \right). \qquad ◀$$

It is assumed that whenever f is written, it is restricted only to those which are measurable in the space $(X, \mathcal{F})$. Note that whilst a symmetric distance is not necessarily required, the use of Δ_α in both places ensures it would be equivalent to instead using the symmetric distance d' where $d'(s,s') = \min\{d(s,s'), d(s',s)\}$. Hence, w.l.o.g. d is assumed to be symmetric. The next chapter will explore the case where Δ_α is not symmetric and so d is also will not necessarily be symmetric.

Further note that in the general case of $\Delta_\alpha(a,b)$, both $a - \alpha b$ and $b - \alpha a$ could be negative, so the maximum with 0 is taken. However, when taken inside the supremum $\Delta_\alpha(a,b)$ can be assumed to be $\max\{a - \alpha b, b - \alpha a\}$, dropping the maximum with zero. This is because the choice of $f(x) = 0$ for all $x \in X$ entails that the first and second component of the maximum is at least zero.

Remark. The Kantorovich distance is also known under other names (e.g. Hutchinson, Wasserstein distance), having been rediscovered several times in history [DD09]. Chatzikokolakis et al. [CGPX14] studied the Kantorovich distance and related bisimulation distances when the absolute value distance above is replaced with another metric. Note that whilst here Δ_α is considered, Δ_α is not a metric and d need not be a pseudometric.

Note that the choice of $\alpha = 1$ gives the standard Kantorovich distance (Definition 4.7). ◀

Now define a function operator, which will be used to define the distance bd_α, using K_α over measures on states, that is $X = Q$.

Definition 4.9. Let $\Gamma_\alpha : [0,1]^{Q\times Q} \to [0,1]^{Q\times Q}$ be defined as follows.

$$\Gamma_\alpha(d)(s,s') = \begin{cases} K_\alpha(d)(\mu_s,\mu_{s'}) & \text{if } \ell(s) = \ell(t) \\ 1 & \text{if } \ell(s) \neq \ell(t) \end{cases}$$ ◀

Note that $[0,1]^{Q\times Q}$ equipped with the pointwise order, written $\sqsubseteq$, is a complete lattice and that Γ_α is monotone with respect that order (larger d permit more functions, thus larger supremum). Consequently, Γ_α has a least fixed point [Tar55]. bd_α is defined to be exactly that point.

Remark. In this work, the ordering on the lattice is such that $d \sqsubseteq d'$ if and only if $d(s,s') \leq d'(s,s')$ for all $s,s' \in Q$; so that a least fixed point corresponds to "smaller" numbers than any other fixed point. This is in contrast to some authors who use $d \sqsubseteq d'$ to indicate $d(s,s') \geq d'(s,s')$ for all $s,s' \in Q$ and then study greatest fixed points. ◀

Definition 4.10 (Skewed Bisimilarity Distance). Let $bd_\alpha : Q \times Q \to [0,1]$ be the least fixed point of Γ_α. ◀

Remark. Recall that the least fixed point is equal to the least pre-fixed point ($\min\left\{d \in [0,1]^{Q\times Q} \mid \Gamma_\alpha(d) \sqsubseteq d\right\}$). ◀

Recall the initial remarks about the Kantorovich distance $K_\alpha(tv_\alpha)(\mu_s,\mu_{s'})$ over-approximating $tv_\alpha(s,s')$. This can be summarised by $tv_\alpha \sqsubseteq K_\alpha(tv_\alpha)$, i.e. tv_α is a post-fixed point of K_α. Since the intention is to bound tv_α as closely as possible, using more technical analysis it is shown that the *least fixed point* bd_α also bounds tv_α from above.

Theorem 4.11. $tv_\alpha \sqsubseteq bd_\alpha$.

Remark. The theorem is an analogue of Theorem 2 in [CGPX14]. Its proof in [Xu15] relied on the fact that the counterpart of Δ_α was a metric, which is not true here (unless $\alpha = 1$). ◀

Just like Δ_α, tv_α is anti-monotone with respect to α, so is bd_α. This means that $bd_\alpha \sqsubseteq bd_1$. The definition of bd_1 coincides with the definition of the classic bisimilarity pseudometric bm (see e.g. [CBW12]), which satisfies $bm(s, s') = 0$ if and only if s and s' are bisimilar. Consequently, entailing the following corollary.

Corollary 4.12. *For any $\alpha \geq 1$, if $s \sim s'$ then $bd_\alpha(s, s') = 0$.*

As in the case of tv_α, and in contrast to [CGPX14], the converse does not hold in the setting here. Example 4.6 shows that $s_0 \not\sim s_1$ but observe that $bd_{1.5}(s_0, s_1) = 0$.

$$\begin{aligned}
&bd_{1.5}(s_0, s_1) \quad \leq \\
&\max_f \Big(\sum_{s \in Q} f(s)(\mu_{s_0}(s) - 1.5 \cdot \mu_{s_1}(s)), \sum_{s \in Q} f(s)(\mu_{s_1}(s) - 1.5 \cdot \mu_{s_0}(s)), 0 \Big) \\
&= \max_f \big(f(s_2)(0.6 - 1.5 \cdot 0.4) + f(s_3)(0.4 - 1.5 \cdot 0.6), \\
&\qquad f(s_2)(0.4 - 1.5 \cdot 0.6) + f(s_3)(0.6 - 1.5 \cdot 0.4), 0 \big) \\
&\hfill = 0.
\end{aligned}$$

Notice the coefficients of $f(s)$ are all non-positive. Consequently, regardless of the restrictions on f, the maximising allocation will be $f(s) = 0$ and, thus, $bd_{1.5}(s_0, s_1) = 0$.

4.3 Proving Theorem 4.11

Recall Theorem 4.11 states that $tv_\alpha \sqsubseteq bd_\alpha$. To prove the theorem, a new distance will be introduced, the skewed Kantorovich distance $K_\alpha(\mathbb{1}_{\neq})$ over measures on traces, which can be planted in between tv_α and bd_α, using the following lemmas, which together entail Theorem 4.11.

Lemma 4.13. *$tv_\alpha(s, s') = K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'})$.*

Lemma 4.14. *$K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'}) \leq bd_\alpha(s, s')$.*

Definition and properties of $K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'})$

Recall Definition 4.7 is defined over any object X, thus consider the lifting instantiated on traces, i.e. $X = \Sigma^\omega$, to measure the difference between measures on traces, e.g. the measures defined from states $\nu_s, \nu_{s'}$. It is assumed that whenever f is written, it is restricted only to those which are measurable in the space $(\Sigma^\omega, \mathcal{F})$.

Definition 4.15. Let $\mathbb{1}_{\neq}(t,t') : \Sigma^\omega \times \Sigma^\omega \to \{0,1\}$ be the inequality indicator function. This function is one if the arguments are not the same and zero if they are. Also define $\mathbb{1}^h_{\neq}$ as a restriction considering only the prefix of length h, where t^h is the prefix of length h of trace t.

$$\mathbb{1}_{\neq}(t,t') = \begin{cases} 1 & \text{if } t \neq t' \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \mathbb{1}^h_{\neq}(t,t') = \begin{cases} 1 & \text{if } t^h \neq t'^h \\ 0 & \text{otherwise} \end{cases} \qquad \blacktriangleleft$$

The consider $K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'})$. The use of $\mathbb{1}_{\neq}$ is so that $\Delta_\alpha(f(t), f(t')) \leq \mathbb{1}_{\neq}(t,t)$ is rather no restriction at all, since either a trace is the same, thus $\Delta_\alpha(f(t), f(t')) = 0$ or the traces are different and $\mathbb{1}_{\neq}(t,t') = 1$. So $K_\alpha(\mathbb{1}_{\neq})$ takes supremum over all measurable f.

Proof of Lemma 4.13. $tv_\alpha(s,s') = K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'})$

Note that only $tv_\alpha(s,s') \leq K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'})$ is actually required for Theorem 4.11. However equality is shown, giving an alternative representation of tv_α.

Lemma 4.16. $tv_\alpha(s,s') = \sup_{f:\Sigma^\omega \to \{0,1\}} \Delta_\alpha\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right)$.

Proof. Note that each measurable $f : \Sigma^\omega \to \{0,1,\}$ is of the form $\mathbb{1}_E$ for some $E \in \mathcal{F}$. Then consider $E \in \mathcal{F}$ then $\mathbb{1}_E$ is a measurable function in the argument of the supremum of $K_\alpha(\mathbb{1}_{\neq})$ with $\int \mathbb{1}_E d\nu_s = \nu_s(E)$ thus $\Delta_\alpha(\nu_s(E), \nu_{s'}(E)) = \Delta_\alpha\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right)$. □

Lemma 4.17. *The following are equivalent:*

- $T(\mu,\mu') = \sup_{f:X\to[0,1]} \hat{f}(\mu) - \alpha\hat{f}(\mu')$
- $T(\mu,\mu') = \sup_{f:X\to\{0,1\}} \hat{f}(\mu) - \alpha\hat{f}(\mu')$.

Proof. The proof proceeds by first considering simple functions, and then extending this to other functions with the Monotone convergence theorem. The proof closely follows the technique of [Xu15].

A *simple function* f can be described by a finite sum $\sum_i a_i \mathbb{1}_{A_i}$, where $\mathbb{1}_{A_i}$ is the characteristic function on $A_i \in \mathcal{F}$. This ensures f takes on finitely many values in its range.

Then it is necessary to show that if $f(a) \in (0,1)$ then value of $\hat{f}(\mu) - \alpha\hat{f}(\mu')$ can only be increased by changing $f(a)$ to 0 or 1. Note this will maintain that f is valid in the restriction of the supremum, since the only restriction is to stay measurable.

Since f is simple then $img(f)$ is finite. Let $n = |img(f) \setminus \{0,1\}|$ and $f_0 = f$.

An indicator function f_n is constructed by induction. Consider $v \in img(f_i) \setminus \{0,1\}$. Let $A_i = \{x : f_i(x) = v\} \in \mathcal{F}$.

Then define $f_{i+1} = f_i + g_i$ with $g_i = t \cdot \mathbb{1}_{A_i}$ where t is defined as follows: if $\mu(A_i) - \alpha\mu'(A_i) \geq 0$ set $t = 1 - v$, this ensures that $f_{i+1}(x) = 1$ for all $x \in A_i$. Otherwise $t = -v$ which ensures that $f_{i+1}(x) = 0$ for all $x \in A_i$.

Then f_{i+1} gives no smaller value in the supremum:

$$\begin{aligned}
\hat{f}_{i+1}(\mu) - \alpha\hat{f}_{i+1}(\mu') &= \int f_{i+1} d\mu - \alpha \int f_{i+1} d\mu' \\
&= \int (f_i + g_i) d\mu - \alpha \int (f_i + g_i) d\mu' \\
&= \int f_i d\mu - \alpha \int f_i d\mu' + \int g_i d\mu - \alpha \int g_i d\mu' \\
&= \hat{f}_i(\mu) - \alpha\hat{f}_i(\mu') + t\mu(A_i) - \alpha t\mu'(A_i) \\
&\geq \hat{f}_i(\mu) - \alpha\hat{f}_i(\mu'),
\end{aligned}$$

where the last inequality holds because, either $t > 0$ and $\mu(A_i) - \alpha\mu'(A_i) \geq 0$ or $t < 0$ and $\mu(A_i) - \alpha\mu'(A_i) < 0$.

In particular, the indicator f_n is not worse than the simple function f, that is,

$$\hat{f}_n(\mu) - \alpha\hat{f}_n(\mu') \geq \hat{f}(\mu) - \alpha\hat{f}(\mu').$$

Consider a function f which is not simple. This can be approximated a sequence of simple functions $h_1, h_2, \ldots$ which are simple, converging point-wise to f. Then by monotone convergence principle [Che08, Theorems 2.4.10 and 3.1.1], with $d(a,b) = a - \alpha b$ continuous the following limit exists:

$$\lim_{n\to\infty} \hat{h}_n(\mu) - \alpha\hat{h}_n(\mu') = \hat{f}(\mu) - \alpha\hat{f}(\mu').$$

For every simple function h in the sequence, the difference is no smaller when replaced by an indicator function, thus the difference in the limit is no smaller as limit of indicator functions. Note that this limit exists as all functions considered are valid under the in the supremum of $\sup_{f:X\to[0,1]} \hat{f}(\mu) - \alpha\hat{f}(\mu')$, hence the new limit is sandwiched between $\hat{h}_n(\mu) - \alpha\hat{h}_n(\mu')$ and $\sup_{f:X\to[0,1]} \hat{f}(\mu) - \alpha\hat{f}(\mu')$ at every point.

Therefore indicator functions, which have 1-1 correspondence with events are sufficient in the supremum. $\square$

Using Lemma 4.16 and Lemma 4.17 one can derive the desired result $tv_\alpha(s, s') = K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'})$, Lemma 4.13.

Proof of Lemma 4.13. Let $\nu_s = \mu$ and $\nu_{s'} = \mu'$. In Lemma 4.16 it is observed that there is a 1–1 correspondence between indicator functions and events. Hence $tv_\alpha(s, s') = \sup_{f:X\to\{0,1\}} \Delta_\alpha\left(\hat{f}(\mu), \hat{f}(\mu')\right)$.

By Lemma 4.17 observe that extending f to non-indicator functions $f : X \to [0, 1]$ does not obtain a larger supremum.

$$\begin{aligned}
K_\alpha(\mathbb{1}_{\neq})(\mu, \mu') &= \max\left\{\sup_{f:X\to[0,1]} \hat{f}(\mu) - \alpha\hat{f}(\mu'), \sup_{f:X\to[0,1]} \hat{f}(\mu') - \alpha\hat{f}(\mu)\right\} \\
&= \max\left\{T(\mu, \mu'), T(\mu', \mu)\right\} \\
&= \max\left\{\sup_{f:X\to\{0,1\}} \hat{f}(\mu) - \alpha\hat{f}(\mu'), \sup_{f:X\to\{0,1\}} \hat{f}(\mu') - \alpha\hat{f}(\mu)\right\} \\
&= \sup_{f:X\to\{0,1\}} \max\left\{\hat{f}(\mu) - \alpha\hat{f}(\mu'), \hat{f}(\mu') - \alpha\hat{f}(\mu)\right\} \\
&= \sup_{f:X\to\{0,1\}} \Delta_\alpha\left(\hat{f}(\mu), \hat{f}(\mu')\right) \\
&= tv_\alpha(s, s').
\end{aligned}$$

$\square$

Corollary 4.18. *When using $\mathbb{1}_{\neq}$ on traces it is necessary only to consider the supremum over indicator functions with no restrictions:*

$$K_\alpha(\mathbb{1}_{\neq})(\mu, \mu') = \sup_{f:X\to\{0,1\}} \Delta_\alpha\left(\hat{f}(\mu), \hat{f}(\mu')\right).$$

Proving Lemma 4.14. $K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'}) \le bd_\alpha$

The proof strategy generally follows the strategy of [Xu15]. For all prefixes of a trace up to length h, it is shown restricting $\mathbb{1}_{\neq}$ to this prefix will provide the required bound. The main change in this argument compared to [Xu15] is a difference in the base case, where an additional result (Lemma 4.19) is needed to compensate for the fact that Δ_α is not a metric. The induction step is similar, changing only the distance function to Δ_α.

The proof is then extend from $\mathbb{1}^h_{\neq}$ to the supremum over h, i.e. $K_\alpha(\mathbb{1}_{\neq})$. To do this, the events in $K_\alpha(\mathbb{1}_{\neq})$ will be approximated by cylinders and it will be shown that for some h, $K_\alpha(\mathbb{1}^h_{\neq})$ is ε close to $K_\alpha(\mathbb{1}_{\neq})$. An additional lemma to support this is provided in Lemma 4.20 and the extension is shown in Lemma 4.22.

This extension from $K_\alpha(\mathbb{1}^h_{\neq})$ to $K(\mathbb{1}_{\neq})$ differs from the strategy of [Xu15] who argue it is enough to show continuity of $K_V(m)$ with respect to m. Continuity *does* hold in here, but it is unclear to the author that this actually shows the result. Since $\mathbb{1}^h_{\neq}$ are discrete the ε, δ formulation of continuity says at $\mathbb{1}_{\neq}$ there exists m such that $\max_{a,b} |\mathbb{1}_{\neq}(a,b) - m(a,b)| \leq \delta$, however no such $m = \mathbb{1}^h_{\neq}$ satisfies this because if it differs in any point, the difference is 1. That is, as $\left\|\mathbb{1}_{\neq} - \mathbb{1}^h_{\neq}\right\| = 1$ for all h, the sequence $\mathbb{1}^h_{\neq}$ does not converge to $\mathbb{1}_{\neq}$.

Auxiliary Lemmas for proving Lemma 4.14

Note that in this case, $\Delta_\alpha(f(x), f(x')) = 0$ cannot be used to conclude $f(x) = f(x')$, because Δ_α is not a metric. To compensate for this, a weaker result is shown that will turn out to be sufficient.

Lemma 4.19. *Consider $f : X \to [0,1]$ such that $\forall x, x' : \Delta_\alpha(f(x), f(x')) = 0$. Then $\forall \mu, \mu'$ measures on $(X, \mathcal{F})$ we have $\Delta_\alpha\left(\hat{f}(\mu), \hat{f}(\mu')\right) = 0$.*

Proof. Consider the range of values f could take, and let $a = \inf_x \{f(x)\}$. Thus, by definition, $f(x) \geq a$. It is known for all x, $\Delta_\alpha(f(x), a) \leq 0$. So, in particular, $f(x) - \alpha a \leq 0$. Therefore $f(x) \in [a, \alpha a]$ for all x.

Now consider $\int_X f d\mu$ where $f(x) \in [a, \alpha a]$ and $\int_X d\mu = 1$.

$$\hat{f}(\mu) = \int_X f d\mu \geq \int_X a d\mu = a \int_X d\mu = a$$
$$\hat{f}(\mu) = \int_X f d\mu \leq \int_X \alpha a d\mu = \alpha a \int_X d\mu = \alpha a$$

So for all x and μ then $\hat{f}(\mu) = \int_X f d\mu \in [a, \alpha a]$, hence the expectation must also lie in this range. Next notice that any two numbers in this range give distance zero, in particular the expectations. Consider $\hat{f}(\mu), \hat{f}(\mu') \in [a, \alpha a]$:

$$\begin{aligned} 0 \leq \Delta_\alpha\left(\hat{f}(\mu), \hat{f}(\mu')\right) &= \max\left\{\hat{f}(\mu) - \alpha\hat{f}(\mu'), \hat{f}(\mu') - \alpha\hat{f}(\mu), 0\right\} \\ &\leq \max\{\alpha a - \alpha a, \alpha a - \alpha a, 0\} \\ &= \max\{0, 0, 0\} = 0. \end{aligned}$$ ☐

The following lemma generalises the classic result [Hal74, Page 56. Theorem D.] that measure on any measurable event can be approximated by events from the generating set. In this case, the generating set (algebra) corresponds to finite unions of cylinders, which are themselves determined by sequences from Σ^*. Here it is shown that simultaneous approximation to the same degree of accuracy is possible for two different measures.

Lemma 4.20. *Let $(B, \mathcal{B}, \mu)$ and $(B, \mathcal{B}, \mu')$ be measure spaces over the σ-algebra $(B, \mathcal{B})$. Let $\mathcal{A} \subset \mathcal{B}$ be an algebra generating $\mathcal{B}$. Then*

$$\mathcal{S} = \left\{ X \in \mathcal{B} \quad | \quad \forall \varepsilon \ \exists A \in \mathcal{A} \text{ such that } \mu(A \ominus X) < \varepsilon \text{ and } \mu'(A \ominus X) < \varepsilon \right\}$$

also forms a σ-algebra.

Proof. The following proof is adapted from a proof for the classical single measure case [Gir12].

To show that a set S is a σ-algebra requires the inclusion of the empty set, closure under complement, and closure under countable unions. Case 1 shows the full set rather than the empty set, which shows the empty set by complement in Case 2. Case 3 shows finite unions and extends to countable union in Case 4.

Case 1. Since $B \in \mathcal{A}$, $B \in \mathcal{S}$.

Case 2. Complement: If $X \in \mathcal{S}$ and $\varepsilon > 0$ then there $\exists A \in \mathcal{A}$ such that $\mu(A \ominus X) < \varepsilon$ and $\mu'(A \ominus X) < \varepsilon$

Then $A^c \in \mathcal{A}$ and $\mu(A^c \ominus X^c) = \mu(A \ominus X) < \varepsilon$, so $X^c \in S$

Case 3. Finite Union: Let $X_1, X_2 \in S$ then $\exists A_i \in \mathcal{A}$ such that $\mu(A_i \ominus X_i) < \frac{\varepsilon}{2}$ and $\mu'(A_i \ominus X_i) < \frac{\varepsilon}{2}$.

$\mu(X_1 \cup X_2 \ominus A_1 \cup A_2) \leq \mu(X_1 \cup A_1 \ominus X_2 \cup A_2) \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$, similarly for μ' and $A_1 \cup A_2 \in \mathcal{A}$ so $X_1 \cup X_2 \in S$.

Case 4. Countable Union. Let $\{X_k\} \subset S$, pairwise disjoint and $\varepsilon > 0$. For each k, let $A_k \in \mathcal{A}$ such that $\mu(A_k \ominus X_k) \leq \frac{\varepsilon}{2^k}$ and $\mu'(A_k \ominus X_k) \leq \frac{\varepsilon}{2^k}$

Take N such that $\mu(\bigcup_{j>N} X_j) \leq \frac{\varepsilon}{2}$ and $\mu'(\bigcup_{j>N} X_j) \leq \frac{\varepsilon}{2}$. (Make $\mu(\bigcup_{j>N} X_j) \leq \sum_{j>N} \mu(X_j)$ arbitrarily small due to finite measure, take N big enough so both are μ and μ' sufficiently small).

Let $A = \bigcup\limits_{j=1}^{N} A_j \in \mathcal{A}$.

Then $(\bigcup_k X_k) \ominus A \subset \bigcup\limits_{j=1}^{N} (X_j \ominus A_j) \cup \bigcup_{j>N} X_j$

Then $\mu(\bigcup_k X_k) \ominus A) \leq \mu(\bigcup\limits_{j=1}^{N} (X_j \ominus A_j) \cup \bigcup_{j>N} X_j) \leq \sum_{j=1}^{N} \frac{\varepsilon}{2^j} + \frac{\varepsilon}{2} \leq \varepsilon$ and also for μ'.

Then $\bigcup_k X_k \in S$ $\square$

The strategy will be to prove Lemma 4.14 by induction. The following lemma shows that with the induction in place, the result will be complete as results on $K_\alpha(\mathbb{1}^h_{\neq})$ extends to $K_\alpha(\mathbb{1}_{\neq})$, by invoking Lemma 4.20.

Lemma 4.21. *$\forall \varepsilon > 0 \; \exists h \in \mathbb{N}$ such that $\left\| K_\alpha(\mathbb{1}_{\neq}) - K_\alpha(\mathbb{1}^h_{\neq}) \right\| \leq \varepsilon$.*

Proof. First observe that $K_\alpha(\mathbb{1}_{\neq}) \geq K_\alpha(\mathbb{1}^h_{\neq})$ and let $V = K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'})$, then either:

$$V = \sup_{f:\Sigma^\omega \to \{0,1\}} \int f d\nu_s - \alpha \int f d\nu_{s'}$$

or

$$V = \sup_{f:\Sigma^\omega \to \{0,1\}} \int f d\nu_{s'} - \alpha \int f d\nu_s.$$

Assume w.l.o.g. the first case and recall by Corollary 4.18, $f : \Sigma^\omega \to \{0,1\}$ are indicator functions. Let $\varepsilon \geq 0$. Consider such an f with $V - (\int f d\nu_s - \alpha \int f d\nu_{s'}) \leq \varepsilon/2$ (made possible by supremum).

If it is shown that there exists h such that $\int f d\nu_s - \alpha \int f d\nu_{s'} - K(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) \leq \varepsilon/2$ then $V - K(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) \leq \varepsilon$ as required.

Let $A = \{x \in A^\omega \mid f(x) = 1\}$. Since f measurable then $A \in \mathcal{F}$ then:

$$\int f d\nu_s - \alpha \int f d\nu_{s'} = \nu_s(A) - \alpha \nu_{s'}(A)$$

By Lemma 4.20, it is known $\nu_s(A)$ and $\nu_{s'}(A)$ can be approximated by cylinder sets since $\mathcal{F} \subseteq S$ as $\mathcal{F}$ is the smallest σ-algebra generated by combinations of cylinders, so contained in S approximable by cylinders. Let C be a cylinder set such that

$$\nu_s(A) - \nu_s(C) \leq \frac{\varepsilon}{4\alpha} \text{ and } \nu_{s'}(A) - \nu_{s'}(C) \leq \frac{\varepsilon}{4\alpha}$$

Let h be the length of the longest prefix $c \in C$. Assume without loss of generality that all prefixes in C are of length h. If not generalise all prefixes shorter than the maximum length h by adding all possible suffixes to make length h. Let

$$g(c) = \begin{cases} 1 & \text{if } \nu_s(c) - \alpha \nu_{s'}(c) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Note that for all $c, c' \in C$ since $|c| \leq h$ then if $c \neq c'$ we have $\mathbb{1}^h_{\neq}(c, c') = 1$, so

g is a valid function in $K_\alpha(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'})$. Thus observe that:

$$\begin{aligned}
\nu_s(C) - \alpha\nu_{s'}(C) &= \sum_{c\in C} \nu_s(c) - \alpha \sum_{c\in C} \nu_{s'}(c) \\
&\leq \sum_{c\in C} g(c)(\nu_s(c) - \alpha\nu_{s'}(c)) \\
&\leq K_\alpha(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}).
\end{aligned}$$

If $K_\alpha(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) > \nu_s(A) - \alpha\nu_{s'}(A)$ then as $K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'}) \geq K_\alpha(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'})$ so $K_\alpha(\mathbb{1}_{\neq})(\nu_s, \nu_{s'}) - K_\alpha(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) \leq \frac{\varepsilon}{2} \leq \varepsilon$ already.

Otherwise

$$\begin{aligned}
&\nu_s(A) - \alpha\nu_{s'}(A) - K_\alpha(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) \\
&\leq \nu_s(A) - \alpha\nu_{s'}(A) - (\nu_s(C) - \alpha\nu_{s'}(C)) \\
&\leq (\nu_s(A) - \nu_s(C)_{+}\alpha(\nu_{s'}(A) - \nu_{s'}(C)) \\
&\leq \frac{\varepsilon}{4\alpha} + \alpha\frac{\varepsilon}{4\alpha} \leq \frac{\varepsilon}{2}
\end{aligned}$$

□

This implies the following lemma:

Lemma 4.22. *If $K_\alpha(\mathbb{1}^h_{\neq}) \leq bd_\alpha$ for all h then $K_\alpha(\mathbb{1}_{\neq}) \leq bd_\alpha$.*

In the next chapter Δ_α will be replaced with another function. Thus the following is proven in full generality for a function F, here playing the role of Δ_α.

Lemma 4.23. *Let $F : [0,1] \times [0,1] \to [0,1]$ be a function such that if $f : X \to [0,1]$ is such that $\forall x, x' : F(f(x), f(x')) = 0$, then $\forall \mu, \mu' : F\left(\hat{f}(\mu), \hat{f}(\mu')\right) = 0$. Let K^F be such that*

$$K^F(d)(\mu, \mu') = \sup_{\substack{f:X\to[0,1] \\ \forall x,x'\in X \;\; F(f(x),f(x'))\leq d(x,x')}} F\left(\hat{f}(\mu), \hat{f}(\mu')\right)$$

Let d be the least fixed point of an operator $\Gamma^F : [0,1]^{S\times S} \to [0,1]^{S\times S}$ such that

$$\Gamma^F(d)(s, s') = \begin{cases} K^F(d)(\mu_s, \mu_{s'}) & \text{if } \ell(s) = \ell(t) \\ 1 & \text{if } \ell(s) \neq \ell(t) \end{cases}$$

Recall the function is monotone on the lattice $[0,1]^{X\times X}$, thus has a least fixed point.

Then for all $h \in \mathbb{N}$:

$$K^F(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) \leq d(s, s)$$

Proof. The proof proceeds by induction on h.

Base Case: $h = 0$ The base case shows that $K^F(\mathbb{1}^0_{\neq}) = 0$. Thus it is necessarily smaller or equal to any value of d.

$$K^F(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) = \sup_{\substack{f:A^\omega \to [0,1] \\ \forall t,t' \in A^\omega \ F(f(t),f(t')) \leq \mathbb{1}^h_{\neq}(t,t)}} F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right)$$

But since $\mathbb{1}^h_{\neq} = 0$, then

$$K^F(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) = \sup_{\substack{f:A^\omega \to [0,1] \\ \forall t,t' \in A^\omega \ F(f(t),f(t')) \leq 0}} F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right)$$

Hence $F(f(t), f(t')) \leq 0$ for all $t, t' \in A^\omega$ then $F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right) = 0$, resulting in:

$$K^F(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) = \sup_{f:A^\omega \to [0,1] \ | \ F(f(t),f(t')) \leq 0} 0 = 0$$

Therefore $K^F(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) = 0 \leq d(s, s')$.

Induction Case: For the induction, it is assumed that $K^F(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) \leq d(s, s')$ in order to show that $K^F(\mathbb{1}^{h+1}_{\neq})(\nu_s, \nu_{s'}) \leq d(s, s')$.

Case 1. $\ell(s) \neq \ell(s')$

This case is trivial as $d(s, s') = 1 \geq K^F(\mathbb{1}^{h+1}_{\neq})(\nu_s, \nu_{s'})$.

Case 2. $\ell(s) = \ell(s') = a \in \Sigma$.

The strategy is to consider a function $f : \Sigma^\omega \to [0, 1]$ valid in the supremum of $K^F(\mathbb{1}^{h+1}_{\neq})$. A function $g : S \to [0, 1]$ is constructed which is valid under the expansion of d with the same difference of expectations. This will lead to the conclusion that the sumpremum in $K^F(\mathbb{1}^{h+1}_{\neq}) \leq d$

First consider a function that is valid under $\mathbb{1}^{h+1}_{\neq}$ to find a function valid under $\mathbb{1}^h_{\neq}$. Consider an f such that $F(f(t), f(t')) \leq \mathbb{1}^{h+1}_{\neq}(t, t')$ for all $t, t' \in A^\omega$ and hence valid under $\mathbb{1}^{h+1}_{\neq}$ and let $f_a(t) = f(at)$.

Claim 4.24. $F(f_a(t), f_a(t')) \leq \mathbb{1}^h_{\neq}(t, t')$.

$$\begin{aligned}
F(f_a(t), f_a(t')) &= F\left(f(at), f(at')\right) && \text{(by defn of } f_a\text{)}\\
&\leq \mathbb{1}_{\neq}^{h+1}(at, at') \qquad = \begin{cases} 0 & (at)^{h+1} = (at')^{h+1} \\ 1 & \text{otherwise} \end{cases} && \text{(by assumption on } f\text{)}\\
&= \mathbb{1}_{\neq}^{h}(t, t') \qquad = \begin{cases} 0 & t^h = t'^h \\ 1 & \text{otherwise} \end{cases}
\end{aligned}$$

This function is used to define a function g which is valid in the restrictions in $K^F(d) = d$ thus the difference of expectations of g is below d.

Let

$$g(s) = \hat{f}_a(\nu_s) = \int_{A^\omega} f_a d\nu_s.$$

Claim 4.25. *For all s, s': $F(g(s), g(s')) \leq d(s, s')$*

$$\begin{aligned}
F(g(s), g(s')) &= F(\hat{f}_a(\nu_s), \hat{f}_a(\nu'_s) && \text{(by definition of g)}\\
&\leq \sup_{\substack{f: A^\omega \to [0,1] \\ \forall t,t' \in A^\omega \;\; F(f(t),f(t')) \leq \mathbb{1}_{\neq}^{h}(t,t')}} F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right) && \text{(} f_a \text{ is one such } f\text{, Claim 4.24)}\\
&= K^F(\mathbb{1}_{\neq}^{h})(\nu_s, \nu_{s'})\\
&\leq d(s, s') && \text{(induction assumption)}
\end{aligned}$$

Then the difference of expectation of g is equal to the difference of expectation of f:

Claim 4.26. $\hat{f}(\nu_s) = \hat{g}(\mu_s)$

Note that by expanding a single step, when $\ell(s) = a$,

$$\nu_s(at) = \sum_{s_i \in Q} \mu_s(s_i) \nu_{s_i}(t). \tag{4.1}$$

Thus,

$$\begin{aligned}
\hat{f}(\nu_s) &= \int_{A^\omega} f d\nu_s && \text{(by definition)}\\
&= \int_{a.A^\omega} f d\nu_s && \text{(since } \ell(s) = a \text{, the first character is } a\text{)}\\
&= \int_{A^\omega} f_a \sum_{s_i \in S} \mu_s(s_i) d\nu_{s_i} && \text{(Expanding a single step by (4.1))}\\
&= \sum_{s_i \in S} \mu_s(s_i) \int_{A^\omega} f_a d\nu_{s_i} && \text{(Exchanging sum and integral.)}\\
&= \sum_{s_i \in S} \mu_s(s_i) g(s_i) && \text{(definition of } g\text{)}\\
&= \hat{g}(\mu_s) && \text{(expectation of } g\text{)}
\end{aligned}$$

Hence the difference of expectation of g (and thus f) can be planted below the supremum defining the fixed point and thus below the fixed point.

Claim 4.27. $F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right) \leq d(s, s')$

$$\begin{aligned}
F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right) &= F\left(\hat{g}(\mu_s), \hat{g}(\mu_{s'})\right) && \text{(Claim 4.26)}\\
&\leq K^F(d)(\mu_s, \mu_{s'}) = \sup_{\substack{f:S\to[0,1]\\ \forall s,s' \in S\ F(f(s),f(s')) \leq d(s,s')}} F\left(\hat{f}(\mu_s), \hat{f}(\mu_{s'})\right)\\
& && \text{(g is such an } f\text{, by Claim 4.25)}\\
&\leq d(s, s') && (\Gamma^F(d) \leq d \text{ by definition})
\end{aligned}$$

Since by Claim 4.27 $F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right) \leq d(s, s')$ for all f valid for $\mathbb{1}_{\neq}^{h+1}$ then this also holds in the supremum.

Thus:

$$\begin{aligned}
K^F(\mathbb{1}_{\neq}^{h+1})(\nu_s, \nu_{s'}) &= \sup_{\substack{f:A^\omega\to[0,1]\\ \forall t,t' \in A^\omega\ F(f(t),f(t')) \leq \mathbb{1}_{\neq}^{h+1}(t,t')}} F\left(\hat{f}(\nu_s), \hat{f}(\nu_{s'})\right)\\
&\leq d(s, s'). \qquad \square
\end{aligned}$$

Proof of Lemma 4.14

Proof. Lemma 4.19 and Definitions 4.8 and 4.10 show that Δ_α, K_α and bd_α satisfy the restrictions in Lemma 4.23 giving

$$K(\mathbb{1}_{\neq}^{h})(\nu_s, \nu_{s'}) \leq bd_\alpha(s, s') \text{ for all } s, s' \in Q$$

for all h and the result follows from Lemma 4.22. $\square$

4.4 Skewed Kantorovich distances

This section discusses how to calculate the skewed variant of the Kantorovich distance. This will inform the next section, which looks into computing bd_α.

Recall that in the definition of $K_\alpha(d)(\mu, \mu')$ from Definition 4.8, the definition of Δ_α can be simplified to omit the 0 case when used inside K_α.

If $\alpha = 1$ then Δ_α is the absolute value function and it is known that the distance corresponds to a single instance of a linear programming problem [BW01]. However, this is no longer true here due to the shape of $\Delta_\alpha(x, y) = \max(x - \alpha y, y - \alpha x)$. Still, the calculation can be presented taking the maximum of a pair of linear programs. This formulation will be referred to as the "primal form" of $K_\alpha(d)$. The first program is given below, the other is its symmetric variant with μ, μ' reversed. Below f_i is written for $f(i)$ and it is assumed that d is symmetric.

$$\max_{f \in [0,1]^Q} \Big(\sum_{i \in Q} f_i \mu(i) - \alpha \sum_{i \in Q} f_i \mu'(i) \Big) \qquad \text{subject to} \qquad \forall i, j \in Q \quad f_i - \alpha f_j \leq d_{i,j}$$

Whilst Definition 4.8 is presented using the supremum, here the objective and constrains are finite (because the set of states Q is finite), and the constraints are linear, hence the objective is attained and maximum can be used.

The standard Kantorovich distance ($\alpha = 1$) is often presented in the following dual form when m is a pseudometric, based on the minimum coupling between the two distributions μ and μ', weighted by the distance function.

$$K(m)(\mu, \mu') = \min_{\omega \in [0,1]^{Q \times Q}} \sum_{i,j \in Q} \omega_{i,j} \cdot m_{i,j} \quad \text{subject to} \quad \begin{array}{l} \forall i \in Q \ \sum_{j \in Q} \omega_{i,j} = \mu(i) \\ \forall j \in Q \ \sum_{i \in Q} \omega_{i,j} = \mu'(j) \end{array}$$

Remark. The dual form can be viewed as an optimal transportation problem in which an arbitrarily divisible cargo must be transferred from one set of locations (represented by a copy Q^L of Q) to another (represented by a different copy Q^R of Q). Each state $s^R \in Q^R$ must receive $\mu(s)$, while each state $s^L \in Q^L$ must send $\mu'(s)$. If $\omega_{i,j}$ is taken to represent the amount that gets sent from j^L to i^R then the above conditions restrict ω in accordance with the sending and receiving budgets. If $d_{i,j}$ represents the cost of sending from j^L to i^R then the objective function $\sum_{i,j} \omega_{i,j} \cdot d_{i,j}$ corresponds to the overall cost of transport. Consequently, the problem is often referred to as a mass transportation problem [Kan42]. ◀

To achieve a similar "dual form" in the skewed case, the dual form of each of the linear programs can be found. Then the distance can be calculated by taking the maximum of the two minima. The shape of the dual is given below on the right.

Lemma 4.28.

$$\begin{array}{rcl}
\displaystyle\max_{f\in[0,1]^Q}\sum_{i\in Q} f_i\mu(i)-\alpha f_i\mu'(i) & = & \displaystyle\min_{\substack{\omega\in[0,1]^{Q\times Q}\\ \tau,\gamma,\eta\in[0,1]^Q}}\sum_{i,j\in Q}\omega_{i,j}\cdot d_{i,j}+\sum_{i\in Q}\eta_i \\
\text{subject to} & & \text{subject to} \\
\forall i,j\in Q\ f_i-\alpha f_j\le d_{i,j} & & \forall i\in Q: \sum\limits_{j\in Q}\omega_{i,j}+\tau_i-\gamma_i+\eta_i=\mu(i) \\
 & & \forall j\in Q: \sum\limits_{i\in Q}\omega_{i,j}+\frac{\tau_j-\gamma_j}{\alpha}\le\mu'(j)
\end{array}$$

The dual form presented above is a simplified (but equivalent) form of the immediate dual obtained via the standard linear programming recipe. The dual of the other linear program is obtained by swapping μ,μ'. In the skewed case, the optimisation is over the polytope, $\Omega_{\mu,\mu'}=$

$$\left\{(\omega,\eta)\in[0,1]^{Q\times Q}\times[0,1]^Q \;\middle|\; \begin{array}{c}\exists\gamma,\tau\in[0,1]^Q\\ \forall i\in Q:\sum\limits_{j\in Q}\omega_{i,j}+\tau_i-\gamma_i+\eta_i=\mu(i)\\ \forall j\in Q:\sum\limits_{i\in Q}\omega_{i,j}+\frac{\tau_j-\gamma_j}{\alpha}\le\mu'(j)\end{array}\right\}.$$

Note that this polytope being optimised over is independent of d, which appears only in the objective function.

Proof of Lemma 4.28. Since Q is assumed to be finite, relabel the elements Q in the form $\{1,\dots,n\}$. Then the primal form can be expressed as the following linear program:

$$\max_{\vec f\in[0,1]^n}\vec f\cdot(\mu-\alpha\mu')\quad\text{subject to}\quad f_i-\alpha f_j\le d_{i,j}\text{ for all }i,j.$$

So that the result is in the desired presentation, split $\vec f$ into two vectors $\vec a$ and $\vec b$, both intended to be equal to $\vec f$, resulting in the following formulation:

$$\max_{\vec a,\vec b\in[0,1]^n}\vec a\cdot\mu-\vec b\cdot(\alpha\mu')\quad\text{subject to } a_i-\alpha b_j\le d_{i,j}\text{ for all }i,j\text{ and }\vec a=\vec b$$

This is then translated in representation to the following standard matrix form:

$$\max_{\vec a,\vec b\in\mathbb{R}^n_{\ge0}}\begin{pmatrix}\mu & -\alpha\mu'\end{pmatrix}\begin{pmatrix}\vec a\\ \vec b\end{pmatrix}\quad\text{subject to}\quad\begin{pmatrix}A & A'\\ I & -I\\ -I & I\\ I & 0\end{pmatrix}\begin{pmatrix}\vec a\\ \vec b\end{pmatrix}\le\begin{pmatrix}\vec d\\ \vec 0\\ \vec 0\\ \vec 1\end{pmatrix}$$

where I is the standard identity matrix, 0 is the zero matrix and A, A' are $(n \times n) \times n$ matrices where for all $i, j, k \in [n], i \neq j, j \neq k, k \neq i$:

- $A_{(i,j),i} = 1$
- $A'_{(i,j),j} = -\alpha$
- $A_{(i,j),j} = 0$, $A'_{(i,j),i} = 0$ and $A_{(i,j),k} = A'_{(i,j),k} = 0$.

These matrices A, A' are visualised as follows (interpreting blank spaces to be zeros):

$$\begin{array}{cc} A & A' \\ \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & \vdots & \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} & \begin{bmatrix} -\alpha & & & \\ -\alpha & & & \\ \vdots & & & \\ -\alpha & & & \\ & -\alpha & & \\ & -\alpha & & \\ & \vdots & & \\ & -\alpha & & \\ & & \ddots & \\ & & & -\alpha \\ & & & -\alpha \\ & & & \vdots \\ & & & -\alpha \end{bmatrix} \end{array}$$

The constraints can be interpreted as follows:

- $\begin{pmatrix} A & A' \end{pmatrix} \begin{pmatrix} \vec{a} \\ \vec{b} \end{pmatrix} \leq \vec{d}$ specify the skewed distance must be less than d.
- $\begin{pmatrix} I & -I \end{pmatrix} \begin{pmatrix} \vec{a} \\ \vec{b} \end{pmatrix} \leq \vec{0}$ and $\begin{pmatrix} -I & I \end{pmatrix} \begin{pmatrix} \vec{a} \\ \vec{b} \end{pmatrix} \leq \vec{0}$ specify $\vec{a} = \vec{b}$.
- $\begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} \vec{a} \\ \vec{b} \end{pmatrix} \leq \vec{1}$ specifies $\vec{a} \leq \vec{1}$ (from the condition $\vec{f} \in [0, 1]$).

Through the standard dualisation recipe, the following dual form is obtained, where $A^\top$ is the transpose of the matrix A:

$$\min_{\omega \in \mathbb{R}^{n \times n}_{\geq 0}, \gamma, \tau, \eta \in \mathbb{R}^n_{\geq 0}} \begin{pmatrix} d & \vec{1} \end{pmatrix} \begin{pmatrix} \omega \\ \eta \end{pmatrix}$$

$$\text{subject to} \quad \begin{pmatrix} A^\top & I & -I & I \\ A'^\top & -I & I & 0 \end{pmatrix} \begin{pmatrix} \omega \\ \tau \\ \gamma \\ \eta \end{pmatrix} \geq \begin{pmatrix} \mu \\ -\alpha\mu' \end{pmatrix}.$$

Finally, through routine manipulations the following presentation is obtained.

$$\min_{\omega \in [0,1]^{n \times n}, \gamma, \tau, \eta \in [0,1]^n} \sum_{i,j} \omega_{i,j} \cdot d_{i,j} + \sum_i \eta_i$$

Subject to:

$$\forall i : \sum_j \omega_{i,j} + \tau_i - \gamma_i + \eta_i = \mu(i)$$

$$\forall j : \sum_i \omega_{i,j} + \frac{\tau_j - \gamma_j}{\alpha} \leq \nu(j)$$

Note the first equation is equality, rather than $\geq$, since an equivalent (or better) solution can always be found by not over-satisfying it: reduce $\omega_{i,j}, \tau_i, \eta_i$ or increase γ_i with no violation of other constraints. $\square$

This formulation would be equivalent to a standard coupling were γ, τ, η to be removed. It would be possible to merge $\tau_i, \gamma_i \in [0,1]$ into a single variable in $[-1,1]$, but for consistency with the standard requirement of linear programs and the following transportation problem intuition they are kept separate.

As before, cargo can be transferred through the standard routes with ω at a cost d; and additionally new resource η can be obtained from an extra location at a cost of 1.

There are also additional, cost-free routes between corresponding pairs s^L and s^R (represented by τ_s) and back (represented by γ_s). These extra routes are quite peculiar. En route from s^L to s^R the cargo "grows": when $\frac{\tau_s}{\alpha}$ is sent from s^L, a larger amount of τ_s is received at s^R. Overall, the total amount of cargo sent may be less than that received, so the sending constraints are now inequalities. From s^R to s^L the cargo "shrinks": when γ_s is sent from s^R, only $\frac{\gamma_s}{\alpha}$ is received by s^L.

It is immediate that τ routes can be useful. The γ routes may be useful for optimisation under two conditions. Firstly the shrinkage of the cargo must be made up elsewhere, i.e., through "growing" τ routes. Additionally the cost $\alpha \times d(s_1^L, s^R) + d(s^L, s_2^R)$ is lower than $d(s_1^L, s_2^R)$, which may well be the case due to the lack of triangle inequality.

This results in the following formulation, the "dual form", $K_\alpha(d)(\mu,\mu') =$

$$\max\left\{\min_{\omega,\eta\in\Omega_{\mu,\mu'}}\sum_{i,j}\omega_{i,j}\cdot d_{i,j}+\sum_i\eta_i,\ \min_{\omega,\eta\in\Omega_{\mu',\mu}}\sum_{i,j}\omega_{i,j}\cdot d_{i,j}+\sum_i\eta_i\right\}.$$

Note that $K_\alpha(d)(\mu,\mu')$ can be computed in polynomial time as a pair of linear programs in either primal or dual form, and taking the maximum (in either case). In the calculations related to bd_α, the distributions μ,μ' will always be taken to be $\mu_s,\mu_{s'}$ respectively, for some $s,s'\in Q$. The ability to switch between primal and dual form will play a useful role in the complexity-theoretic arguments of the next section.

4.5 Computing bd_α

This section will show the computability of $bd_\alpha(s,s')$ by exhibiting a logical formulation of the distance. By suitably exploiting these formulations, Theorem 4.32 shows that the distance can be computed exactly in polynomial time using an **NP** oracle.

The first order theory of the reals are the true sentences of first order logic with universal and existential quantifiers and logical combinations of inequalities of arithmetic expressions of real variables. The decision question asks given a sentence whether it is in the theory. The first order fragment of *linear* real arithmetic (LRA) is when these arithmetic expressions are restricted to the form $c_1x_1+\cdots+c_mx_m$ where c_i's are constant and x_i's are real variables. Sontag [Son85] relates the alternation hierarchy within LRA to the polynomial hierarchy **PH**: formulae of the form $\exists x_1\forall x_2\ldots Qx_kF(x_1\ldots x_k)$ (with quantifier-free F) correspond to $\Sigma_k^{\mathbf{P}}$ (and formulae starting with $\forall$ to $\Pi_k^{\mathbf{P}}$). Recall that $\Sigma_1^{\mathbf{P}}=\mathbf{NP}$.

The first result will be to observe that all distances $bd_\alpha(s,s')$ are rational and can be expressed in polynomial size with respect to $\mathcal{M}$. To that end, a result by Sontag [Son85] is exploited, which states that, without affecting satisfiability, quantification in the first order fragment of linear real arithmetic can be restricted to rationals of polynomial size with respect to formula length (as long as all coefficients present in the formula are rational). Consequently, expressing "there exists a least fixed point d of Γ_α" in this fragment (with a polynomial increase in size), enables the intended conclusion.

The relevant LRA formula is given in Figure 4.2. The formula asserts the existence of a distance d, which is a pre-fixed point of Γ_α ($\forall f.\phi(d,f)$) such that any other pre-fixed point d' of Γ_α is greater than or equal to d. Note that $\forall f.\phi(f,d)$ exploits the fact that $\max_f A(f)\le d(s,s')$ is equivalent to

$$\begin{aligned}
\exists d \in & [0,1]^{Q\times Q} \text{ s.t.} \\
& (\forall f \in [0,1]^Q \phi(d,f) \\
& \quad \wedge \forall d' \in [0,1]^{Q\times Q} \left\{ \forall f \in [0,1]^Q \phi(d',f) \implies \bigwedge_{s,s'\in Q} d_{s,s'} \le d'_{s,s'}) \right\}
\end{aligned}$$

$$\phi(d,f) = \bigwedge_{s,s'} \begin{cases} d_{s,s'} = 1 & \ell(s) \neq \ell(s') \\ (\bigwedge_{i,j} f_i - \alpha f_j \le d_{i,j} \wedge f_j - \alpha f_i \le d_{i,j}) & \ell(s) = \ell(s') \\ \quad \implies (\sum_i f_i \mu_s(i) - \alpha \sum_i f_i \mu_{s'}(i) \le d_{s,s'} & \\ \qquad \wedge \sum_i f_i \mu_{s'}(i) - \alpha \sum_i f_i \mu_s(i) \le d_{s,s'}) & \end{cases}$$

Figure 4.2: Logical formulation of least pre-fixed point bd_α.

$\forall f(A(f) \le d(s,s'))$. Sontag's result then implies the following.

Theorem 4.29. *Values of bd_α are rational. There exists a polynomial p such that for any LMC $\mathcal{M}$ and $s,s' \in Q$, the size of bd_α (in binary) can be bounded from above by a polynomial in $|\mathcal{M}|$.*

The second result focuses on the following decision problem for bd_α.

Definition 4.30 (BD_α-THRESHOLD)**.**

INPUT	An LMC $\mathcal{M}$, states $s,s' \in Q$ and a threshold $\theta \in \mathbb{Q} \cap [0,1]$
OUTPUT	is $bd_\alpha(s,s') \le \theta$? ◀

Recall that the analogous problem for tv_α is undecidable (Theorem 3.11). In the case of bd_α, the problem turns out to be decidable and the argument does not depend on whether $<$ or $\le$ is used. To establish decidability observe that $bd_\alpha(s,s') \le \theta$ can be expressed in LRA simply by adding $d(s,s') \le \theta$ to the formula from Figure 4.2.

The formula can, however, be simplified, using $bd_\alpha = \min \{d \,|\, \Gamma_\alpha(d) \sqsubseteq d\}$. Then $bd_\alpha(s,s') \le \theta$ can be specified as the existence of a pre-fixed point d such that $d(s,s') \le \theta$. This can be done as follows, using $\phi(d,f)$ from Figure 4.2.

$$\exists d \in [0,1]^{Q\times Q} \,(\, \forall f \in [0,1]^Q \phi(d,f) \;\wedge\; d(s,s') \le \theta \,)$$

By Sontag's results, this not only yields decidability but also membership in $\Sigma_2^{\mathbf{P}}$. Recall that $\mathbf{NP} \subseteq \Sigma_2^{\mathbf{P}} \subseteq \mathbf{PH} \subseteq \mathbf{PSPACE}$.

Note that the universal quantification over f remains, i.e. this only concludes that the problem is in $\Sigma_2^{\mathbf{P}}$. To overcome this, the dual form is used instead

$$\mathrm{BD}_\alpha\text{-}\textsc{Threshold}(s,s',\theta) =$$

$$\exists (d_{i,j})_{i,j\in Q} \quad \bigwedge_{i,j\in Q} (0 \le d_{i.j} \le 1) \quad \wedge \ \mathit{labelConstraint}(d) \ \wedge \ d_{s,s'} \le \theta$$

$$\mathit{labelConstraint}(d) =$$

$$\bigwedge_{q,q'\in Q} \begin{cases} d_{q,q'} = 1 & \ell(q) \neq \ell(q') \\ \mathit{couplingConstraint}(d, d_{q,q'}, q, q') & \ell(q) = \ell(q') \\ \quad \wedge\ \mathit{couplingConstraint}(d, d_{q,q'}, q', q) & \end{cases}$$

$$\mathit{couplingConstraint}(d,x,q,q') =$$

$$\begin{aligned} &\exists(\omega_{i,j})_{i,j\in Q} \quad \exists(\gamma_i)_{i\in Q} \quad \exists(\tau_i)_{i\in Q} \quad \exists(\eta_i)_{i\in Q} \\ &\quad \sum_{i,j\in Q} \omega_{i,j}\cdot d_{i,j} + \sum_i \eta_i \le x \\ &\wedge \bigwedge_{i,j\in Q} 0 \le \omega_{i,j} \le 1 \\ &\wedge \bigwedge_{i\in Q} 0 \le \gamma_i \le 1 \wedge 0 \le \tau_i \le 1 \wedge 0 \le \eta_i \le 1 \\ &\wedge \bigwedge_{i\in Q} \sum_{j\in Q} \omega_{i,j} - \gamma_i + \tau_i + \eta_i = \mu_q(i) \\ &\wedge \bigwedge_{j\in Q} \sum_{i\in Q} \omega_{i,j} + \frac{\tau_j - \gamma_j}{\alpha} \le \mu_{q'}(j) \end{aligned}$$

Figure 4.3: **NP** formula for BD$_\alpha$-Threshold.

(Lemma 4.28). This will enable the elimination of the universal quantification by replacing it with existential quantifiers using the fact that $\min_\omega A(\omega) \le B$ is equivalent to $\exists\omega(A(\omega) \le B)$. The resultant formula is shown in Figure 4.3, presented in a form similar to a formula used by van Breugel, Sharma and Worrell [BSW07] who used it to show the approximation standard bisimilarity distance is in **PSPACE**.

Note the formula is not linear due to $\omega_{i,j} \cdot d_{i,j}$. However, it is known, by Theorem 4.29, that bd_α corresponds to an assignment of poly-sized rationals. Supposing that these values of bd_α are known consider the formula again, but with the references to d replaced with the value in bd_α. Then it does become an LRA formula (of polynomially bounded length with respect to $|\mathcal{M}|$) and again the conclusion can be drawn that the assignments of ω, γ, τ must also involve rationals whose size is polynomially bounded. Consequently, the formula implies membership of the BD$_\alpha$-Threshold problem in $\Sigma_1^{\mathbf{P}} = \mathbf{NP}$: it suffices to guess the satisfying assignment, guaranteed to be rational and of polynomial size.

Theorem 4.31. BD_α-THRESHOLD *is in* $\mathbf{NP}$.

The decidability of BD_α-THRESHOLD makes it possible to approximate $bd_\alpha(s, s')$ to arbitrary (rational) precision γ by binary search. This will involve $O(|\gamma|)$ calls to the oracle for BD_α-THRESHOLD (where $|\gamma|$ is the number of bits required to represent γ in binary).

What's more, assuming the oracle, one can actually find the exact value of $bd_\alpha(s, s')$ in polynomial time (wrt $\mathcal{M}$). This exploits the fact that the value of bd_α is rational and its size is polynomially bounded, so one can find it by approximation to a carefully chosen level of precision and then finding the relevant rational with the continued fraction algorithm (see e.g. [GLS88, Section 5.1] or [EY10]).

Theorem 4.32. bd_α *can be calculated in polynomial time with an* $\mathbf{NP}$ *oracle.*

As a consequence, the problem of computing bd_α reduces to propositional satisfiability, i.e., can be encoded in SAT. This justifies, for instance, the following approach: treat every variable as a ratio of two integers from an exponential range, and give the system of resulting constraints to an Integer Arithmetic or SAT solver. While this might look like resorting to a general-purpose "hammer", Theorem 4.32 is necessary for this method to work: it is not, in fact, possible to solve general polynomial constraint systems relying just on SAT. More precisely, the existence of such a procedure would be a breakthrough in the computational complexity theory, showing that the existential fragment of the first order theory of the reals is complete for $\mathbf{NP}$. This would imply that a multitude of problems in computational geometry could be solved using SAT solvers [SS17; Car15]. Unlike for bd_α, variable assignments in these problems may need to be irrational, even if all numbers in the input data are integer or rational.

This directed approach is, however, expected to be inferior to the following observation. Theorem 4.29 reveals that the variables in the relevant constraint system need not assume irrational values or have large bit representations. Thus, one can give the system to a more powerful theory solver, or an optimisation tool, but to expect that the existence of simple and small models (solutions) will help the SMT heuristics (resp. optimisation engines) to find them quickly.

4.6 Examples

Example 4.33 (Dining Cryptographers). In the dining cryptographer model [Cha88], a ring of diners want to determine whether one of the diners paid or an outside body. If it was one of the diners that paid, it is undesirable to reveal which of them it was. The protocol proceeds with each adjacent pair privately flipping a coin, each diner then reports the XOR of the two coin flips they observe, however if the diner paid he would report the negation of this.

Whether one of the diners paid, or an outside body, can be determined by observing the XOR of the announcements. With perfectly fair coins, the protocol guarantees total privacy of the paying diner, but it is differentially private if the coins are biased. If an outside body paid, there is no privacy to maintain so it is only necessary to simulate the scenarios in which one of the diners did pay.

In the following two-diner scenario, the diners would know which of them had paid but an external observer of the output would only learn that one of them paid, not which. Thus the problem is considered from the perspective of an outside observer.

The protocol is formalised in the code in Figure 4.4. This is translated in to the labelled Markov chain in Figure 4.5 instantiated for the 2-person case, using weighted coins with $p = \frac{49}{100}$. This is achieved by states capturing the five variables required by the algorithm in a tuple:

$(\texttt{previousFlip}, \texttt{thisFlip}, \texttt{firstFlip}, \texttt{cryptographer}, \texttt{payingCryptographer})$.

Once the announcement has been made thisFlip becomes previousFlip and thisFlip is temporarily not used, so in the state where the next flip is ready to be performed the state stores the following:

$$(\texttt{previousFlip}, \texttt{firstFlip}, \texttt{cryptographer}, \texttt{payingCryptographer}).$$

The scenario where cryptographer 0 paid must have similar output distribution to cryptographer 1 paying, so that it can be determined that one of them did pay (with 100% accuracy), but not which. The internal configuration of the machine is always assumed to be hidden, but the announcements are made public whilst maintaining the privacy of the participating cryptographer (and the internal states).

The states of the machine encode the 5 variables that need to be tracked. To achieve (ε, δ)-differential privacy with $\alpha = e^{\varepsilon} = 1.0002$ the minimal (true) value of δ is 0.00030004. The methods of this chapter generate a correct upper bound $bd_\alpha(s_0, s_1) = 0.0004$, showing $(\ln(1.0002), 0.0004)$-differential privacy. The protocol could be played with n players, requiring $O(n^2)$ states, for all possible assignments of paying cryptographer and current cryptographer. ◀

```
diningCrypto(payingCryptographer):
   firstFlip = flip(p, 1−p)
   previousFlip = firstFlip
   for cryptographer = 0 → n−1:
      if cryptographer == n−1:
         thisFlip = firstFlip
      else:
         thisFlip = flip(p, 1−p)
      if (cryptographer == payingCryptographer):
         announce(previousFlip == thisFlip)
      else:
         announce(previousFlip != thisFlip)
      previousFlip = thisFlip
```

Figure 4.4: Simulation of dining cryptographers protocol

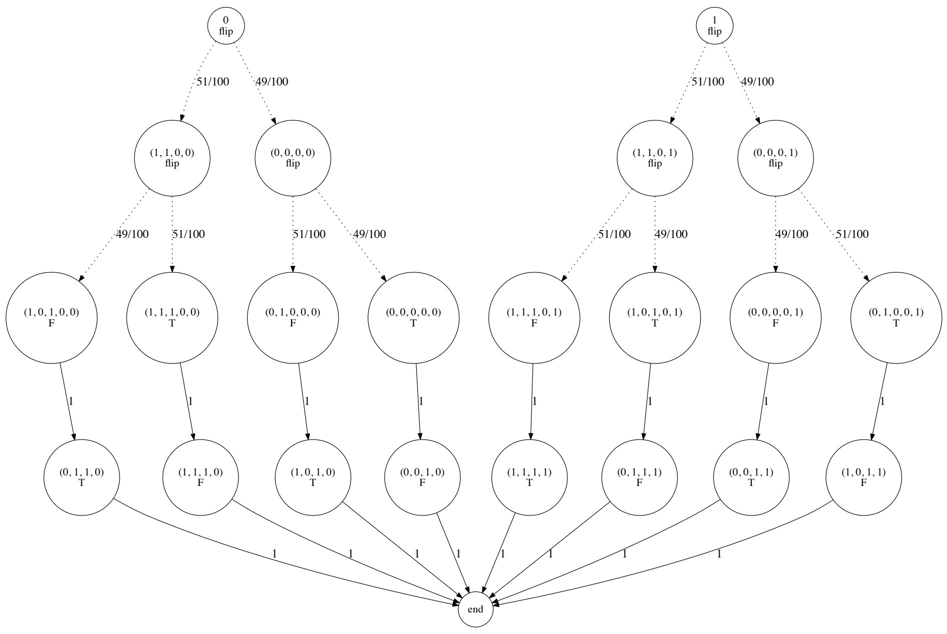
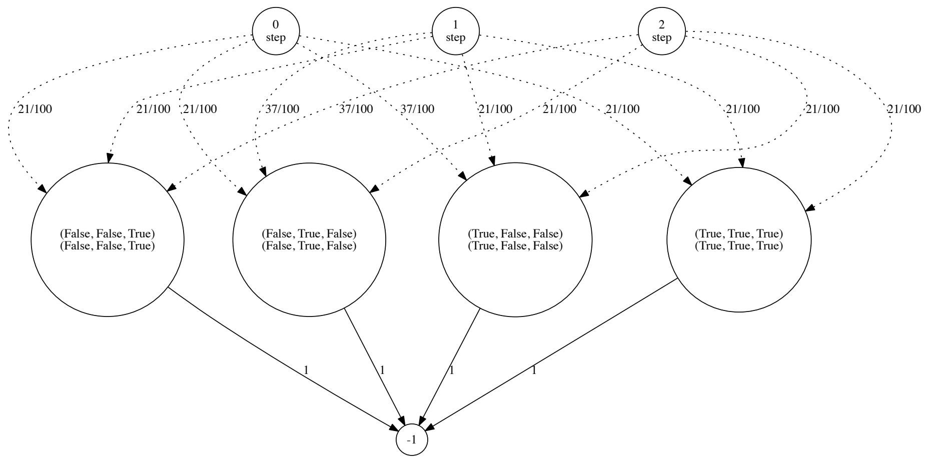


Figure 4.5: Markov chain for 2 dining cryptographers: state 0 (resp. 1) denotes cryptographer 0 (resp. 1) paid. The first line of a node is the state name, the second line is the label of the state.

Example 4.34 (Simplified Dining Cryptographers). It is possible to describe the dining cryptographers example using a Markov chain with lower depth, but in principal more states ($O(2^n)$) and larger alphabet. In this version the announcements are determined and announced together. One can consider this as the sequence of labels on each possible trace, or branch, of the Markov chain in Figure 4.5 as having been combined into a single state.

The model is demonstrated in Figure 4.6. In this case the results are much more accurate because in a single step $tv_\alpha = bd_\alpha$, thus $bd_\alpha(0, 1) = tv_\alpha(0, 1)$, the values for which are shown in Figure 4.7. ◀



0 step
1 step
2 step
21/100
37/100
(False, False, True) (False, False, True)
(False, True, False) (False, True, False)
(True, False, False) (True, False, False)
(True, True, True) (True, True, True)
1
-1

Figure 4.6: Three person dining cryptographers, with $p = \frac{3}{10}$. The first line of a node is the state name, the second line is the label of the state.

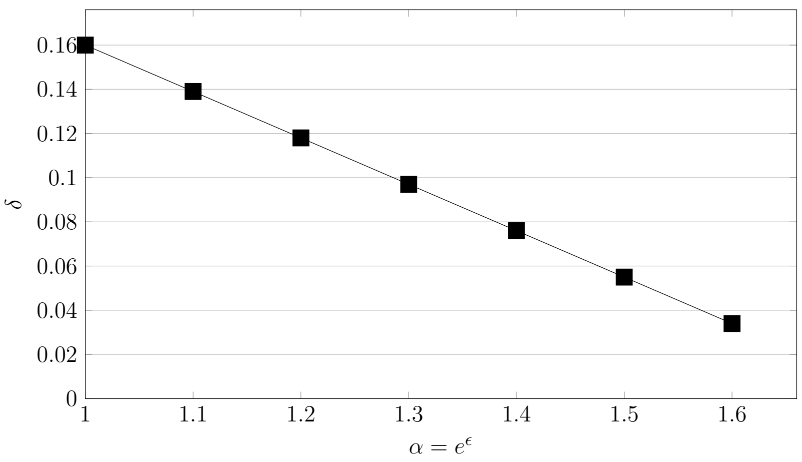
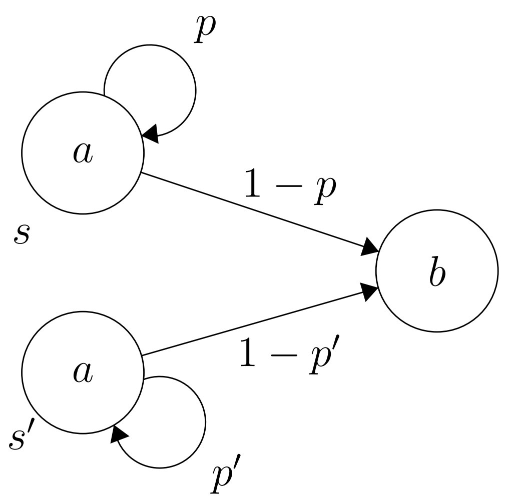


Figure 4.7: Plot of δ against α for the model in Figure 4.6, between states 0 and 1, 0 and 2, and 1 and 2.



$$bd_\alpha = \begin{bmatrix} 0 & v & 1 \\ & 0 & 1 \\ & & 0 \end{bmatrix}$$

Figure 4.8: Simple Markov chain.

Example 4.35 (bd_α can be unresponsive to α). Consider the Markov chain depicted in Figure 4.8 instantiated with $p = 0.8, p' = 0.9$. Whilst bd_α provides a sound upper bound on tv_α, no guarantees have been provided on how good the distance can be. Whilst it is true that bd_α is anti-monotone with respect to α (larger α, cannot give larger value), one would expect this to be the case in a strong sense. Naturally, by the structure of the system this may only hold up to a point, that there is a minimum value of δ. The following example demonstrates that anti-monotonicity is not always the case for bd_α; it produces the value $bd_\alpha(s, s') = 0.5$ for all values of $\alpha \geq 1$, despite $tv_1(s, s') \approx 0.269297$ (and thus $tv_\alpha(s, s') \leq 0.269297$ for all $\alpha \geq 1$).

Consider bd_α of Figure 4.8, since bd_α is symmetric consider only the upper triangle. Any state is distance zero to itself, thus the diagonals are fixed to zero. The difference between the third state and s or s' is always 1, due to different labels. Thus only the difference between states s, s' needs to be considered, let $bd_\alpha(s, s') = v$.

By the fixed point definition, $bd_\alpha(s, s') = v$ satisfies the following:

$$v = \text{maximise}_{x,y,z}\ 0.9x - 0.8\alpha y + (0.1 - 0.2\alpha)z$$

$$\begin{array}{rccc} & x \leq 1 & y \leq 1 & z \leq 1 \\ \text{subject to} & x - \alpha y \leq v & x - \alpha z \leq 1 & y - \alpha z \leq 1 \\ & y - \alpha x \leq v & z - \alpha x \leq 1 & z - \alpha y \leq 1 \end{array}$$

Note the conditions between x, z and y, z are redundant. Then by choice of $x = 1, z = 0$, it can be guaranteed that $v = \frac{1}{2}$, i.e. when $\alpha = 2$ let $y = 1/4$ and get $v = 1/2$ and when $\alpha = 4$ let $y = 1/8$ and get $v = 1/2$. The first condition $x - \alpha y \leq v$ is equivalent to $1 - \alpha y \leq 0.5$, or $0.5 \leq \alpha y$. So set $y = 0.5/\alpha$ and

$v = 0.5$ can always be achieved.

This example in part motivates the further study, leading to ld_α in the next chapter, which is more robust to this phenomena (as demonstrated in Example 5.28), although not in the example. ◀

4.7 Conclusion

This chapter has demonstrated that bisimilarity distances can be used to determine differential privacy parameters, despite their non-metric properties. The complexity of finding these values has been established to be polynomial, relative to an **NP** oracle. Yet, it may still be possible to obtain a polynomial algorithm—although much like in the case of the classical bisimilarity distances and linear programming, it may not necessarily outperform theoretically slower procedures.

The standard bisimilarity distance is characterised by the *unique* fixed point of an operator and it's possible this applies to bd_α also. That is bd_α, which is defined as the least fixed point of the operator Γ_α, may in fact be characterised as the unique fixed point of a similar operator. By the results of Etessami and Yannakakis [EY10], it would then follow that bd_α can be computed in **PPAD**, a smaller complexity class, improving upon the **NP** upper bound and matching the complexity of a bisimilarity distances on probabilistic systems with non-deterministic choice. The reason is the continuity of Γ_α, which follows from the properties of the polytope over which f ranges (in the definition of $K_\alpha(d)$). Whether bd_α can in fact be computed in polynomial time or is hard for some class above polynomial time is challenging open question.

The next chapter will revisit the total variation distance tv_α and its corresponding bisimilarity distances bd_α and develop a new variant which performs better in practice (in the sense of a more accurate answer) and no worse in theory (in the sense that the value is never less accurate and the complexity of computing the values is the same).

Chapter 5

Asymmetric Distances for δ

This chapter further the studies of bounds on δ by defining new bisimilarity distances. The bisimilarity distance of Chapter 4, inspired by the work of Chatzikokolakis et al. [CGPX14; Xu15], transpired to be computable in polynomial time with an **NP** oracle. There, the distance was defined using the Kantorovich lifting, the associated bisimilarity distance based on a fixed point and the effect of replacing the absolute value function with the skewed distance Δ_α considered. For the purposes of (ε, δ)-differential privacy the distance required is not a metric, nor even a pseudometric, so the methods of Chatzikokolakis et al. are adapted in Chapter 4 to account for this; resulting in a distance function bd_α which can be used to bound the δ parameter in differential privacy from above. The function, however, retained the symmetry property, that is, $bd_\alpha(s, s') = bd_\alpha(s', s)$. This chapter further studies distances to bound differential privacy in labelled Markov chains, but drops this symmetry property to discover a tighter bound, which can be computed at the same computational complexity. A weaker bisimilarity distance, lgd_α, is also defined for bounding δ that can be computed in polynomial time.

The privacy parameter in question, δ, can be expressed as a variant of the total variation distance tv_α. In particular lv_α will be defined as a single component of tv_α (which is a maximum over two functions). This distance is a way of measuring the maximum difference of probabilities between any two states. Total variation distance is usually expressed using absolute difference, but for differential privacy a skew is introduced into this distance. These exact distances transpire to be very difficult to compute: it will be confirmed that the threshold distance problem for lv_α, which asks whether the distance is below a given threshold, is undecidable (Corollary 5.4). Further it is also shown that approximating it is #**P**-hard, but for finite words it can be approximated in **PSPACE** (Theorem 5.5). These results match the results of Kiefer [Kie18] for standard total variation distances.

The distance lv_α will be bounded from above by a distance ld_α (Theorem 5.12) which will turn out to be computable, in a similar manner to how bd_α bounds

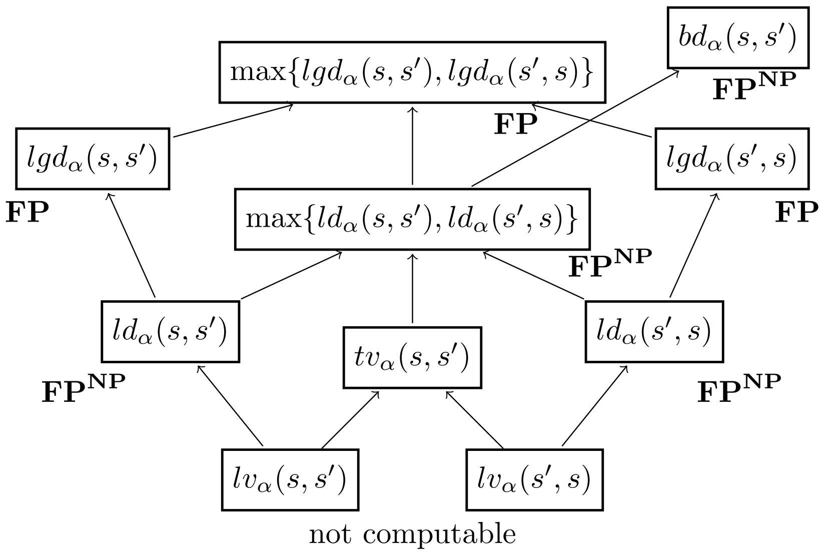


Figure 5.1: Partial order of distances, such that $a \to b$ if it is known that $a \leq b$. **FP** is the functional counterpart of **P**, where the value of the function can be computed in polynomial time. $\mathbf{FP^{NP}}$ indicates polynomial time with **NP** oracle. tv_α and bd_α are introduced in Chapter 4 and recalled in Sections 5.1 and 5.4, respectively. The remaining distances are the contribution of this chapter.

tv_α in Chapter 4. This distance ld_α can be computed in polynomial time with an **NP** oracle; that is, with the same complexity as bd_α (Theorem 5.15). ld_α is then generalised to a new distance lgd_α, computable in polynomial time (Theorem 5.26). Whilst this new distance is no smaller than ld_α it is conjectured that it might be equal. The distances $\max\{ld_\alpha(s,s'), ld_\alpha(s',s)\}$ and $\max\{lgd_\alpha(s,s'), lgd_\alpha(s',s)\}$ can then be used as sound upper bounds on δ. Thus defining the first non-trivial estimate of the δ parameter that can be computed in polynomial time (trivially, always returning 1 is technically correct).

The results in this chapter show that taking the maximum over two calls to ld_α is a better approximation than bd_α from Chapter 4. This is confirmed using several case studies, which also demonstrate, on a randomised response mechanism, that the estimates based on ld_α can beat standard differential privacy composition theorems. The relationships between distances are summarised in Figure 5.1.

5.1 Defining lv_α to capture δ

Like in Chapter 4, the aim is to capture the value of δ required to satisfy the differential privacy property for a given ε. That is, given a LMC $\mathcal{M}$, a symmetric relation R and $\alpha = e^\varepsilon \geq 1$, determine the smallest δ such that

$\mathcal{M}$ is (ε, δ)-differentially private with respect to R. The value δ is captured by the skewed total variation function between two measures ν, ν' on $(\Sigma^\omega, \mathcal{F})$ as follows: $tv_\alpha(\nu, \nu') = \sup_{E \in \mathcal{F}} \Delta_\alpha(\nu(E), \nu'(E))$. When used on $\nu_s, \nu_{s'}$ and $\alpha = e^\varepsilon$, $tv_\alpha(s, s')$ gives the required δ between states s, s'.

This chapter observes that significant simplification occurs by splitting the two main parts of the maximum, taking only the 'left variant'. Whilst Δ_α is symmetric, this property is broken to introduce a new distance function Λ_α (similarly to [BO13]). An analogous total variation distance lv_α is the defined, which will be the main object of study.

Definition 5.1 (Asymmetric skewed total variation distance). Let $\alpha \geq 1$. Given two measures ν, ν' on $(\Sigma^\omega, \mathcal{F})$, let

$$lv_\alpha(\nu, \nu') = \sup_{E \in \mathcal{F}} \Lambda_\alpha\left(\nu(E), \nu'(E)\right),$$

where $\Lambda_\alpha(a, b) = \max\{a - \alpha b, 0\}$. ◀

For notational simplicity $lv_\alpha(s, s')$ is written for $lv_\alpha(\nu_s, \nu_{s'})$. Note that it is not required to take the maximum with zero, that is $lv_\alpha(\nu, \nu') = \sup_{E \in \mathcal{F}} \nu(E) - \alpha\nu'(E)$, since there is always an event such that $\nu'(E) = 0$, in particular $\nu(\emptyset) = 0$. Observe that Δ_α and Λ_α are not metrics as $\Delta_\alpha(a, b) = 0 \not\Longrightarrow a = b$, and not a pseudometric as the triangle inequality does not hold. The new distance Λ_α (and lv_α) is not symmetric, while Δ_α and tv_α are.

If $\alpha = 1$, then $lv_1 = tv_1 = tv$, since if ν, ν' are probability measures and we have $\nu(E) = 1 - \nu(\overline{E})$ then $\sup_{E \in \mathcal{F}} |\nu(E) - \nu'(E)| = \sup_{E \in \mathcal{F}} \nu(E) - \nu'(E) = \sup_{E \in \mathcal{F}} \nu'(E) - \nu(E)$, i.e., despite the use of the absolute value in the definition of tv, it is not required.

Differential privacy can be reformulated in terms of tv_α and lv_α.

Proposition 5.2. *Given a labelled Markov chain $\mathcal{M}$ and a symmetric relation $R \subseteq Q \times Q$, the following properties are equivalent for $\alpha = e^\varepsilon$:*

- *$\mathcal{M}$ is (ε, δ)-differentially private w.r.t. R,*
- *$\max_{(s,s') \in R} tv_\alpha(s, s') \leq \delta$, and*
- *$\max_{(s,s') \in R} lv_\alpha(s, s') \leq \delta$.*

The next sections focus on computing lv_α to determine the 'level' of differential privacy for a given ε. Like in Chapter 4, e^ε is referred to as α. For the purposes of the complexity arguments, α is assumed to be rational with $O(\mathit{size}(\mathcal{M}))$-bit representation.

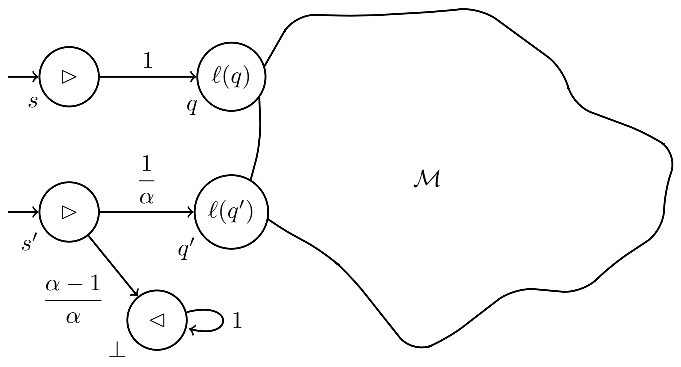


Figure 5.2: Markov chain $\mathcal{M}'$ in the reduction from $tv(q,q')$ to $lv_\alpha(s,s')$.

5.2 lv_α is not computable

As noted earlier (Theorem 3.11), $tv(s,s')$ turns out to be surprisingly difficult to compute: the threshold distance problem (whether the distance is strictly greater than a given threshold) is undecidable, and the non-strict variant of the problem ("greater or equal") is not known to be decidable [Kie18]. The undecidability result is shown by reduction from the emptiness problem for probabilistic automata to the threshold distance problem for finite-word transition-labelled Markov chains. Recall that such chains are a special case of the more general definition of infinite-word state-labelled Markov chains. Thus, the problem is undecidable in this case also.

Chapter 4 argued that tv_α was not computable on the basis that the threshold problem on $tv_1 = tv$ is undecidable. Since $tv = lv_1$, then also $lv_1(s,s') > \theta$ is undecidable. This is not special, that is, the problem remains undecidable for any fixed $\alpha > 1$. In other words, no value of the privacy parameter ε makes it possible to compute the optimal δ exactly.

Theorem 5.3. *Finding a value of tv reduces in polynomial time to finding a value of lv_α for any fixed $\alpha > 1$.*

Proof. Given a labelled Markov chain $\mathcal{M} = \langle Q, \Sigma, \mu, \ell\rangle$, and states q, q' for which the answer of $tv(q,q')$ is required, a new labelled Markov chain $\mathcal{M}'$ can be constructed containing states s, s', for which $lv_\alpha(s,s') = tv(q,q')$.

Define $\mathcal{M}' = \langle Q \cup \{s, s', \perp\}, \Sigma', \mu', \ell'\rangle$, with $\ell'(s) = \ell'(s') = \triangleright$, $\ell'(\perp) = \triangleleft$, $\ell'(x) = \ell(x)$ for all $x \in Q$, $\Sigma' = \Sigma \cup \{\triangleright, \triangleleft\}$,

$$\mu'_s(q) = 1, \quad \mu'_{s'}(q') = \frac{1}{\alpha}, \quad \mu'_{s'}(\perp) = \frac{\alpha - 1}{\alpha}, \quad \mu'_\perp(\perp) = 1, \qquad \text{and}$$

$\mu'_x(y) = \mu_x(y)$ for all $x, y \in Q$. The reduction, sketched in Figure 5.2, adds three new states, so can be done in polynomial time.

It remains to show that $lv_\alpha(s, s') = tv(q, q')$.

Consider $E \in \mathcal{F}_\Sigma$, observe that $\nu_q(E) = \nu_s(E')$ and $\nu_{q'}(E) = \alpha\nu_{s'}(E')$, where $E' = \{\triangleright w \mid w \in E\} \in \mathcal{F}_{\Sigma'}$. Then $\nu_q(E) - \nu_{q'}(E) = \nu_s(E') - \alpha\nu_{s'}(E')$ and $lv_\alpha(s, s') \geq tv(q, q')$.

Conversely, consider an event $E' \in \mathcal{F}_{\Sigma'}$. Since the character $\triangleleft$ can only be reached from s', any word using it contributes negatively to the difference. Hence intersecting the event with $\triangleright\Sigma^\omega$, to remove $\triangleleft$, can only increase the difference. The character $\triangleright$ must occur (only) as the first character of every (useful) word in E'. Let $E = \{w \mid \triangleright w \in E' \cap \triangleright\Sigma^\omega\} \in \mathcal{F}_\Sigma$, then $\nu_q(E) - \nu_{q'}(E) \geq \nu_s(E') - \alpha\nu_{s'}(E')$. Thus $tv(q, q') \geq lv_\alpha(s, s')$. □

Since an oracle to solve decision problems for lv_α would solve problems for tv, obtaining the following result.

Corollary 5.4. *Given $s, s' \in Q$ and $\theta \in [0, 1]$, deciding if $lv_\alpha(s, s') > \theta$ is undecidable for all $\alpha \geq 1$.*

It is not clear that lv_α reduces easily to tv. Arguments along the lines of the proof of Theorem 5.3 may not result in a Markov chain due to non-stochastic transitions, or modifications to the $s \to q$ branch may result in new maximising events.

5.3 Approximation of lv_α

Given that lv_α cannot be computed exactly, approximation is considered; that is, the problem, given $\gamma > 0$, of finding some x such that $|x - lv_\alpha(s, s')| \leq \gamma$. For $\alpha = 1$, Chen and Kiefer [CK14; Kie18] show that approximating $tv = lv_1$ is possible in **PSPACE** and is #**P**-hard. In Theorem 5.5 it is shown that the case $\alpha = 1$ is not special and the theorems of Chen and Kiefer generalise; that is, when $\alpha > 1$, lv_α can also be approximated and the same complexity bounds apply.

Remark. Typically one might suggest being ε close ($|x - lv_\alpha(s, s')| \leq \varepsilon$). To avoid confusion with the differential privacy parameter, γ is used to denote the level of accuracy. ◀

Theorem 5.5. *For finite-word Markov chains, approximation of $lv_\alpha(s, s')$ within γ can be performed in* **PSPACE** *and is* #**P**-*hard.*

Observe *additive* approximation is considered here. This is the natural approximation scheme for δ, the *additive* error term of differential privacy. Thus, once it is known that $|x - lv_\alpha(s, s')| \leq \gamma$, it is known that $x + \gamma$ is a sound estimate for δ. The *multiplicative* variant of the approximation problem remains open.

First, the following claim of Kiefer [Kie18] is recalled as a useful component of the proof.

Claim 5.6. *[Kie18, Lemma 13] Fix $\theta \in (0, 1)$ and $n \in \mathbb{N}$, for a word $w \in \Sigma^{\leq n}$ then a value $\tilde{\nu}_s(w) \in [\nu_s(w)(1 - \theta), \nu_s(w)(1 + \theta)]$, exists and can be computed.*

Proof of Theorem 5.5. For the lower bound, note that approximating *tv* is #**P**-hard [Kie18], by a reduction from the #*NFA*, a #**P**-complete problem [KSM95]. That is, given a non-deterministic finite automaton $\mathcal{N}$ and $n \in \mathbb{N}$ in unary, determine the number of accepted words of $\mathcal{N}$ up to length n, $|\Sigma^n \cap L(A)|$. Since computing *tv* can be reduced to computing lv_α (Theorem 5.3), approximating lv_α is #**P**-hard as well. The hardness result applies to finite-word transition-labelled Markov chains, thus also applies to the more general infinite word labelled Markov chains.

The argument for the upper bound shows that the i^{th} bit of an x such that $|x - lv_\alpha(s, s')| \leq \gamma$ can be found in **PSPACE**. The approach, inspired by [Kie18], is to consider the maximising event of $lv_\alpha(s, s') = \sup_{E \subseteq \Sigma^*} \nu_s(E) - \alpha\nu_{s'}(E)$, which turns out to be $W = \{w \mid \nu_s(w) \geq \alpha\nu_{s'}(w)\}$, so that $lv_\alpha(s, s') = \nu_s(W) - \alpha\nu_{s'}(W)$. This choice of the maximising event only applies to finite-word Markov chains, thus the proof does not extend in full generality to infinite-word Markov chains. The shape of the event is the key difference between the proof here and [Kie18], which uses events of the form $\{w \mid \nu_s(w) \geq \nu_{s'}(w)\}$.

Let $\overline{W}$ denote the complement of W and let $\nu_s(\overline{W})$ be approximated by a number X and $\nu_{s'}(W)$ by a number Y. Normally, one would expect X to be close to $\nu_s(\overline{W})$ and Y to be close to $\nu_{s'}(W)$. Here, like in [Kie18], the trick is to require only that $\nu_s(\overline{W}) + \alpha\nu_{s'}(W)$ be close to $X + \alpha Y$. It is then argued that, for specific X, Y with this property, one can find any bit of $X + \alpha Y$.

The details are formalised below:

Due to the countable nature of finite word automata, the function $lv_\alpha(s, s')$ to be approximated, can be rewritten in the following form: $lv_\alpha(s, s') = \sup_{E \subseteq \Sigma^*} \nu_s(E) - \alpha \cdot \nu_{s'}(E) = \nu_s(W_2) - \alpha \cdot \nu_{s'}(W_2)$ where $W_2 = \{w \in \Sigma^* | \ \nu_s(w) \geq \alpha \cdot \nu_{s'}(w)\}$.

To see this, consider $E \subseteq \Sigma^*$ as $E = E_1 \cup E_2$, where $E_1 = E \cap W_2$ and $E_2 = E \setminus W_2$. Note the value of $\nu_s(E)$ is a (possibly infinite) sum over the measure on individual words. Then if $E_1 \subsetneq W_2$, then the event $E' = E_2 \cup W_2$ results only in at least as large a difference since $\nu_s(w) - \alpha \cdot \nu_{s'}(w) \geq 0$ for all

$w \in W_2$. If $E_2 \neq \emptyset$ then $E' = E_1$ results in at least as large a differences since $\nu_s(w) - \alpha \cdot \nu_{s'}(w) < 0$ for all $w \in E_2$.

Then, let $W_1 = \Sigma^* \setminus W_2 = \{w \in \Sigma^* \mid \nu_s(w) < \alpha \cdot \nu_{s'}(w)\}$, then we have

$$lv_\alpha(s, s') = \nu_s(W_2) - \alpha \cdot \nu_{s'}(W_2) = 1 - \nu_s(W_1) - \alpha \cdot \nu_{s'}(W_2). \tag{5.1}$$

It is therefore equivalent to focus on approximation of $\nu_s(W_1) + \alpha \cdot \nu_{s'}(W_2)$.

Given λ, one can compute n such that $\nu_s(\Sigma^{>n}) \leq \lambda$ and $\alpha \cdot \nu_{s'}(\Sigma^{>n}) \leq \lambda$ [Kie18, Lemma 12]. Restrict sets W_1, W_2 only to those defined on words shorter than n, to that end, let $W_1' = W_1 \cap \Sigma^{\leq n}$ and $W_2' = W_2 \cap \Sigma^{\leq n}$. This allows approximation as follows:

$$\nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') \leq \nu_s(W_1) + \alpha \cdot \nu_{s'}(W_2) \leq \nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') + 2\lambda. \tag{5.2}$$

It remains therefore to approximate $\nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2')$. Fix θ, then by Claim 5.6 $\tilde{\nu}_s(w)$ can be computed as an approximation of $\nu_s(w)$, using this let $\widetilde{W_1} = \left\{w \in \Sigma^{\leq n} \mid \tilde{\nu}_s(w) < \alpha \cdot \tilde{\nu}_{s'}(w)\right\}$ and $\widetilde{W_2} = \left\{w \in \Sigma^{\leq n} \mid \tilde{\nu}_s(w) \geq \alpha \cdot \tilde{\nu}_{s'}(w)\right\}$ Note that $\widetilde{W_1} \cup \widetilde{W_2} = \Sigma^{\leq n}$. It is then necessary to estimate the measures of the events $\widetilde{W_1} \cap W_2'$ and $\widetilde{W_2} \cap W_1'$, which can be interpreted as the error when words are placed in the 'wrong' event due to approximation error.

Suppose $w \in \widetilde{W_1} \cap W_2'$. Since $w \in W_2'$ then $\alpha \cdot \nu_{s'}(w) \leq \nu_s(w)$, and since $w \in \widetilde{W_1}$ then $\nu_s(w)(1-\theta) \leq \tilde{\nu}_s(w) < \alpha \cdot \tilde{\nu}_{s'}(w) \leq \alpha \cdot \nu_{s'}(w)(1+\theta)$ so $\nu_s(w) < \alpha \cdot \nu_{s'}(w) + \theta\alpha \cdot \nu_{s'}(w) + \theta \cdot \nu_s(w)$.

Suppose $w \in \widetilde{W_2} \cap W_1'$. Since $w \in W_1'$ then $\nu_s(w) < \alpha \cdot \nu_{s'}(w)$, and since $w \in \widetilde{W_2}$ then $\alpha \cdot \nu_{s'}(w)(1-\theta) \leq \alpha\tilde{\nu}_s(w) \leq \tilde{\nu}_s(w) \leq \nu_s(w)(1+\theta)$ so $\alpha \cdot \nu_{s'}(w) \leq \nu_s(w) + \theta \cdot \nu_s(w) + \theta\alpha \cdot \nu_{s'}(w)$.

Lifting this to sets:

$$\begin{aligned}
\alpha \cdot \nu_{s'}(\widetilde{W_1} \cap W_2') &\leq \nu_s(\widetilde{W_1} \cap W_2') \\
&\leq \alpha \cdot \nu_{s'}(\widetilde{W_1} \cap W_2') + \theta\alpha \cdot \nu_{s'}(\widetilde{W_1} \cap W_2') + \theta \cdot \nu_s(\widetilde{W_1} \cap W_2') \\
&\leq \alpha \cdot \nu_{s'}(\widetilde{W_1} \cap W_2') + (1+\alpha)\theta
\end{aligned}$$

and

$$\begin{aligned}
\nu_s(\widetilde{W_2} \cap W_1') &< \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_1') \\
&\leq \nu_s(\widetilde{W_2} \cap W_1') + \theta \cdot \nu_s(\widetilde{W_2} \cap W_1') + \theta\alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_1') \\
&\leq \nu_s(\widetilde{W_2} \cap W_1') + (1+\alpha)\theta.
\end{aligned}$$

Recall that $\nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2')$ appears twice in the inequality in (5.2), which is now bounded:

$$\begin{aligned}
&\nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') && (5.3)\\
&\quad = \nu_s(\widetilde{W_1} \cap W_1') + \nu_s(\widetilde{W_2} \cap W_1') + \alpha \cdot \nu_{s'}(\widetilde{W_1} \cap W_2') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_2')\\
&\quad \le \nu_s(\widetilde{W_1} \cap W_1') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_1') + \nu_s(\widetilde{W_1} \cap W_2') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_2')\\
&\quad = \nu_s(\widetilde{W_1} \cap W_1') + \nu_s(\widetilde{W_1} \cap W_2') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_1') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_2')\\
&\quad = \nu_s(\widetilde{W_1}) + \alpha \cdot \nu_{s'}(\widetilde{W_2}) && (5.4)\\
&\quad = \nu_s(\widetilde{W_1} \cap W_1') + \nu_s(\widetilde{W_1} \cap W_2') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_1') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_2')\\
&\quad \le \nu_s(\widetilde{W_1} \cap W_1') + \alpha \cdot \nu_{s'}(\widetilde{W_1} \cap W_2') + (1+\alpha)\theta +\\
&\qquad\qquad \nu_s(\widetilde{W_2} \cap W_1') + (1+\alpha)\theta + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_2')\\
&\quad = \nu_s(\widetilde{W_1} \cap W_1') + \nu_s(\widetilde{W_2} \cap W_1') + \alpha \cdot \nu_{s'}(\widetilde{W_1} \cap W_2') + \alpha \cdot \nu_{s'}(\widetilde{W_2} \cap W_2') + 2(1+\alpha)\theta\\
&\quad = \nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') + 2(1+\alpha)\theta. && (5.5)
\end{aligned}$$

In particular from (5.3), (5.4) and (5.5):

$$\begin{aligned}
\nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') &\le \nu_s(\widetilde{W_1}) + \alpha \cdot \nu_{s'}(\widetilde{W_2}) && (5.6)\\
&\le \nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') + 2(1+\alpha)\theta. && (5.7)
\end{aligned}$$

Thus:

$$\begin{aligned}
\nu_s(\widetilde{W_1}) + \alpha \cdot \nu_{s'}(\widetilde{W_2}) - 2(1+\alpha)\theta &\\
&\le \nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') && \text{by } (5.7)\\
&\le 1 - lv_\alpha(s,s') && \text{by } (5.2) \text{ and } (5.1)\\
&\le \nu_s(W_1') + \alpha \cdot \nu_{s'}(W_2') + 2\lambda && \text{by } (5.2)\\
&\le \nu_s(\widetilde{W_1}) + \alpha \cdot \nu_{s'}(\widetilde{W_2}) + 2\lambda. && \text{by } (5.6)
\end{aligned}$$

So to obtain a γ approximation, choose θ, λ such that $2\lambda \le \frac{\gamma}{2}$ and $2(1+\alpha)\theta \le \frac{\gamma}{2}$. Recall that it is also necessary to ensure that $\nu_s(\Sigma^{>n}) \le \lambda$ and $\alpha \cdot \nu_{s'}(\Sigma^{>n}) \le \lambda$, by appropriate choice of n. This gives

$$\left|(\nu_s(\widetilde{W_1}) + \alpha \cdot \nu_{s'}(\widetilde{W_2})) - (1 - lv_\alpha(s,s'))\right| \le \frac{\gamma}{2}. \tag{5.8}$$

Then compute $\nu_s(\widetilde{W_1})$ and $\nu_{s'}(\widetilde{W_2})$ with accuracy $\frac{\gamma}{4}$ and $\frac{\gamma}{4\alpha}$, finding x s.t. $\left|x - \nu_s(\widetilde{W_1})\right| \le \frac{\gamma}{4}$ and y s.t. $\left|y - \nu_{s'}(\widetilde{W_2})\right| \le \frac{\gamma}{4\alpha}$. Combining the two results to obtain $\left|x + \alpha y - (\nu_s(\widetilde{W_1}) + \alpha \cdot \nu_{s'}(\widetilde{W_2}))\right| \le \frac{\gamma}{2}$ then combining with (5.8) to obtain

$$\left|x + \alpha y - (1 - lv_\alpha(s,s'))\right| \le \gamma.$$

It remains to show how to compute $\nu_s(\widetilde{W_1})$ within γ'. This can be computed similarly for $\nu_{s'}(\widetilde{W_2})$. This method is analogous to [Kie18, Lemma 14]. Recall

$\widetilde{W_1}$ contains words that are no longer than n and for which $\tilde{\nu}_s(w) < \alpha\tilde{\nu}_{s'}(w)$. Construct a probabilistic Turing machine, to sample a word w, letter by letter, according the transition probabilities of $\mathcal{M}$. If $|w|$ exceeds n, then the Turing machine rejects. Along the way compute $\tilde{\nu}_s(w)$ and $\tilde{\nu}_{s'}(w)$, which can be maintained in polynomial space. If at the end of the word $\tilde{\nu}_s(w) < \alpha\tilde{\nu}_{s'}(w)$, then the Turing machine accepts and otherwise rejects. Overall the Turing machine does not need more than polynomial space, as it does not need to remember the whole word. $\nu_s(\widetilde{W_1})$ is therefore the probability that the Turing machine accepts, which can be computed to accuracy γ' in polynomial space (**PSPACE**) [Lad89].

Technically this process produces only a single bit of the answer, so that in **PSPACE** the i^{th} bit of either p and q can be found, where $\left|\frac{p}{q} - \nu_s(\widetilde{W_1})\right| \leq \frac{\gamma}{4}$ and we have a procedure for the i^{th} bit of r and t for $\left|\frac{r}{t} - \nu_{s'}(\widetilde{W_2})\right| \leq \frac{\gamma}{4\alpha}$. In fact, the i^{th} bit of u or v is required, where $\frac{u}{v} = \frac{p}{q} + \frac{\alpha r}{t}$. Such arithmetic can be done in **NC** using *logspace*-uniform circuits [Koz92], thus the computations require only polylogarithmically many bits at any moment of the computation. Since there are exponentially many bits in p, q, r, t, α, accessing polylogarithmically many of these at any one time requires only polynomially many bits overall at any one time, for which each bit can be computed in **PSPACE**. □

5.4 A least fixed point bound ld_α

The exact computation of lv_α is not possible, and the approximation of lv_α is seemingly out of reach, thus it is desirable to bound lv_α from above by a computable quantity. For this purpose a new distance function ld_α will be defined. First a new variant of the Kantorovich lifting is required, as a technique to measure the distance between probability distributions on a set X, given a distance function between objects of X. lv_α can be reformulated using such a distance over the (infinite) trace distributions $\nu_s, \nu_{s'}$. The new distance function ld_α is then defined as the fixed point of the Kantorovich lifting of distances from individual states using (finite) transition distributions. This distance ld_α will be computable and acts as a sound bound on lv_α.

Hence this distance can be used to determine (ε, δ)-differential private w.r.t. relation R by bounding δ with $\max_{(s,s')\in R} ld_\alpha(s, s')$. This can be achieved in polynomial time with access to an **NP** oracle, by computing $ld_\alpha(s, s')$ exactly in this time ($|R|$ is polynomial with respect to the size of $\mathcal{M}$). This suggests a complexity lower than approximation (which is #**P**-hard by Theorem 5.5), although formally $\mathbf{P}^{\mathbf{NP}}$ and #**P** are not known to be comparable.

Definition 5.7 (Asymmetric Skewed Kantorovich Lifting)**.** For a set X, given $d : X \times X \to [0,1]$ a distance function and measures μ, μ', define

$$K^{\Lambda}_{\alpha}(d)(\mu,\mu') = \sup_{\substack{f:X\to[0,1] \\ \forall x,x'\in X\ \Lambda_\alpha(f(x),f(x'))\le d(x,x')}} \Lambda_\alpha\left(\int_X f d\mu, \int_X f d\mu'\right)$$

where f ranges over functions which are measurable w.r.t. μ and μ'. ◀

Remark. Asymmetric Skewed Kantorovich Lifting is based on the standard Kantorovich distance Definition 4.7, with the absolute value function replaced with the distance function Λ_α. Note that $K^{\Lambda}_{\alpha}(d)$ is equivalent to the standard Kantorovich distance for $\alpha = 1$ and d symmetric [Kan42; DD09]. If $|X| < \infty$ (for example when X is a finite set of states, Q), we have $\int_X f d\mu = \sum_{x\in X} f(x)\,\mu(x)$. Since Λ_α is not a metric, it does not fit in the framework of Chatzikokolakis et al. [CGPX14]. ◀

The interest in K^{Λ}_{α} is that it allows the reformulation of definition of the distance function lv_α. The goal is to measure the difference between measures over infinite traces $\nu_s, \nu_{s'}$, and so distance function over infinite words is lifted ($d : \Sigma^\omega \times \Sigma^\omega \to [0,1]$). In particular, by lifting the discrete metric $\mathbb{1}_{\neq}$ as defined in Definition 4.15.

Lemma 5.8. *$lv_\alpha(s,s') = K^{\Lambda}_{\alpha}(\mathbb{1}_{\neq})(\nu_s,\nu_{s'})$.*

Proof. The use of $\mathbb{1}_{\neq}$ makes it so that $\Lambda_\alpha(f(t), f(t')) \le \mathbb{1}_{\neq}(t,t)$ is rather no restriction at all, since either $t = t'$, thus $\Lambda_\alpha(f(t), f(t')) = 0$ or $t \neq t'$, thus $\mathbb{1}_{\neq}(t,t') = 1$. So $K^{\Lambda}_{\alpha}(\mathbb{1}_{\neq})$ takes supremum over all measurable $f : \Sigma^\omega \to [0,1]$.

Using the argument of Lemma 4.16 of Chapter 4 it is clear there is a 1–1 correspondence between indicator functions and events entailing that $lv_\alpha(s,s') = \sup_{f:\Sigma^\omega\to\{0,1\}} \hat{f}(\mu) - \alpha\hat{f}(\mu')$. Then by Lemma 4.17 of Chapter 4 observe that extending f to non-indicator functions $f : \Sigma^\omega \to [0,1]$ does not obtain larger supremum. □

Corollary 5.9. *Let $\nu_s, \nu_{s'}$ be measures on $(\Sigma^\omega, \mathcal{F})$, then*

$$K^{\Lambda}_{\alpha}(\mathbb{1}_{\neq})(\nu_s,\nu_{s'}) = \sup_{f:\Sigma^\omega\to\{0,1\}} \Lambda_\alpha\left(\int f d\nu_s, \int f d\nu_{s'}\right).$$

Since computing lv_α, or now $K^{\Lambda}_{\alpha}(\mathbb{1}_{\neq})(\nu_s,\nu_{s'})$, is difficult, an an upper bound on lv_α is introduced, inspired by bisimilarity distances, called ld_α. This will be the least fixed point of $\Gamma^{\Lambda}_{\alpha}$, a function which measures (relative to a distance function d) the distance between the transition distributions of s, s' where s, s' share a label, or 1 when they do not.

Definition 5.10. Let $\Gamma_\alpha^\Lambda : [0,1]^{Q\times Q} \to [0,1]^{Q\times Q}$ be defined as follows.

$$\Gamma_\alpha^\Lambda(d)(s,s') = \begin{cases} K_\alpha^\Lambda(d)(\mu_s,\mu_{s'}) & \text{if } \ell(s)=\ell(s') \\ 1 & \text{otherwise} \end{cases} \qquad \blacktriangleleft$$

The utility of this function is that rather than using the Kantorovich lifting over infinite trace distributions, it is now over finite transition distributions ($\mu_s \in \mathit{Dist}(Q)$).

As $[0,1]^{Q\times Q}$ is a complete lattice with the pointwise order $\sqsubseteq$, and Γ_α is monotone with respect to that order, Γ_α^Λ has a least fixed point [Tar55]. ld_α is defined to be exactly that point, for which it can then be shown that ld_α is a sound upper bound on lv_α, entailing the desired privacy guarantee.

Definition 5.11. Let $ld_\alpha : Q \times Q \to [0,1]$ be the least fixed point of Γ_α^Λ. ◀

A sound upper bound on lv_α is required to provide a guarantee of privacy, ld_α provides this bound.

Theorem 5.12. *$lv_\alpha(s,s') \le ld_\alpha(s,s')$ for every $s,s' \in Q$.*

The proof of Theorem 5.12 proceeds similarly to Theorem 4.11 in Chapter 4, indeed much of the machinery developed there is reused. Firstly, note using Lemma 5.8, it is sufficient to show $K_\alpha^\Lambda(\mathbb{1}_{\neq})(\nu_s,\nu_{s'}) \le ld_\alpha(s,s')$ for every $s,s' \in Q$.

It is necessary to take care of the base case using an additional result (Lemma 5.13) to compensate for the fact that Λ_α is not a metric (it is not possible to conclude $\Lambda_\alpha(f(x),f(x')) = 0$ entails $f(x) = f(x')$).

Lemma 5.13. *Consider $f : X \to [0,1]$ such that $\forall x,x' : \Lambda_\alpha(f(x),f(x')) = 0$. Then $\forall \nu,\nu' : \Lambda_\alpha\left(\hat{f}(\nu),\hat{f}(\nu')\right) = 0$.*

Proof. Consider the range of values f could take, let $a = \inf_x f(x)$. Since for all x, $\Lambda_\alpha(f(x),a) \le 0$, we have $f(x) - \alpha a \le 0$ i.e. $f(x) \le \alpha a$. By definition of a we have $f(x) \ge a$ concluding $f(x) \in [a,\alpha a]$ for all x.

Now consider $\int_X f d\nu$ where $f(x) \in [a,\alpha a]$ and $\int_X d\nu = 1$.

$$\hat{f}(\nu) = \int_X f d\nu \ge \int_X a d\nu = a\int_X d\nu = a$$
$$\hat{f}(\nu) = \int_X f d\nu \le \int_X \alpha a d\nu = \alpha a\int_X d\nu = \alpha a$$

So for all ν then $\hat{f}(\nu) = \int_X f d\nu \in [a,\alpha a]$, i.e. the expectation must also lie in this range. Next notice that any two numbers in this range give distance zero, in particular the expectations.

Consider $\hat{f}(\mu), \hat{f}(\mu') \in [a, \alpha a]$

$$\begin{aligned} 0 \leq \Lambda_\alpha(\hat{f}(\mu), \hat{f}(\mu')) &= \max\left\{\hat{f}(\mu) - \alpha\hat{f}(\mu'), 0\right\} \\ &\leq \max\{\alpha a - \alpha a, 0\} \\ &= \max\{0, 0\} = 0. \end{aligned}$$ □

Theorem 5.12 is then shown by invoking Lemma 4.23 and then relaxing the restriction up to length h ($\mathbb{1}^h_{\neq}$) to the supremum over h ($\mathbb{1}_{\neq}$).

Proof of Theorem 5.12. By Lemma 5.13, Definition 5.7 and Definition 5.10 we have that Λ_α, K^Λ_α and ld_α satisfy Lemma 4.23 giving

$$K^\Lambda_\alpha(\mathbb{1}^h_{\neq})(\nu_s, \nu_{s'}) \leq ld_\alpha(s, s) \text{ for all } s, s' \in Q \quad \text{ for all } h \in \mathbb{N}.$$

Then recall the proof of Lemma 4.21, for which it is shown: $\forall\varepsilon\ \exists h$ such that $K_\alpha(\mathbb{1}_{\neq}) - K_\alpha(\mathbb{1}^h_{\neq}) \leq \varepsilon$. Here the two sides of Δ_α are already considered separately and so it also implies that $\forall\varepsilon > 0\ \exists h$ such that $K^\Lambda_\alpha(\mathbb{1}_{\neq}) - K^\Lambda_\alpha(\mathbb{1}^h_{\neq}) \leq \varepsilon$, implying if $K^\Lambda_\alpha(\mathbb{1}^h_{\neq}) \leq ld_\alpha$ for all h then $K^\Lambda_\alpha(\mathbb{1}_{\neq}) \leq ld_\alpha$, entailing the result. □

5.4.1 Comparison with bd_α from Chapter 4

This upper bound on lv_α is stronger (or at least no worse) than the bound obtained in Chapter 4. Recall from Chapter 4 that bd_α is defined as the least fixed point of

$$\Gamma^\Delta_\alpha(d)(s, s') = \begin{cases} K^\Delta_\alpha(d)(\mu_s, \mu_{s'}) & \text{if } \ell(s) = \ell(s') \\ 1 & \text{otherwise} \end{cases}$$

where $K^\Delta_\alpha(d)$ behaves as $K^\Lambda_\alpha(d)$, but uses $\Delta_\alpha(a, b) = \max\{a - \alpha b, b - \alpha a, 0\}$ rather than $\Lambda_\alpha(a, b) = \max\{a - \alpha b, 0\}$.

Remark. In Chapter 4, Γ^Δ_α was simply Γ_α and K^Δ_α was simply K_α. Here these distances are annotated with Δ to avoid confusion with similar distances defined in this chapter. ◀

Theorem 5.14. *$\max\{ld_\alpha(s, s'), ld_\alpha(s', s)\} \leq bd_\alpha(s, s')$ for every $s, s' \in Q$.*

Proof. Given a matrix A, let A^T be its transpose. Consider bd_α and ld_α as matrices. bd_α is the least fixed point of Γ^Δ_α so $\Gamma^\Delta_\alpha(bd_\alpha)(s, s') = bd_\alpha(s, s')$. Also notice that $\Gamma^\Lambda_\alpha(bd_\alpha)(s, s') \leq \Gamma^\Delta_\alpha(bd_\alpha)(s, s')$, since $K^\Lambda_\alpha(bd_\alpha) \sqsubseteq K^\Delta_\alpha(bd_\alpha)$. To see this, note that, because $bd_\alpha = bd_\alpha^\intercal$, the relevant set of functions is the same, but the objective function in the supremum is smaller.

Hence $\Gamma_\alpha^\Lambda(bd_\alpha) \sqsubseteq bd_\alpha$, i.e. bd_α is also a pre-fixed point of Γ_α^Λ. Since ld_α is the least pre-fixed point of Γ_α^Λ hence $ld_\alpha \sqsubseteq bd_\alpha$. By symmetry, $bd_\alpha = bd_\alpha^\intercal$ giving $ld_\alpha \sqsubseteq bd_\alpha^\intercal$ and then $ld_\alpha^\intercal \sqsubseteq bd_\alpha$. Entailing $\max\{ld_\alpha(s,s'), ld_\alpha(s',s)\} \leq bd_\alpha(s,s')$ for every $s, s' \in Q$. □

Remark. Example 5.29 on page 89 demonstrates the inequality in Theorem 5.14 can be strict. ◀

5.4.2 Computing ld_α

The standard variant of the Kantorovich metric is often presented in its dual formulation. Recall from Lemma 4.28, in the case of finite distributions, the asymmetric skewed Kantorovich distance exhibits a dual form. This is obtained through the standard recipe for dualising linear programming. Interestingly, this technique yields a linear optimisation problem over a polytope independent of d, and that will prove useful in the computation of ld_α.

Remark. Let X be finite and given $d : X \times X \to [0,1]$ a distance function, $\mu, \mu' \in \mathit{Dist}(X)$ we have

$$K_\alpha^\Lambda(d)(\mu,\mu') = \min_{(\omega,\eta)\in\Omega^\alpha_{\mu,\mu'}} \left(\sum_{s,s'\in X} \omega_{s,s'} \cdot d(s,s') + \sum_{s\in X} \eta_s \right),$$

where $\Omega^\alpha_{\mu,\mu'} =$

$$\left\{ (\omega,\eta) \in [0,1]^{X\times X} \times [0,1]^X \quad \middle| \quad \begin{array}{l} \exists \gamma,\tau \in [0,1]^X \\ \forall i \in X : \sum_{j\in X} \omega_{i,j} + \tau_i - \gamma_i + \eta_i = \mu(i) \\ \forall j \in X : \sum_{i\in X} \omega_{i,j} + \frac{\tau_j - \gamma_j}{\alpha} \leq \mu'(j) \end{array} \right\}.$$

◀

When the distance is taken between states, i.e. $X = Q$, $\Omega^\alpha_{s,s'}$ is written to mean $\Omega^\alpha_{\mu_s,\mu_{s'}}$. Take $V(\Omega^\alpha_{s,s'})$ to be the vertices of the polytope.

Theorem 5.15. *ld_α can be computed in polynomial time with access to an* **NP** *oracle.*

Definition 5.16 (LD$_\alpha$-Threshold).

INPUT	An LMC $\mathcal{M}$, states $s, s' \in Q$ and a threshold $\theta \in [0,1] \cap \mathbb{Q}$
OUTPUT	is $ld_\alpha(s,s') \leq \theta$? ◀

$$\text{LD}_\alpha\text{-}\textsc{Threshold}(s, s', \theta) =$$

$$\exists (d_{i,j})_{i,j\in Q} \quad \bigwedge_{i,j\in Q} (0 \leq d_{i.j} \leq 1) \quad \wedge \; d_{s,s'} \leq \theta$$
$$\wedge \bigwedge_{q,q'\in Q} \begin{cases} d_{q,q'} = 1 & \text{if } \ell(q) \neq \ell(q') \\ \mathit{couplingConstraint}(d, q, q') & \text{if } \ell(q) = \ell(q') \end{cases}$$

$$\mathit{couplingConstraint}(d, q, q') =$$

$$\exists (\omega_{i,j})_{i,j\in Q} \quad \exists (\gamma_i)_{i\in Q} \quad \exists (\tau_i)_{i\in Q} \quad \exists (\eta_i)_{i\in Q}$$
$$\sum_{i,j\in Q} \omega_{i,j} \cdot d_{i,j} + \sum_i \eta_i \leq d_{q,q'} \quad \wedge \quad \bigwedge_{i,j\in Q} (0 \leq \omega_{i,j} \leq 1)$$
$$\wedge \bigwedge_{i\in Q} \begin{cases} 0 \leq \gamma_i \leq 1 \\ 0 \leq \tau_i \leq 1 \\ 0 \leq \eta_i \leq 1 \end{cases}$$
$$\wedge \bigwedge_{i\in Q} \sum_{j\in Q} \omega_{i,j} - \gamma_i + \tau_i + \eta_i = \mu_q(i)$$
$$\wedge \bigwedge_{j\in Q} \sum_{i\in Q} \omega_{i,j} + \frac{\tau_j - \gamma_j}{\alpha} \leq \mu_{q'}(j)$$

Figure 5.3: **NP** formula for LD$_\alpha$-Threshold.

First note, the LD$_\alpha$-Threshold problem is in **NP**. This is achieved through the formula shown in Figure 5.3, based on Figure 4.3 and that of van Breugel, Sharma and Worrell [BSW07] who used a similar formula for approximating standard bisimilarity distances. The problem can be solved in **NP** as each of the variables can be shown to be satisfied in the optimal solution with rational numbers that are of polynomial size (see Theorems 4.29 and 4.31). It suffices to guess these numbers (non-deterministically) and verify the correctness of the formula in polynomial time.

Like for bd_α, since the threshold problem can be solved in **NP**, it is possible to approximate the value using binary search with polynomial overhead to arbitrary accuracy γ, finding a value x such that $|x - ld_\alpha(s, s')| \leq \gamma$. One can then find the exact value of $ld_\alpha(s, s')$ in polynomial time assuming the oracle. Using the fact that ld_α is rational and its size is polynomially bounded, one can find a sufficiently close approximation to a carefully chosen level of precision and then finding the relevant rational with the continued fraction algorithm (recall [GLS88, Section 5.1] or [EY10]), entailing Theorem 5.15.

5.5 A greatest fixed point bound lgd_α

The previous section used the least fixed point of Γ^Λ_α, which finds the fixed point closest to the objective lv_α. Now consider relaxing this requirement, to find a fixed point in polynomial time. To do this lgd_α will be introduced, expressing the greatest fixed point of another operator which can be represented as a linear program that can be solved in polynomial time. Relaxing to any fixed point could of course be much worse than ld_α, so first the fixed point function, Γ^Λ_α, is refined to reduce the potential gap. This is done by characterising the elements which are zero in ld_α and fixing these as such; so that they cannot be larger in the greatest fixed point.

Refinement of Γ^Λ_α

In the case of standard bisimulation distances the kernel of ld_1, that is $\{(s,s') \mid ld_1(s,s') = 0\}$, is exactly bisimilarity. The kernel of ld_α is considered next, to define a new relation $\sim_\alpha$, which will be called skewed bisimilarity, which captures zero distance.

Definition 5.17. Let a relation $R \subseteq Q \times Q$ have the property

$$\begin{aligned}(s,s') \in R \iff{}& \ell(s) = \ell(s') \land \\ & \exists\ (\omega,\eta) \in \Omega^\alpha_{s,s'} \text{ s.t. } (\omega_{u,v} > 0 \implies (u,v) \in\ R)\ \land\ \forall u\ \eta_u = 0.\end{aligned}$$

Arbitrary unions of such relations also maintain the property, thus a largest such relation exists. Let $\sim_\alpha$ be the largest relation with this property. ◀

Remark. When $\alpha = 1$ the formulation corresponds to an alternative characterisation of bisimilarity [JL91; TB16], so $\sim_1 = \sim$. ◀

Lemma 5.18. *$ld_\alpha(s,s') = 0$ if and only if $s \sim_\alpha s'$.*

Proof.

Only if direction: $ld_\alpha(s,s') = 0 \implies s \sim_\alpha s'$

Let $X = \{(s,s') \mid ld_\alpha(s,s') = 0\}$ and consider $(s,s') \in X$. By definition, we have $ld_\alpha(s,s') = \inf_{\omega,\eta\in\Omega_{s,s'}} \sum_{u,v} \omega_{u,v} ld_\alpha(u,v) + \sum_i \eta_i = 0$. So $\eta = \vec{0}$ and whenever $\omega_{u,v} > 0$ then $ld_\alpha(u,v) = 0$, so $(u,v) \in X$. Hence, concluding X satisfies the requirements of Definition 5.17, so $X \subseteq \sim_\alpha$.

If direction: $s \sim_\alpha s' \implies ld_\alpha(s,s') = 0$

For notation, recall $\mathbb{1}_{!\sim_\alpha}(s,s') = 0$ if $(s,s') \in \sim_\alpha$ and 1 otherwise. Next, it is shown that $\mathbb{1}_{!\sim_\alpha}$ is a prefixed point of Γ^Λ_α, i.e. $\Gamma^\Lambda_\alpha(\mathbb{1}_{!\sim_\alpha}) \sqsubseteq \mathbb{1}_{!\sim_\alpha}$.

Firstly, if $\mathbb{1}_{!\sim_\alpha}(s,s') = 1$ then $\Gamma^\Lambda_\alpha(\mathbb{1}_{!\sim_\alpha})(s,s') \le 1$. Conversely, suppose $\mathbb{1}_{!\sim_\alpha}(s,s') = 0$ then $\Gamma^\Lambda_\alpha(\mathbb{1}_{!\sim_\alpha})(s,s') = \inf_{\omega,\eta\in\Omega_{s,s'}} \sum_{u,v} \omega_{u,v}\mathbb{1}_{!\sim_\alpha}(u,v) + \sum_i \eta_i$. By $s \sim_\alpha s'$, there exists $\omega, \eta_i \in \Omega_{s,s'}$ where $\omega_{u,v} > 0$ implies $(u,v) \in \sim_\alpha$, so $\mathbb{1}_{!\sim_\alpha}(u,v) = 0$ and $\eta_i = 0$. Thus $\Gamma^\Lambda_\alpha(\mathbb{1}_{!\sim_\alpha})(s,s') = 0$.

Therefore $\mathbb{1}_{!\sim_\alpha}$ is indeed a prefixed point of Γ^Λ_α. Since ld_α is the *least* fixed point then $ld_\alpha \sqsubseteq \mathbb{1}_{!\sim_\alpha}$. Hence, concluding $s \sim_\alpha s' \implies \mathbb{1}_{!\sim_\alpha}(s,s') = 0 \implies ld_\alpha(s,s') = 0$. □

Since $ld_\alpha(s,s') = 0$ implies $lv_\alpha(s,s') = 0$ by Theorem 5.12, this also provides a way to show that δ is zero, that is, to show ε-differential privacy holds. However, note this is not a complete method to do this, and there are bisimilarity distances focused on finding ε [CGPX14], which is explored further in the next chapter.

Lemma 5.19. *If $s \sim_\alpha s'$ then $lv_\alpha(s,s') = 0$.*

The pairs of states related by $\sim_\alpha$ need to be computed quickly and independently. In fact this can be done in polynomial time using a closure procedure, which will terminate after polynomially many rounds.

Proposition 5.20. *$\sim_\alpha$ can be computed in time polynomial in size$(\mathcal{M})$.*

Proof. The following is a standard refinement algorithm, let $A_0 = Q \times Q$ and $n = |Q|$. At each step compute

$$A_{i+1} = \left\{(s,s') \in A_i \mid \exists(\omega,\eta) \in \Omega^\alpha_{s,s'} : \eta = \mathbf{0} \ \land \ (\omega_{u,v} > 0 \implies (u,v) \in A_i)\right\}.$$

To find this, define $\mathbb{1}_{!A_i}$, a matrix such that $\mathbb{1}_{!A_i}(s,s') = 0$ if $(s,s') \in A_i$ and 1 otherwise. Apply Γ^Λ_α to $\mathbb{1}_{!A_i}$, which amounts to computing n^2 linear programs. Take A_{i+1} to be indices of the matrix where $\Gamma^\Lambda_\alpha(\mathbb{1}_{!A_i})$ is zero. At each step, at least one element is removed, or the procedure has stabilised so that the set will not change in subsequent rounds. After n^2 steps it is either stable (and possibly empty).

$A_{n^2} \subseteq \sim_\alpha$: after convergence there is a set such that $(s,s') \in A_{n^2} \implies \exists(\omega,\eta) \in \Omega^\alpha_{s,s'} : \eta = \mathbf{0} \ \land \ (\omega_{u,v} > 0 \implies (u,v) \in A_{n^2})$. $\sim_\alpha$ is the largest such set, so it contains A_{n^2}.

$\sim_\alpha \subseteq A_{n^2}$: by induction the process starts with $\sim_\alpha \subseteq A_0$ and only removes pairs not in $\sim_\alpha$. □

Recall that ld_α was defined as the least fixed point of Γ^Λ_α. Next, Γ^Λ_α is refined so the gap between the least fixed point and the greatest is as small as possible. This is done by fixing the known values of the least fixed point in the function, in particular the zero cases. Let

$$\Gamma'^\Lambda_\alpha(d)(s,s') = \begin{cases} 0 & \text{if } s \sim_\alpha s' \\ \Gamma^\Lambda_\alpha(d)(s,s') & \text{otherwise} \end{cases}$$

and observe that ld_α is also the least fixed point of Γ'^Λ_α.

Lemma 5.21. *ld_α is the least fixed point of Γ'^Λ_α.*

Proof. Let d' be the least fixed point of Γ'^Λ_α, and d be the least fixed point of Γ^Λ_α. The proof shows $d = d'$.

Case 1 ($d' \sqsubseteq d$).

To show d is a fixed point of Γ'^Λ_α:

$$\begin{aligned} d(s,s') &= \Gamma^\Lambda_\alpha(d)(s,s') \\ &= \begin{cases} K^\Lambda_\alpha(d)(\mu_s,\mu_{s'}) & \text{if } \ell(s)=\ell(s') \\ 1 & \text{otherwise} \end{cases} \end{aligned}$$

By Lemma 5.18, $d(s,s') = 0 \iff s \sim_\alpha s'$, which is separated into its own case:

$$\begin{aligned} &= \begin{cases} 0 & \text{if } s \sim_\alpha s' \\ K^\Lambda_\alpha(d)(\mu_s,\mu_{s'}) & \text{if } \ell(s)=\ell(s') \\ 1 & \text{otherwise} \end{cases} \\ &= \Gamma'^\Lambda_\alpha(d)(s,s'). \end{aligned}$$

So d is a fixed point of Γ'^Λ_α, but d' is the *least* fixed point of Γ'^Λ_α, hence $d' \sqsubseteq d$.

Case 2 ($d \sqsubseteq d'$).

If $s \sim_\alpha s'$ then

$$\begin{aligned} \Gamma^\Lambda_\alpha(\mathbb{1}_{!\sim_\alpha})(s,s') &= K^\Lambda_\alpha(\mathbb{1}_{!\sim_\alpha})(\mu_s,\mu_{s'}) \\ &= \inf_{\omega,\eta\in\Omega_{s,s'}} \sum_{u,v} \omega_{u,v}\mathbb{1}_{!\sim_\alpha}(u,v) + \sum_i \eta_i \\ &= 0. \end{aligned}$$

Clearly $d' \sqsubseteq \mathbb{1}_{!\sim_\alpha}$ giving $\Gamma^\Lambda_\alpha(d')(s,s') \le \Gamma^\Lambda_\alpha(\mathbb{1}_{!\sim_\alpha})(s,s') = 0$.

Concluding:

$$s \sim_\alpha s' \implies K_\alpha^\Lambda(d')(\mu_s, \mu_{s'}) = 0. \tag{5.9}$$

To show d' is a fixed point of Γ_α^Λ:

$$d' = \Gamma_\alpha'^\Lambda(d')(s, s')$$

$$= \begin{cases} 0 & \text{if } s \sim_\alpha s' \\ K_\alpha^\Lambda(d')(\mu_s, \mu_{s'}) & \text{if } \ell(s) = \ell(s') \\ 1 & \text{otherwise} \end{cases}$$

By (5.9) the values at $s \sim_\alpha s'$ can be replaced by $K_\alpha^\Lambda(d')(\mu_s, \mu_{s'})$:

$$= \begin{cases} K_\alpha^\Lambda(d')(\mu_s, \mu_{s'}) & \text{if } s \sim_\alpha s' \\ K_\alpha^\Lambda(d')(\mu_s, \mu_{s'}) & \text{if } \ell(s) = \ell(s') \\ 1 & \text{otherwise} \end{cases}$$

Combining cases with equivalent outcomes (noting $s \sim_\alpha s' \implies \ell(s) = \ell(s')$):

$$= \begin{cases} K_\alpha^\Lambda(d')(\mu_s, \mu_{s'}) & \text{if } \ell(s) = \ell(s') \\ 1 & \text{otherwise} \end{cases}$$

$$= \Gamma_\alpha^\Lambda(d')(s, s').$$

So d' is a fixed point of Γ_α^Λ, but d is the *least* fixed point of Γ_α^Λ, so $d \sqsubseteq d'$. □

Definition and Computation of lgd_α

Towards a more efficiently computable function, the greatest fixed point of $\Gamma_\alpha'^\Lambda$ is now studied.

Definition 5.22. Let lgd_α be the greatest fixed point of $\Gamma_\alpha'^\Lambda$. ◀

It is equivalent to consider the greatest *post*-fixed point. It turns out that when $\alpha = 1$, $lgd_1 = ld_1$ [CBW12]. It is not known if this holds for $\alpha > 1$, although it appears to do so on all tested examples, leading to the conjecture that it might (Conjecture 5.27). Whilst it may not necessarily be as tight a bound on lv_α as ld_α, lgd_α can also be used to bound lv_α, thus the δ parameter of (ε, δ)-differential privacy. Because $ld_\alpha(s, s') \le lgd_\alpha(s, s')$ for every $s, s' \in Q$, then Theorem 5.12 implies that $lv_\alpha(s, s') \le lgd_\alpha(s, s')$, for every $s, s' \in Q$.

Next, it is shown that lgd_α can be computed in polynomial time using the ellipsoid method for solving a linear program of exponential size, matching the result of Chen, van Breugel and Worrell [CBW12] for standard bisimilarity distances. It will not be necessary to express the entire linear program in one go (which would be of exponential size), but it will be necessary to represent any one constraint at a time, so each constraint must be expressible in polynomially many bits. To do this the representation of vertices of $\Omega^{\alpha}_{s,s'}$ is shown to be small.

Lemma 5.23. *Each $(\omega,\eta) \in V(\Omega^{\alpha}_{s,s'})$ are rational numbers requiring a number of bits polynomial in size$(\mathcal{M})$.*

Proof. Consider the polytope $\Omega'^{\alpha}_{\mu,\mu'} =$

$$\left\{ (\omega,\tau,\gamma,\eta) \in [0,1]^{Q\times Q} \times ([0,1]^{Q})^3 \quad \middle| \quad \begin{array}{l} \forall i : \sum_j \omega_{i,j} + \tau_i - \gamma_i + \eta_i = \mu(i) \\ \forall j : \sum_i \omega_{i,j} + \frac{\tau_j - \gamma_j}{\alpha} \le \mu'(j) \end{array} \right\}.$$

Each vertex is the intersection of hyperplanes defined in terms of μ, μ' (rationals given in the input $\mathcal{M}$), thus vertices of $\Omega'^{\alpha}_{\mu,\mu'}$ are rationals with representation size polynomial in the input. Vertices of $\Omega^{\alpha}_{\mu,\mu'} = \left\{(\omega,\eta) \mid \exists \tau,\gamma\ (\omega,\tau,\gamma,\eta) \in \Omega'^{\alpha}_{\mu,\mu'}\right\}$ require only fewer bits. $\square$

The following linear program expresses the greatest post-fixed point. It has polynomially many variables but exponentially many constraints (for each s, s' one constraint for each $\omega \in V(\Omega^{\alpha}_{s,s'})$). Since linear programs can be solved in polynomial time, the greatest fixed point can be found in exponential time using the exponential size linear program.

Proposition 5.24. *lgd_α is the optimal solution, $d \in [0,1]^{Q\times Q}$ of the following linear program: $\max_{d\in[0,1]^{Q\times Q}} \sum_{(u,v)\in Q\times Q} d_{u,v}$ subject to: for all $s, s' \in Q$:*

$$\begin{array}{lll} d_{s,s'} = 0 & & \textit{if } s \sim_\alpha s', \\ d_{s,s'} = 1 & & \textit{if } \ell(s) \neq \ell(s'), \\ d_{s,s'} \le \sum\limits_{(u,v)\in Q\times Q} \omega_{u,v} d_{u,v} + \sum\limits_{u\in Q} \eta_u & \textit{for all } (\omega,\eta) \in V(\Omega^{\alpha}_{s,s'}) & \textit{otherwise.} \end{array}$$

Proof. The $s \sim_\alpha s'$ and $\ell(s) \neq \ell(s')$ cases follow by definition. Observe that by the definition of lgd_α as a post-fixed point it is required that $d(s,s') \le \Gamma'^{\Lambda}_{\alpha}(d)(s,s') = K^{\Lambda}_{\alpha}(d)(s,s') = \min_{(\omega,\eta)\in\Omega^{\alpha}_{s,s'}} \sum_{(u,v)\in Q\times Q} \omega_{u,v} d_{u,v} + \sum_{u\in Q} \eta_u$ or equivalently, for all $(\omega,\eta) \in \Omega^{\alpha}_{s,s'}$: $d(s,s') \le \sum_{(u,v)\in Q\times Q} \omega_{u,v} d_{u,v} + \sum_{u\in Q} \eta_u$ $\square$

In the spirit of Chen et al. [CBW12], the exponential-size linear program given in Proposition 5.24 can be solved using the ellipsoid method, in polynomial time. Whilst the linear program has exponentially many constraints, it has only polynomially many variables. Therefore, the ellipsoid method can be used to solve the linear program in polynomial time, provided a polynomial-time separation oracle can be given [Sch99, Chapter 14]. The separation oracle takes as argument $d \in [0,1]^{Q\times Q}$, a proposed solution to the linear program and must decide whether d satisfies the constraints or not. If not then it must provide $\theta \in \mathbb{Q}^{|Q\times Q|}$ as a separating hyperplane such that, for every d' that does satisfy the constraints, $\sum_{u,v} d_{u,v}\theta_{u,v} < \sum_{u,v} d'_{u,v}\theta_{u,v}$.

The separation oracle will perform the following: for every $s, s' \in Q$ check that $d(s,s') \leq \min_{(\omega,\eta)\in\Omega^{\alpha}_{s,s'}} \omega\cdot d + \eta\cdot\mathbf{1}$. This is done by solving $\min_{(\omega,\eta)\in\Omega^{\alpha}_{s,s'}} \omega\cdot d + \eta\cdot\mathbf{1}$ using linear programming. If every check succeeds, return YES. If some check fails for s, s' return NO and

$$\theta_{u,v} = \begin{cases} \omega_{u,v} - 1 & (u,v) = (s,s') \\ \omega_{u,v} & \text{otherwise} \end{cases} \qquad \text{where } (\omega,\eta) = \operatorname*{argmin}_{(\omega,\eta)\in V(\Omega^{\alpha}_{s,s'})} d\cdot\omega + \eta\cdot\mathbf{1}.$$

Lemma 5.25. *θ is a separating hyperplane, i.e., it separates the unsatisfying d and all satisfying d':*

$$\sum_{u,v} d_{u,v}\theta_{u,v} < \sum_{u,v} d'_{u,v}\theta_{u,v}.$$

Proof. Recall each constraint is of the form

$$d'_{s,s'} \leq \sum_{u,v} d'_{u,v}\omega_{u,v} + \sum_{u} \eta_u.$$

Since d does not satisfy the constraint then

$$-\sum_{u} \eta_u > \sum_{u,v} d_{u,v}\omega_{u,v} - d_{s,s'} = \sum_{u,v} d_{u,v}\theta_{u,v}.$$

For each d' satisfying the constraint then

$$-\sum_{u} \eta_u \leq \sum_{u,v} d'_{u,v}\omega_{u,v} - d'_{s,s'} = \sum_{u,v} d'_{u,v}\theta_{u,v}.$$

Thus concluding for satisfying d':

$$\sum_{u,v} d_{u,v}\theta_{u,v} < -\sum_{i} \eta_i \leq \sum_{u,v} d'_{u,v}\theta_{u,v}.$$ □

Theorem 5.26. *lgd_{α} can be found in polynomial time in the size of $\mathcal{M}$.*

Proof. Checking $d(s,s') \leq \min_{\omega,\eta\in\Omega^{\alpha}_{s,s'}} \omega\cdot d + \eta\cdot\mathbf{1}$ is polynomial time. The linear program is of polynomial size, so runs in polynomial time in the size of the encoding of the linear program. Similarly finding θ is polynomial time by running essentially the same linear program and reading off the minimising result.

Because pairs (ω, η) are in $V(\Omega^{\alpha}_{s,s'})$, they are polynomial size in the size of $\mathcal{M}$, independent of d, by Lemma 5.23. Note that, unlike in Chen et al. [CBW12], the oracle procedure is not strongly polynomial, so the time to find θ may depend on the size of d, but the output θ and d remain polynomial in the size of the initial system.

Hence, concluding there is a procedure for computing lgd_{α} running in polynomial time [Sch99, Theorem 14.1, Page 173]. There exists a polynomial ψ where the ellipsoid algorithm solves the linear program in time $T \cdot \psi(size(\mathcal{M}))$, where T is the time the separation algorithm takes on inputs of size $\psi(size(\mathcal{M}))$. Since the $T \in poly(\psi(size(\mathcal{M})))$ and $\psi(size(\mathcal{M})) \in poly(size(\mathcal{M}))$ then $T \in poly(size(\mathcal{M}))$. Overall we have $T \cdot \psi(size(\mathcal{M})) \in poly(size(\mathcal{M}))$. $\square$

5.5.1 A unique fixed point?

ld_{α} has been defined as the least fixed point of $\Gamma'^{\Lambda}_{\alpha}$, whilst lgd_{α} is defined as the greatest fixed point.

Conjecture 5.27. *$ld_{\alpha} = lgd_{\alpha}$, that is, $\Gamma'^{\Lambda}_{\alpha}$ has a unique fixed point.*

The consequences of Conjecture 5.27 would entail that ld_{α}, thus the best known estimate of δ, is decidable in polynomial time. This would match the knowledge for the standard bisimilarity distance, which, defined as a unique fixed point, is computable in polynomial time.

In all of the examples tested in the next section, and attempts to specifically construct a counter example, the least fixed point has been equal to the greatest fixed point. Due to the more complex nature of the coupling capturing the distance, the proof used in the case of standard bisimilarity distances does not translate, and a direct proof of such remains elusive.

5.6 Examples

This section considers the effectiveness of bd_{α} and ld_{α} indicating examples which lead to successful verifications and examples which demonstrate their respective limitations.

Techniques for computation

Before considering these examples note that the practicality of the algorithms for complexity analysis is limited. Hence each of the examples was computed using alternative techniques.

Iterative methods The easiest method for computing bisimilarity distances is to use the properties of the fixed point. Starting at the bottom element of the lattice ($d(s,s')=0$ for all s,s'), iteratively compute $F(F(\dots F(d)))$ to convergence. There is no guarantee of fast convergence, but in practice the method provides an approximation quickly. In the case that the fixed point is unique, the same can by applied from the top element of the lattice ($d(s,s')=1$ for all s,s'), and close convergence can be detected by ensuring the difference between the two is small. Such a unique fixed point is not shown to exist, however it is this method that leads to the conjecture that it does exist for ld_α, i.e. that $ld_\alpha = lgd_\alpha$.

SMT based methods The formula given in Figure 5.3 can be encoded directly into an SMT solver, such as Z3. Further Z3 allows the specification of an objective function, so that ld_α (and bd_α using the respective formula from Chapter 2) can be found as the satisfying solution minimising $\sum_{s,s'} ld_\alpha(s,s')$.

However in practice the non-linearities cause Z3 some issues, but surprising success can be found on small examples by encoding in Z3 the formula akin to Figure 4.2. However, it is not required to specify the *least* fixed point explicitly as this can be expressed with an objective function. This appears to work well, despite this employing universal quantification (and thus being a level higher in the polynomial hierarchy), the fact that it is in *linear* real arithmetic appears to help. The encoding, whilst polynomial, is in practice quite large and so the method fails to terminate in reasonable time once the number of states exceeds 10 to 12.

Examples

This section demonstrates the methods are a sound technique for determining the δ privacy parameter satisfying (ε,δ)-differential privacy, given e^ε.

Example 5.28 (PIN Checker). Take as an example, in Figure 5.4, a PIN checking system studied by Xu et al. [XCL14; Xu15]. Intuitively, the machine accepts or rejects a code (a or b). Instead of accepting a code deterministically, it probabilistically decides whether to accept. The machine allows an attempt with the other code if it is not accepted. The system is modelled to accepts more often on the the pin-code a, from state 0, and more often the code b, from state 1. The chain simulates attempts to gain access to the system by trying code a then b until the system accepts (reaching the 'end' state). Pen-and-paper analysis can determine that the system is $(\ln(\frac{2809}{2209}),0)$-differentially private, or at the other extreme $(0,\frac{200}{2503})$-differentially private ($\frac{2809}{2209}\approx 1.27$, $\frac{200}{2503}\approx 0.0799$).

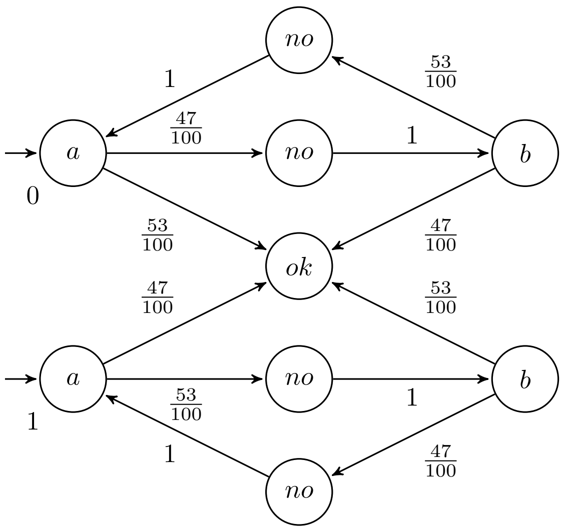


Figure 5.4: Labelled Markov chain for PIN checker example: each state denotes its label, transition probabilities on arrows.

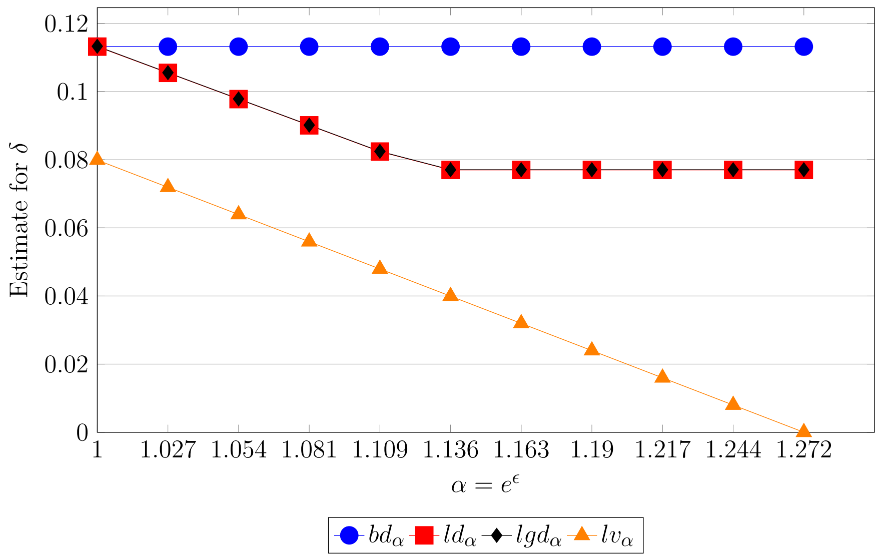
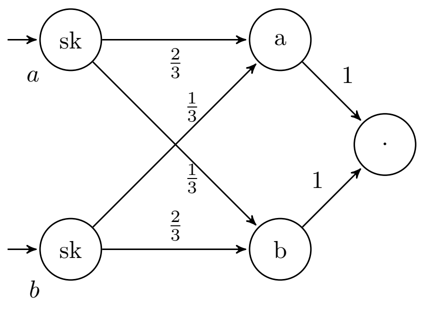
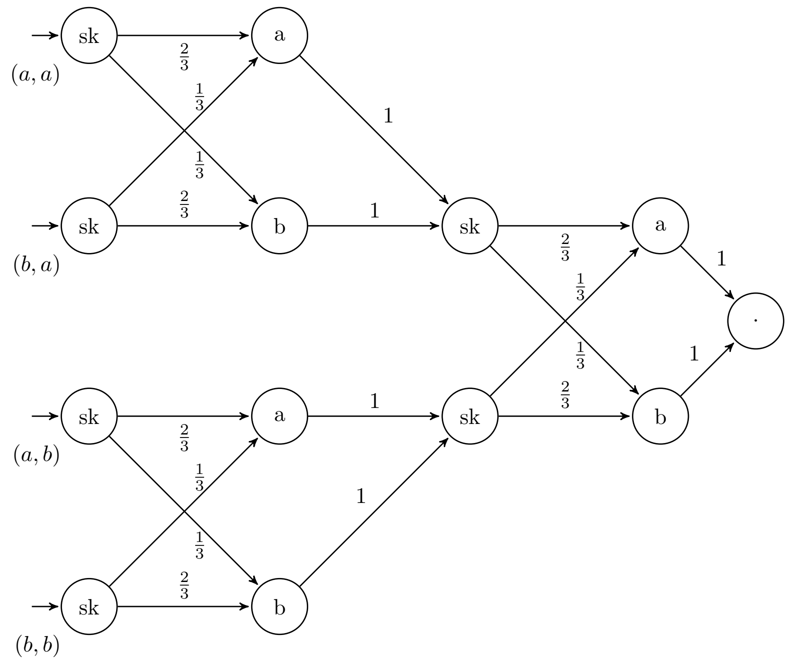


Figure 5.5: Calculated approximations of δ given ε for Figure 5.4.

In Figure 5.5, the true privacy, lv_α is shown along the orange line (▲). In the blue line (●) shows the estimate bd_α as defined in Chapter 4; which correctly bounds the true privacy, but is unresponsive to α. Using the methods introduced in this chapter ld_α is computed on the red line (■) and lgd_α on the black line (♦), which coincide. Observe that this is an improvement and is within approximately 1.5 times the true privacy for $\alpha \leq 1.035$. In this example observe that $ld_\alpha = lgd_\alpha$; suggesting lgd_α, which can be computed in polynomial time is as good as ld_α. The results do eventually suffer, as increasing α cannot find a better δ, despite a lower value existing. ◀



(a) Single-input, single-output.



(b) Two-input, two-output.

Figure 5.6: Randomised response. Every second label is the outcome of the randomised response mechanism and alternately `sk` (for 'skip'). The left most state represents the sensitive input.

Example 5.29 (Randomised Response). The randomised response mechanism allows a data subject to reveal a secret answer to a potentially humiliating or sensitive question honestly with some degree of plausible deniability. This is achieved by flipping a biased coin and providing the wrong answer with some probability based on the coin toss. If there are two answers a or b, answering truthfully with probability $\frac{\beta}{1+\beta}$ and otherwise with $\frac{1}{1+\beta}$ leads to ε-differential privacy where $e^{\varepsilon} = \beta$ and such a bound is tight (there is no smaller ε' such that answering in this way gives ε'-differential privacy). However, it can be (ε', δ)-differentially private for $\varepsilon' < \varepsilon$ and some δ.

Consider the single-input, single-output randomised response mechanism shown in Figure 5.6a with $\beta = 2$, hence $\ln(2)$-differentially private, alternatively it is $(\ln(\frac{6}{5}), \frac{4}{15})$-differential privacy ($\ln(\frac{6}{5}) \approx \frac{\ln(2)}{4}$). Consider the application of composing automata to determine more complex properties automatically.

Differential privacy enjoys multiple composition theorems [DR14]. When applied to disjoint datasets, differential privacy allows the results of (ε, δ)-differentially private mechanism applied to each independently to be combined with no additional loss in privacy. Consider the two-input, two-output labelled Markov chain (Figure 5.6b), by considering each input to be from two independent respondents, using these methods verifies that the privacy does not increase on the partitioned data. Hence, consider the adjacency relation as the symmetric closure of $R = \{((a,a),(a,b)), ((a,a),(b,a)), ((b,b),(a,b)), ((b,b),(b,a))\}$. Computing $\max_{(s,s')\in R} ld_{6/5}(s,s') = \frac{4}{15}$, determines $(\ln(\frac{6}{5}), \frac{4}{15})$-differential privacy, verifying there is no privacy loss from composition. Because randomised response is finite, it is possible to compute lv_α for adjacent inputs in exponential time for comparison. In this instance, the technique provides the optimal solution, in the sense $\max_{(s,s')\in R} ld_{6/5}(s,s') = \max_{(s,s')\in R} lv_{6/5}(s,s')$; indicating that ld_α and lgd_α can provide a good approximation.

The basic composition theorems suggest that if a mechanism that is (ε, δ)-differentially private is used k times, one achieves $(k\varepsilon, k\delta)$-differential privacy [Dwo+06]. However, this is not necessarily optimal. More advanced composition theorems may enable tighter analysis, although this can be computationally difficult (#**P**-complete) [MV16]. Even this may not be exact when allowed to look inside the composed mechanisms. Instead, if it is assumed that the responses are from two questions answered by the same respondent and let $R' = R \cup \{((a,a),(b,b))\}$, naïvely applying basic composition concludes $(\ln(\frac{36}{25}), \frac{8}{15})$-differential privacy. The methods from this chapter can find a better bound than basic composition since $\max_{(s,s')\in R'} ld_{36/25}(s,s') = \frac{103}{225} < \frac{8}{15}$. However, in this case, this technique is not optimal either. ◀

5.7 Future work: extended models

The work presented in Chapters 4 and 5 is limited to labelled Markov chains, or 'fully probabilistic' automata. Characterising programs with labelled Markov chains, captures the semantics of programs with finite memories, because each configuration of the program needs to be explicitly described in a state. A natural direction is to extend to programs without this restriction.

The standard bisimulation distances can also be defined on non-deterministic systems, where their computational complexity is **PPAD** [BW14]. In this thesis so far, the privacy can only be analysed between two start states, but it is also reasonable to allow an input in the form of a trace or sequence of actions, the output would also be a trace, in the style of Tschantz et al. [TKD11]. Here the choice of labels (at a specific state) would correspond to decisions taken by the user/environment/input, and the output can be announced either via a labelling of the state, or by the action being chosen fully-probabilistically (i.e. there being no choice of action available). This setting would support a broader range of scenarios that could be modelled and verified as differentially private.

One such direction is to extend the model to some variant of *probabilistic automata*, *Markov decision processes* or *Input-Output automata*. Each of these have their own technical distinctions, but the common theme is that the operation of the machine is not fully probabilistic, instead some input from the environment is required. This is usually by an *action* or character from a distinct *input alphabet*. In such models, one induces a labelled Markov chain by a choice of how the environment will behave, typically called a scheduler. The exact semantics of the scheduler require careful choice, particularly with relation to differential privacy so that the choice made by the scheduler does not reveal information which is supposed to be private [CP10]. A common model of scheduler could be positional, so that when in some particular state, some particular action will always be chosen. However, such a positional schedule could be used to distinguish the states and so is not an appropriate model for the scheduler in the context of privacy.

Such operations could be used to describe *streaming* or *online algorithms*; for which the input is revealed during the computation instead of being known at the start of the computation (as assumed in the work of this thesis). Such a model could then be used to describe differential privacy over input streams which differ in exactly one position, in a similar fashion to Tschantz et al. [TKD11]. To ensure the available actions does not describe the possible positions, one should ensure every action is available at every choice.

This model could be particularly susceptible to automated analysis by the following construction. Assume the input alphabet is two characters a and b (the construction generalises to further characters). Let $\mathcal{M}$ be a non-deterministic model which takes actions and produces outputs (for example by labelled states or labelled transitions). Take three copies of this machine. The first machine, at every place there is a choice between a and b, there will be a further choice ?, from which the transition will move to the third copy of the machine, mimicking the operations as if an a was chosen. In the second machine, at every place there is a choice between a and b, there will be a further choice ?, from which the transition will move to the third copy of the machine, mimicking the operations as if an b was chosen. The third copy only has the choice of a or b.

For example the operation of the machine from the first copy on $aaabbaba?bababa$ would operate as $aaabbaba\mathbf{a}bababa$, where as the second machine would behave as $aaabbaba\mathbf{b}bababa$. The input sequence could have up to one position with ? and comparing the starting from the either of the two machines compares the difference on input sequences with one difference. It is then necessary to compare the maximal difference between the probabilities over inputs of the form $\{a, b\}^*?\{a, b\}^*$, but since these are the only accepted words, it can be quantified over $\{a, b, ?\}^*$; that is, find

$$\sup_{d\in\{a,b,?\}^*} \sup_{E\in\Sigma^*} \Delta_\alpha(\mathbb{P}[\mathcal{M}_1(d) \in E], \mathbb{P}[\mathcal{M}_2(d) \in E]).$$

Bisimilarity distances can be used on such models, by extension with Hausdorff distance. Assume the transition function $T : Q \times A \to \mathit{Dist}(Q)$, where A is a set of actions. Then the bisimilarity distance characterised by

$$\Gamma^{\Delta}_{\alpha}(d)(s, s') = \begin{cases} K^{\Delta}_{\alpha}(d)(\mu_s, \mu_{s'}) & \text{if } \ell(s) = \ell(s') \\ 1 & \text{otherwise} \end{cases}$$

can be generalised to

$$\Gamma^{\Delta}_{\alpha}(d)(s, s') = \begin{cases} \max\limits_{a\in A} \left\{ \begin{matrix} \sup_{\mu\in T(s,a)} \inf_{\mu'\in T(s',a)} K^{\Delta}_{\alpha}(d)(\mu,\mu'), \\ \sup_{\mu'\in T(s',a)} \inf_{\mu\in T(s,a)} K^{\Delta}_{\alpha}(d)(\mu,\mu') \end{matrix} \right\} & \text{if } \ell(s) = \ell(s') \\ 1 & \text{otherwise} \end{cases}$$

A further direction of study is to determine the correct notion of scheduler for a suitable choice of formal model and verify that such a bisimilarity distance would bound the privacy found in such a model. Further one should understand the computability of such a distance; note that in the classical case of bisimilarity distance (where the absolute value function is used, not Δ_α) then the distance can be computed in **PPAD**; but as with labelled Markov chains, there is no guarantee this result will transfer when complicated by Δ_α or Λ_α.

5.8 Conclusion

The results for δ are summarised in Figure 5.1 on page 67. The privacy parameter δ is captured by the value of lv_α, but it is not computable and difficult to approximate. Hence an upper bound ld_α is defined and shown to be more accurate than the previously bound bd_α from Chapter 4 and just as easy to compute (in polynomial time with an **NP** oracle). Further lgd_α is defined, a distance based on the greatest fixed point, which has the same flavour but can be computed in polynomial time.

When considering lv_α directly, the distance can be approximated to arbitrary precision in **PSPACE** and is #**P**-hard (which generalises a known result on tv). It is left open whether the least fixed point bisimilarity distance (or any refinement smaller than lgd_α) can be computed in polynomial time, or even if $lgd_\alpha = ld_\alpha$. It is also open whether approximation can be resolved to be in #**P**, **PSPACE**-hard, or complete for some intermediate class.

This thesis does not address the extension of the models further (as sketched in Section 5.7), and leaves this open as a direction for future work. The next chapter considers the problem of estimating the ε parameter in the context of pure differential privacy.

Chapter 6

Distances for ε

In the spirit of Chapters 4 and 5, the parameter ε can be captured by the following variant of the total variation distance defined in [Xu15; Smi08]:

$$tv_{\ln}(s, s') = \sup_{E \in \Sigma^*} d_{\ln}\left(\nu_s(E), \nu_{s'}(E)\right),$$

where $d_{\ln}(x, x') = |\ln(x) - \ln(x')|$.

Then a labelled Markov chain is ε-differentially private with respect to a relation $R \subseteq Q \times Q$ if and only if

$$\max_{(s,s') \in R} tv_{\ln}(s, s') \leq \varepsilon.$$

This chapter studies this distance through a problem called the big-O problem, which has analogy to the big-O notation in the analysis of algorithms; a state s will be big-O of s' if the probability of every finite word from s is not greater than some constant multiple of the probability from the second (see Definition 6.1). There is a direct connection between this concept and the distance $tv_{\ln}$, or rather the exponential of this distance, which will be called $tv_{\otimes}$ and studied throughout this chapter; the optimal constant of big-O will correspond directly to distances of this form. However, in full generality, the decision questions of interest on this distance (e.g. whether the distance is bounded or below a given threshold) will turn out to be undecidable and the distances inapproximable (Theorem 6.12), so further restrictions will be considered. First the big-O problem will be shown to be **coNP**-complete on unary automata (Theorem 6.26). One may expect the concatenation of multiple unary automata (with different characters, forming a letter-bounded automaton) would be a simple generalisation of the methods. However, evidence is provided that this is not the case, and the problem will only be shown decidable subject to a conjecture in number theory, leaving the decidability question partly open (Theorem 6.59).

Finally the bisimilarity distance of Chatzikokolakis et al. [CGPX14] will be considered which bounds $tv_{\ln}(s, s')$, and for a slightly modified version, that is, by studying the distance which bounds its exponent $tv_{\otimes}$, the threshold problem for the bisimilarity distance will be shown to be in **PSPACE** (Theorem 6.74).

In contrast to the previous chapters the main problem here (the big-O problem) can be defined on weighted automata, a generalisation of labelled Markov chain, and the positive results will also apply in this setting, although this chapter will not consider infinite word models.

6.1 The big-O problem

The problem is presented using three equivalent formulations. The first is an analogue to the big-O notation typically used when classifying the complexity of algorithms; and applies in full generality to weighted automata (and thus to labelled Markov chains). The following formulations of the problem, based on variants on the total variation distance, are defined only on labelled Markov chains.

Definition 6.1. Given a weighted automaton $\mathcal{W} = \langle Q, \Sigma, M, F \rangle$ and $s, s' \in Q$, we say that s *is big-O of* s' if there exists $C > 0$ such that for all $w \in \Sigma^*$: $\nu_s(w) \le C \cdot \nu_{s'}(w)$. ◀

This can be used to compare two weighted automata (or Markov chains) by notionally combining them into a single system and comparing their respective start states.

Definition 6.2 (Big-O Problem).

INPUT	Weighted automaton $\langle Q, \Sigma, M, F \rangle$ and $s, s' \in Q$
OUTPUT	Is s big-O of s'?

◀

The second formulation is an analogue to the total variation distance (recall by Definition 3.9 $tv(s, s') = \sup_{E \in \Sigma^*} \nu_s(E) - \nu_{s'}(E))$, which takes the supremum of a distance between two events. Here the distance is a ratio between the two events. The distance is presented in its asymmetric and symmetric variants.

Definition 6.3. The asymmetric ratio variation function is

$$tv_{\oslash}(s, s') = \sup_{E \subseteq \Sigma^*} \frac{\nu_s(E)}{\nu_{s'}(E)}. \quad ◀$$

In this chapter, consider $\frac{0}{0} = 0$, and for $x > 0$ consider $\frac{x}{0} = \infty$.

Definition 6.4. The symmetric ratio variation function is:

$$tv_{\otimes}(s, s') = \max\left\{ tv_{\oslash}(s, s'), tv_{\oslash}(s', s) \right\}. \quad ◀$$

Remark. The distance is equivalent to the exponential of the "multiplicative variant of the total variation distance" $tv_{\ln}$. ◀

Definition 6.5 (TV-BOUNDED).

INPUT	LMC $\langle Q, \Sigma, M, F\rangle$ and $s, s' \in Q$
OUTPUT	$tv_{\oslash}(s, s') < \infty$? ◀

The problem is set out above in the asymmetric case; the symmetric problem asks if $tv_{\otimes}(s, s') < \infty$. Clearly the symmetric problem can be solved by reduction to two instances of the asymmetric instances, but will show hardness for variants.

Proposition 6.6. *For $tv_{\oslash}$, and $tv_{\otimes}$, on word terminating labelled Markov chains, is sufficient to consider the supremum over $w \in \Sigma^*$, rather than $E \subseteq \Sigma^*$.*

Proof. The strategy of the proof is to approximate any event by a finite subset, then observe that an event with more than one word can be simplified, and the ratio will not decrease.

Claim 6.7. *For $a, b, c, d > 0$ we have $\max(\frac{a}{c}, \frac{b}{d}) \geq \frac{a+b}{c+d}$.*

Proof. Suppose $\frac{a+b}{c+d} > \frac{a}{c}$ and $\frac{a+b}{c+d} > \frac{b}{d}$.

- By the first we have $ac + bc > ac + dc = bc > ad \implies \frac{b}{d} > \frac{a}{c}$.
- By the second we have $ad + bd > bc + bd = ad > bc \implies \frac{a}{c} > \frac{b}{d}$.

Contradiction. ■

Hence, given $f, g : E \to \mathbb{R}_{\geq 0}$ and a *finite* set E for the purposes of maximisation, such a set can always be simplified by repeated application of this concept. That is, there exists $e' \in E$ such that,

$$\frac{\sum_{e\in E} f(e)}{\sum_{e\in E} g(e)} \leq \frac{f(e')}{g(e')}. \tag{6.1}$$

Consider an event $E \subseteq \Sigma^*$, then for every $\lambda > 0$ there is a k such that $\nu_s(E \cap \Sigma^{>k}) \leq \lambda$. Then $\nu_s(E \cap \Sigma^{\leq k}) \leq \nu_s(E) \leq \nu_s(E \cap \Sigma^{\leq k}) + \lambda$ [Kie18, Lemma 12]. For any ε, by choice of sufficiently small λ there is a finite set E' such that $\frac{\nu_s(E')}{\nu_{s'}(E')} - \varepsilon \leq \frac{\nu_s(E)}{\nu_{s'}(E)} \leq \frac{\nu_s(E')}{\nu_{s'}(E')} + \varepsilon$.

Consider $\sup_{E\subseteq\Sigma^*} \frac{\nu_s(E)}{\nu_{s'}(E)}$, this is equivalent to $\lim_{k\to\infty} \sup_{E\subseteq\Sigma^{\leq k}} \frac{\nu_s(E)}{\nu_{s'}(E)}$ and by Equation (6.1) this is equivalent to $\lim_{k\to\infty} \sup_{w\in\Sigma^{\leq k}} \frac{\nu_s(w)}{\nu_{s'}(w)} = \sup_{w\in\Sigma^*} \frac{\nu_s(w)}{\nu_{s'}(w)}$. □

The third analogue of the problem is using the skewed variant of the total variation distance from Chapter 5, by asking if there exists an α such that $lv_\alpha(s, s') = 0$. This chapter will also address the question of deciding whether, for a given α, $lv_\alpha(s, s') = 0$; this done by equivalently asking if $tv_\oslash(s, s') \leq \alpha$.

Proposition 6.8. *On word terminating labelled Markov chains, the following conditions are equivalent:*

- *s is big-O of s'*
- $\textsc{Tv-Bounded}(s, s')$
- $\exists \alpha : lv_\alpha(s, s') = 0$.

Further the minimal constant of the big-O problem is exactly $tv_\oslash(s, s')$ and the minimal α.

6.1.1 The relation to differential privacy

Recall that, in the case of pure differential privacy, and given an LMC $\mathcal{M}$ and a symmetric relation $R \subseteq Q \times Q$ then $\mathcal{M}$ is ε-differentially private (wrt R) if, for any $s, s' \in Q$ such that $(s, s') \in R$, we have

$$\nu_s(E) \leq e^\varepsilon \cdot \nu_{s'}(E)$$

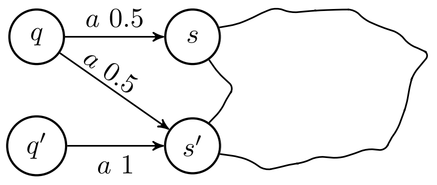
for any observable set of traces $E \subseteq \Sigma^*$.

Then note there is such an ε if and only if s is big-O of s' for every $(s, s') \in R$. Further the minimal such ε is such that $e^\varepsilon = \max_{(s,s') \in R} tv_\oslash(s, s')$.

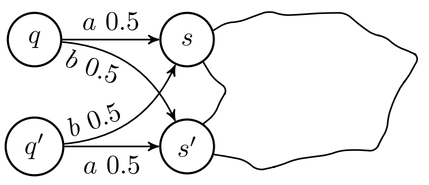
6.1.2 The big-Θ problem

One could consider whether s is big-Θ of s', defined as s is big-O of s' and s' is big-O of s; or equivalently, whether $tv_\otimes(s, s') < \infty$ for labelled Markov chains. However, Lemma 6.9 notes that these two notions reduce to each other, justifying the consideration of only the big-O problem. There is an obvious Cook reduction from big-Θ to big-O (ask if s is big-O of s' and s' is big-O of s), but this is strengthened to a Karp reduction (that is a single call to the big-O problem, preserving the answer), although this does require at least two characters.

Lemma 6.9. *The big-O problem is inter-reducible with the big-Θ problem.*



(a) Reduction to big-Θ.



(b) Reduction to big-O.

Figure 6.1: Reductions between big-O and big-Θ.

Proof of Lemma 6.9.

Case 1 (big-O problem reduces to the big-Θ problem)**.** To ask if s is big-O of s' use the construction of Figure 6.1a then ask if q is big-Θ of q'. Then note q' is big-O of q in all cases, so q is big-O of q' if and only if q is big-Θ of q':

$$\frac{\nu_q(aw)}{\nu_{q'}(aw)} = \frac{0.5\nu_s(w) + 0.5\nu_{s'}(w)}{\nu_{s'}(w)} < C \iff \frac{\nu_s(w)}{\nu_{s'}(w)} < 2C - 1$$

$$\frac{\nu_{q'}(aw)}{\nu_q(aw)} = \frac{\nu_{s'}(w)}{0.5\nu_s(w) + 0.5\nu_{s'}(w)} \leq 2$$

Case 2 (big-Θ problem reduces to the big-O problem)**.** To ask if s is big-Θ of s' use the construction of Figure 6.1b then ask if q is big-O of q'. Then s is big-Θ of s' if and only if s is big-O of s' and s' is big-O of s if and only if q is big-O of q':

$$\frac{\nu_q(aw)}{\nu_{q'}(aw)} = \frac{0.5\nu_s(w)}{0.5\nu_{s'}(w)} < C \iff \frac{\nu_s(w)}{\nu_{s'}(w)} < C$$

$$\frac{\nu_q(bw)}{\nu_{q'}(bw)} = \frac{0.5\nu_{s'}(w)}{0.5\nu_s(w)} < C \iff \frac{\nu_{s'}(w)}{\nu_s(w)} < C$$

Each of the reductions adds a constant number of bits, as such operates in logspace. □

6.2 Big-O, threshold and approximation problems are undecidable

This section shows that the big-O problem is undecidable, by reduction from the emptiness problem for probabilistic automata. The results in this section are presented as results on ratio total variation distances on *word terminating labelled Markov chains*, and thus apply to the big-O problem and the more general weighted automata. In the case of distances, it is not only interesting to consider whether distance is bounded, but also to consider its actual value. However, the distances are inapproximable.

In the context of Markov chains, the problem whether $tv_{\otimes}(s, s')$ is 1 is equivalent to the problem of whether $lv_1(s, s')$ is 0 and in the case of the classical total variation distance whether $tv(s, s')$ is 0; that is, whether the measures are identical $\nu_s = \nu_{s'}$. This question is decidable in polynomial time [Sch61; Tze92; Kie+13].

The classical total variation distance threshold problem, whether $tv(s, s') \le \theta$, is undecidable [Kie18] and the same applies to the distance here. In the case of classical total variation distances it is not known if the strict variant is undecidable (whether $tv(s, s') < \theta$); here the reductions are agnostic and both cases are undecidable. However approximation of classical total variation distances is possible [Kie18; CK14], but this will not be the case for the distances $tv_{\oslash}$ and $tv_{\otimes}$.

Definition 6.10. The asymmetric threshold problem takes s, s', θ and asks if $tv_{\oslash}(s, s') \le \theta$. The variant under the promise of boundedness; promises that $tv_{\oslash}(s, s')$ is not infinity. The strict variant of each problem replaces $\le$ with $<$.

The asymmetric additive approximation problem takes s, s', γ and asks for x such that $|tv_{\oslash}(s, s') - x| \le \gamma$. The asymmetric multiplicative approximation problem takes s, s', γ and asks for x such that $1 - \gamma \le \frac{x}{tv_{\oslash}(s,s')} \le 1 + \gamma$.

The symmetric variant of each problem replaces $tv_{\oslash}$ with $tv_{\otimes}$. ◀

Undecidability of each of these problems is established by the is emptiness problem for probabilistic automata, which is undecidable [Paz14; Fij17].

Definition 6.11 (Empty).

INPUT	A probabilistic automaton $\mathcal{A}$
OUTPUT	is $\mathbb{P}_{\mathcal{A}}(w) \le \frac{1}{2}$ for all words w? ◀

The problem equivalently asks if the language $\{w \in \Sigma^* \mid \mathbb{P}_{\mathcal{A}}(w) > \frac{1}{2}\}$ is empty.

The results are summarised in the following theorem:

Theorem 6.12.

- *The problem* Tv-Bounded *(and the big-O problem) is undecidable.*
- *The symmetric and asymmetric strict and non-strict threshold problems are undecidable, and undecidable even under the promise of boundedness.*
- *Symmetric and asymmetric approximation problems are undecidable under the promise of boundedness.*

The following auxiliary lemma is used to show the result.

Lemma 6.13.

1. *Given a constant c such that it is sure that $tv_{\oslash}(s,s') \le c$ or $tv_{\oslash}(s,s') = \infty$, it is undecidable to distinguish between $tv_{\oslash}(s,s') \le c$ or $tv_{\oslash}(s,s') = \infty$.*

2. *Given two numbers c and C, such that $c < C$ and that it is sure that $tv_{\oslash}(s,s') \le c$ or $C \le tv_{\oslash}(s,s') < \infty$, it is undecidable to distinguish between $tv_{\oslash}(s,s') \le c$ or $C \le tv_{\oslash}(s,s') < \infty$.*

Observe that Theorem 6.12 is implied by the lemma. TV-BOUNDED, or the threshold problem could be used to distinguish between $tv_{\oslash}(s,s') \le c$ or $tv_{\oslash}(s,s') = \infty$. The question $tv_{\oslash}(s,s') \le \frac{c+C}{2}$ or $tv_{\oslash}(s,s') < \frac{c+C}{2}$, could distinguish between $tv_{\oslash}(s,s') \le c$ or $C \le tv_{\oslash}(s,s') < \infty$, thus are both undecidable. Additionally it is not possible to approximate, since finding x such that $|tv_{\oslash}(s,s') - x| \le \frac{C-c}{2}$ tells us which side of the gap by comparing x and $\frac{c+C}{2}$. Similarly finding x such that $1 - \frac{C-c}{2C} \le \frac{x}{tv_{\oslash}(s,s')} \le 1 + \frac{C-c}{2C}$. All of the results hold when $tv_{\oslash}$ is replaced by $tv_{\otimes}$.

Proof of Lemma 6.13. The proof will reduce EMPTY to each of the two promise problems. The construction will result in two branches of a Markov chain, the first will simulate the probabilistic automaton, resolving each character with equal weight. In the other branch, from s', simulate every word equally, so that its weight is proportional to the weight a word would be if it were accepted by the probabilistic automaton with probability one half; hence if there is a word above $\frac{1}{2}$ the ratio between these two branches is greater than 1. The words are then repeated so that this ratio can be pumped unboundedly.

The construction is shown for $\Sigma = \{a, b\}$, but the procedure can be generalised to arbitrary alphabets. Assume a probabilistic automaton $\mathcal{A} = \langle Q, \Sigma, M, F\rangle$ with starting state s and construct the Markov chain $\langle Q', \Sigma', \delta, F'\rangle$ as follows. Let $Q' = Q \cup \{s, s', s'', s_0, t\}$ and $\Sigma' = \{a, b, acc, rej, \vdash\}$. First simulate the probabilistic automaton, with equal weight $\frac{1}{4}$ on each character $\{a, b, acc, rej\}$:

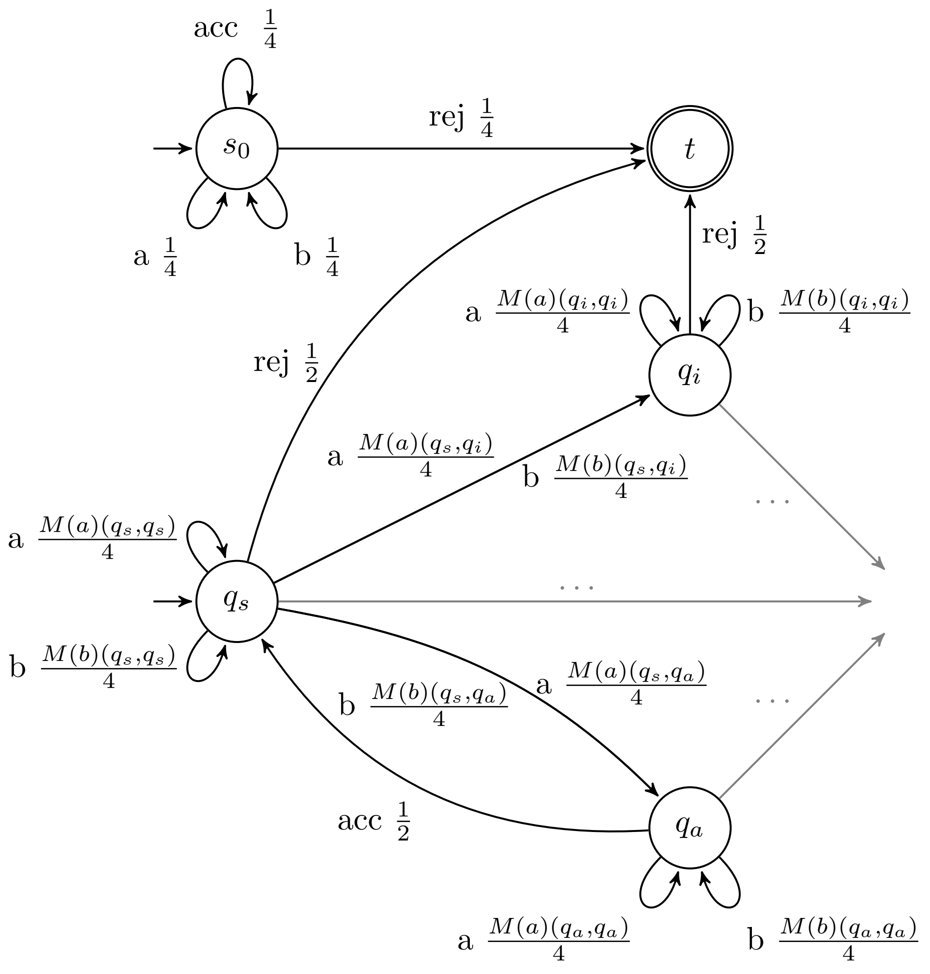
For all $q \in Q$:

$$\forall q' \in Q : q \xrightarrow[a]{\frac{1}{4}M(a)(q,q')} q' \qquad q \xrightarrow[b]{\frac{1}{4}M(b)(q,q')} q'$$

$$\text{if } q \in F : q \xrightarrow[acc]{\frac{1}{2}} q_s \quad \text{ and } \quad \text{if } q \notin F : q \xrightarrow[rej]{\frac{1}{2}} t$$

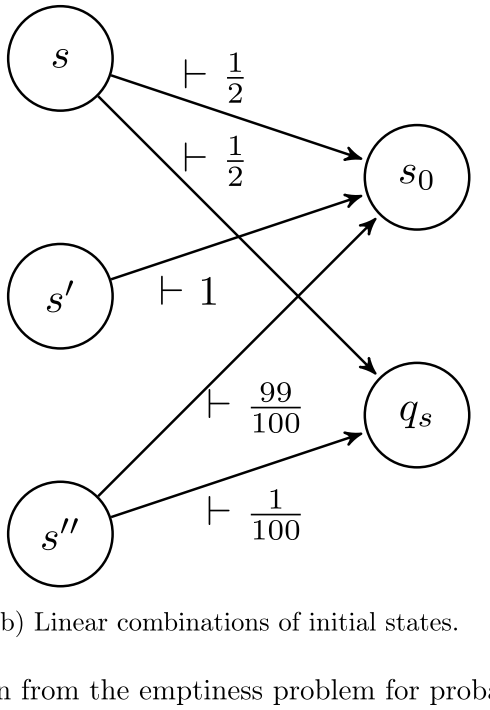
Then consider the part of the chain which behaves equally, rather than according to the probabilistic automaton:

$$s_0 \xrightarrow[a]{\frac{1}{4}} s_0 \qquad s_0 \xrightarrow[b]{\frac{1}{4}} s_0 \qquad s_0 \xrightarrow[acc]{\frac{1}{4}} s_0 \qquad s_0 \xrightarrow[rej]{\frac{1}{4}} t$$



acc $\frac{1}{4}$
s_0
rej $\frac{1}{4}$
t
a $\frac{1}{4}$
b $\frac{1}{4}$
rej $\frac{1}{2}$
a $\frac{M(a)(q_i,q_i)}{4}$
b $\frac{M(b)(q_i,q_i)}{4}$
rej $\frac{1}{2}$
q_i
a $\frac{M(a)(q_s,q_i)}{4}$
b $\frac{M(b)(q_s,q_i)}{4}$
...
a $\frac{M(a)(q_s,q_s)}{4}$
...
q_s
b $\frac{M(b)(q_s,q_s)}{4}$
a $\frac{M(a)(q_s,q_a)}{4}$
b $\frac{M(b)(q_s,q_a)}{4}$
...
acc $\frac{1}{2}$
q_a
a $\frac{M(a)(q_a,q_a)}{4}$
b $\frac{M(b)(q_a,q_a)}{4}$

(a) Main reduction: where q_a represents accepting states of the probabilistic automaton, q_i represents rejecting states and q_s represents the start state (assumed to be rejecting).



(b) Linear combinations of initial states.

Figure 6.2: Reduction from the emptiness problem for probabilistic automata to an LMC for the big-O problem.

The reduction can be seen in Figure 6.2a; the following states are used to argue on the distances, depicted in Figure 6.2b:

$$s \xrightarrow[\vdash]{\frac{1}{2}} s_0 \qquad s \xrightarrow[\vdash]{\frac{1}{2}} q_s \qquad s' \xrightarrow[\vdash]{1} s_0 \qquad s'' \xrightarrow[\vdash]{\frac{99}{100}} s_0 \qquad s'' \xrightarrow[\vdash]{\frac{1}{100}} q_s$$

The following claims on the reduction complete the lemma.

Claim 6.14 (Bounded vs Unbounded Problem).

- $\mathcal{A} \notin \textsc{Empty} \implies tv_{\oslash}(s,s') = \infty$ *and* $tv_{\otimes}(s,s') = \infty$
- $\mathcal{A} \in \textsc{Empty} \implies tv_{\oslash}(s,s') \leq 2$ *and* $tv_{\otimes}(s,s') \leq 2.$

Claim 6.15 (Gap Problem).

- $\mathcal{A} \notin \textsc{Empty} \implies 49 < tv_{\oslash}(s,s'') \leq 51$ *and* $49 < tv_{\otimes}(s,s'') \leq 51$
- $\mathcal{A} \in \textsc{Empty} \implies tv_{\oslash}(s,s'') \leq 2$ *and* $tv_{\otimes}(s,s'') \leq 2.$

Proof of Claim 6.14. First observe that

$$\frac{\nu_s(\vdash w')}{\nu_{s'}(\vdash w')} = \frac{\frac{1}{2}\nu_{s_0}(w') + \frac{1}{2}\nu_{q_s}(w')}{\nu_{s_0}(w')} = \frac{1}{2} + \frac{1}{2}\frac{\nu_{q_s}(w')}{\nu_{s_0}(w')}. \tag{6.2}$$

If there is a word w that is accepted by the automaton with probability $> \frac{1}{2}$, then let $w' = (w\ acc)^i\ rej$ and we have

$$\frac{\nu_{q_s}(w')}{\nu_{s_0}(w')} = \frac{((\frac{1}{4})^{|w|}\mathbb{P}_{\mathcal{A}}(w)\frac{1}{2})^i}{((\frac{1}{4})^{|w|}\frac{1}{4})^i} = (2\mathbb{P}_{\mathcal{A}}(w))^i. \tag{6.3}$$

Since $\mathbb{P}_{\mathcal{A}}(w) > \frac{1}{2}$ then $2\mathbb{P}_{\mathcal{A}}(w) > 1$ and we have:

$$\lim_{i\to\infty} \frac{\nu_s(\vdash (w\ acc)^i\ rej)}{\nu_{s'}(\vdash (w\ acc)^i\ rej)} = \infty \qquad \text{and} \qquad tv_{\oslash}(s,s') = tv_{\otimes}(s,s') = \infty.$$

If there is no such word ($\forall w \in \Sigma^* : \mathbb{P}_{\mathcal{A}}(w) \leq \frac{1}{2}$) then probability ratio of all words is bounded. All words start with $\vdash$ and are terminated by rej, so in general all words take the form $w =\vdash (w_1\ acc)\dots(w_n\ acc)(w_{n+1}\ rej)$. Consider the probability of $w' = (w_1\ acc)\dots(w_n\ acc)(w_{n+1}\ rej)$ from s_0 and q_s:

$$\begin{aligned}
&\frac{\nu_{q_s}(w')}{\nu_{s_0}(w')} && (6.4)\\
&= \frac{(\prod_{i=1}^n \frac{1}{2}(\frac{1}{4})^{|w_i|}\mathbb{P}_{\mathcal{A}}(w_i))((\frac{1}{4})^{|w_{n+1}|}(1-\mathbb{P}_{\mathcal{A}}(w_{n+1}))\frac{1}{2})}{(\frac{1}{4})^{|w_1|+\dots+|w_n|}(\frac{1}{4})^n(\frac{1}{4})^{|w_{n+1}|}\frac{1}{4}} && (6.5)\\
&\leq \frac{((\frac{1}{4})^{|w_1|+\dots+|w_n|}(\frac{1}{2})^n(\frac{1}{2})^n)((\frac{1}{4})^{|w_{n+1}|}\frac{1}{2})}{(\frac{1}{4})^{|w_1|+\dots+|w_n|+n}(\frac{1}{4})^{|w_{n+1}|}\frac{1}{4}} && (\forall i : \mathbb{P}_{\mathcal{A}}(w_i) \leq \tfrac{1}{2})\\
&= 2. && (6.6)
\end{aligned}$$

Then using Equation (6.2) we have for every word w: $\frac{1}{2} \leq \frac{\nu_s(w)}{\nu_{s'}(w)} \leq \frac{3}{2}$ and $tv_{\oslash}(s, s') \leq \frac{3}{2}$ and $tv_{\otimes}(s, s') \leq 2$. ∎

Proof of Claim 6.15. First observe that the direction of $\frac{\nu_{s''}(\vdash w)}{\nu_s(\vdash w)}$ is always ≤ 2:

$$\begin{aligned}
\frac{\nu_{s''}(\vdash w)}{\nu_s(\vdash w)} &= \frac{\frac{99}{100}\nu_{s_0}(w) + \frac{1}{100}\nu_{q_s}(w)}{\frac{1}{2}\nu_{s_0}(w) + \frac{1}{2}\nu_{q_s}(w)} \\
&= \frac{\frac{99}{100}\nu_{s_0}(w)}{\frac{1}{2}\nu_{s_0}(w) + \frac{1}{2}\nu_{q_s}(w)} + \frac{\frac{1}{100}\nu_{q_s}(w)}{\frac{1}{2}\nu_{s_0}(w) + \frac{1}{2}\nu_{q_s}(w)} \\
&\leq \frac{\frac{99}{100}\nu_{s_0}(w)}{\frac{1}{2}\nu_{s_0}(w)} + \frac{\frac{1}{100}\nu_{q_s}(w)}{\frac{1}{2}\nu_{q_s}(w)} \\
&= \frac{2 \cdot 99}{100} + \frac{2}{100} = 2
\end{aligned}$$

It will turn out the only interesting direction is $\frac{\nu_s(\vdash w)}{\nu_{s''}(\vdash w)}$, considered next:

Observe that for all words $\vdash w$, $tv_{\oslash}$ and $tv_{\otimes}$ is bounded:

$$\begin{aligned}
\frac{\nu_s(\vdash w)}{\nu_{s''}(\vdash w)} &= \frac{\frac{1}{2}\nu_{s_0}(w) + \frac{1}{2}\nu_{q_s}(w)}{\frac{99}{100}\nu_{s_0}(w) + \frac{1}{100}\nu_{q_s}(w)} \\
&= \frac{\frac{1}{2}\nu_{s_0}(w)}{\frac{99}{100}\nu_{s_0}(w) + \frac{1}{100}\nu_{q_s}(w)} + \frac{\frac{1}{2}\nu_{q_s}(w)}{\frac{99}{100}\nu_{s_0}(w) + \frac{1}{100}\nu_{q_s}(w)} \\
&\leq \frac{\frac{1}{2}\nu_{s_0}(w)}{\frac{99}{100}\nu_{s_0}(w)} + \frac{\frac{1}{2}\nu_{q_s}(w)}{\frac{1}{100}\nu_{q_s}(w)} \\
&\leq \frac{100}{2 \cdot 99} + \frac{100}{2} \leq 51.
\end{aligned}$$

If there is a word w that is accepted by the automaton with probability $> \frac{1}{2}$, then consider the word $\vdash (w\ acc)^i\ rej)$, let $w' = (w\ acc)^i\ rej)$.

$$\begin{aligned}
\frac{\nu_s(\vdash (w\ acc)^i\ rej)}{\nu_{s''}(\vdash (w\ acc)^i\ rej)} &= \frac{\frac{1}{2}\nu_{s_0}(w') + \frac{1}{2}\nu_{q_s}(w')}{\frac{99}{100}\nu_{s_0}(w') + \frac{1}{100}\nu_{q_s}(w')} \\
&\geq \frac{\frac{1}{2}\nu_{q_s}(w')}{\frac{99}{100}\nu_{s_0}(w') + \frac{1}{100}\nu_{q_s}(w')}.
\end{aligned}$$

By Equation (6.3) of the previous proof we have $\frac{\nu_{q_s}(w')}{\nu_{s_0}(w')} \xrightarrow[i\to\infty]{} \infty$, thus $\frac{\nu_{s_0}(w')}{\nu_{q_s}(w')} \xrightarrow[i\to\infty]{} 0$. Consider

$$\frac{\frac{99}{100}\nu_{s_0}(w') + \frac{1}{100}\nu_{q_s}(w')}{\frac{1}{2}\nu_{q_s}(w')} = \frac{2}{100} + \frac{2 \cdot 99}{100}\left[\frac{\nu_{s_0}(w')}{\nu_{q_s}(w')}\right] \xrightarrow[i\to\infty]{} \frac{2}{100}.$$

Then $\frac{\frac{1}{2}\nu_{q_s}(w')}{\frac{99}{100}\nu_{s_0}(w')+\frac{1}{100}\nu_{q_s}(w')} \xrightarrow[i\to\infty]{} \frac{100}{2} = 50$. So for all ε there exists an i such that $\frac{\nu_s(\vdash(w\ acc)^i\ rej)}{\nu_{s''}(\vdash(w\ acc)^i\ rej)} \geq 50 - \varepsilon$. In particular for example $tv_{\oslash}(s, s'') \geq 49$.

If there is no such word then $\forall w \in \Sigma^* : \mathbb{P}_{\mathcal{A}}(w) \leq \frac{1}{2}$, then the total variation distance will be small. All words start with $\vdash$ and are terminated by rej, so in general all words take the form $w = \vdash ((w_1\ acc) \dots (w_n\ acc)(w_{n+1}\ rej)$. Let us consider the probability of such words from s, s''.

$$\begin{aligned}
\frac{\nu_s(w)}{\nu_{s''}(w)} = \frac{\frac{1}{2}\nu_{s_0}(w') + \frac{1}{2}\nu_{q_s}(w')}{\frac{99}{100}\nu_{s_0}(w') + \frac{1}{100}\nu_{q_s}(w')} &\leq \frac{\frac{1}{2}\nu_{s_0}(w') + \frac{1}{2}\nu_{q_s}(w')}{\frac{99}{100}\nu_{s_0}(w')} \\
&\leq \frac{100}{99} \cdot \left[\frac{1}{2} + \frac{1}{2}\frac{\nu_{q_s}(w')}{\nu_{s_0}(w')}\right] \\
&\leq \frac{100}{99} \cdot \frac{3}{2} \qquad \text{(by Equation (6.6))} \\
&\leq 2.
\end{aligned}$$

This creates a significant gap between the case where there is a word with probability greater than one half and not; in particular if $\exists w : \mathbb{P}_{\mathcal{A}}(w) > \frac{1}{2}$ then $49 < tv_{\oslash}(s, s'') \leq 51$ and $49 < tv_{\otimes}(s, s'') \leq 51$ and if not then $tv_{\oslash}(s, s'') \leq 2$ and $tv_{\otimes}(s, s'') \leq 2$. ■

The gaps established in the claims complete the proof of Lemma 6.13. □

6.3 The relation to the VALUE-1 problem

The previous section showed undecidability of the big-O problem via the emptiness problem for probabilistic automata. Another undecidable problem for probabilistic automata is the VALUE-1 problem. The VALUE-1 problem asks whether some word of a probabilistic automaton is one, or at least arbitrarily close to 1. This section shows that there is a close, but not complete, connection between the VALUE-1 problem and big-O problem by reducing in both directions between the two, the results are shown in Lemmas 6.17 and 6.18.

Definition 6.16 (VALUE-1 problem).

INPUT	A probabilistic automaton $\mathcal{A}$
OUTPUT	for all $\delta > 0$ is there a word w such that $\mathbb{P}_{\mathcal{A}}(w) > 1 - \delta$? ◀

Lemma 6.17. VALUE-1 *problem reduces to the big-O problem.*

Lemma 6.18. *The big-O problem reduces to* VALUE-1 *problem.*

Proof of Lemma 6.17: The VALUE-1 *problem reduces to the big-O problem.*

Given a probabilistic automaton $\mathcal{A} = \langle Q, \Sigma, M, F \rangle$ and a dedicated starting state $q_0 \in Q$, which accepts words with probability $\mathbb{P}_{\mathcal{A}}(w)$, first construct $\mathcal{A}'$ in which words are accepted with probability $\mathbb{P}_{\mathcal{A}'}(w) = 1 - \mathbb{P}_{\mathcal{A}}(w)$, by inverting accepting states.

The proof uses a two letter alphabet, $\Sigma = \{a, b\}$, but the procedure can be generalised to arbitrary alphabets. Construct a Markov chain $\mathcal{M}_{\mathcal{A}} = \langle Q', \Sigma', M', F'\rangle$, where $Q' = Q \cup \{s, s', s_0, rej, acc\}$, $\Sigma' = \{a, b, c\}$ and $F' = \{acc\}$. The probabilistic automaton will be simulated by $\mathcal{M}_{\mathcal{A}}$. The relation M' is described by the notation $\xrightarrow[a]{p}$:

For all $q \in Q$:

$$\forall q' \in Q : q \xrightarrow[a]{\frac{1}{3}M(a)(q,q')} q' \qquad q \xrightarrow[b]{\frac{1}{3}M(b)(q,q')} q'$$

$$\text{if } q \in F : q \xrightarrow[c]{\frac{1}{3}} acc \quad \text{ and } \quad \text{if } q \notin F : q \xrightarrow[c]{\frac{1}{3}} rej$$

$$s' \xrightarrow[c]{1} q_0 \qquad s \xrightarrow[c]{1} s_0 \qquad s_0 \xrightarrow[a]{\frac{1}{3}} s_0 \qquad s_0 \xrightarrow[b]{\frac{1}{3}} s_0 \qquad s_0 \xrightarrow[c]{\frac{1}{3}} acc$$

Note the only words with positive probability are words of the form $c\Sigma^*c \subseteq \Sigma'^*$. Then given a word $w \in \Sigma^*$, $\nu_s(cwc) = (\frac{1}{|\Sigma|+1})^{|wc|}$ and $\nu_{s'}(cwc) = (\frac{1}{|\Sigma|+1})^{|wc|}(1 - \mathbb{P}_{\mathcal{A}}(w))$.

Then if there is a sequence of words for which $\mathbb{P}_{\mathcal{A}}(w)$ tends to 1 then $\frac{\nu_s(cwc)}{\nu_{s'}(cwc)}$ is unbounded.

However, if there exists some $\gamma > 0$ so that for all $w \in \Sigma^*$ $\mathbb{P}_{\mathcal{A}}(w) \le (1 - \gamma)$ then $(1 - \mathbb{P}_{\mathcal{A}}(w)) \ge \gamma$, and so $\frac{\nu_s(cwc)}{\nu_{s'}(cwc)} \le \frac{1}{\gamma}$. $\square$

Proof of Lemma 6.18: The big-O problem reduces to the VALUE-1 *problem.*

Given a labelled Markov chain $\mathcal{M} = \langle Q, \Sigma, M, F\rangle$ and $s, s' \in Q$, construct a probabilistic automaton $\mathcal{A} = \langle Q', \Sigma', M', F'\rangle$. Each state of Q will be duplicated, once for s and once for s'; $Q_s = \{q_s \mid q \in Q\}$, $Q_{s'} = \{q_{s'} \mid q \in Q\}$. Let $Q' = Q_s \cup Q_{s'} \cup \{q_0, acc, rej, sink\}$, $\Sigma' = \Sigma \cup \{\$\}$ and $F' = \{acc\}$.

Each transition of $\mathcal{M}$ will be simulated in each of the copies according the probability in $\mathcal{M}$. For every $q, q' \in Q, a \in \Sigma$, let $M'(a)(q_s, q'_s) = M(a)(q, q')$ and $M'(a)(q_{s'}, q'_{s'}) = M(a)(q, q')$. A probabilistic automaton should be stochastic for every $a \in \Sigma$, so there is unused probability for each character, which will divert to a sink. For every $q \in Q$ and $a \in \Sigma$, let

$$M'(a)(q_s, sink) = 1 - \sum_{q' \in Q} M(a)(q, q')$$

and

$$M'(a)(q_{s'}, sink) = 1 - \sum_{q' \in Q} M(a)(q, q').$$

There will be an additional character \$.

From q_0 the machine will pick either of the two machines with equal probability; $M(\$)(q_0, s_s) = M(\$)(q_0, {s'}_{s'}) = \frac{1}{2}$. If in the accepting or rejecting state the system will stay there forever $M'(\$)(acc, acc) = 1$ and $M'(\$)(rej, rej) = 1$.

The behaviour on \$ will differ in the two copies of $\mathcal{M}$. If in an s state the system will preference the accepting state when accepting and otherwise restart. If in an s' state the system will preference the rejecting state when accepting and otherwise restart. Formally,

$$M'(\$)(q_s, acc) \text{ when } q_s \in F \text{ and } M'(\$)(q_s, q_0) \text{ when } q_s \notin F$$

and

$$M'(\$)(q_{s'}, rej) \text{ when } q_{s'} \notin F \text{ and } M'(\$)(q_{s'}, q_0) \text{ when } q_{s'} \in F.$$

When in the sink state, the system restarts on \$, $M'(\$)(sink, q_0) = 1$, or for all $a \in \Sigma$ stays there $M'(a)(sink, sink) = 1$. A partial sketch of the reduction can be seen in Figure 6.3.

The idea is that if $\nu_s(w)$ is much larger than $\nu_{s'}(w)$ then, by repeated reading of the word w, nearly all of the probability mass will eventually move to acc; otherwise a sufficiently large amount of mass will be lost to rej.

Claim 6.19. *s is not big-O of s' if and only if $\mathcal{A} \in$ VALUE-1.*

Denote by $\mathbb{P}_\mathcal{A}(w)$ the probability of a word w in the probabilistic automaton, from state q_0, i.e. $\nu_{q_0}(w)$. However, ν will be used to refer to the probability in the labelled Markov chain $\mathcal{M}$. Further the notation $\mathbb{P}[q \xrightarrow{w} q']$ is used to denote $(M'(w_1) \times \cdots \times M'(w_{|w|})_{q,q'}$, i.e. the probability of transitioning from state q to q' after reading w in $\mathcal{A}$.

Case 1 (Not big-O implies VALUE-1)**.** The proof shows that $\forall \delta \exists C, i \in \mathbb{N}, w \in \Sigma^*$ such that $\nu_s(w) > C\nu_{s'}(w)$ and $\mathbb{P}_\mathcal{A}((w\$)^i) > 1 - \delta$.

Hence given δ, choose C such that $(1 - \frac{\delta}{2})\frac{C}{C+1} > 1 - \delta$. Then by the big-O property, choose a word such that $\nu_s(w) = C'\nu_{s'}(w)$, with $C' > C$. Then $(1 - \frac{\delta}{2})\frac{C'}{C'+1} > (1 - \frac{\delta}{2})\frac{C}{C+1} > 1 - \delta$.

Given the fixed sequence $(\$w\$)^i$, this induces a (unary) Markov chain, represented by the Matrix A, representing states q_0, acc and rej in the three positions

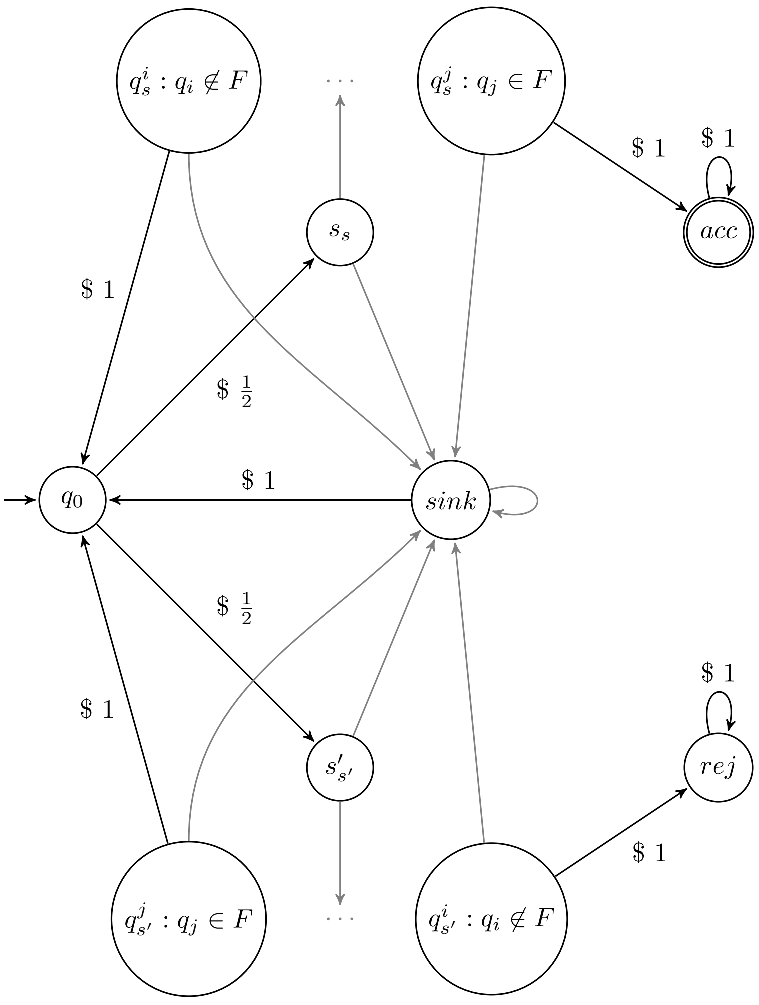


Figure 6.3: Reduction from the big-O problem to Value-1. Only the effect of transitions on the \$ symbol are shown in black, with the possibility to transition to the sink state depicted in grey (on symbols in Σ). All remaining transitions are omitted.

respectively:

$$A = \begin{bmatrix} 0.5(1-\nu_s(w)) + 0.5(1-\nu_{s'}(w)) & 0.5\nu_s(w) & 0.5\nu_{s'}(w) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then in the long run, starting from state 0, observe:

$$[\ 1 \quad 0 \quad 0\ ]A^i \xrightarrow{i\to\infty} [\ 0 \quad Cx \quad x\ ] \text{ with } Cx + x = 1.$$

Clearly, $A^i(0,1) + A^i(0,2) + A^i(0,0) = 1$, and choose i such that $A^i(0,0) \leq \frac{\delta}{2}$. Then $A^i(0,1) + A^i(0,2) \geq 1 - \frac{\delta}{2}$, using the fact that $A^i(0,1) = C'A^i(0,2)$, obtaining $A^i(0,1) + \frac{A^i(0,1)}{C'} \geq 1 - \frac{\delta}{2}$.

Hence $A^i(0,1) \geq (1-\frac{\delta}{2})\frac{C'}{C'+1} > 1-\delta$, as required.

Case 2 (big-O implies Not VALUE-1). We have there exists C such that $\forall w$ $\nu_s(w) \leq C\nu_{s'}(w)$ and should show there exists $\delta > 0$ such that $\forall w \in (\Sigma \cup \{\$\})*$ $\mathbb{P}_{\mathcal{A}}(w) \leq 1-\delta$.

To move probability from q_0 to acc it is necessary to use words of the form $\$\Sigma^*\$$ where Σ is the alphabet of $\mathcal{M}$. Hence any word can be decomposed into $\$w_1\$\$w_2\$...\$w_m\$$.

After reading w_1 the probability is such that

$$\begin{aligned}
x_1 &= \mathbb{P}[q_0 \xrightarrow{\$w_1\$} acc] = \nu_s(w_1)\\
y_1 &= \mathbb{P}[q_0 \xrightarrow{\$w_1\$} rej] = \nu_{s'}(w_1)\\
\mathbb{P}[q_0 &\xrightarrow{\$w_1\$} q_0] = 1 - x_1 - y_1.
\end{aligned}$$

Since $\exists C \forall w_i : \nu_s(w_i) \leq C\nu_{s'}(w_i)$, we have $x_1 \leq Cy_1$. By induction, repeating this process we have for all i: $x_i \leq Cy_i$.

$$\begin{aligned}
x_i &= \mathbb{P}[q_0 \xrightarrow{\$w_1\$...\$w_i\$} acc] = (1-\nu_s(w_i)-\nu_{s'}(w_i))x_{i-1} + \nu_s(w_i)\\
y_i &= \mathbb{P}[q_0 \xrightarrow{\$w_1\$...\$w_i\$} acc] = (1-\nu_s(w_i)-\nu_{s'}(w_i))y_{i-1} + \nu_{s'}(w_i)\\
\mathbb{P}[q_0 &\xrightarrow{\$w_1\$...\$w_i\$} q_0] = \prod_{j=1}^{i}(1-x_j+y_j).
\end{aligned}$$

Hence

$$\begin{aligned}
x_i &= (1-\nu_s(w_i)-\nu_{s'}(w_i))x_{i-1} + \nu_s(w_i)\\
&\leq (1-\nu_s(w_i)-\nu_{s'}(w_i))Cy_{i-1} + C\nu_{s'}(w_i)\\
&= C[(1-\nu_s(w_i)-\nu_{s'}(w_i))y_{i-1} + \nu_{s'}(w_i)]\\
&\leq Cy_i.
\end{aligned}$$

In the extreme $x_m + y_m = 1$, then $x_m \leq \frac{C}{C+1} < 1$, so the probability of reaching acc is bounded away from 1 for every word. □

The VALUE-1 problem is undecidable in general, however it is decidable in the unary case in **coNP** [CKV14a] and for *leaktight automata* [FGO12]. Note, however, that the construction combined with these decidability results *does not* entail any decidability results for the big-O problem. Firstly note that the construction adds an additional character, and such a unary instance of the big-O problem always has at least two characters when translated to the VALUE-1 problem. Further the construction does not result in a leaktight automaton, to see this the definition of leaktight automata is recalled from [FGO12]. The following, does not, of course, preclude the existence of a construction which does maintain these properties.

Definition 6.20. A finite word u is idempotent if reading once or twice the word u does not change qualitatively the transition probabilities. That is $\mathbb{P}_{\mathcal{A}}[q \xrightarrow{u} q'] > 0 \iff \mathbb{P}_{\mathcal{A}}[q \xrightarrow{uu} q'] > 0$.

Let u_n be a sequence of idempotent words. Assume that the sequence of matrices $\mathbb{P}_{\mathcal{A}}(u_n)$ converges to a limit M, that this limit is idempotent and denote M the associated Markov chain. The sequence u_n is a *leak* if there exist $r, q \in Q$ such that the following three conditions hold:

1. r and q are recurrent in M,
2. $\lim \mathbb{P}_{\mathcal{A}}[r \xrightarrow{u_n} q] = 0$,
3. for all n, $\mathbb{P}_{\mathcal{A}}[r \xrightarrow{u_n} q] > 0$.

An automaton is leaktight if there is no leak. ◀

If there were no leak in the probabilistic automaton then decidability would follow. However, this is not the case, and the reduction does not solve any cases by reduction to known decidable fragment of the VALUE-1 problem.

Claim 6.21. *The resulting automaton from the reduction of the big-O problem to the* VALUE-1 *problem has a leak.*

Proof. Consider some infinite sequence of words w_i growing in length, such that $\nu_s(w_i) > 0$ for every i. Let $u_i = \$w_i\$$.

Observe that this word is idempotent. For each starting state, consider the possible states with non-zero probability and from each of these the set of reachable states. Observe that in all cases the set reachable after one application is equal to the set reachable after two.

- $acc \xrightarrow{\$w_i\$} acc \xrightarrow{\$w_i\$} acc$
- $rej \xrightarrow{\$w_i\$} rej \xrightarrow{\$w_i\$} rej$
- $q_0 \xrightarrow{\$w_i\$} q_0, acc, rej \xrightarrow{\$w_i\$} q_0, acc, rej$
- $q_0 \xrightarrow{\$w_i\$} q_0, acc, rej \xrightarrow{\$w_i\$} q_0, acc, rej$
- For q accepting in Q_s: $q \xrightarrow{\$w_i\$} acc \xrightarrow{\$w_i\$} acc$
- For q rejecting in Q_s: $q \xrightarrow{\$w_i\$} \emptyset \xrightarrow{\$w_i\$} \emptyset$

- For q accepting in $Q_{s'}$: $q \xrightarrow{\$w_i\$} rej \xrightarrow{\$w_i\$} rej$
- For q rejecting in $Q_{s'}$: $q \xrightarrow{\$w_i\$} \emptyset \xrightarrow{\$w_i\$} \emptyset$.

Assume that the Markov chain $\mathcal{M}$ is a word terminating finite word chain, that is the decision to terminate the word must be made by probability (for example by transitioning to a final state). Then $\forall \lambda > 0$ there exists n such that $\nu_s(\Sigma^{>n}) < \lambda$ and $\nu_{s'}(\Sigma^{>n}) < \lambda$ [Kie18, Lemma 12.].

Suppose limit $\mathbb{P}_{\mathcal{A}}(u_n)$ converges to a limit M and let $r = q_0$ and $q = acc$.

Hence for longer and longer words the probability of reaching acc is diminishing. Thus $\lim \mathbb{P}_{\mathcal{A}}[r \xrightarrow{u_n} q] = 0$, and in M we have r and q in different SCCs. acc is clearly recurrent as it is deterministically looping on every character. Since the probability of reaching acc is diminishing for longer and longer words, whenever \$ is read the state returns to r, hence all words return to r with probability 1 in the limit. By the choice of words in the sequence, for every word $\nu_s(w_n) > 0$, we have $\mathbb{P}_{\mathcal{A}}[r \xrightarrow{u_n} q] > 0$ for all n.

Hence a leak has been defined, even in the case where $\mathcal{M}$ is unary. $\square$

6.4 The language containment condition

This section identifies a simple condition which is necessary but not sufficient for s being big-O of s'. Recall that $\frac{x}{0} = \infty$ for $x > 0$. Hence, if there is a word with $\nu_s(w) > 0$ whilst $\nu_{s'}(w) = 0$ then s is not big-O of s'.

Definition 6.22 (Language containment condition). A weighted automaton $\mathcal{W} = \langle Q, \Sigma, M, F\rangle$ and $s, s' \in Q$ satisfy the language containment condition if for all words w with $\nu_s(w) > 0$ it is the case $\nu_{s'}(w) > 0$. ◀

The condition can be verified by constructing a non-deterministic finite automaton which accepts the language of words with non-zero probability.

Definition 6.23. Let $NFA_i(\mathcal{W})$ be the NFA with the same set of states (and final states) as $\mathcal{W}$, start state i, and transitions $q \xrightarrow{a} q'$ whenever $M(a)(q, q') > 0$. ◀

The name is due to the fact that the condition is equivalent to $L(NFA_s(\mathcal{W})) \subseteq L(NFA_{s'}(\mathcal{W}))$. Recall that NFA language containment is **NL**-complete if the automaton is in fact deterministic, in **P** if the NFA is unambiguous [Col15, Theorem 3], is **coNP**-complete for unary NFAs and in general **PSPACE**-complete (see e.g. [KMT17]); in all cases this will match, or be easier than, the respective algorithm for the big-O problem.

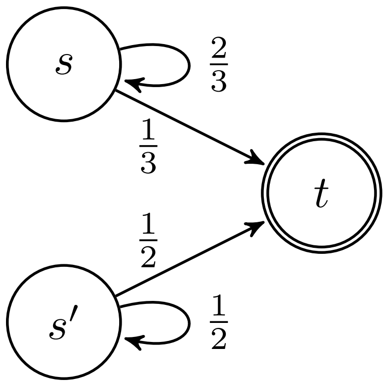


Figure 6.4: Language equivalent but not big-O. Transition labels are omitted as the alphabet is unary.

The language containment condition will be the first step in each of the verification routines to be presented. However, the following example shows that the condition alone is not sufficient to solve the big-O problem, because there may be two states that admit the same set of words with non-zero weight, but have unbounded ratio.

Example 6.24. In Figure 6.4 the states s, s' admit the same languages (both are $\{a^n \mid n \geq 1\}$) but the ratio is unbounded:

$$\frac{\nu_s(a^n)}{\nu_{s'}(a^n)} = \frac{(\frac{2}{3})^n \frac{1}{3}}{(\frac{1}{2})^n \frac{1}{2}} = \frac{2}{3}\left(\frac{4}{3}\right)^n \xrightarrow[n\to\infty]{} \infty. \qquad ◀$$

Remark. The classic big-O notation refers to functions $f, g : \mathbb{N} \to \mathbb{N}$, stating that f is $O(g)$ if $\exists C > 0 : \forall n > 0\ f(n) \leq C\, g(n)$. In many formulations, another definition is used, which excludes finitely many numbers: $\exists C > 0, k > 0 : \forall n > k\ f(n) \leq C\, g(n)$. The notions are equivalent when g is bounded away from 0, by taking a sufficiently larger C to deal with the finite prefix.

The definitions for weighted automata can thus be amended to $\exists C > 0, k > 0, \forall w \in \Sigma^{\geq k} : \nu_s(w) \leq C \cdot \nu_{s'}(w)$ provided the language containment condition holds. In the definition of big-O, s not big-O s' if there exists even a single word w such that $\nu_s(w) > 0$ and $\nu_s(w') = 0$; however the natural extension to allowing finitely many exceptions to this can be handled by simple modifications to the algorithms. ◀

6.4.1 Unambiguous weighted automata

This section presents the first decidability result, in the case where the weighted automaton is unambiguous (note that this also entails the deterministic case). In this case there is polynomial-time solvability.

Lemma 6.25. *For unambiguous weighted automata, the big-O problem is decidable in polynomial time.*

Proof. Let $\mathcal{W} = \langle Q, \Sigma, M, F\rangle$ be a unambiguous weighted automaton. Suppose $s, s' \in Q$ and t is a unique final state. If $\mathcal{W}$ fails the language containment condition (recall that it can be checked in polynomial time), return NO. Otherwise, let us construct another weighted automaton, call it $\mathcal{W}'$, via the product construction involving two copies of $\mathcal{W}$: for all $q_1, q_2, q_1', q_2' \in Q$, add edges $(q_1, q_1') \xrightarrow[a]{p} (q_2, q_2')$ where $p = \frac{M(a)(q_1,q_2)}{M(a)(q_1',q_2')}$. Now, observe that s is not big-O of s' (for $\mathcal{W}$) if and only there is a path (using edges with positive weights) in $\mathcal{W}'$ from (s, s') to (t, t) which contains a cycle such that the product of the weights in that cycle is greater than 1. Note that some edges may be ∞, but thanks to the language containment condition, the weights are well-defined (i.e. not ∞) on all paths from (s, s') to (t, t). The latter can also be checked in polynomial time, for instance, by a modified version of the Bellman-Ford algorithm, which can be used to find negative cycles. The natural approach would be to take the negative of the log of each edge weight so that the the presence of a negative cycle indicates the presence of a positive cycle when edges are multiplied together. To avoid inaccuracies from taking logs, the logs can be represented by their exponent, using multiplication instead of addition in the Bellman-Ford algorithm. □

Remark. Observe that transitions which are taken once in any run are of very little significance to the big-O problem. Such transitions have at most a constant multiplicative effect on the ratio. This will hold true whether the system is unambiguous or otherwise. The relevant behaviours are those on cycles. ◀

6.5 The big-O problem for unary weighted automata is **coNP**-complete

This section shows that the big-O problem for unary weighted automata is **coNP**-complete. The upper bound is shown for the big-O problem on *unary weighted automata* and the lower bound for *unary word-terminating labelled Markov chains.*

For the upper bound, the analysis will refine the analysis of the growth of powers of non-negative matrices of Friedland and Schneider [Sch86] which gives the asymptotic order of growth of $A^n_{s,s'} + A^{n+1}_{s,s'} + \cdots + A^{n+q}_{s,s'}$ for some appropriate q. The refinement is able to give an accurate answer to the asymptotic value of $A^n_{s,s'}$, without the smoothing effect of $A^{n+1}_{s,s'} + \cdots + A^{n+q}_{s,s'}$.

Theorem 6.26. *The big-O problem for unary non-negative weighted automata is* **coNP***-complete.*

6.5.1 Preliminaries

Let $\mathcal{W}$ be a unary non-negative weighted automaton with states Q, transition matrix A and a unique final state t.

Definition 6.27. Recall the greatest common divisor, the gcd of $\{x_1, \dots, x_n\}$ is the largest integer c such $\frac{x_i}{c}$ is an integer for every i.

Recall the least common multiple, the lcm of $\{x_1, \dots, x_n\}$ is the smallest integer c such $\frac{c}{x_i}$ is an integer for every i. ◀

Whenever a *path* of a weighted automaton is referred to, states on the path may repeat. Formally, a path is in the NFA of $\mathcal{W}$, that is, paths only use transitions with non-zero weights. A state q can *reach* q' if there is a path from q to q'. In particular, any state q can always reach itself.

Definition 6.28. A strongly connected component (SCC) $\varphi \subseteq Q$ is a maximal set of states such that for each $q, q' \in \varphi$, q can reach q' with non-zero weight. Denote by A^φ, the $|\varphi| \times |\varphi|$ transition matrix of φ, and by $\mathrm{SCC}(q)$ the SCC in which q is a member. ◀

Note that every state is in a SCC, even if it is a singleton.

Definition 6.29. The DAG of $\mathcal{W}$ is the directed acyclic graph of strongly connected components. Components φ, φ' are connected by an edge if there exists $q \in \varphi$ and $q' \in \varphi'$ with $A(q, q') > 0$. ◀

Definition 6.30. The spectral radius of an $m \times m$ matrix A is the largest absolute value of the eigenvalues of A (the eigenvalues of A are the set $\{\lambda \in \mathbb{C} \mid \text{ exists vector } \vec{x} \in \mathbb{C}^m, \vec{x} \neq 0 \text{ with } A\vec{x} = \lambda\vec{x}\}$). The spectral radius of φ, denoted by ρ_φ, is the spectral radius of A^φ. By $\rho(q)$ denote the spectral radius of the SCC in which q is a member. If $A^\varphi = [0]$ then $\rho_\varphi = 0$. ◀

The spectral radius of an SCC is an algebraic number, as the absolute value of a root of a polynomial with rational coefficients; as such can be represented using the tuple to record a polynomial and a sufficiently close approximation (recall the representation described in Section 3.1.2). Henceforth, when the spectral radius is referred to this form is meant implicitly. In this section, only the relative ordering of the spectral radii of the SCCs are needed. However in Section 6.6 the exact value is relied upon.

Lemma 6.31. *Given A^{φ}, a representation of the value ρ_{φ} can be found in polynomial time. This representation will admit polynomial time testing of $\rho_{\varphi} > \rho_{\varphi'}$ and $\rho_{\varphi} = \rho_{\varphi'}$.*

Proof. Any coefficient of the characteristic polynomial of an *integer* matrix can be found in **GapL** [HT03]. **GapL** is the difference of two **#L** calls, each of which can be found in $\mathbf{NC}^2 \subseteq \mathbf{P}$. Here the matrix will be rational; but it can be normalised to an integer matrix by a scaler, the least common multiple of the denominator of each rational. Whilst the number may be exponential, its representation will be polynomial. Once the eigenvalues of the integer matrix have been found, they can be renormalised by this constant.

The characteristic polynomial of an $n \times n$ matrix has degree at most n, since each coefficient can be found in polynomial time, the whole characteristic polynomial can be found in this time. Thus by enumerating its roots (at most n), taking the modulus of each, and sorting them with a comparison based sort($a > b \iff a + -1 \times b > 0$) it is possible to find the spectral radius in this form (p_z, a, b, r).

Algebraic numbers can be complex, but note that the spectral radius is a real number, so that given the spectral radius in the form (p_z, a, b, r) then in fact $b = 0$. Then the number can be encoded exactly in the first order theory of the reals using $\exists z : p_z(z) = 0 \wedge z - a \leq r \wedge a - z \leq r$. □

Definition 6.32. Denote by q^{φ} the period of the SCC φ to be the greatest common divisor of return times for some state in the SCC, i.e. $\gcd\left\{t \in \mathbb{N} \mid A^t(s,s) > 0\right\}$. It is known that any choice of state in the SCC will give the same value (see e.g. [Ser13, Theorem 1.20]). If there is no cycle, then the period is zero. ◀

Definition 6.33. Let $\mathscr{P}(s, s')$ be the set of paths from the SCC of s to the SCC of s' in the DAG of $\mathcal{W}$. Thus a path $\pi \in \mathscr{P}(s, s')$ is a sequence of SCCs $\varphi_1, \ldots, \varphi_m$. ◀

Definition 6.34. Let $q(s, s')$ be the local period between s and s'; let

$$q(s, s') = \operatorname*{lcm}_{\pi \in \mathscr{P}(s,s')} \operatorname*{gcd}_{\varphi \in \pi} \quad q^{\varphi}. \qquad ◀$$

Definition 6.35. The spectral radius between states s and s' is the largest spectral radius seen of any SCC seen on a path from s to s, formally:

$$\rho(s, s') = \max_{\pi \in \mathscr{P}(s,s')} \rho(\pi), \quad \text{where } \rho(\pi) = \max_{\varphi \in \pi} \rho_{\varphi} \text{ for } \pi \in \mathscr{P}(s, s'). \qquad ◀$$

The following function captures the number of SCCs which attain the largest spectral radius on the path which has the most SCCs of maximal spectral radius.

Definition 6.36. Let $k(s, s')$ be the integer such that

$$k(s, s') + 1 = \max_{\pi \in \mathscr{P}(s,s')} k(\pi)$$

where, for $\pi \in \mathscr{P}(s, s')$, $k(\pi) = |\{\varphi \in \pi \mid \rho_\varphi = \rho(s, s')\}|$. ◀

The asymptotic behaviours of weighted automata will be characterised using the following key definition.

Definition 6.37. A (ρ, k)-pair is an element of $\mathbb{R} \times \mathbb{N}$. The ordering on $\mathbb{R} \times \mathbb{N}$ is lexicographic, i.e.

$$(\rho_1, k_1) \leq (\rho_2, k_2) \iff \rho_1 < \rho_2 \vee (\rho_1 = \rho_2 \wedge k_1 \leq k_2). \quad ◀$$

Friedland and Schneider [FS80; Sch86] essentially use (ρ, k)-pairs to show the asymptotic behaviour of the powers of non-negative matrices. In particular they smooth the behaviour of the matrix over the local period and then show the asymptotic behaviour of this.

Theorem 6.38 (Friedland and Schneider [FS80; Sch86])**.** *Let A be an $m \times m$ non-negative matrix, inducing a unary weighted automaton $\mathcal{W}$ with states $Q = \{1, \dots, m\}$. Given $s, t \in Q$, let $B^n_{s,t} = A^n_{s,t} + A^{n+1}_{s,t} + \cdots + A^{n+q(s,t)-1}_{s,t}$. Then*

$$\lim_{n\to\infty} \frac{B^n_{s,t}}{\rho(s,t)^n n^{k(s,t)}} = c, \quad 0 < c < \infty.$$

In the case where the local period is one ($q(s, t) = q(s', t) = 1$), Theorem 6.38 can already be used to solve the big-O problem (in particular if the matrix A is aperiodic). In this case $A^n_{s,t} = B^n_{s,t} = \Theta(\rho(s, t)^n n^{k(s,t)})$. Then to establish that s is big-O of s' verify that the language containment condition holds and

$$(\rho(s, t), k(s, t)) \leq (\rho(s', t), k(s', t)).$$

However this is not sufficient if the local period if not one.

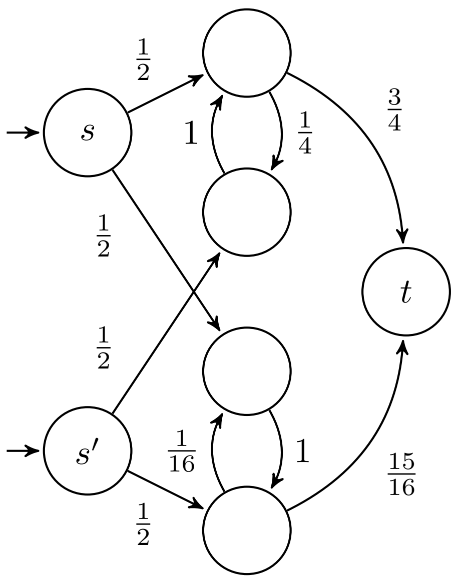


Figure 6.5: Different rates for different phases.

Example 6.39. Consider chains shown in Figure 6.5 with local period 2. The behaviour is:

$$A^n_{s,t} = \begin{cases} \Theta(\frac{1}{2}^n) & \text{n even} \\ \Theta(\frac{1}{4}^n) & \text{n odd} \end{cases} \quad \text{and} \quad A^n_{s',t} = \begin{cases} \Theta(\frac{1}{4}^n) & \text{n even} \\ \Theta(\frac{1}{2}^n) & \text{n odd} \end{cases}$$

However Theorem 6.38 tells us $B^n_{s,t} = \Theta(\frac{1}{2}^n)$ and $B^n_{s',t} = \Theta(\frac{1}{2}^n)$ suggesting s is big-O of s', but actually

$$\frac{A^{2n}_{s,t}}{A^{2n}_{s',t}} \xrightarrow[n\to\infty]{} \infty.$$ ◀

Non-deterministic finite automata The computability arguments will rely upon the following normalised forms of non-deterministic finite automata. In particular these will be necessary for the arguments on eventual inclusion (to be defined in Definition 6.42) and when proving Theorem 6.26.

Definition 6.40. A unary NFA $\mathcal{N} = \langle Q, \to, q_s, F\rangle$ is in *Chrobak normal form* [Chr86] if

- Q can be partitioned so that $Q = S \uplus C^1 \uplus \cdots \uplus C^k$, where $\uplus$ denotes disjoint union,
- S forms only a path $s_1 \xrightarrow{a} s_2 \xrightarrow{a} \ldots \xrightarrow{a} s_m$,
- C^i forms only a cycle $c^i_1 \xrightarrow{a} c^i_2 \xrightarrow{a} \ldots \xrightarrow{a} c^i_{|c^i|} \xrightarrow{a} c^i_1$ for all $i \in [k]$,
- $s_m \xrightarrow{a} c^i_1$ for all $i \in [k]$. ◀

Any unary NFA can be translated to this representation with at most quadratic blow up in the size of the machine [Chr86], despite work to find efficiencies [SJ05], all that is needed here is that a representation can be found in polynomial time [To09; Mar02]. Consider the following restriction on Chrobak normal form, which will simplify the argument:

Definition 6.41. A unary NFA $\mathcal{N}$ is in *restricted Chrobak normal form* if

- it is in Chrobak normal form, and
- there is exactly one accepting state in each cycle. ◀

This restricted form can be found with at most a further quadratic blow up over Chrobak normal form, by duplicating once for each accepting state in the cycle.

6.5.2 Eventual inclusion

As part of the computability argument in the next section will require a new relaxed notion of inclusion, called *eventual inclusion.* This relation requires that one set is included in another "eventually", that is, with finitely many exceptions.

Definition 6.42. Given two sets A, B, A is *eventually included* in B, denoted $A \lesssim B$, if and only if $A \setminus B$ is finite. ◀

The following results consider the complexity of deciding this property on NFAs, showing that for bounded (and thus for unary NFAs) the problem is in **coNP** and more generally in **PSPACE**.

Theorem 6.43. *Given two NFA, $\mathcal{N}_1, \mathcal{N}_2$, deciding whether $L(\mathcal{N}_1) \lesssim L(\mathcal{N}_2)$ is in* **coNP** *if $L(\mathcal{N}_1)$ and $L(\mathcal{N}_2)$ are bounded and in* **PSPACE** *in the general case.*

Proof. One could construct M, a *deterministic* finite automaton (DFA) for the language $L(\mathcal{N}_1) \cap L(\overline{\mathcal{N}_2})$, by explicitly producing the DFAs for $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively and then applying a standard product construction. The size of the resulting automaton is $|M| = 2^{|\mathcal{N}_2|+|\mathcal{N}_1|}$

Then $L(\mathcal{N}_1) \lesssim L(\mathcal{N}_2)$ if and only if $L(M)$ is finite. However $L(M)$ is infinite if and only if there exists some word $w \in L(D_2) \cap L(\overline{D_2})$, with $|M| \leq w \leq 2\,|M|$.

When the word is bounded, one can guess $n_1, n_2, \ldots, n_m \in \mathbb{N}^m$, such that $n_1 + n_2 + \cdots + n_m \in [|M|, 2\,|M|]$ and verify the word $w_1^{n_1}, w_2^{n_2} \ldots w_m^{n_m} \in L(\mathcal{N}_1)$

and $w_1^{n_1}, w_2^{n_2} \dots w_m^{n_m} \notin L(\mathcal{N}_1)$. If such a word exists then $L(\mathcal{N}_1) \not\subsetneq L(\mathcal{N}_2)$ and if no such word exists then $L(\mathcal{N}_1) \subsetneq L(\mathcal{N}_2)$.

However the DFA M does not need to be constructed explicitly, the check $w_1^{n_1}, w_2^{n_2} \dots w_m^{n_m} \in L(\mathcal{N}_1)$ can be done directly on the machine $\mathcal{N}_1$ without construction of D_1 by repeated squaring [IRS76].

When the word is not bounded, one cannot guess the word in this minimised form. Instead non-deterministically simulate a word with length in this bound and accept if accepted by $\mathcal{N}_1$ and rejected by $\mathcal{N}_2$ in **NPSPACE**. Recall **PSPACE** = **NPSPACE**. □

From the proof one can see that if $L(\mathcal{N}_1) \cap \overline{L(\mathcal{N}_1)}$ is finite, it is of at most exponential size. For unary NFAs, an alternative proof can show this size is at most polynomial.

Lemma 6.44. *if $L(\mathcal{N}_1) \cap \overline{L(\mathcal{N}_2)}$ is finite, it is of polynomial size.*

Proof. The strategy is to provide an alternative proof of that given two unary NFA's, $\mathcal{N}_1, \mathcal{N}_2$, the problem $L(\mathcal{N}_1) \subsetneq L(\mathcal{N}_2)$ is in **coNP**. However, this proof, using Chrobak normal form will observe that the violations must occur within the finite path.

Convert both NFA's to Chrobak normal form. Normalise these so the finite path is the same length in each machine. To do this make the shorter one longer by repeatedly increasing the length by one and transitioning to one state later in each cycle. The new state in the finite path should be accepting if any of the first states in the cycles were accepting. Let z be length of these paths.

Assume in $\mathcal{N}_1$, there are n cycles, of lengths $a_1, \dots, a_n$, and in $\mathcal{N}_2$, m cycles of lengths $b_1, \dots, b_m$. For any word longer than z, it is either in both, one of, or neither of $L(\mathcal{N}_1)$ and $L(\mathcal{N}_2)$. Then the behaviour repeats every $q = a_1 \cdot \ldots \cdot a_n \cdot b_1 \cdot \ldots \cdot b_m$ steps. So for every word a^t for $t \in [z, z+q]$ the same condition holds for $t+q, t+2q, t+3q.....$

So if for some t in $[z, z+q]$ there is $a^t \in L(\mathcal{N}_1)$ but $a^t \notin L(\mathcal{N}_2)$ then there are infinitely many words like this. Hence it is necessary that for $t \in [z, z+q]$ that a^t be in both, neither or in $L(\mathcal{N}_2)$ but not $L(\mathcal{N}_1)$.

This test can be performed in **coNP**, by first guessing the violating $t \in [z, z+q]$ and checking $a^t \in L(\mathcal{N}_1)$ but $a^t \notin L(\mathcal{N}_2)$. Checking whether a word $a^t \in L(\mathcal{N}_1)$ can be checked in polynomial time by repeated squaring of the transition matrix, and similarly for $\mathcal{N}_2$.

There are at most finitely many violations, and all of these exceptions occur within the finite paths for a^t, $t \in [0, z]$. Further z is polynomially bounded; it's at most the number of states in the normalised Chrobak normal form, which is polynomial in the size of $\mathcal{N}_1$ or $\mathcal{N}_2$ before conversion to Chrobak normal form. □

6.5.3 The big-O problem for unary weighted automata is in **coNP**

This section proves the computability result of Theorem 6.26. Further Lemma 6.46 shows a refined analysis of the growth in powers of non-negative matrices, without smoothing required in Theorem 6.38. This is done by exhibiting a value of ρ and k for every word.

Let $\mathcal{W}$ be a unary weighted automaton and the question whether state s is big-O of s'. Recall, without loss of generality, it is assumed that there is a unique final state t with no outgoing transitions. Assume, without loss of generality, s, s' do not appear on any cycle. If this is not the case, add two new states $\hat{s}, \hat{s}'$, transitions $\hat{s} \xrightarrow{1} s$, $\hat{s}' \xrightarrow{1} s'$, and consider the big-O problem for $\hat{s}, \hat{s}'$ instead (note that $\nu_{\hat{s}}(a^n) = \nu_s(a^{n-1})$ and $\nu_{\hat{s}'}(a^n) = \nu_{s'}(a^{n-1})$).

The following defines a "degree function", which captures, by way of a value ρ and k, the asymptotic behaviour of each word a^n.

Definition 6.45. Let $f_{s,t} : \mathbb{N} \to \mathbb{R} \times \mathbb{N}$, such that $f_{s,t}(n) = (\rho, k)$ if and only if

- The largest spectral radius of any vertex visited on the any length n path from s to t is ρ.
- The path from s to t which visits the most SCCs of spectral radius ρ visits $k + 1$ such SCCs.
- If there is no path of length n from s to t, then $(\rho, k) = (0, 0)$. ◀

Further, where only one component is necessary, assume the functions $\rho(n), k(n)$, defined by $f_{s,t}(n) = (\rho(n), k(n))$.

Let $s, t \in Q$ be fixed. The following lemma is the key technical lemma of this subsection (compare to Theorem 6.38, Friedland and Schneider [FS80; Sch86]).

Lemma 6.46. *There exists $c, C \in \mathbb{R}$ such that for every $n \in \mathbb{N}$, with $n \geq |Q|$,*

$$c \cdot \rho(n)^n n^{k(n)} \leq A^n_{s,t} \leq C \cdot \rho(n)^n n^{k(n)}.$$

The set of *admissible* (ρ, k)-pairs is the image of $f_{s,t}$. Observe that this set is finite and of size at most $|Q|^2$: there can be no more than $|Q|$ values of ρ (if at worst each state were its own SCC) and the value of k is also bounded by the number of SCCs and thus $|Q|$.

In order to prove Lemma 6.46 and the computability results, the (ρ, k)-annotated version of $\mathcal{W}$ is defined, which, in each state records the relevant value of (ρ, k) corresponding to the current run to the state.

Definition 6.47 (The weighted automaton $\mathcal{M}^\dagger$). Given $\mathcal{W} = \langle Q, \Sigma, M, \{t\}\rangle$ and $s \in Q$, the weighted automaton $\mathcal{M}^\dagger$ has states of the form (q, ρ, k) for all $q \in Q$ and all admissible (ρ, k)-pairs, the same Σ and no final states. For every transition $q \xrightarrow{p} q'$ from $\mathcal{W}$, include the following transition in $\mathcal{M}^\dagger$ for each admissible (ρ, k):

- $(q, \rho, k) \xrightarrow{p} (q', \rho, k)$ if $\mathrm{SCC}(q) = \mathrm{SCC}(q')$
- $(q, \rho, k) \xrightarrow{p} (q', \rho, k+1)$ if $\mathrm{SCC}(q) \neq \mathrm{SCC}(q')$ and $\rho = \rho(q')$
- $(q, \rho, k) \xrightarrow{p} (q', \rho, k)$ if $\mathrm{SCC}(q) \neq \mathrm{SCC}(q')$ and $\rho > \rho(q')$
- $(q, \rho, k) \xrightarrow{p} (q', \rho(q'), 0)$ if $\mathrm{SCC}(q) \neq \mathrm{SCC}(q')$ and $\rho(q') > \rho$ ◀

Note that the accepting behaviour has not been specified, which will depend on the application of $\mathcal{M}^\dagger$, and be specified on use.

Lemma 6.48. *$\mathcal{M}^\dagger$ can be constructed in polynomial time given $\mathcal{W}$.*

Proof. It suffices to note that spectral radii of all SCCs can be computed and compared to each other in time polynomial in the size of $\mathcal{W}$ (see Lemma 6.31). □

Proof of Lemma 6.46. For values $n < |Q|$ the value of ρ may be incorrectly characterised as 0, by going through only strongly connected components with no loops. If $n \geq |Q|$ some state repeats and ρ must be non-zero.

Case 1 (Lower Bound). Fix (ρ', k') and let $\pi = \varphi_1 \dots \varphi_k \in \mathscr{P}(s, t)$ be a sequence of SCCs with exactly $k' + 1$ SCCs of spectral radius ρ' and no SCC with a larger spectral radius.

Additionally, from each φ_i, specify an entry point s_i and an exit point e_i witnessing π, i.e. $s = s_1$, $SCC(s_i) = SCC(e_i) = \varphi_i$ $(1 \leq i \leq k)$, there is a transition (of positive weight) from e_i to s_{i+1} $(1 \leq i < k)$ and $e_k = t$.

Define a new unary weighted automaton $\mathcal{W}^\blacktriangle$ to be a restriction of $\mathcal{W}$ so that the only entry points to its SCCs are s_i's and the only exit point are e_i's, i.e. the probability is reduced to zero for any violating transition. Let D be the transition matrix of $\mathcal{W}^\blacktriangle$.

Clearly $A^n_{s,t} \geq D^n_{s,t}$, since $\mathcal{W}^\blacktriangle$ is a restriction of $\mathcal{W}$. Note that, in $\mathcal{W}^\blacktriangle$, $\rho(s, t) = \rho'$ and $k(s, t) = k'$, because all paths from s to t must visit $k' + 1$ SCCs with spectral radius ρ'. Hence, by Theorem 6.38, $D^n_{s,t} + D^{n+1}_{s,t} + \cdots + D^{n+q-1}_{s,t} \geq c_1(\rho')^n n^{k'}$, for some $c_1 > 0$, where q is the local period from s to t in $\mathcal{W}^\blacktriangle$. Finally it is required that $D^{n+1}_{s,t} + \cdots + D^{n+q-1}_{s,t} = 0$, which implies $D^n_{s,t} \geq c_1(\rho')^n n^{k'}$ and, hence, $A^n_{s,t} \geq c_1(\rho')^n n^{k'}$.

Let L be the length of the shortest path from s to t in $\mathcal{W}^\blacktriangle$. Observe that paths from s to t in $\mathcal{W}^\blacktriangle$ can only have lengths from

$$\Big\{L + n_1 \cdot q^{SCC(s_1)} + \cdots + n_k \cdot q^{SCC(s_k)} \;\mid\; n_1, \ldots, n_k \in \mathbb{N}\Big\}$$

and, thus, $\{L + n \cdot gcd\,\{q^{SCC(s_1)}, \ldots, q^{SCC(s_k)}\} \;\mid\; n \in \mathbb{N}\}$. As $\mathscr{P}(s,t) = \{\pi\}$ in $\mathcal{W}^\blacktriangle$, $q(s,t) = gcd\,\{q^{SCC(s_1)}, \ldots, q^{SCC(s_k)}\}$. Consequently, all paths from s to t in $\mathcal{W}^\blacktriangle$ have lengths of the form $L + nq$. Hence, since $D^n_{s,t}$ is positive, there are no paths which can contribute positive value to $D^{n+1}_{s,t} + \cdots + D^{n+q-1}_{s,t}$.

Case 2 (Upper bound)**.** Let $N_{(\rho',k')} = \{n \mid f_{s,t}(n) = (\rho', k')\}$. This gives a finite partition of $\mathbb{N}$ as $\bigcup_{(\rho,k)} N_{(\rho,k)}$. For each (ρ', k'), the proof will find a value $C_{(\rho',k')}$ so that, for $n \in N_{(\rho',k')}$, we have $A^n_{s,t} \leq C_{(\rho',k')}(\rho')^n n^{k'}$. Then, to have $A^n_{s,t} \leq C\rho(n)^n n^{k(n)}$ for all $n \in \mathbb{N}$, it will suffice to take C to be the maximum over all $C_{(\rho',k')}$.

Fix (ρ', k') and consider $\mathcal{W}^\bullet$ to be $\mathcal{M}^\dagger$ in which, for every $(\rho, k) \leq (\rho', k')$, states (t, ρ, k) are merged into a single final state t' (recall there are no outgoing edges from t). Finally, rename the state $(s, 0, 0)$ to s'. Let E be the corresponding transition matrix of $\mathcal{W}^\bullet$. Note that all paths from s' to t' in $\mathcal{W}^\bullet$ go through at most $k' + 1$ SCCs with spectral radius ρ'.

Claim 6.49. *For all $n \in N_{(\rho',k')}$, we have $A^n_{s,t} = E^n_{s',t'}$.*

Consider any path $s \to q_1 \to \cdots \to q_m \to t$ in $\mathcal{W}$. There is a corresponding path in $\mathcal{W}^\bullet$, however the states q_i are annotated as (q_i, ρ, k), where ρ is the largest spectral radius seen so far, and $k + 1$ is the number of SCCs of that radius number seen so far. The only paths removed are those terminating at (t, ρ, k) with $(\rho, k) > (\rho', k')$. Since $f_{s,t}(n) = (\rho', k')$, then no path visits more than $k' + 1$ SCCs of spectral radius ρ', or an SCC of spectral radius greater than ρ'. Consequently, no such path is disallowed in $\mathcal{W}^\bullet$. No paths were added either. Because every SCC in $\mathcal{W}$ remains a strongly connected component in $\mathcal{W}^\bullet$ (duplicated with various (ρ, k)) and its transition probability matrix (and hence the spectral radius) remains the same, concluding that $A^n_{s,t} = E^n_{s',t'}$.

Claim 6.50. *There exists $C_{(\rho',k')}$ such that $A^n_{s,t} \leq C_{(\rho',k')}(\rho')^n n^{k'}$.*

We have $A^n_{s,t} = E^n_{s',t'} \leq E^n_{s',t'} + E^{n+1}_{s',t'} + \cdots + E^{n+q(s',t')-1}_{s',t'}$, where $q(s', t')$ is the local period between states s' and t' in $\mathcal{W}^\bullet$. By Theorem 6.38, there exists $C_{(\rho',k')}$ such that this quantity is bounded by $C_{(\rho',k')}(\rho')^n n^{k'}$. Thus, for $n \in N_{(\rho',k')}$, we have $A^n_{s,t} \leq C_{(\rho',k')}(\rho')^n n^{k'}$. $\square$

For the following lemma, recall the language containment condition from Definition 6.22 and the ordering on (ρ, k)-pairs from Definition 6.37.

Lemma 6.51. *A state s is big-O of s' if and only if the language containment condition holds and, for all but finitely many $n \in \mathbb{N}$, we have $f_{s,t}(n) \leq f_{s',t}(n)$.*

Proof. First note some consequences of $f_{s,t}(n) \leq f_{s',t}(n)$. Suppose $f_{s,t}(n) = (\rho, k)$ and $f_{s',t}(n) = (\rho', k')$. Thanks to Lemma 6.46, we have $\nu_s(a^n) \leq (\frac{C}{c}(\frac{\rho}{\rho'})^n n^{k-k'}) \cdot \nu_{s'}(a^n)$. If $f_{s,t}(n) \leq f_{s',t}(n)$ then there are two cases: either $(\rho, k) = (\rho', k')$ or $(\rho, k) < (\rho', k')$.

- In the former case, $(\frac{\rho}{\rho'})^n n^{k-k'} = 1$ and, thus, $\nu_s(a^n) \leq (\frac{C}{c}) \cdot \nu_{s'}(a^n)$.
- In the latter case, we have $\lim_{m\to\infty}(\frac{\rho}{\rho'})^m m^{k-k'} = 0$ and, thus, $(\frac{\rho}{\rho'})^m m^{k-k'} < 1$ for all but finitely many m. Consequently, for all but finitely many n, $\nu_s(a^n) \leq (\frac{C}{c}) \cdot \nu_{s'}(a^n)$.

Thanks to the above analysis, if $f_{s,t}(n) \leq f_{s',t}(n)$ holds for all but finitely many n, it follows that $\nu_s(a^n) \leq (\frac{C}{c}) \cdot \nu_{s'}(a^n)$ for all but finitely many n. Moreover, the language containment condition implies that $\nu_s(a^n) \leq C' \cdot \nu_{s'}(a^n)$ for some C' in the remaining (finitely many) cases. Hence, s is big-O of s', which shows the right-to-left implication.

For the converse, recall that it is already established that "s is big-O of s'" implies the language containment condition. For the remaining part, suppose that there are infinitely many n with $f_{s,t}(n) > f_{s',t}(n)$. As there are finitely many values in the range of $f_{s,t}$ and $f_{s',t}$, there exist (ρ, k) and (ρ', k') such that $(\rho, k) > (\rho', k')$ and, for infinitely many n, $f_{s,t} = (\rho, k)$ and $f_{s',t} = (\rho', k')$. For such n, Lemma 6.46 yields $\nu_s(a^n) \geq (\frac{c}{C}(\frac{\rho}{\rho'})^n n^{k-k'}) \cdot \nu_{s'}(a^n)$. But $(\rho, k) > (\rho', k')$ implies

$$\lim_{m\to\infty}(\frac{\rho}{\rho'})^m m^{k-k'} = \infty,$$

i.e. $(\frac{\rho}{\rho'})^n n^{k-k'}$ is unbounded. Thus, s cannot be big-O of s'. $\square$

The following result relates Lemma 6.51 to eventual inclusion.

Lemma 6.52. *Suppose $f_1, f_2 : \mathbb{N} \to X$, where $(X, \leq)$ is a finite total order. Then $f_1(n) \leq f_2(n)$ for all but finitely many n if and only if $\{n \mid f_1(n) \geq x\} \subsetneq \{n \mid f_2(n) \geq x\}$ for all $x \in X$.*

Proof. The left-to-right implication is clear. For the opposite direction, observe that, because the order on X is total, $f_1(n) > f_2(n)$ implies the existence of $x \in X$ such that $f_1(n) \geq x$ and $f_2(n) < x$ (it suffices to take $x = f_1(n)$). Because X is finite, $f_1(n) > f_2(n)$ for infinitely many n implies failure of $\{n \mid f_1(n) \geq x\} \subsetneq \{n \mid f_1(n) \geq x\}$ for some x. □

Lemma 6.53. *Given a unary weighted automaton $\mathcal{W}$, the associated problem whether $f_{s,t}(n) \leq f_{s',t}(n)$ for all but finitely many $n \in \mathbb{N}$ is in* **coNP***.*

Proof. Given an admissible pair $x = (\rho, k)$, construct an NFA $\mathcal{N}_{s,x}$ accepting $\{a^n \mid f_{s,t}(n) \geq x\}$, taking the NFA $NFA_s(\mathcal{M}^\dagger)$ (Definitions 6.23 and 6.47) with a suitable choice of accepting states. Recall that states in $\mathcal{M}^\dagger$ are of the form (q, ρ', k'), where q is a state from $\mathcal{W}$ and (ρ', k') is admissible. Designate states (t, ρ', k') with $(\rho', k') \geq x$ as accepting, so it will accept $\{a^n \mid f_{s,t}(n) \geq x\}$. Clearly, this is a polynomial-time construction. An analogous automaton, let us call it $\mathcal{N}_{s',x}$, can be constructed for s'.

Then, by Lemma 6.52, the problem whether $f_{s,t}(n) \leq f_{s',t}(n)$ for all but finitely many $n \in \mathbb{N}$ is equivalent to $L(\mathcal{N}_{s,x}) \subsetneq L(\mathcal{N}_{s',x})$ for all admissible x. As there are at most $|Q|^2$ many values of x and each can be verified non-deterministically by **coNP**, it suffices to show that $L(\mathcal{N}_{s,x}) \subsetneq L(\mathcal{N}_{s',x})$ is in **coNP** for each x. This is the case by Theorem 6.43. □

Remark. Lemma 6.52 remains true if using the condition $\{n \mid f_1(n) = x\} \subsetneq \{n \mid f_2(n) \geq x\}$ instead. However, although this appears simpler, it does not seem possible to construct an NFA for $\{a^n \mid f_{s,t}(n) = x\}$ in polynomial time. For instance, taking accepting states to be (t', ρ, k) would not be correct, because there could be paths of the same length ending in (t', ρ', k') with $(\rho', k') > (\rho, k)$. This problem is avoided if one uses $\geq$ instead of $=$, as done in Lemma 6.53. ◀

Lemma 6.51 and Lemma 6.53 together complete computability result for Theorem 6.26. The required hardness results is established next.

6.5.4 Tv-Bounded is **coNP**-hard for unary LMCs

Given a unary NFA, $\mathcal{N}$, the *NFA universality problem* asks if is it the case that $L(\mathcal{N}) = \{a^n \mid n \in \mathbb{N}\}$. This problem is **coNP**-complete [SM73].

Lemma 6.54. Tv-Bounded(s, s') *is* **coNP***-hard on unary Markov chains.*

Proof of Lemma 6.54. The proof reduces the NFA universality problem to Tv-Bounded(s, s'). This will produce a labelled Markov chain with a strongly

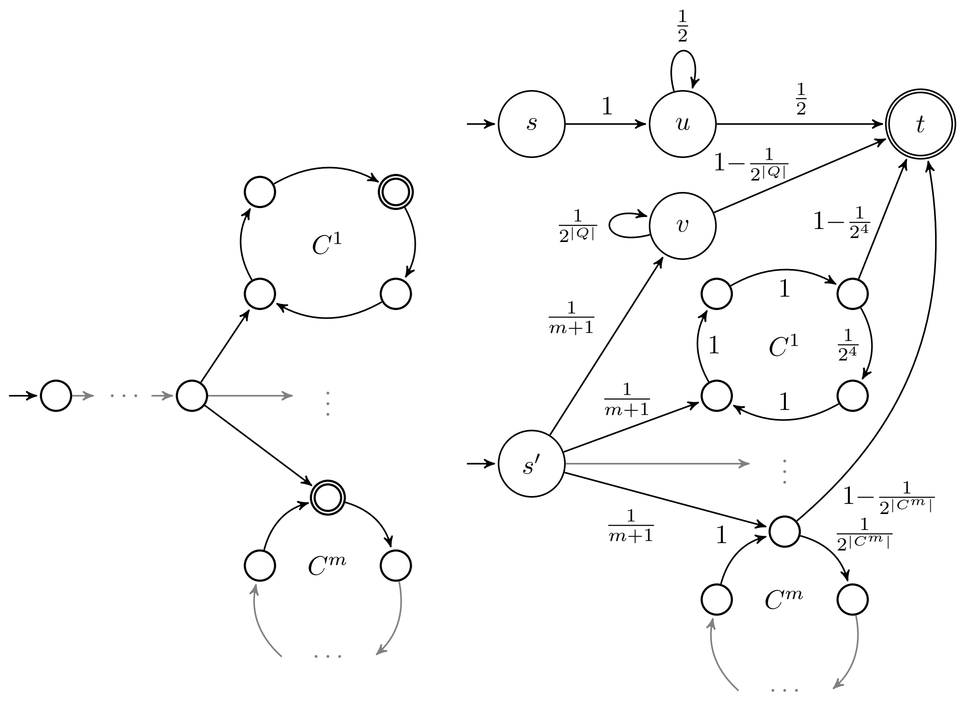


Figure 6.6: Reduction from NFA (left) to LMC (right).

connected component for each cycle of the NFA, given in Chrobak normal form. Each strongly connected component will be set to have spectral radius $\frac{1}{2}$, so that in the case when the word is in the language $\nu_{s'}(a^n) = \Theta((\frac{1}{2})^n)$ and when the word is not in the language it will have a much smaller probability. When there are infinitely many words not in the language the ratio with a branch which always has $\nu_s(a^n) = \Theta((\frac{1}{2})^n)$, will diverge. The remaining finite case will be considered separately.

Formally, assume $\mathcal{N} = \langle Q, \rightarrow, s_1, F\rangle$ is given in restricted Chrobak normal form (Definition 6.41), or convert to this representation in polynomial time. Then there is some path $s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n$ and for $1 \leq k \leq m$, we have a cycle $C^k = c^k_1 \rightarrow c^k_2 \rightarrow \cdots \rightarrow c^k_{|C^k|} \rightarrow c^k_1$, with $s_n \rightarrow c^k_1$. All transitions are labelled with the unique character (say a). There is exactly one accepting state in each cycle.

If any of $s_1, \ldots s_n$ are not accepting then $\mathcal{N}$ is not universal; simply reject. The path can now be discarded, and $\mathcal{N}$ has is a single accepting vertex s' as the path, so $Q = \{s'\} \uplus C^1 \uplus \ldots \uplus C^m$.

A unary Markov chain $\mathcal{M}$, depicted in Figure 6.6 is constructed, with states $Q' = Q \cup \{s, u, v, t\}$ and t the final state.

First consider the fixed branch from s, behaving as $\nu_s(a^n) = \Theta((\frac{1}{2})^n)$ for all

$n \geq 2$:

$$s \xrightarrow{1} u \qquad u \xrightarrow{\frac{1}{2}} u \qquad u \xrightarrow{\frac{1}{2}} t.$$

Then consider the remaining branch derived from $\mathcal{N}$. First ensure that every word has non-zero probability with a branch having very small probability.

$$s' \xrightarrow{\frac{1}{m+1}} v \qquad v \xrightarrow{\frac{1}{2^{|Q|}}} v \qquad v \xrightarrow{1-\frac{1}{2^{|Q|}}} t.$$

The remainder ensures that if the cycle can be used, it will cause the probability to behave according to $\nu_{s'}(a^n) = \Theta((\frac{1}{2})^n)$: First enable a jump to each cycle, for $1 \leq k \leq m$

$$s' \xrightarrow{\frac{1}{m+1}} c_1^k$$

And then simulate the cycle: if $c_j^k \notin F$ we have

$$c_{j-1}^k \xrightarrow{1} c_j^k \qquad \text{(if } j = 1 \text{ then } j - 1 = \left|C^k\right|\text{)}$$

and, if $c_j^k \in F$ we have

$$c_{j-1}^k \xrightarrow{\frac{1}{2}^{\left|C^k\right|}} c_j^k \qquad c_{j-1}^k \xrightarrow{1-\frac{1}{2}^{\left|C^k\right|}} t \qquad \text{(if } j = 1 \text{ then } j - 1 = \left|C^k\right|\text{)}.$$

Let $\varepsilon = 1 - \frac{1}{2}^{\left|C^k\right|}$, that is, the probability to leave the cycle instead of entering the state corresponding to an accepting state in the NFA.

Claim 6.55. *The spectral radius ρ_{C_i} of each cycle SCC is $(1-\varepsilon)^{\frac{|C_i \cap F|}{|C_i|}}$.*

Proof of Claim 6.55. Let A be the transition matrix of the cycle,

$$A = \begin{bmatrix} 0 & x_1 & 0 & \cdots & 0 \\ & 0 & x_2 & & 0 \\ & & 0 & x_3 & \vdots \\ \vdots & & & \ddots & x_{n-1} \\ x_n & 0 & 0 & \dots & 0 \end{bmatrix}$$

then we have

$$A - \lambda I = \begin{bmatrix} -\lambda & x_1 & 0 & \cdots & 0 \\ & -\lambda & x_2 & & 0 \\ & & -\lambda & x_3 & \vdots \\ \vdots & & & \ddots & x_{n-1} \\ x_n & 0 & 0 & \dots & -\lambda \end{bmatrix}.$$

The eigenvalue of A are the roots of $det(A - \lambda I) = 0$. Using Leibniz formula $det(A) = \Sigma_{\pi \in S_n} sgn(\pi) \Pi_{i=1}^{n} A_{\pi(i),i}$, where S_n is the set of all permutation functions. However note the only permutations which result in non-zero products are the identity $[1, \dots n]$ and $[2, \dots, n, 1]$. This obtains $(-\lambda)^n + (-1)^{n-1}(x_1 \cdot \dots \cdot x_n) = 0$. Note that x_i is 1 where the only option is to stay in the cycle and $(1-\varepsilon)$ where there is some possibility to leave. Thus $x_1 \cdot \dots \cdot x_n = (1-\varepsilon)^{\left|C^i \cap F\right|}$, obtaining $\lambda^n = (1-\varepsilon)^{\left|C^i \cap F\right|}$ and so the only real dominant eigenvalue is $\lambda = (1-\varepsilon)^{\frac{\left|C^i \cap F\right|}{\left|C^i\right|}}$. ■

Claim 6.56. *$\mathcal{N}$ universal* $\iff$ $tv_{\oslash}(s, s') < \infty \iff tv_{\otimes}(s, s') < \infty$.

Proof. Let $f_{s,t}$ be defined as in Definition 6.45, then $f_{s,t}(0) = f_{s,t}(1) = f_{s',t}(0) = f_{s',t}(1) = (0, 0)$. For the s branch, observe then that $f_{s,t}(n) = (\frac{1}{2}, 0)$ for all $n \geq 2$. Also note, using the claim above, that every cycle's spectral radius, ρ_{C^i}, is $\frac{1}{2}$:

$$\rho_{C^k} = (1-\varepsilon)^{\frac{\left|C^k \cap F\right|}{\left|C^k\right|}} = (1-\varepsilon)^{\frac{1}{\left|C^k\right|}} = \left((\frac{1}{2})^{\left|C^k\right|} \right)^{\frac{1}{\left|C^k\right|}} = \frac{1}{2}.$$

Thus for all $n \geq 2$ we have:

$$f_{s',t}(n) = \begin{cases} (\frac{1}{2}, 0) & \text{some cycle accepts } a^{n-1} \text{ in } \mathcal{N} \\ (\frac{1}{2^{|Q|}}, 0) & \text{otherwise} \end{cases}$$

Due to the branches $s \to u \to t$ and $s' \to v \to t$, both machines accept all words of length 2 or more with some probability; the language containment condition holds without testing. Then recall, by Lemma 6.51, $\frac{\nu_s(s,s')}{\nu_{s'}(s,s')}$ is bounded when all but finitely many n give $f_{s',t}(n) \geq f_{s,t}(n)$. Note then $f_{s',t}(n) \leq f_{s,t}(n)$ for all $n \geq 2$ so $\frac{\nu_{s'}(s,s')}{\nu_s(s,s')}$ is always bounded.

Suppose the automaton is universal, all a^n are accepted. Then $f_{s',t}(n) = f_{s,t}(n) = (\frac{1}{2}, 0)$ for all $n \geq 2$ so $\frac{\nu_s(s,s')}{\nu_{s'}(s,s')}$ is bounded.

Suppose the automaton is not universal, so some $a^t \notin L(\mathcal{N})$. In Chrobak

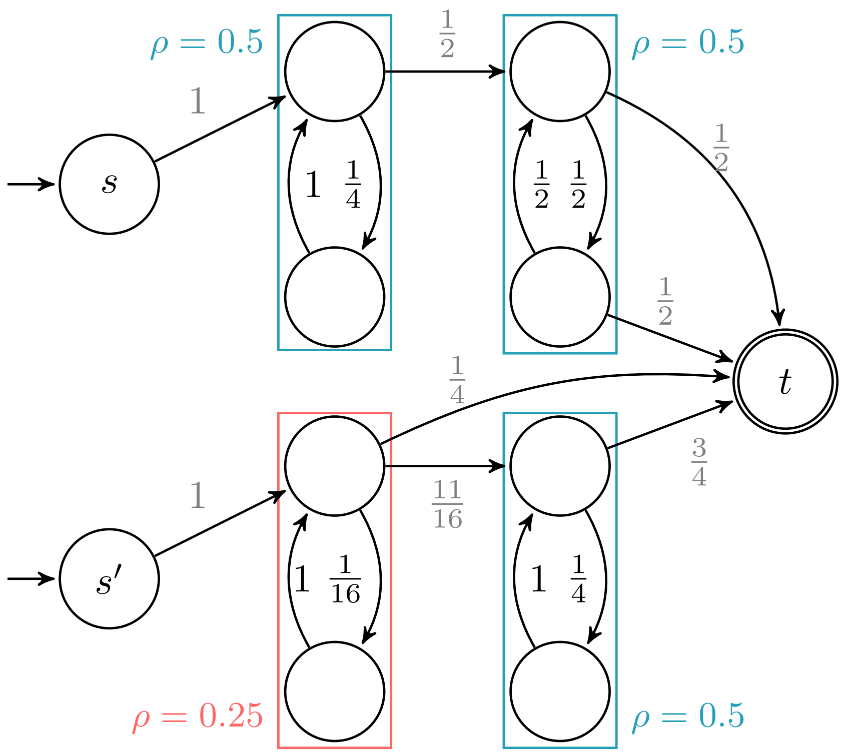


Figure 6.7: Example unary labelled Markov chain, with the spectral radius of each SCC indicated.

normal form, the behaviour is periodic for $q = lcm\left\{\left|C^1\right|, \ldots, |C^m|\right\}$. Hence $a^{t+q\cdot z} \notin L(\mathcal{N})$ for all $z \in \mathbb{N}$, so giving an infinite sequence with $f_{s',t}(n) < f_{s,t}(n)$, so $\frac{\nu_s(s,s')}{\nu_{s'}(s,s')}$ unbounded. ■

Claim 6.56 entails the required hardness for Lemma 6.54. □

Example 6.57. Consider the Markov chain depicted in Figure 6.7.

$$\nu_s(a^n) = \begin{cases} \Theta(0.5^n n) & n \geq 3 \\ 0 & n = 1, 2 \end{cases}$$

$$\nu_{s'}(a^n) = \begin{cases} \Theta(0.25^n) & n \geq 2 \text{ and even} \\ \Theta(0.5^n) & n \geq 3 \text{ and odd} \\ 0 & n = 1 \end{cases}$$

Then s *not* big-O of s' as $\dfrac{\nu_s(a^n)}{\nu_{s'}(a^n)} \xrightarrow{n\to\infty} \infty$.

Also s' *not* big-O of s, as $\nu_{s'}(a^2) > 0$ but $\nu_s(a^2) = 0$. ◀

6.6 Decidability for weighted automata with bounded languages

This section considers the big-O problem on weighted automata with bounded languages, which can be considered a generalisation of unary languages.

Definition 6.58. Let $L \subseteq \Sigma^*$.

- L is *bounded* [GS64] if $L \subseteq w_1^* w_2^* \cdots w_m^*$ for some $w_1, \ldots, w_m \in \Sigma^*$.
- L is *letter-bounded* if $L \subseteq a_1^* a_2^* \ldots a_m^*$ for some $a_1, \ldots, a_m \in \Sigma$.
- L is *plus-letter-bounded* if $L \subseteq a_1^+ a_2^+ \ldots a_m^+$ for some $a_1, \ldots, a_m \in \Sigma$. ◀

The problem is first discussed for the plus-letter-bounded case, and in the subsequent subsections this is generalised to weighted automata with bounded languages. The decidability result in this section uses the first order theory of the real numbers with exponential function. This theory is only known to be decidable subject to Schanuel's conjecture [MW96], and so the result is conditional on this conjecture.

Theorem 6.59. *Given a weighted automaton $\mathcal{W} = \langle Q, \Sigma, M, F\rangle$, $s, s' \in Q$, with $L(NFA_s(\mathcal{W}))$ and $L(NFA_{s'}(\mathcal{W}))$ bounded, it is decidable whether $\mathcal{W}, s, s'$ form a positive instance of the big-O problem, subject to Schanuel's conjecture.*

In the unary case it is sufficient to consider the relative order between spectral radii at various points, with careful handling of the periodic behaviour. Such an approach is not sufficient in the bounded case as the actual values of the spectral radii have to be examined, as demonstrated in Example 6.60.

Example 6.60 (Relative orderings are insufficient). Consider the LMC in Figure 6.8, with $0.61 \le p \le 0.62$, we have $\nu_s(a^n b^m) = \Theta(0.6^n \cdot 0.4^m)$ and initially $\nu_{s'}(a^n b^m) = \Theta(p^n \cdot 0.39^m + 0.59^n \cdot 0.41^m)$. Note that neither $0.59^n \cdot 0.41^m$ nor $p^n \cdot 0.39^m$ dominate, nor are dominated by, $0.6^n \cdot 0.4^m$ for any value of $0.61 \le p \le 0.62$. That is, there are values of n and m where $0.59^n \cdot 0.41^m \gg 0.6^n \cdot 0.4^m$ and values of n and m where $0.59^n \cdot 0.41^m \ll 0.6^n \cdot 0.4^m$; and similarly for $p^n \cdot 0.39^m$ and $0.6^n \cdot 0.4^m$. However the values $p = 0.61$ and $p = 0.62$ have different outcomes with respect to the big-O problem; despite the same relative ordering between values.

When $p = 0.62$, the ratio $\frac{\nu_s(a^n b^m)}{\nu_{s'}(a^n b^m)}$ is bounded for all n, m. However, when $p = 0.61$, observe there is a solution to x with $0.61 \cdot 0.39^x < 0.6 \cdot 0.4^x$ and $0.59 \cdot 0.41^x < 0.6 \cdot 0.4^x$, e.g. $x = 0.66$, then let $m = xn$ and observe $\frac{\nu_s(a^n b^{0.66n})}{\nu_{s'}(a^n b^{0.66n})} \xrightarrow[n\to\infty]{} \infty$. Whilst useful for illustration in this example, this effect is not limited to a linear relation between the characters, and so heavier machinery is required. ◀

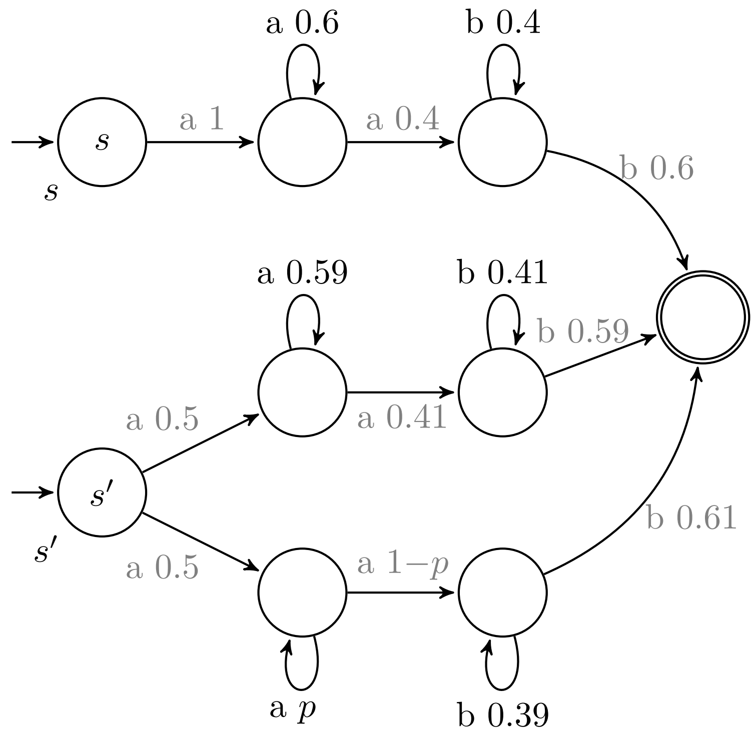


Figure 6.8: For $p = 0.61$ and $p = 0.62$ the relative orderings of spectral radii are the same, but the big-O status is different.

The strategy to prove Theorem 6.59 will be as follows. First detector automata will be defined, which characterise behaviours generalising (ρ, k)-pairs considered in the unary case. Once the behaviours can be characterised, the technical meat of the argument will show the theorem restricted to plus-letter-bounded languages (in Lemma 6.64). The generalisations of letter-bounded languages and bounded languages will then be reduced to this case.

6.6.1 Detector automata

Like in Definition 6.45, a degree function is used which will associate the asymptotic behaviour of each word, but (ρ, k) values are associated to each of the m characters. However there may be multiple, incomparable behaviours, hence $f : \mathbb{N}^m \to \mathcal{P}\{(\mathbb{R} \times \mathbb{N})^m\}$, such that for elements $f(n_1, \dots, n_m)$ are $((\rho_1, k_1), \dots, (\rho_m, k_m))$ pairs that are incomparable, first consider how to compare elements.

Recall Lemma 6.46 does not capture the asymptotics when $n \leq |Q|$. In the unary case this is inconsequential as small words are covered by the *finitely many* exceptions and the language containment condition. However, here, a small number of one character may be used to enable access to a particular part of the automaton in another character. For this case, a new number

$\delta = \frac{1}{2}\min_{\varphi:\rho_\varphi>0}\rho_\varphi$ is introduced which is strictly smaller than the spectral radius of every non-zero SCC. The purpose of this value is to assign non-zero weight to these small paths in such a way that they do not dominate in the partial order. Since the path is a small finite path, thus in each path δ will occur a finite number of times the asymptotic characterisation will be correct no matter the true value of δ.

Definition 6.61. Let $\leq$ be a partial on $(\mathbb{R}\times\mathbb{N})^m$, using the lexicographic order used on $\mathbb{R}\times\mathbb{N}$ such that

$$(\rho_1,k_1),\dots,(\rho_m,k_m) \leq (\rho'_1,k'_1),\dots,(\rho'_m,k'_m) \\ \iff \forall i\in\{1,\dots,m\} : (\rho_i,k_i)\leq(\rho'_i,k'_i). \quad \blacktriangleleft$$

Then $((\rho_1,k_1),\dots,(\rho_m,k_m)) \in f_s(n_1,\dots,n_m)$ if and only if

1. There is a path labelled with $a_1^{n_1}a_2^{n_2}\dots a_m^{n_m}$ such that for each $i\in\{1,\dots,m\}$, either:
   - visits k_i+1 SCCs with spectral radius ρ_i whilst reading the character a_i, or
   - the path visits only singleton SCCs (with no loops) whilst reading a_i, in which case $(\rho_i,k_i)=(\delta,0)$.
2. There is no path according to (1) that could be labelled with $(\rho'_1,k'_1),(\rho'_2,k'_2),\dots,(\rho'_m,k'_m)$ and $(\rho_1,k_1),\dots,(\rho_m,k_m) \leq (\rho'_1,k'_1),\dots,(\rho'_m,k'_m)$

Let $\mathcal{D}$ be the set of possible such sequences, i.e. $f_s(n_1,\dots,n_m)\subseteq\mathcal{D}$, observe that the size of $\mathcal{D}$ is less than $(|Q|^2)^m$.

Lemma 6.62. *There exists $c,C\in\mathbb{R}$ such that*

$$c\sum_{X\in f_s(n_1,\dots,n_m)}\prod_{(\rho_i,k_i)\in X}(\rho_i^{n_i}n_i^{k_i}) \\ \leq \nu_s(a_1^{n_1}a_2^{n_2}\dots a_m^{n_m}) \leq \\ C\sum_{X\in f_s(n_1,\dots,n_m)}\prod_{(\rho_i,k_i)\in X}(\rho_i^{n_i}n_i^{k_i}).$$

Proof of Lemma 6.62.

$$\begin{aligned}
\nu_s(a_1^{n_1}a_2^{n_2}\dots a_m^{n_m}) &= (M(a_1)^{n_1}\times M(a_2)^{n_2}\times\cdots\times M(a_m)^{n_m})_{s,t}\\
&= \sum_{q_1\in Q} M(a_1)^{n_1}_{s,q_1}(\times M(a_2)^{n_2}\times\cdots\times M(a_m)^{n_m})_{q_1,t}\\
&\vdots\\
&= \sum_{(q_1,\dots,q_{m-1})\in Q^{m-1}} M(a_1)^{n_1}_{s,q_1}\times M(a_2)^{n_2}_{q_1,q_2}\times\cdots\times M(a_m)^{n_m}_{q_{m-1},t}
\end{aligned}$$

In the case $n_i \geq |Q|$, by Lemma 6.46 in the unary case, for each $M(a_i)^{n_i}_{q_{i-1},q_i}$, there is a $(\rho_{q_{i-1},q_i}, k_{q_{i-1},q_i}), c, C$, such that

$$c\rho^{n_i}_{q_{i-1},q_i} n_i^{k_{q_{i-1},q_i}} \leq M(a_i)^{n_i}_{q_{i-1},q_i} \leq C\rho^{n_i}_{q_{i-1},q_i} n_i^{k_{q_{i-1},q_i}}.$$

Otherwise if $n_i \leq |Q|$, since there are at most $|Q|$ instances it is clear there exists c, C,

$$c\delta^{n_i} \leq M(a_i)^{n_i}_{q_{i-1},q_i} \leq C\delta^{n_i}.$$

Take c, C so that C is maximised over all such C and c is minimised over all such c.

$$\begin{aligned}
&c^{m-1}\sum_{(q_1,\dots,q_{m-1})\in Q^{m-1}} \rho^{n_1}_{s,q_1} n_1^{k_{s,q_1}}\cdot\dots\cdot\rho^{n_m}_{q_{m-1},t} n_m^{k_{q_{m-1},t}}\\
&\leq \nu_s(a_1^{n_1}a_2^{n_2}\dots a_m^{n_m}) \leq\\
&\qquad C^{m-1}\sum_{(q_1,\dots,q_{m-1})\in Q^{m-1}} \rho^{n_1}_{s,q_1} n_1^{k_{s,q_1}}\cdot\dots\cdot\rho^{n_m}_{q_{m-1},t} n_m^{k_{q_{m-1},t}} \qquad (6.7)
\end{aligned}$$

By standard manipulations, any such that if for all i $(\hat{\rho}_i, \hat{k}_i) \leq (\rho_1, k_1)$, then $\hat{\rho}_1^{n_1} n_1^{\hat{k}_1}\cdot\dots\cdot\hat{\rho}_m^{n_m} n_m^{\hat{k}_m} + \rho_1^{n_1} n_1^{k_1}\cdot\dots\cdot\rho_m^{n_m} n_m^{k_m} = \Theta(\rho_1^{n_1} n_1^{k_1}\cdot\dots\cdot\rho_m^{n_m} n_m^{k_m})$ and by sufficient modification of C, c, paths admitting $(\hat{\rho}_1, \hat{k}_1), \dots, (\hat{\rho}_m, \hat{k}_m)$ can be omitted.

Since the sum is finite, any two sums with the same ρ, k values can be reduced to a single one, changing c, C by a factor of two.

The remaining (ρ, k) paths correspond exactly with $f_s(n_1, \dots, n_m)$. □

Given a set of sequences

$$\mathcal{X} = \left\{((\rho^1_1, k^1_1), \dots, (\rho^1_m, k^1_m)), \dots, ((\rho^h_1, k^h_1), \dots, (\rho^h_m, k^h_m))\right\} \subseteq \mathcal{D}.$$

An automaton accepting all $a_1^{n_1}, \dots, a_m^{n_m}$, where $f_s(n_1, \dots, n_m) = \mathcal{X}$, can be constructed. First, let $X_i = ((\rho_1^i, k_1^i), \dots, (\rho_m^i, k_m^i))$ and construct $\mathcal{N}_s^{\geq X_i}$ with the language,

$$L(\mathcal{N}_s^{\geq X_i}) = \left\{ a_1^{n_1}, \dots, a_m^{n_m} \mid {\scriptstyle \exists((\rho'_1,k'_1),\dots,(\rho'_m,k'_m)) \in f_s(n_1,\dots,n_m) \atop \text{such that } ((\rho'_1,k'_1),\dots,(\rho'_m,k'_m)) \geq X_i} \right\}.$$

This is done by tracking for each state the current maximum spectral radius seen and the number of different SCCS with this spectral radius. Permit passage from states reading a_j to states reading a_{j+1} only if this tracked value is at least (ρ_j^i, k_j^i) and states should only be final if the tracked value of a_m is at least (ρ_m^i, k_m^i).

Similarly, by replacing ensuring at least one such inequality is strict, with one extra bit of information, the machine $\mathcal{N}_s^{>X_i}$ can be constructed such that:

$$L(\mathcal{N}_s^{>X_i}) = \left\{ a_1^{n_1}, \dots, a_m^{n_m} \mid {\scriptstyle \exists((\rho'_1,k'_1),\dots,(\rho'_m,k'_m)) \in f_s(n_1,\dots,n_m) \atop \text{such that } ((\rho'_1,k'_1),\dots,(\rho'_m,k'_m)) > X_i} \right\}.$$

Which can then be used to make $\mathcal{N}_s^{X_i}$ with the property

$$L(\mathcal{N}_s^{X_i}) = L(\mathcal{N}_s^{\geq X_i}) \setminus L(\mathcal{N}_s^{>X_i}).$$

Then for the sequence $\mathcal{X} = \{X_1, \dots, X_h\}$, construct $\mathcal{N}_s^{\mathcal{X}}$ such that:

$$L(\mathcal{N}_s^{\mathcal{X}}) = \bigcap_{X_i \in \mathcal{X}} L(\mathcal{N}_s^{X_i}) \cap \bigcap_{X_i \in \mathcal{D} \setminus \mathcal{X}} \overline{L(\mathcal{N}_s^{X_i})}.$$

Finally, given two sets of sequences $\mathcal{X}$ and $\mathcal{Y}$ one can construct $\mathcal{N}_s^{\mathcal{X}} \cap \mathcal{N}_{s'}^{\mathcal{Y}}$ requiring words to satisfy $\mathcal{X}$ from state s, and $\mathcal{Y}$ from s'.

Definition 6.63. $\mathcal{N}_s^{X_i}$ and $\mathcal{N}_s^{\mathcal{X}}$ are called *detector automata.* ◀

6.6.2 The plus-letter-bounded case

The problem will be addressed first when the language is of the form $a_1^+ \dots a_m^+$ with $a_i \neq a_{i+1}$, so that any word can be uniquely identified as $a_1^{n_1} \dots a_m^{n_m}$ using a vector $(n_1, \dots, n_m) \in \mathbb{N}^m$ with $n_i > 0$ for every $i \in [m]$. Note that one can translate this into a machine such that $a_i \neq a_j$ for all $i \neq j$, by relabelling the transitions.

Lemma 6.64. *Given a weighted automaton $\mathcal{W} = \langle Q, \Sigma, M, F \rangle$, $s, s' \in Q$, with $L(NFA_s(\mathcal{W}))$ and $L(NFA_{s'}(\mathcal{W}))$ plus-letter-bounded it is decidable whether $\mathcal{W}, s, s'$ form a positive instance of the big-O problem, subject to Schanuel's conjecture.*

Proof. Again, assume that the language containment condition is satisfied. Then, to violate big-O, there needs to exists a infinite sequence of words such that for all $C > 0$, there exists a word such that $\frac{\nu_s(w)}{\nu_{s'}(w)} > C$. The following will define a procedure to detect if s is big-O of s', described as a non-deterministic procedure with a universal acceptance condition, that is, looking for a single branch to detect s is *not* big-O of s' and if no such branch detects it then s is big-O of s'. Using the language of Lemma 6.62, the procedure will decide if there is a sequence $n : \mathbb{N} \to \mathbb{N}^m$ such that:

$$\frac{\sum_{X \in f_s(n(t)_1,\ldots,n(t)_m)} \prod_{(\rho_i,k_i)\in X} (\rho_i^{n(t)_i} n(t)_i^{k_i})}{\sum_{X' \in f_{s'}(n(t)_1,\ldots,n(t)_m)} \prod_{(\rho_i,k_i)\in X'} (\rho_i^{n(t)_i} n(t)_i^{k_i})} \xrightarrow[t\to\infty]{} \infty.$$

If such a sequence exists, given a property of a word P, then there is either an infinite subsequence satisfying P or an infinite subsequence satisfying its complement $\overline{P}$. Generalising this, given a set of properties $P_1, \ldots, P_k$, such that any word satisfies exactly one of P_i; then there is an infinite subsequence satisfying one choice of P_i. This idea will be used repeatedly to restrict the set of sequence that need to be considered when detecting violations to big-O.

Since the image of f_s and $f_{s'}$ is finite, the search can be restricted to some explicit choice of $f_s(n_1, \ldots, n_m) = \mathcal{X}$ and $f_s(n_1, \ldots, n_m) = \mathcal{Y}$. Let

$$\begin{aligned}
&\mathcal{X} = ((\rho_1^1, k_1^1), \ldots, (\rho_m^1, k_m^1), \ldots, (\rho_1^g, k_1^g), \ldots, (\rho_m^g, k_m^g)), \qquad \text{and}\\
&\mathcal{Y} = ((\sigma_1^1, \ell_1^1), \ldots, (\sigma_m^1, \ell_m^1)), \ldots, ((\sigma_1^h, \ell_1^h), \ldots, (\sigma_m^h, \ell_m^h))
\end{aligned}$$

Now the question asks if there is a sequence $n : \mathbb{N} \to \mathbb{N}^m$ such that $f_s(n(t)_1, \ldots, n(t)_m) = \mathcal{X}$ and $f_{s'}(n(t)_1, \ldots, n(t)_m) = \mathcal{Y}$ and

$$\frac{\sum_{X \in \mathcal{X}} \prod_{(\rho_i,k_i)\in X} (\rho_i^{n(t)_i} n(t)_i^{k_i})}{\sum_{X' \in \mathcal{Y}} \prod_{(\sigma_i,\ell_i)\in X'} (\sigma_i^{n(t)_i} n(t)_i^{\ell_i})} \xrightarrow[t\to\infty]{} \infty$$

But note it would be necessary for one of the summand in the numerator to go to infinity, hence only one choice of $X \in \mathcal{X}$ is required such that $X \in f_s(n_1, \ldots, n_m)$.

$$\begin{aligned}
&X = ((\rho_1, k_1), \ldots, (\rho_m, k_m)), \quad \text{and}\\
&\mathcal{Y} = ((\sigma_1^1, \ell_1^1), \ldots, (\sigma_m^1, \ell_m^1)), \ldots, ((\sigma_1^h, \ell_1^h), \ldots, (\sigma_m^h, \ell_m^h)).
\end{aligned} \tag{6.8}$$

Hence the question amounts to whether there is a sequence $n : \mathbb{N} \to \mathbb{N}^m$ such that $X \in f_s(n(t)_1, \ldots, n(t)_m)$ and $f_{s'}(n(t)_1, \ldots, n(t)_m) = \mathcal{Y}$ and

$$\frac{\prod_{(\rho_i,k_i)\in X} (\rho_i^{n(t)_i} n(t)_i^{k_i})}{\sum_{X' \in \mathcal{Y}} \prod_{(\sigma_i,\ell_i)\in X'} (\sigma_i^{n(t)_i} n(t)_i^{\ell_i})} \xrightarrow[t\to\infty]{} \infty.$$

Thus, non-deterministically, guess such a combination of realisable elements of the form in Equation (6.8). An automaton to capture X and $\mathcal{Y}$ can be built, let $\mathcal{N}$ be a detector automaton such that $L(\mathcal{N}) = L(\mathcal{N}_s^X) \cap L(\mathcal{N}_{s'}^{\mathcal{Y}})$ as per Definition 6.63.

Further, observe that by taking the reciprocal and requiring each of the resulting summands to go to zero, the question asks if there is a sequence $n : \mathbb{N} \to \mathbb{N}^m$ such that $a_1^{n(t)_1} a_2^{n(t)_2} \ldots a_m^{n(t)_m} \in L(\mathcal{N})$ for all t and simultaneously for every $j \in \{1, \ldots, h\}$:

$$\frac{\prod_{(\sigma_i^j, \ell_i^j) \in X'} ((\sigma_i^j)^{n(t)_i} n(t)_i^{\ell_i^j})}{\prod_{(\rho_i, k_i) \in X} (\rho_i^{n(t)_i} n(t)_i^{k_i})} \xrightarrow[t\to\infty]{} 0.$$

Simplifying the equation, so that for all $j \in \{1, \ldots, h\}$ we have:

$$\prod_{i=1}^{m} \left(\frac{\sigma_i^j}{\rho_i}\right)^{n(t)_i} n(t)_i^{\ell_i^j - k_i} \xrightarrow[t\to\infty]{} 0.$$

Then by taking logarithms let $\alpha_{j,i} = \log\left(\frac{\sigma_i^j}{\rho_i}\right)$ and $p_{j,i} = \ell_i^j - k_i$. Now ask if is there a sequence $n : \mathbb{N} \to \mathbb{N}^m$ such that

- $a_1^{n(t)_1} a_2^{n(t)_2} \ldots a_m^{n(t)_m} \in L(\mathcal{N})$ for all t, and
- $\displaystyle\bigwedge_{j=1}^{h} \sum_{i=1}^{m} \alpha_{j,i} \cdot n(t)_i + p_{j,i} \log(n(t)_i) \xrightarrow[t\to\infty]{} -\infty.$

Note that, to go to $-\infty$, it is equivalent to find a sequence such that:

$$\bigwedge_{j=1}^{h} \sum_{i=1}^{m} \alpha_{j,i} \cdot n(t)_i + p_{j,i} \log(n(t)_i) < -t.$$

The condition to characterise s not big-O of s' is currently characterised by a condition on automata and a condition expressible as a logical formula. The next claim will characterise the condition on automata as a condition in logic so that the two conditions can be merged into a single logical condition.

Claim 6.65. *The language of $\mathcal{N}$ can be effectively decomposed as a finite union $\bigcup_j \mathcal{L}_j$, where each $\mathcal{L}_j$ is defined by two vectors $\vec{b} = (a_1, \ldots, a_m), \vec{r} = (b_1, \ldots, b_m) \in \mathbb{N}^m$, such that*

$$\mathcal{L}_j = \{a^{n_1} a^{n_2} \ldots a^{n_m} \ \mid \ \exists \lambda \in \mathbb{N}^m \ s.t. \ \forall i \in [m] \ n_i = b_i + r_i \cdot \lambda_i\}.$$

It is well known that the Parikh image of an NFA, that is, a set of vectors indicating the number of occurrences of each character in each accepted word, is a semi-linear set, which can be described as a finite union of linear sets.

A linear set $\mathcal{L}$ can be described by the base vector $\vec{b} \in \mathbb{N}^m$ and the period vectors $\vec{r^1} \dots \vec{r^s} \in \mathbb{Z}^m$ forming the set $\left\{\vec{b} + \lambda_1 \vec{r^1} + \dots + \lambda_s \vec{r^s} \mid \lambda_1, \dots, \lambda_s \in \mathbb{Z}\right\}$. However since $\mathcal{N}$ is bounded over $a_1^+ a_2^+ \dots a_m^+$ and under the assumption that $a_i \neq a_j$, the Parikh image describes exactly a unique word. Further note Claim 6.65 enables the linear set to be of a particular form, where each $\vec{r^i}$ is a constant multiple of the ith unit vector, enabling a contracted representation.

Proof of Claim 6.65. Consider the machine $\mathcal{N}$, accepting a language which is a subset of $a_1^+ a_2^+ \dots a_m^+$, with any state not reachable from the starting state or not leading to an accepting state removed. To induce a form with the property required property, intersect M with the standard DFA[1] for $a_1^+ a_2^+ \dots a_m^+$, without changing the language.

Hence every state corresponds to reading from exactly one character block of $a_1, a_2, \dots, a_m$. At each state there can be at most two characters enabled, either the character to remain in the current character block, or the character to move to the next. Every state can be labelled as

- *only having transition for* a_i; or
- *also having transition with* a_{i+1}.

Consider all possible choices of automaton formed by restricting $\mathcal{N}$ so that there is a single state which is allowed to transition from a_i to a_{i+1} for each i and any other state which had this property in $\mathcal{N}$ has its a_{i+1} transitions removed (but keeps its a_i transitions). Each such choice corresponds with a partition of the accepting runs of $\mathcal{N}$.

Thus $L(\mathcal{N})$ is the finite union over the languages induced by all such machines. Such machines can further be expressed as a finite union of linear sets in the form prescribed.

Assume $\mathcal{N}_j$ is such a machine with a single state capable of transitioning from a_i to a_{i+1} for each i, and again remove any state not reachable from the starting state or not leading to an accepting state. The part of the machine reading a_i has a single starting state and a single final state, which is a unary NFA when the transitions to a_{i+1} are discarded.

This unary NFA can be converted to Chrobak normal form; the section of $\mathcal{N}_j$ corresponding to a_i can be replaced with this unary NFA, and any accepting state has additionally the transitions for transitioning from a_i to a_{i+1} of the single such state in $\mathcal{N}_j$.

[1]By DFA allow 0 or 1 transition for each character from every state, rather than exactly 1; that is, the transition function may be partial.

Let us repeat the process above for all i, decomposing $\mathcal{N}_j$ into the subsets of languages where there are exactly one state transitioning from a_i to a_{i+1}. Let $\mathcal{N}_j = \bigcup_k \mathcal{N}_{j,k}$, a finite union; where each k corresponds to a selection of accepting states $(q_1, \ldots, q_m)$ with q_l being the accepting state in the Chrobak normal form for a_l.

Consider such an $\mathcal{N}_{j,k}$. The steps spent in each block corresponding to a_i is either formed by the finite path or the a single cycle at the end of the path. If the transition occurs in the finite path then b_i is the length of the path to that transition and r_i is zero. If the transition occurs in the cycle at the end of the path, then b_i is the length of the path to that transition from the start of the path and r_i is the length of the cycle. In $\mathcal{N}_{j,k}$ the time spent in block a_i has no influence on the time spent in a_j for $j \neq i$. Then $L(\mathcal{N}_{j,k}) = \{a^{n_1}a^{n_2}\ldots a^{n_m} | \exists \lambda \in \mathbb{N}^m \text{ s.t. } \forall i \in [m] n_i = b_i + r_i \cdot \lambda_i\}$. The language $L(\mathcal{N})$ is the union over all $L(\mathcal{N}_{j,k})$. ■

Now guess one such set described by $\vec{b}$ and $\vec{r}$; since there are finitely many there must be a subsequence of $n(t)$ conforming to one of them maintaining the unbounded ratio. By expressing each $n(t)_i$ as a choice of $\lambda \in \mathbb{N}^m$, the condition can thus be expressed using the following characterisation:

s is not big-O of s' if and only if, for some choice of $X \in \mathcal{D}$, $\mathcal{Y} \subseteq \mathcal{D}$ and $\vec{b}, \vec{r}$ the following holds:

$$\forall C\ \exists \lambda \in \mathbb{N}^m \quad \bigwedge_j \sum_{i \in [m]} \alpha_{j,i} \cdot (b_i + r_i \cdot \lambda_i) + p_{j,i} \log(b_i + r_i \cdot \lambda_i) < C. \tag{6.9}$$

Next it is argued that discarding offset component $\vec{b}$ and relaxing the restriction of λ from naturals to positive reals maintains the satisfiability of the formula. The advantage here is that this relaxation can be solved with the first order theory of the reals with exponential function; which is decidable subject to Schanuel's conjecture.

Claim 6.66. *Equation* (6.9) *holds if and only if the following holds:*

$$\forall C\ \exists x \in \mathbb{R}^U_{\geq \max_i b_i} \quad \bigwedge_j \sum_{i \in U} \alpha_{j,i} \cdot r_i \cdot x_i + \sum_{i \in U} p_{j,i} \log(x_i) < C \tag{6.10}$$

for some $U \subseteq \{i \in [m] \mid r_i > 0\}$.

Proof. First equivalence of Equation (6.9) is shown with the following logical characterisation:

$$\forall C\ \exists \lambda \in \mathbb{N}^{U}_{\geq \max_i b_i} \bigwedge_j \sum_{i\in U} \alpha_{j,i}\cdot(b_i + r_i\cdot\lambda_i) + p_{j,i}\log(b_i + r_i\cdot\lambda_i) < C \quad (6.11)$$

for some $U \subseteq \{i \in [m] \mid r_i > 0\}$.

First note that Equation (6.11) immediately implies Equation (6.9). It is required to show the converse.

Note that in the sequence $n(t)$ some components may be bounded. Either because $r_i = 0$, or the choice of $n(t)$ makes it so. Suppose there exists a $\theta > 0$ such that $n(t)_x \leq \theta$ for some $x \in [m]$, then $\sum_{i=1}^{m} \alpha_{j,i}\cdot n(t)_i + p_{j,i}\log(n(t)_i) \leq \sum_{i=1, i\neq x}^{m} \alpha_{j,i}\cdot n(t)_i + p_{j,i}\log(n(t)_i) + |\alpha_{j,i}|\cdot\theta + |p_{j,i}|\,\theta$. Hence the sequence $\sum_{i=1,i\neq x}^{m} \alpha_{j,i}\cdot n(t)_i + p_{j,i}\log(n(t)_i)$ goes to $-\infty$ as well.

Consider each choice of components $B \subseteq [m]$ which will be bounded. For some components there will be no choice as $r_{ki} = 0$. Assume that the chosen set is maximal with respect to set-inclusion; that is, there should be no subsequence maintaining the property with fewer components unbounded. Let the remaining unbounded components be $U = [m] \setminus B$.

Since each remaining component is not bounded, there is always a later point in the sequence in which the value is larger; thus one can take a subsequence of $n(t)$ so that $n(t)_i \leq n(t+1)_i$ for every t. Repeat for every remaining component $i \in U$; this can be done as the minimal choice of unbounded components has been selected. Hence, without loss of generality if there exists some sequence, then for any θ, there exists a subsequence of $n(t)$, such that $n(t)_i > \theta$ for all $i \in U$. To enable a more succinct analysis later, restrict $n(t)$ to those in which $\lambda_i \geq \max_i b_{ki}$ where $n(t)_i = b_{ki} + r_{ki}\cdot\lambda_i$ for some λ_i.

The characterisation of Equation (6.11) allows the removal of the base vector $\vec{b}$. Observe that

$$\sum_{i\in U}\alpha_{j,i}\cdot(b_i + r_i\cdot\lambda_i) = \sum_{i\in U}\alpha_{j,i}\cdot b_i + \sum_{i\in U}\alpha_{j,i}\cdot r_i\cdot\lambda_i$$

and that $\sum_{i\in U}\alpha_{j,i}\cdot b_i$ is constant so it does not affect whether the sequence goes to $-\infty$, hence Equation (6.11) holds if and only if :

$$\forall C\ \exists \lambda \in \mathbb{N}^{U}_{\geq \max_i b_i} \bigwedge_j \sum_{i\in U} \alpha_{j,i}\cdot r_i\cdot\lambda_i + p_{j,i}\log(b_i + r_i\cdot\lambda_i) < C. \quad (6.12)$$

The log component can be extracted by using the following rewriting

$$\log(b_i + r_i \cdot \lambda_i) = \log(\lambda_i \cdot (\frac{b_i}{\lambda_i} + r_i)) = \log(\lambda_i) + \log(\frac{b_i}{\lambda_i} + r_i).$$

Since $r_i \geq 1$ and $\lambda_i \geq b_i$ we have $\log(\frac{b_i}{\lambda_i} + r_i) \leq \log(r_i + 1)$, which is constant. Hence Equation (6.11) is equivalent to:

$$\forall C' \; \exists \lambda \in \mathbb{N}^U_{\geq \max_i b_i}$$
$$\bigwedge_j \sum_{i \in U} \alpha_{j,i} \cdot r_i \cdot \lambda_i + \sum_{i \in U} p_{j,i} \log(\lambda_i) < C' \quad (6.13)$$

Clearly a natural assignment (Equation (6.13)) implies a real assignment (Equation (6.10)). Now consider Equation (6.10) holding, and it is required that Equation (6.13) is satisfied, by exhibiting a choice of $\lambda \in \mathbb{N}^U_{\geq \max_i b_i}$ for every C'.

Given $C' < 0$, let $C = C' - \max_j \sum_{i \in U} |\alpha_{j,i}| \, r_i - \max_j \sum_{i \in U} |p_{j,i}|$, and choose $x \in \mathbb{R}^{|U|}_{\geq \max_i b_i}$ satisfying Equation (6.10).

Now let $x_i = \lambda_i + y_i$, with $y_i < 1, \lambda_i = \lfloor x_i \rfloor$. First observe that since $x_i \geq \max_i b_i$, an integer, also $\lambda_i \geq \max_i b_i$.

Observe that $\left|\sum_{i \in U} \alpha_{j,i} \cdot r_i \cdot y_i\right| \leq \sum_{i \in U} |\alpha_{j,i}| \, r_i$. Since

$$\sum_{i \in U} \alpha_{j,i} \cdot r_i \cdot \lambda_i + \sum_{i \in U} \alpha_{j,i} \cdot r_i \cdot y_i + \sum_{i \in U} p_{j,i} \log(\lambda_i + y_i) < C$$

we have

$$\sum_{i \in U} \alpha_{j,i} \cdot r_i \cdot \lambda_i + \sum_{i \in U} p_{j,i} \log(\lambda_i + y_i) < C + \sum_{i \in U} |\alpha_{j,i}| \, r_i$$

Again rewrite $\log(\lambda_i + y_i) = \log(\lambda_i(1 + \frac{y_i}{\lambda_i})) = \log(\lambda_i) + \log(1 + \frac{y_i}{\lambda_i})$. Then since $\lambda_i > y_i$, $\log(1 + \frac{y_i}{\lambda_i}) \leq 1$, so $\left|\sum_{i \in U} p_{j,i} \log(1 + \frac{y_i}{\lambda_i})\right| \leq \sum_{i \in U} |p_{j,i}|$. Thus we have

$$\sum_{i \in U} \alpha_{j,i} \cdot r_i \cdot \lambda_i + \sum_{i \in U} p_{j,i} \log(\lambda_i) < C + \sum_{i \in U} |\alpha_{j,i}| \, r_i + \sum_{i \in U} |p_{j,i}| \leq C'$$

and hence, Equation (6.13) holds. ■

This completes the proof of Lemma 6.64. To check that s is not big-O of s', first, check the language containment condition which may immediately conclude that s is not big-O of s'. It then suffices to check all choices of $X \in \mathcal{D}$, $\mathcal{Y} \subseteq \mathcal{D}$, choice of $\vec{b}, \vec{r}$ defining one of the linear sets making up $L(\mathcal{N}) = L(\mathcal{N}^X_s) \cap L(\mathcal{N}^{\mathcal{Y}}_{s'})$, and choice of $U \subseteq \{i \in [m] \mid r_i > 0\}$ and then verify Equation (6.10) using the first order theory of the reals with exponential function (to encode the logarithm). If any of these choices leads to the formula being satisfied then s is not big-O of s'. Otherwise s is big-O of s'. $\square$

6.6.3 The letter-bounded case

In this section the restriction of the language as $a_1^+ \dots a_m^+$ is relaxed to $a_1^* \dots a_m^*$. Again assume that the language containment condition is satisfied. In this case, great care must be taken in how a word is represented. For example, given a language letter-bounded over $a^*b^*a^*$, the word a^n must refer to all paths representing $a^{n_1}b^0a^{n_2}$ for $n_1 + n_2 = n$.

On a more extreme example a word $a^{n_1}b^{n_2}a^{n_3}$ over the language $a^*b^*a^*b^*a^*$ then the b^{n_2} could refer to either the first or second block of b's, indeed it could refer to both blocks of b's with no central a's. So all of the following decompositions must be dealt with

- $a^{n_1}b^{n_2'}a^0b^{n_2''}a^{n_3}$ such that $n_2 = n_2' + n_2''$; and
- $a^{n_1}b^{n_2}a^{n_3'}b^0a^{n_3''}$ such that $n_3 = n_3' + n_3''$; and
- $a^{n_1'}b^0a^{n_1''}b^{n_2}a^{n_3}$ such that $n_1 = n_1' + n_1''$.

The analysis must account for all such paths, so that when a block of letters is characterised by a (ρ, k)-pair it should capture all contiguous letters, no matter how long is spent in each block.

Lemma 6.67. *The big-O problem for $\mathcal{W}, s, s'$ with $L(NFA_s(\mathcal{W}))$ and $L(NFA_{s'}(\mathcal{W}))$ letter-bounded reduces to the plus-letter-bounded case.*

Proof. Suppose the language containment condition holds and $L(NFA_s(\mathcal{W})) \subseteq L(NFA_{s'}(\mathcal{W})) \subseteq a_1^* \cdots a_m^*$. Let I be the set of strictly increasing sequences $\vec{\imath} = i_1 \cdots i_k$ of integers between 1 and m. Given $\vec{\imath} \in I$, let $\mathcal{W}_{\vec{\imath}}$ be the weighted automaton obtained by intersecting $\mathcal{W}$ with a DFA for $a_{i_1}^+ \cdots a_{i_k}^+$ whose initial state is q. Note that s is big-O of s' (in $\mathcal{W}$) if and only if (s, q) is big-O of (s', q) in $\mathcal{W}_{\vec{\imath}}$ for all $\vec{\imath} \in I$, because $a_1^* \cdots a_m^* = \bigcup_{\vec{\imath} \in I} a_{i_1}^+ \cdots a_{i_k}^+$. Because the big-O problem for each $\mathcal{W}_{\vec{\imath}}$, (s, q), (s', q) falls into the plus-letter-bounded case, it is decidable by Lemma 6.64 and the result follows. $\square$

On each of these new weighted automaton, the analysis of Lemma 6.64 will apply to each block of letters, where each block may refer to more states which referred to more than one set of characters in $a_1^* \dots a_m^*$, after skipping some characters. Then for this more complicated block, Friedland and Schneider can correctly characterise the relevant decompositions. For example on $a^{n_1}b^0a^{n_2}$ for $n_1 + n_2 = n$; if one block of a's dominate the (ρ, k)-choice will correspond to this, but if there is a path going through (ρ, k) in the first block of a's and

(ρ, k') in the second, both dominating, then the k's will be added suitably to account for the additional polynomial factor, i.e. $(\rho, k+k')$.

6.6.4 The bounded case

In this section the big-O problem is considered in the case where $L(\mathit{NFA}_s(\mathcal{W}))$ and $L(\mathit{NFA}_{s'}(\mathcal{W}))$ are bounded, which is a relaxation of letter-boundedness (see Definition 6.58): $L(\mathit{NFA}_s(\mathcal{W}))$ and $L(\mathit{NFA}_{s'}(\mathcal{W}))$ are subsets of $w_1^* \dots w_m^*$ for some $w_1, \dots, w_m \in \Sigma^*$. A reduction to the letter-bounded case (of Section 6.6.3) is shown, entailing Theorem 6.59.

Here the situation is somewhat more complicated again, consider the language $(abab)^*a^*b^*(ab)^*$, then the word $(ab)^4$ can be decomposed in a number of ways; $(abab)^2a^0b^0(ab)^0$, $(abab)^1a^1b^1(ab)^1$, $(abab)^1a^0b^0(ab)^2$, $(abab)^0a^1b^1(ab)^3$, $(abab)^0a^0b^0(ab)^4$. One must be sure that when a word is referred to, its weight is associated with all paths of any different decomposition.

Lemma 6.68. *The big-O problem for $\mathcal{W}, s, s'$ with $L(\mathit{NFA}_s(\mathcal{W}))$ and $L(\mathit{NFA}_{s'}(\mathcal{W}))$ bounded reduces to the letter-bounded case.*

Proof. Let $\mathcal{W} = \langle Q, \Sigma, M, F\rangle$. Then there exists $w_1, \dots, w_m$ such that for all w with $\nu_s(w) > 0$, $w = w_1^{n_1} \dots w_m^{n_m}$ for some $n_1, \dots, n_m \in \mathbb{N}$. Assume that $w_i = b_{i,1}b_{i,2}, \dots, b_{i,|w_i|}$.

Given a word w, there may be multiple paths $\pi_1, \pi_2, \dots$ from s to t respecting that word. Further there may be multiple decomposition vectors $\vec{n}_1, \vec{n}_2, \dots \in \mathbb{N}^m$ such that $\vec{n}_i = (n_1, \dots, n_m)$ and $w = w_1^{n_1} \dots w_m^{n_m}$. The goal will be to construct a weighted automaton $\mathcal{W}'$ with states s and s' letter-bounded over $a_1^* \dots a_m^*$ such that for every word w the weight of $a_1^{n_1} \dots a_m^{n_m}$ in $\mathcal{W}'$ (for every valid decomposition vector $\vec{n} \in \mathbb{N}^m$) is the sum of the weights of all paths $\pi_1, \pi_2, \dots$ respecting w in $\mathcal{W}$. To compute $\mathcal{W}'$, a transducer will be defined and applied to the automaton $\mathcal{W}$.

A non-deterministic finite transducer is an NFA with transitions labelled by pairs from $\Sigma \times (\Sigma' \cup \{\varepsilon\})$. In the construction only edges of this form are required, that is a definition with transitions labelled with ε in the first component (e.g. ε/a) are not considered. A transducer induces a translation $\mathcal{T} : \Sigma^* \to \Sigma'^*$.

Consider the set of regular expressions $w_{i_1}^+ \dots w_{i_{m'}}^+$ each induced by a sequence $(i_1, \dots, i_{m'}) \in \mathbb{N}^{m'}$, $m' \le m$, with $1 \le i_1 < \dots < i_{m'} \le m$. Note that two sequences $(i_1', \dots, i_{m'}')$, $(i_1'', \dots, i_{m''}'')$ may induce the same expression $w_{i_1}^+ \dots w_{i_m}^+$ in which case one need not consider more than one. The transducer $\mathcal{T}$ will be defined as follows.

For each such sequence $I = (i_1, \dots, i_{m'})$ build the following automaton. For each i_j, construct the following section, which simply reads the word:

$$f_j^I \xrightarrow{b_{i_j,1}} s_j^I \xrightarrow{b_{i_j,2}} \cdot \xrightarrow{b_{i_j,3}} \cdot \dots \cdot \xrightarrow{b_{i_j,|w_i|-1}} e_j^I.$$

Then on the final character, non-deterministically restart, or move to the next word, emitting a character representing the word:

$$e_j^I \xrightarrow{b_{i_j,\left|w_{i_j}\right|}/a_{i_j}} f_j^I \quad \text{and} \quad e_j^I \xrightarrow{b_{i_j,\left|w_{i_j}\right|}/a_{i_j}} f_{j+1}^I$$

The transducer $\mathcal{T}$ is the union across all the transducers for each sequence. For the global start state, non-deterministically move to f_1^I for each I. To avoid ε transitions, duplicate the first transition $f_1^I \xrightarrow{x} s_1^I$ with $q_0 \xrightarrow{x} s_1^I$ for a global start state q_0. Observe the valid output sequences are $(\varepsilon^* a_1)^*(\varepsilon^* a_2)^* \dots (\varepsilon^* a_m)^*$. However there can be a finite number of ε's in a row; at most $r = \max_{i\in[m]} |w_i| - 1$.

Assume $\mathcal{W} = \langle Q, \Sigma, M, \{t\}\rangle$ and $\mathcal{T} = \langle Q', \Sigma \times (\Sigma' \cup \{\varepsilon\}), \rightarrow, q_0\rangle$. Then construct the weighted automaton $\mathcal{T}(\mathcal{W}) = \langle Q \times Q', \Sigma', M^{\mathcal{T}}, \{t\} \times Q'\rangle$ using a product construction. The probability is associated in the following way $M^{\mathcal{T}}(a)((s,q),(s',q')) = p$ if there is a transition $q \xrightarrow{b/a} q'$ in $\mathcal{T}$ and $s \xrightarrow[b]{p} s'$ in $\mathcal{W}$. Note that, by this definition, there is a matrix $M^{\mathcal{T}}(\varepsilon)$; however, in every run of $\mathcal{T}(\mathcal{W})$ at most r many ε's in a row are produced, where $r = \max_{i\in[m]} |w_i| - 1$.

Now let $\mathcal{W}'$ be a copy of $\mathcal{T}(\mathcal{W})$ with ε removed: $M'(a_i) = (\sum_{x=0}^{r} M^{\mathcal{T}}(\varepsilon)^x) M^{\mathcal{T}}(a^i)$. Then $\nu_{\mathcal{W}}(w) = \nu_{\mathcal{W}'}(a_1^{n_1} \dots a_m^{n_m})$ for all $n_1, \dots, n_m$ such that $w = w_1^{n_1} \dots w_m^{n_m}$. Hence $\mathcal{W}'$ is a weighted automaton with letter-bounded languages from (s, q_0) and (s', q_0) such that (s, q_0) is big-O of (s', q_0) in $\mathcal{W}'$ if and only if s is big-O of s' in $\mathcal{W}$, which can be tested using Lemma 6.67. $\square$

6.7 Bisimilarity distances for ε

As discussed in Section 2.4.1, Chatzikokolakis et al. [CGPX14] defines a bisimilarity distance as an upper bound on ε, which they called $bm_{\otimes}$. To avoid confusion with the use of $\otimes$ here as the symmetric ratio variation distance, here the same distance is called $bd_{\ln}$. The distance was considered in the context of labelled concurrent Markov chains, here the definition is restricted to labelled Markov chains (state-labelled and infinite-word). Recalling their definition, recast in the notation of this thesis, let $bd_{\ln} : Q \times Q \to [0, \infty]$ be the least fixed point of $\Gamma_{\ln} : [0, \infty]^{Q\times Q} \to [0, \infty]^{Q\times Q}$ defined as

$$\Gamma_{\ln}(d)(s, s) = \begin{cases} K_{\ln}(d)(\mu_s, \mu_s) & \text{if } \ell(s) = \ell(s') \\ \infty & \text{if } \ell(s) \neq \ell(s') \end{cases}$$

where

$$K_{\ln}(d)(\mu, \mu') = \sup_{\substack{f:Q\to[0,1] \\ d_{\ln}(f(q),f(q'))\leq d(q,q')}} d_{\ln}\left(\int f d\mu, \int f d\mu'\right),$$

using the metric $d_{\ln}(x, x') = |\ln(x) - \ln(x')|$. This distance $bd_{\ln}$ serves as an upper bound on the multiplicative total variation distance $tv_{\ln}(s, s')$.

This section considers relevant simplifications, recasts the bisimilarity distance using the distance function $d_{\otimes}(x, y) = \max\{x/y, y/x\}$ directly and shows that the threshold problem on this new distance, $bd_{\otimes}$, can be computed in **PSPACE**; matching a similar early result for the approximation of the classical bisimilarity distance on labelled Markov chains [BSW07].

6.7.1 Dual form and simplification

In the case where μ, μ' are finite distributions, [CGPX14; Xu15] present a dual form characterisation of $K_{\ln}(d)(\mu, \mu')$, being the optimal value of the following optimisation problem. This is nearly a linear program, if the values of $e^{m_{i,j}}$ are considered to be rational, except that the objective function contains ln. It is equivalent to solve the linear program optimising over z and then take the logarithm of the optimal value of z.

Presentation of [CGPX14]: Linear Program:

$$\min_{\omega\in[0,1]^{Q\times Q},\gamma\in[0,1]^Q,z\in\mathbb{R}_+} \ln(z)$$

Subject to:

$$\forall i : \sum_j \omega_{i,j} - \gamma_i = \mu_i$$

$$\forall j : \sum_i \omega_{i,j} e^{m_{i,j}} - \gamma_j \leq z\mu'_j$$

$$\min_{\omega\in[0,1]^{Q\times Q},\gamma\in[0,1]^Q,z\in\mathbb{R}_+} z$$

Subject to:

$$\forall i : \sum_j \omega_{i,j} - \gamma_i = \mu_i$$

$$\forall j : \sum_i \omega_{i,j} e^{m_{i,j}} - \gamma_j \leq z\mu'_j$$

This can be interpreted as an optimal transportation problem. For each $s \in Q$ there are two locations, a factory and a destination.

Each destination i requires exactly μ_i produce at the end of the transportation. Destination i receives $\sum_j \omega_{i,j}$ produce, with $\omega_{i,j}$ coming from factory j. Additionally destination i can send produce back to its own factory for redistribution, this is free and represented by γ_i.

When a factory sends produce, some of the produce gets lost on the way so to be sure $\omega_{i,j}$ is received at destination i from factory j it has to send $\omega_{i,j}e^{m_{i,j}}$, where $e^{m_{i,j}}$ represents the percentage overproduction required. The standard production for a factory i is μ'_i, the cost of the transportation is the percentage overproduction required z for the worst case factory.

Note that, in the case m is a pseudometric, this can actually be optimised further, there is never a requirement to send produce back from destination to factory for redistribution, that is, all γ_i are zero in some optimal solution and therefore can be removed.

Lemma 6.69. *Assume $m : Q\times Q \to [0,\infty]$ is a pseudo-metric, then the following linear programs have the same optimal value:*

$$\min_{\omega\in[0,1]^{Q\times Q},\gamma\in[0,1]^Q,z\in\mathbb{R}_+} \ln(z)$$

Subject to:

$$\forall i : \sum_j \omega_{i,j} - \gamma_i = \mu_i \qquad (1)$$

$$\forall j : \sum_i \omega_{i,j} e^{m_{i,j}} - \gamma_j \leq z\mu'_j \qquad (2)$$

$$\min_{\omega\in[0,1]^{Q\times Q},z\in\mathbb{R}_+} \ln(z)$$

Subject to:

$$\forall i : \sum_j \omega_{i,j} = \mu_i$$

$$\forall j : \sum_i \omega_{i,j} e^{m_{i,j}} \leq z\mu'_j$$

Proof. The proof shows that γ can be set to 0 in the optimal solution to the first linear program. Suppose amongst all optimal solutions, choose the solution such that $\sum_i \gamma_i$ is minimised. When $\sum_i \gamma_i > 0$ a contradiction will be obtained, by exhibiting another optimal solution with smaller $\sum_i \gamma_i$, contradicting minimality. Thus, concluding $\sum_i \gamma_i = 0$ in some optimal solution.

Suppose $\gamma_j > 0$. Then there is some path $\mu'_i \xrightarrow{\omega_{j,i}} \mu_j \xrightarrow{\gamma_j} \mu'_j \xrightarrow{\omega_{k,j}} \mu_k$, with positive flow. Let $x > 0$ be the minimal flow on any section of this path, so we have $\gamma_j \geq x, \omega_{j,i} \geq x$ and $\omega_{k,j} \geq \frac{x}{e^{m_{k,j}}}$.

Define a new assignment $\omega' \in [0,1]^{Q\times Q}, \gamma' \in [0,1]^{Q}$, with $\omega = \omega'$ and $\gamma = \gamma'$ except that:

$$\gamma'_j = \gamma_j - x \qquad \omega'_{j,i} = \omega_{j,i} - x \qquad \omega'_{k,j} = \omega_{k,j} - \frac{x}{e^{m_{k,j}}} \qquad \omega'_{k,i} = \omega_{k,i} + \frac{x}{e^{m_{k,j}}}.$$

It is necessary to verify the constraints hold whenever there has been a change:

- **Constraint (2) at i:** By satisfiability of the initial linear program, we have $\sum_a \omega_{a,i} e^{m_{a,i}} - \gamma_i \leq z\mu'_i$ and extracting the relevant variables for some N we have $\omega_{j,i}e^{m_{j,i}} + \omega_{k,i}e^{m_{k,i}} + N = z\mu'_i$.

  By triangle inequality we have $m_{k,i} \leq m_{k,j} + m_{j,i}$, hence $e^{m_{k,i}} \leq e^{m_{k,j}+m_{j,i}}$ and $\frac{e^{m_{k,i}}}{e^{m_{k,j}}} \leq e^{m_{j,i}}$. Thus:

$$\begin{aligned}\sum_a \omega'_{a,i} e^{m_{a,i}} - \gamma_i &= \omega'_{k,i}e^{m_{k,i}} + \omega'_{j,i}e^{m_{j,i}} + N\\ &= (\omega_{k,i} + \frac{x}{e^{m_{k,j}}})e^{m_{k,i}} + (\omega_{j,i} - x)e^{m_{j,i}} + N\\ &= \frac{x}{e^{m_{k,j}}}e^{m_{k,i}} - xe^{m_{j,i}} + \omega_{k,i}e^{m_{k,i}} + \omega_{j,i}e^{m_{j,i}} + N\\ &\leq xe^{m_{j,i}} - xe^{m_{j,i}} + \omega_{k,i}e^{m_{k,i}} + \omega_{j,i}e^{m_{j,i}} + N\\ &= \omega_{k,i}e^{m_{k,i}} + \omega_{j,i}e^{m_{j,i}} + N\\ &= z\mu'_i.\end{aligned}$$

- **Constraint (2) at j:** By satisfiability of the initial linear program, we have $\sum_a \omega_{a,j} e^{m_{a,j}} - \gamma_j = z\mu'_j$ and extracting the relevant variables for some N we have $\omega_{k,j}e^{m_{k,j}} - \gamma_j + N \leq z\mu'_j$. Thus:

$$\begin{aligned}\sum_a \omega'_{a,i} e^{m_{a,i}} - \gamma_i &= \omega'_{k,j}e^{m_{k,j}} - \gamma'_j + N\\ &= (\omega_{k,j} - \frac{x}{e^{m_{k,j}}})e^{m_{k,j}} - (\gamma_j - x) + N\\ &= \omega_{k,j}e^{m_{k,j}} - x - \gamma_j + x + N\\ &= \omega_{k,j}e^{m_{k,j}} - \gamma_j + N\\ &= z\mu'_j.\end{aligned}$$

- **Constraint (2) at $b \neq i, j$:** There is no change of ω, γ to ω', γ' relevant to $\sum_a \omega_{a,b}e^{m_{a,b}} - \gamma_b \leq z\mu'_b$ and $\sum_a \omega'_{a,b}e^{m_{a,b}} - \gamma'_b \leq z\mu'_b$.
- **Constraint (1) at j:** At $\sum_a \omega_{j,a} - \gamma_j = \mu_j$ we have $\omega'_{j,i}$ decrease by x complemented by γ'_i decreased by x.
- **Constraint (1) at k:** At $\sum_a \omega_{k,a} - \gamma_k = \mu_k$ we have $\omega'_{k,i}$ increase by $\frac{x}{e^{m_{k,j}}}$ complemented by $\omega'_{k,j}$ decrease by $\frac{x}{e^{m_{k,j}}}$.

- **Constraint (1) at $b \neq j, k$:** There is no change of ω, γ to ω', γ' relevant to $\sum_a \omega_{b,a} - \gamma_b = \mu_b$ and $\sum_a \omega'_{k,a} - \gamma'_k = \mu_k$.

Hence all constraints hold but γ_j is smaller. Contradiction. $\square$

6.7.2 Computing $bd_{\ln}$

The distance $d_{\ln}(x, y)$ is of particular interest as it can be directly plugged into the frame work of generalised liftings, due to $d_{\ln}(x, y)$ being a metric. However the computation of $bd_{\ln}$ was not considered. In particular, it does not lend itself to computation due to the exponential function and the necessity that the value may be infinity.

Consider the threshold problem $\mathrm{BD}_{\ln}$THRESHOLD:

Definition 6.70 ($\mathrm{BD}_{\ln}$THRESHOLD).

INPUT	An LMC $\mathcal{M}$, states $s, s' \in Q$ and a threshold $\theta \in [0, \infty) \cap \mathbb{Q}$
OUTPUT	is $bd_{\ln}(s, s') \leq \theta$? ◀

One approach would be to reuse the approach to solving LD_α-THRESHOLD and BD_α-THRESHOLD by encoding the problem into a decidable logic. In contrast to the estimators of δ, where the answer is in the range $[0, 1]$, the estimate of ε can be infinity. Since infinity cannot be handled directly, the formula uses a variable, $b_{i,j}$ which is either 0 or 1, where 1 indicates that $d_{i,j}$ should be treated as ∞ and otherwise as a real number.

The natural encoding, as demonstrated in Figure 6.9, would use first order theory of reals with exponential function (that is, the same logic used to show conditional decidability of the big-O problem in the bounded language case). This logic is only known to be quasi-decidable [FRZ11], or decidable subject to Schanuel's conjecture [MW96]. In the next section this conditional decidability is strengthened by removing the exponential function.

6.7.3 A direct approach avoiding exponentiation and logarithms

The previous section required the use of exponential function, however in this section it can be seen that a similar problem is fully decidable by studying a small modification to the problem. To do this the ratio is considered directly using the distance $d_\otimes(x, y) = \max\{x/y, y/x\}$. This can then be dealt with in much the same way as Δ_α or $d_{\ln}$.

Remark. $d_\otimes$ is not a metric, nor a pseudometric, not least because $d_\otimes(x, x) = 1$ rather than 0. ◀

$$\begin{aligned}
&\textsc{BD}_{\ln}\textsc{Threshold}(s,s',\theta) = \\
&\qquad \exists (d_{i,j})_{i,j\in Q} \quad \bigwedge_{i,j\in Q} (0 \le d_{i,j} \le 1 \land d_{i,j} = d_{j,i}) \\
&\qquad\quad \land\ \exists (b_{i,j})_{i,j\in Q} \quad \bigwedge_{i,j\in Q} (b_{i,j} = 0 \lor b_{i,j} = 1) \land b_{i,j} = b_{j,i} \\
&\qquad\quad \land\ \mathit{labelconstraint}(d,b)\ \land\ d_{s,s'} \le \theta\ \land\ b_{s,s'} = 0
\end{aligned}$$

$$\begin{aligned}
&\mathit{labelconstraint}(d,b) = \\
&\qquad \bigwedge_{q,q'\in Q} \begin{cases} b_{q,q'} = 1 & \text{if } \ell(q) \ne \ell(q') \\ \mathit{couplingconstraint}(d,b,q,q',q,q') & \text{if } \ell(q) = \ell(q') \\ \qquad \land\ \mathit{couplingconstraint}(d,b,q,q',q',q) & \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathit{couplingconstraint}(d,b,x,x',q,q') = \\
&\qquad b_{x,x'} = 1\ \lor \exists (\omega_{i,j})_{i,j\in Q}\quad \exists z \qquad z \le \exp(d_{x,x'}) \\
&\qquad\qquad\qquad \land \bigwedge_{i,j\in Q} 0 \le \omega_{i,j} \le 1 \quad \land \quad \bigwedge_{i\in Q} \sum_{j\in Q} \omega_{i,j} = \mu_q(i) \\
&\qquad\qquad\qquad \land \bigwedge_{j\in Q} \Big(\sum_{i\in Q} \omega_{i,j} \cdot \exp(d_{i,j}) = z\mu_{q'}(j) \\
&\qquad\qquad\qquad\qquad\qquad \land \bigwedge_{i\in Q} \omega_{i,j} > 0 \implies b_{i,j} = 0) \Big)
\end{aligned}$$

Figure 6.9: Logical formulation for $\textsc{BD}_{\ln}\textsc{Threshold}$.

One can also define a ratio based Kantorovich distance; by replacing $d_{\ln}$ with $d_{\otimes}$:

$$K_{\otimes}(d)(\mu,\mu') = \sup_{\substack{f:Q\to[0,1] \\ \forall q,q'\in Q\ \ d_{\otimes}(f(q)),f(q'))\le d(q,q')}} d_{\otimes}\left(\int f d\mu, \int f d\mu' \right).$$

In the dual form, and on finite distributions μ, μ', the distance can be computed using the following linear program:

$$\min_{\omega\in[0,1]^{Q\times Q},\ z\in\mathbb{R}_+} z \qquad \text{subject to:} \qquad \begin{array}{l} \forall i \in Q : \sum_{j\in Q} \omega_{i,j} = \mu_i \\ \forall j \in Q : \sum_{i\in Q} \omega_{i,j} m_{i,j} \le z\mu'_j \end{array}$$

Then let $\Gamma_{\otimes}(d)(s,s') = \begin{cases} K_{\otimes}(d)(\mu_s,\mu_{s'}) & \text{if } \ell(s) = \ell(s') \\ \infty & \text{otherwise} \end{cases}$
and let $bd_{\otimes}$ to be the least fixed point of $\Gamma_{\otimes}(d)$. The following lemma shows that the two distances are related; exactly by taking the exponential function on the distance of [CGPX14]. This directly entails $tv_{\otimes}(s,s') \le bd_{\otimes}(s,s')$, using

the result that $tv_{\ln}(s,s') \leq bd_{\ln}(s,s')$ of Chatzikokolakis et al. [CGPX14].

Lemma 6.71. *$bd_{\otimes}(s,s') = exp(bd_{ln}(s,s'))$*

Proof. Clearly when $\ell(s) \neq \ell(s')$ both distances are ∞. Therefore, assume $\ell(s) = \ell(s')$.

Let d be some distance function such that $K_{\ln}(d)(\mu_s, \mu_{s'}) \sqsubseteq d$ and $m(s,s') = e^{d(s,s')}$. Suppose $f : Q \to [0,1]$ then $\forall q, q' \in Q$:

$$\begin{aligned}
&d_{\ln}(f(q), f(q')) \leq d(q,q')\\
&\iff \left|\ln(f(q)) - \ln(f(q'))\right| \leq d(q,q')\\
&\iff \max\left\{\ln(f(q)) - \ln(f(q')), \ln(f(q')) - \ln(f(q))\right\} \leq d(q,q')\\
&\iff \exp(\max\left\{\ln(f(q)) - \ln(f(q')), \ln(f(q')) - \ln(f(q))\right\}) \leq exp(d(q,q'))\\
&\iff \max\left\{\exp(\ln(f(q)) - \ln(f(q'))), \exp(\ln(f(q')) - \ln(f(q)))\right\} \leq exp(d(q,q'))\\
&\iff \max\left\{\exp(\ln(f(q)/f(q'))), \exp(\ln(f(q')/f(q)))\right\} \leq m(q,q')\\
&\iff \max\left\{f(q)/f(q'), f(q')/f(q)\right\} \leq m(q,q')\\
&\iff d_{\otimes}(f(q), f(q')) \leq m(q,q')
\end{aligned}$$

Hence the set of function f satisfying the restriction in each supremum are the same. Let F be this set of function for s, s'.

Claim 6.72.

$$\sup_{f\in F} d_{ln}\left(\int f d\mu, \int f d\mu'\right) \leq d(s,s') \iff \sup_{f\in F} d_{\otimes}\left(\int f d\mu, \int f d\mu'\right) \leq m(s,s').$$

Proof.

$$\begin{aligned}
\sup_{f\in F} d_{\ln}\left(\int f d\mu, \int f d\mu'\right) &\leq d(s,s')\\
\sup_{f\in F}\left|\ln\left(\int f d\mu\right) - \ln\left(\int f d\mu'\right)\right| &\leq d(s,s')\\
\exp\left(\sup_{f\in F}\max\left\{\ln\left(\frac{\int f d\mu}{\int f d\mu'}\right), \ln\left(\frac{\int f d\mu'}{\int f d\mu}\right)\right\}\right) &\leq exp\left(d(s,s')\right)\\
\sup_{f\in F}\max\left\{\frac{\int f d\mu}{\int f d\mu'}, \frac{\int f d\mu'}{\int f d\mu}\right\} &\leq m(s,s')\\
\sup_{f\in F} d_{\otimes}\left(\int f d\mu, \int f d\mu'\right) &\leq m(s,s') \qquad \blacksquare
\end{aligned}$$

Therefore concluding that if d is a fixed point of $\Gamma_{\ln}$ then m is a fixed point of $\Gamma_{\otimes}$ and that if m is a fixed point of $\Gamma_{\otimes}$ then d is a fixed point of $\Gamma_{\ln}$. Since exp is a monotone function, then d is the *least* fixed point of $\Gamma_{\ln}$ if and only if m is the *least* fixed point of $\Gamma_{\otimes}$. Yielding, $bd_{\otimes}(s,s') = exp(bd_{\ln}(s,s'))$. $\square$

6.7.4 Computing $bd_{\otimes}$

Consider the relevant threshold problem:

Definition 6.73 ($\text{BD}_{\otimes}$THRESHOLD).

INPUT	An LMC $\mathcal{M}$, states $s, s' \in Q$ and a threshold $\theta \in [0, \infty) \cap \mathbb{Q}$
OUTPUT	is $bd_{\otimes}(s, s') \leq \theta$? ◀

It is easy to see that the $\text{BD}_{\otimes}$THRESHOLD problem can be encoded in the existential first order theory of the reals using a formula of polynomial size, as demonstrated in Figure 6.10 using a technique similar to van Breugel, Sharma and Worrell [BSW07] for encoding the standard bisimilarity distance. However, this encoding does not require the exponential function and so falls within the first order theory of the reals. This theory can be decided in **PSPACE** [Can88; Ren92], entailing the following theorem and matching the complexity attained by van Breugel et al. [BSW07] to approximate the standard bisimilarity distance.

Theorem 6.74. $\text{BD}_{\otimes}$THRESHOLD *is in* **PSPACE**.

An approximation of the value can then be found by a variation on binary search. First check there is a solution not requiring infinity, i.e. such that $b_{s,s'} = 0$ but with no constraint on $d_{s,s'}$. Once this is established, the value may be arbitrarily large. First find k such that $d_{s,s'} \leq 2^k$, in k steps. Binary search can then be conducted on the interval $[2^{k-1}, 2^k]$, up to arbitrary precision. The result remains in polynomial space with respect to the output (but not necessarily the input). This is because $\lceil d_{s,s'} \rceil$ may be arbitrarily large.

This technique falls short of the later techniques to compute the standard bisimilarity distance exactly in polynomial time, and the techniques of Chapters 4 and 5 which compute bisimilarity distances in polynomial time with an **NP** oracle. Both of these techniques, at minimum, relied on the polynomial size representation of relevant variables. Note also the formula does not fall into LRA due to the presence of $\omega_{i,j} \cdot d_{i,j}$ in the coupling condition. Thus it remains open whether there are bounds on the size of $d_{s,s'}$, or whether the threshold problem, or the exact computation, can be computed in a complexity class below **PSPACE**.

6.7.5 Looking for a unique fixed point

Chatzikokolakis et al. [CGPX14] emphasised that their resulting bisimilarity pseudometric respected that distance zero corresponded exactly with bisimilarity, i.e. $bd_{\ln}(s, s') = 0 \iff s \sim s'$. Since $bd_{\otimes}(s, s')$ is the exponential of $bd_{\ln}(s, s')$ one can conclude $bd_{\otimes}(s, s') = 1 \iff s \sim s'$. One could ask whether

$$
\begin{aligned}
&\mathrm{BD}_{\otimes}\textsc{Threshold}(s,s',\theta) = \\
&\qquad \exists (d_{i,j})_{i,j\in Q} \quad \bigwedge_{i,j\in Q} (0 \le d_{i,j} \le 1 \wedge d_{i,j} = d_{j,i}) \\
&\qquad \wedge\ \exists (b_{i,j})_{i,j\in Q} \quad \bigwedge_{i,j\in Q} (b_{i,j} = 0 \vee b_{i,j} = 1) \wedge b_{i,j} = b_{j,i} \\
&\qquad \wedge\ \mathit{labels}(d,b) \quad \wedge \quad d_{s,s'} \le \theta \quad \wedge \quad b_{s,s'} = 0
\end{aligned}
$$

$$
\mathit{labels}(d,b) = \bigwedge_{q,q'\in Q} \begin{cases} b_{q,q'} = 1 & \text{if } \ell(q) \ne \ell(q') \\ \mathit{couplings}(d,b,q,q',q,q') & \text{if } \ell(q) = \ell(q') \\ \qquad \wedge\ \mathit{couplings}(d,b,q,q',q',q) & \end{cases}
$$

$$
\begin{aligned}
&\mathit{couplings}(d,b,x,x',q,q') = b_{x,x'} = 1 \vee \\
&\quad \Big(\exists (\omega_{i,j})_{i,j\in Q} \quad \exists z \qquad z \le d_{x,x'} \\
&\quad \wedge \bigwedge_{i,j\in Q} (0 \le \omega_{i,j} \le 1) \\
&\quad \wedge \bigwedge_{j\in Q} \left(\sum_{i\in Q} \omega_{i,j} \cdot d_{i,j} \le z\mu_{q'}(j) \wedge \bigwedge_{i\in Q} (\omega_{i,j} > 0 \implies b_{i,j} = 0) \right) \\
&\quad \wedge \bigwedge_{i\in Q} \sum_{j\in Q} \omega_{i,j} = \mu_q(i) \Big)
\end{aligned}
$$

Figure 6.10: Logical formulation for $\mathrm{BD}_{\otimes}\textsc{Threshold}$.

fixing $\Gamma_{\otimes}$ at the kernel results in a unique fixed point, in the same way as happens for the standard bisimilarity distance. Then one may hope this would lead to a polynomial algorithm for computing $\mathit{bd}_{\otimes}$ using a linear program for the greatest fixed point. Lemma 6.75 shows that this strategy is not enough.

To that end, one can redefine $\mathit{bd}_{\otimes} : Q \times Q \to [0,\infty]$ to be the least fixed point of

$$
\Gamma'_{\otimes}(d) = \begin{cases} 1 & \text{if } s \sim s' \\ K_{\otimes}(d)(\mu_s,\mu_{s'}) & \text{if } \ell(s) = \ell(s') \\ \infty & \text{otherwise} \end{cases} .
$$

Lemma 6.75. $\Gamma'_{\otimes}$ *does not have a unique fixed point.*

Proof. The proof proceeds by showing that the following degenerate distance

$$d(s, s') = \begin{cases} 1 & \text{if } s \sim s' \\ \infty & \text{otherwise} \end{cases}$$

is always a fixed point.

Consider $\Gamma'_\otimes(d)$, if $s \sim s'$ both $\Gamma'_\otimes(d)(s, s')$ and $d(s, s')$ are fixed to 1. Similarly if $\ell(s) \neq \ell(s')$, both are fixed to ∞.

Then consider $s \not\sim s'$ and $\ell(s) = \ell(s')$, so then

$$\Gamma_\otimes(d)(s, s') = \min_{\substack{\omega \in [0,1]^{Q \times Q},\ z \in \mathbb{R}_+ \\ \forall i: \sum_j \omega_{i,j} = \mu_s(i) \\ \forall j: \sum_i \omega_{i,j} d_{i,j} \leq z \mu_{s'}(j)}} z.$$

Consider the pairs i, j, where $\omega_{i,j} > 0$. Either some of them have $d_{i,j} = \infty$ or only edges with $i \sim j$ are used. If any edge with $d_{i,j} = \infty$, then $z = \infty$ and thus $\Gamma_\otimes(d)(s, s') = \infty$. If only $i \sim j$ edges are used, then $z = 1$, and the allocation ω actually forms a proper coupling and implying that $s \sim s'$ [CBW12, Proposition 9] which is a contradiction. So some edges with $d_{i,j} = \infty$ must be used so $z = \infty$.

Hence d is always a fixed point, indeed the greatest fixed point, which is not unique, as in general there is a smaller, useful, least fixed point as demonstrated by Chatzikokolakis et al. [CGPX14]. □

6.8 Conclusion

The problems relating to the big-O problem and the ratio variation distances are, in full generality, undecidable and, unlike for the standard and skewed total variation distances, also inapproximable. This chapter also has implication for the skewed total variation distance of Chapter 5, for which the threshold problem corresponds to the problem of whether $lv_\alpha(s, s') = 0$; implying that this is also undecidable (in contrast to deciding whether $tv(s, s') = 0$, which is decidable). This implies it is not possible to build a bisimilarity distance operator, in the style of Γ_α^Λ which fully captures the zero distance, and some under-approximation (such as $\sim_\alpha$) is required.

Despite the undecidability results, there are positive results for some restricted classes. However, even for automata with letter-bounded languages, the result is conditional on a conjecture from number theory. This leaves open the barrier between decidability and undecidability; in particular, is it (unconditionally) decidable for automata with bounded language, or other classes of weighted automata (such as leaktight probabilistic automata [FGO12]).

Chatzikokolakis et al. [CGPX14] defined the relevant bisimilarity pseudometric for ε, but only considered the complexity of computing the related Kantorovich distance, and not the bisimilarity distance. The question is partially answered here, showing that the threshold problem is in **PSPACE**, providing one is satisfied to instead consider the exponential of the distance, which for practical purposes should be around $1 + \varepsilon$. Whether the distance can be computed in polynomial time, or classified in a complexity class below **PSPACE**, is left as an open question; but the results show that technique of Chen et al. [CBW12], that is, to capture a unique fixed point, will not be sufficient.

Chapter 7

Verifying Differential Privacy in Circuits

This chapter considers the computational complexity of determining whether programs satisfy (ε, δ)-differential privacy. As demonstrated on labelled Markov chains and more generally, the problem is undecidable, and hence the restriction to probabilistic straight line programs is considered, which are part of any reasonable programming language supporting random computations. Equivalently, randomised circuits are considered. The latter are Boolean circuits with input nodes corresponding both to input bits and to uniformly random bits ("coin flips") where the latter allow the circuit to behave probabilistically (see Figure 7.1). Both decision and approximation versions of the problem are considered, where in the case of decision the input consists of a randomised circuit and parameters ε, δ and in the case of approximation the input is a randomised circuit, the desired approximation precision, and one of the two parameters ε, δ. In both cases, complexity is measured as function of the total input length in bits.

Firstly, Theorem 7.9 shows that determining whether a randomised circuit is ε-differentially private is $\mathbf{coNP}^{\#\mathbf{P}}$-complete. To show hardness in $\mathbf{coNP}^{\#\mathbf{P}}$, the complement to the problem E-Maj-Sat [LGM98] is considered, which is complete for $\mathbf{NP}^{\#\mathbf{P}}$ [CDM17].

Turning to the case where $\delta > 0$, this work shows that determining whether a randomised circuit is (ε, δ)-differentially private is $\mathbf{coNP}^{\#\mathbf{P}}$-complete when the number of output bits is small relative to the total size of the circuit and otherwise between $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ (Theorem 7.25).

This work is also motivated by the work by Murtagh and Vadhan [MV16] studying the complexity of optimally compose differentially private algorithms. It is known that the composition of differentially private computations also satisfies differential privacy [DMNS06; DRV10; MV16]. Consider the composition of k differentially private algorithms with privacy parameters $(\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k)$. The resulting program is $(\varepsilon_g, \delta_g)$-differentially private for a multitude of possible $(\varepsilon_g, \delta_g)$ pairs. Murtagh and Vadhan showed that determining the minimal ε_g given δ_g is $\#\mathbf{P}$-complete [MV16]. They also give an efficient, i.e. polynomial time, approximation algorithm that computes ε_g to arbitrary accuracy, giving hope that for “simple” programs deciding differential privacy or approximating of privacy parameters may be tractable. Hence, one might expect the existence of polynomial time algorithms to approximate ε or δ. The results show that this is not the case, by showing this is **NP**-hard and **coNP**-hard, and therefore an efficient algorithm does not exist, unless $\mathbf{P} = \mathbf{NP}$ (Theorem 7.31).

This model is considered because it corresponds to (randomised) straight line programs, i.e., programs with fixed input size, with access to random bits, and with no loops. Notice also that any randomised computation running in polynomial time $p(n)$ can be translated to a sequence of randomised circuits of size $p'(n)$ using standard techniques. For example, if the the computation is specified for a Turing Machine, then $p'(n) = O(p(n) \log p(n))$ [PF79].

7.1 Preliminaries

7.1.1 Randomised circuits

Definition 7.1. A Boolean circuit ψ with n inputs and ℓ outputs is a directed acyclic graph $\psi = (V, E)$ containing n input vertices with zero in-degree, labelled $X_1, \dots, X_n$ and ℓ output vertices with zero out-degree, labelled $O_1, \dots, O_\ell$. Other nodes are assigned a label in $\{\wedge, \vee, \neg\}$, with vertices labelled $\neg$ having in-degree one and all others having in-degree two. The size of ψ, denoted $|\psi|$, is defined to be $|V|$. A *randomised circuit* has m additional random input vertices labelled $R_1, \dots, R_m$.

Given an input string $\boldsymbol{x} = (x_1, \dots, x_n) \in \{0,1\}^n$, the circuit is evaluated as follows. First, the values $x_1, \dots, x_n$ are assigned to the nodes labelled $X_1, \dots, X_n$. Then, m bits $\boldsymbol{r} = (r_1, \dots, r_m)$ are sampled uniformly at random from $\{0,1\}^m$ and assigned to the nodes labelled $R_1, \dots, R_m$. Then, the circuit is evaluated in topological order in the natural way. For example, let v be a node labelled $\wedge$ with incoming edges $(u_1, v), (u_2, v)$ where u_1, u_2 were assigned values z_1, z_2 then v is assigned the value $z_1 \wedge z_2$. The outcome of ψ is $(o_1, \dots, o_\ell)$, the concatenation of values assigned to the ℓ output vertices $O_1, \dots, O_\ell$.

For input $\boldsymbol{x} \in \{0,1\}^n$ and event $E \subseteq \{0,1\}^\ell$ we have

$$\mathbb{P}\left[\psi(\boldsymbol{x}) \in E\right] = \frac{|\{\boldsymbol{r} \in \{0,1\}^m \ : \ \psi(\boldsymbol{x}, \boldsymbol{r}) \in E\}|}{2^m}. \qquad ◀$$

Remark. The operators, $\wedge, \vee$ and $\neg$ are functionally complete. However, $\oplus$ (exclusive or) is also used, such that $p \oplus q \iff (p \vee q) \wedge \neg(p \wedge q)$. ◀

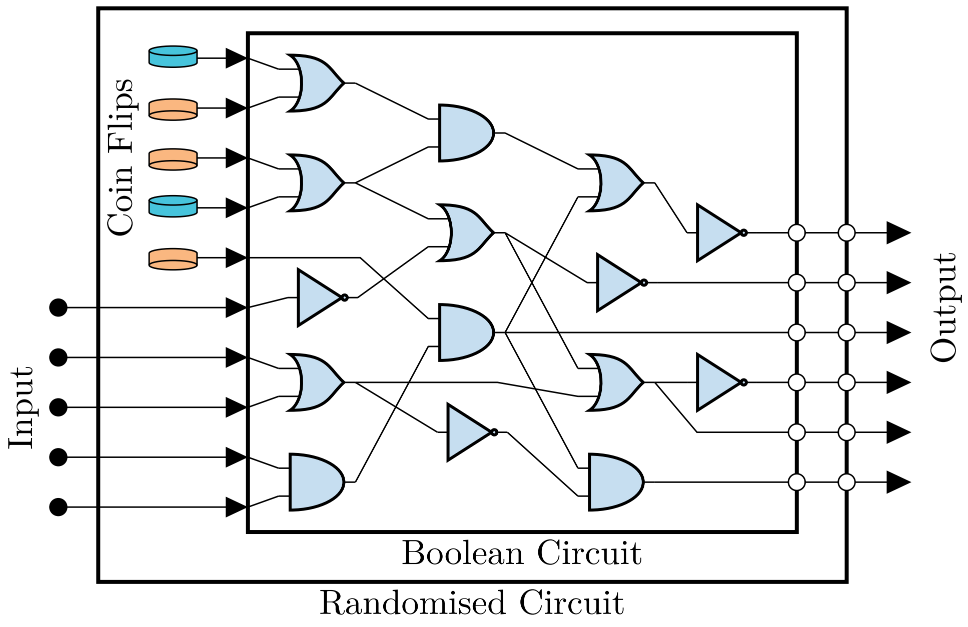


Figure 7.1: Example randomised circuit.

7.1.2 Differential privacy in randomised circuits

Let X be any input domain. An input to a differentially private analysis would generally be an array of elements from X, i.e., $\boldsymbol{x} = (x_1, \ldots, x_n) \in X^n$.

The definition of differential privacy depends on adjacency between inputs, *neighbouring* inputs are defined as follows.

Definition 7.2. Inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{x}' = (x'_1, \ldots, x'_n) \in X^n$ are called *neighbouring* if there exist $i \in [n]$ s.t. $j \neq i$ implies $x_j = x'_j$. ◀

This work considers input domains with finite representation. Without loss of generality set $X = \{0,1\}^k$ and hence an array $x = (x_1, \ldots, x_n)$ can be written as a sequence of nk bits, and given as input to a (randomised) circuit with nk inputs. The lower bounds work already for $k = 1$ and the upper bounds are presented using $k = 1$ but generalise to all k.

Reformulation 7.3 (Differential privacy for randomised circuits). A randomised circuit ψ is (ε, δ)-differentially private if for all neighbouring $\boldsymbol{x}, \boldsymbol{x}' \in X^n$ and for all $E \subseteq \{0,1\}^\ell$,

$$\mathbb{P}\left[\psi(\boldsymbol{x}) \in E\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\psi(\boldsymbol{x}') \in E\right] + \delta. \quad ◀$$

7.1.3 Problems of deciding and approximating differential privacy

The two decision problems, to decide either pure or approximate differential privacy, and the two approximation problems, to approximate either ε or δ, are set out in the following definitions:

Definition 7.4 (Decide-ε-DP).

INPUT	ε and a circuit ψ
OUTPUT	is ψ ε-differentially private?

ε is assumed to be given by the input e^{ε} given in binary[1]. ◀

Definition 7.5 (Decide-ε, δ-DP).

INPUT	ε, δ and a circuit ψ
OUTPUT	is ψ (ε, δ)-differentially private?

ε is assumed to be given by the input e^{ε} given in binary. ◀

Definition 7.6. Given an approximation error $\gamma > 0$, the Approximate-δ problem and the Approximate-ε problem, respectively, ask:

- Given ε, find $\hat{\delta} \in [0, 1]$, such that $0 \leq \hat{\delta} - \delta \leq \gamma$, where δ is the minimal value such that ψ is (ε, δ)-differentially private.
- Given δ, find $\hat{\varepsilon} \geq 0$, such that $0 \leq \hat{\varepsilon} - \varepsilon \leq \gamma$, where ε is the minimal value such that ψ is (ε, δ)-differentially private. ◀

7.1.4 The class $\mathbf{coNP}^{\#\mathbf{P}}$

The complexity class $\#\mathbf{P}$ is the counting analogue of $\mathbf{NP}$ problems. In particular the problem #Sat is complete for $\#\mathbf{P}$, which asks given a Boolean formula ϕ on n variables, how many allocations $\boldsymbol{x} \in \{0, 1\}^n$ does $\phi(\boldsymbol{x})$ evaluate to true. Note that #Sat is not a decision problem, it returns a non-negative integer between 0 and 2^n. Similarly the problem #CircuitSat is $\#\mathbf{P}$-complete, which ask given a circuit with a single output, how many allocations evaluate to true. More generally one asks how many runs of a non-deterministic Turing machine accept.

A language L is in $\mathbf{coNP}^{\#\mathbf{P}}$ if its membership problem can be refuted using a polynomial time non-deterministic Turing machine with access to a $\#\mathbf{P}$ oracle. Alternatively, $x \in L$ if and only if *all* branches of the non-deterministic Turing machine (with a $\#\mathbf{P}$ oracle) accept. It is easy to see that $\mathbf{coNP}^{\#\mathbf{P}} = \mathbf{coNP}^{\mathbf{PP}}$. Finally, $\mathbf{PH} \subseteq \mathbf{coNP}^{\#\mathbf{P}} \subseteq \mathbf{PSPACE}$, where $\mathbf{PH} \subseteq \mathbf{coNP}^{\#\mathbf{P}}$ follows by Toda's theorem ($\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$) [Tod91].

[1]For this specific problem, the results apply if e^{ε} is given as a rational number.

To establish a complete problem (All-Min-Sat) for $\mathbf{coNP}^{\#\mathbf{P}}$, first consider the problem E-Maj-Sat [LGM98], a complete decision problem for the class $\mathbf{NP}^{\#\mathbf{P}}$ [CDM17] (thus also complete for $\mathbf{NP}^{\mathbf{PP}}$ [PD08]).

Definition 7.7. E-Maj-Sat asks, given ϕ a quantifier free formula over $n+m$ variables if there exist an allocation $\boldsymbol{x} \in \{0,1\}^n$ such that there are strictly greater than $\frac{1}{2}$ of allocations to $\boldsymbol{y} \in \{0,1\}^m$ where $\phi(\boldsymbol{x},\boldsymbol{y})$ evaluates to true. ◀

The complementary problem All-Min-Sat, is complete for $\mathbf{coNP}^{\#\mathbf{P}}$: a formula ϕ is All-Min-Sat, if ϕ is not E-Maj-Sat. That is, ϕ a quantifier free formula over $n+m$ variables is All-Min-Sat if for all allocations $\boldsymbol{x} \in \{0,1\}^n$ there are no more than $\frac{1}{2}$ of allocations to $\boldsymbol{y} \in \{0,1\}^m$ where $\phi(\boldsymbol{x},\boldsymbol{y})$ evaluates to true.

Lemma 7.8. *[Wag86, Theorem 4. Point 4.] co*$(\mathbf{NP}^{\#\mathbf{P}}) = \mathbf{coNP}^{\#\mathbf{P}}$.

The remaining sections show the close connection between decision problems for differential privacy and the complexity class $\mathbf{coNP}^{\#\mathbf{P}}$.

7.2 The complexity of deciding pure differential privacy

This section classifies the complexity of deciding ε-differential privacy, showing the following theorem:

Theorem 7.9. Decide-ε-DP *is* $\mathbf{coNP}^{\#\mathbf{P}}$*-complete.*

It will be convenient to consider the well-known simpler reformulation of the definition of pure differential privacy in finite ranges to consider specific outcomes $\boldsymbol{o} \in \{0,1\}^\ell$ rather than events $E \subseteq \{0,1\}^\ell$.

Reformulation 7.10 (Pure differential privacy)**.** A randomised circuit ψ is ε-differentially private if and only if for all neighbouring $\boldsymbol{x},\boldsymbol{x'} \in X^n$ and for all $\boldsymbol{o} \in \{0,1\}^\ell$,

$$\mathbb{P}\left[\psi(\boldsymbol{x}) = \boldsymbol{o}\right] \leq e^\varepsilon \cdot \mathbb{P}\left[\psi(\boldsymbol{x'}) = \boldsymbol{o}\right]. \qquad ◀$$

7.2.1 Decide-ε-DP $\in \mathbf{coNP}^{\#\mathbf{P}}$:

There is a non-deterministic Turing machine which can "refute" ψ being ε-differentially private in polynomial time with a $\#\mathbf{P}$ oracle. A circuit ψ is shown not to be ε-differentially private by exhibiting a combination $\boldsymbol{x},\boldsymbol{x'},\boldsymbol{o}$ such that $\mathbb{P}\left[\psi(\boldsymbol{x}) = \boldsymbol{o}\right] > e^\varepsilon \cdot \mathbb{P}\left[\psi(\boldsymbol{x'}) = \boldsymbol{o}\right]$. The witness to the non-deterministic Turing machine would be a sequence of $2n$ bits parsed as neighbouring inputs

$\boldsymbol{x}, \boldsymbol{x'} \in \{0,1\}^n$ and ℓ bits describing an output $\boldsymbol{o} \in \{0,1\}^\ell$. The constraint can then be checked in polynomial time, using the $\#\mathbf{P}$ oracle to compute $\mathbb{P}[\psi(\boldsymbol{x}) = \boldsymbol{o}]$ and $\mathbb{P}[\psi(\boldsymbol{x'}) = \boldsymbol{o}]$.

To compute $\mathbb{P}[\psi(\boldsymbol{x}) = \boldsymbol{o}]$ in $\#\mathbf{P}$ create an instance to #CircuitSat, which will count the number of allocations to the m probabilistic bits consistent with this output. This is done by extending ψ with additional gates reducing to a single output which is true only when the input is fixed to $\boldsymbol{x}$ and the output of ψ was $\boldsymbol{o}$.

7.2.2 $\mathbf{coNP}^{\#\mathbf{P}}$-hardness of Decide-$\varepsilon$-DP

$\mathbf{coNP}^{\#\mathbf{P}}$-hardness of Decide-$\varepsilon$-DP is shown by a reduction from All-Min-Sat in Lemma 7.11; together with the inclusion result above, this entails that Decide-ε-DP is $\mathbf{coNP}^{\#\mathbf{P}}$-complete (Theorem 7.9).

Randomised Response Randomised response [War65] is a technique for answering sensitive Yes/No questions by flipping the answer with probability $p < 0.5$. Setting $p = \frac{1}{1+e^\varepsilon}$ gives ε-differential privacy.

Lemma 7.11. All-Min-Sat *reduces in polynomial time to* Decide-ε-DP.

Proof. The reduction from All-Min-Sat to Decide-ε-DP will use randomised response, by taking a Boolean formula ϕ and creating a randomised circuit that is ε-differentially private if and only if ϕ is All-Min-Sat.

Consider the circuit ψ which takes as input the value $z \in \{0,1\}$. It probabilistically chooses a value of $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{y} \in \{0,1\}^m$ and one further random bit p_1 and computes $b = z \oplus \neg(p_1 \vee \phi(\boldsymbol{x}, \boldsymbol{y}))$. The circuit outputs $(\boldsymbol{x}, b)$.

Claim 7.12. *ψ is $\ln(3)$-differentially private if and only if ϕ is* All-Min-Sat.

Suppose $\phi \in$ All-Min-Sat then, no matter the choice of $\boldsymbol{x}$,

$$0 \le \mathbb{P}_y[\phi(\boldsymbol{x}, \boldsymbol{y}) = 1] \le \frac{1}{2} \quad \text{and hence} \quad \frac{1}{4} \le \mathbb{P}_{y,p_1}[\neg(p_1 \vee \phi(\boldsymbol{x}, \boldsymbol{y})) = 1] \le \frac{1}{2}.$$

This concludes that the true answer z is flipped between $\frac{1}{4}$ and $\frac{1}{2}$ of the time, recall this is exactly the region in which randomised response gives us the most privacy. In the worst case $p = \frac{1}{4} = \frac{1}{1+e^\varepsilon}$, gives $e^\varepsilon = 3$, so $\ln(3)$-differential privacy.

In the converse, suppose $\phi \notin$ All-Min-Sat and hence $\phi \in$ E-Maj-Sat, then $\mathbb{P}_{y,p_1}[\neg(p_1 \vee \phi(\boldsymbol{x}, \boldsymbol{y})) = 1] < \frac{1}{4}$ for some $\boldsymbol{x}$, in which case the randomised response does not provide $\ln(3)$-differential privacy. $\square$

Remark. The result is skewed so that the proportion of accepting allocations is between $\frac{1}{4}$ and $\frac{1}{2}$ to satisfy privacy, resulting in the choice of $\ln(3)$-differentially privacy. Alternative skews, using more bits akin to p_1, shows hardness for other choices of ε. ◀

Hardness of DECIDE-ε-DP by number of input/output bits

The inclusion proof uses **coNP** to resolve the non-deterministic choice of both input and output. This can be shown to be necessary, in the sense **coNP** is still required for either large input or large output. The hardness proof shows that when $|\psi| = n$ the problem is hard for $O(1)$-bit input and $\Omega(n)$-bit output. It can also be proven (Lemma 7.13) this is hard for $\Omega(n)$-bit input and $O(1)$-bit output. Further the problem is in $\mathbf{P}^{\#\mathbf{P}}$ for $O(\log(n))$-bit input and $O(\log(n))$-bit output, as the choice made by **coNP** can be checked deterministically. In this case it is **PP**-hard, even when there is 1-bit input and 1-bit output.

Lemma 7.13. *Given a circuit ψ, the following hardness results apply:*

# Input Bits	*# Output Bits*	*Hardness*
$\Omega(n)$	*1*	$\mathbf{coNP}^{\#\mathbf{P}}$*-hard*
1	$\Omega(n)$	$\mathbf{coNP}^{\#\mathbf{P}}$*-hard*
1	*1*	**PP***-hard*

Remark. Note that the hardness results entail hardness for any larger number of input and output bits; for example $\Theta(\log n)$-input,$\Theta(\log n)$-output is **PP**-hard and $\Theta(n)$-input,$\Theta(n)$-output is $\mathbf{coNP}^{\#\mathbf{P}}$-hard. ◀

Proof for large input small output. Given $\phi(\boldsymbol{x}, \boldsymbol{y})$, reduce $\phi \in$ ALL-MIN-SAT to DECIDE-ε-DP. The resulting circuit ψ will have 1 output bit but $n+1$ input bits

Let $\psi(\boldsymbol{x}, z) = (z \vee p_1) \wedge (\neg z \vee (p_2 \vee (p_3 \wedge p_4 \wedge \phi(\boldsymbol{x}, \boldsymbol{r}))))$, with $p_1, \ldots, p_4, \boldsymbol{r}$ determined randomly. This circuit has the property:

- If $z = 0$ return 1 w.p. $\frac{1}{2}$.
- If $z = 1$ return 1 w.p. $\frac{1}{2} + \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 1]$.

Claim 7.14. $\phi \in$ ALL-MIN-SAT $\iff$ $\ln(\frac{4}{3})$*-differential privacy holds.*

If $\phi \notin$ ALL-MIN-SAT then for some $\boldsymbol{x}$ with $\mathbb{P}[\phi(\boldsymbol{x}) = 1] > \frac{1}{2}$, $\mathbb{P}[\phi(\boldsymbol{x}) = 0] < \frac{1}{2}$

$$\begin{aligned}\frac{\mathbb{P}\left[\psi(\boldsymbol{x},0)=0\right]}{\mathbb{P}\left[\psi(\boldsymbol{x},1)=0\right]} &= \frac{\frac{1}{2}}{1-\frac{1}{2}-\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=1\right])} \\ &= \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}-\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=1\right])} \\ &= \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}(1-\mathbb{P}\left[\phi(\boldsymbol{x})=1\right])} \\ &= \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}(\mathbb{P}\left[\phi(\boldsymbol{x})=0\right])} \\ &> \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}\frac{1}{2}} = \frac{4}{3},\end{aligned}$$

thus indicating the ratio is too large, and $\ln(\frac{4}{3}$-differential privacy is violated.

If $\phi \in$ ALL-MIN-SAT then for all $\boldsymbol{x}$ it is the case $\mathbb{P}\left[\phi(\boldsymbol{x})=1\right] \leq \frac{1}{2}$, $\mathbb{P}\left[\phi(\boldsymbol{x})=0\right] \geq \frac{1}{2}$. The following demonstrates the ratio is less than $\frac{4}{3}$ for all adjacent inputs and on each output (0 and 1), thus maintaining $\ln(\frac{4}{3}$-differential privacy:

$$\frac{\mathbb{P}\left[\psi(\boldsymbol{x},0)=0\right]}{\mathbb{P}\left[\psi(\boldsymbol{x},1)=0\right]} \leq \frac{4}{3}$$

$$\frac{\mathbb{P}\left[\psi(\boldsymbol{x},1)=0\right]}{\mathbb{P}\left[\psi(\boldsymbol{x},0)=0\right]} = \frac{1-\frac{1}{2}-\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=1\right]}{\frac{1}{2}} = \frac{\frac{1}{4}+\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=0\right]}{\frac{1}{2}} \leq 1$$

$$\frac{\mathbb{P}\left[\psi(\boldsymbol{x},1)=1\right]}{\mathbb{P}\left[\psi(\boldsymbol{x},0)=1\right]} = \frac{\frac{1}{2}+\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=1\right]}{\frac{1}{2}} \leq \frac{\frac{1}{2}+\frac{1}{4}\frac{1}{2}}{\frac{1}{2}} = 1.25$$

$$\frac{\mathbb{P}\left[\psi(\boldsymbol{x},0)=1\right]}{\mathbb{P}\left[\psi(\boldsymbol{x},1)=1\right]} = \frac{\frac{1}{2}}{\frac{1}{2}+\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=1\right]} \leq 1$$

$$\frac{\mathbb{P}\left[\psi(\boldsymbol{x},0)=1\right]}{\mathbb{P}\left[\psi(\boldsymbol{x}',0)=1\right]} = \frac{\frac{1}{2}}{\frac{1}{2}} = 1$$

$$\frac{\mathbb{P}\left[\psi(\boldsymbol{x},1)=1\right]}{\mathbb{P}\left[\psi(\boldsymbol{x}',1)=1\right]} = \frac{\frac{1}{2}+\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=1\right]}{\frac{1}{2}+\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x}')=1\right]} \leq \frac{\frac{1}{2}+\frac{1}{4}\frac{1}{2}}{\frac{1}{2}+\frac{1}{4}0} = 1.25$$

$$\frac{\mathbb{P}\left[\psi(\boldsymbol{x},1)=0\right]}{\mathbb{P}\left[\psi(\boldsymbol{x}',1)=0\right]} = \frac{1-\frac{1}{2}-\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x})=1\right]}{1-\frac{1}{2}-\frac{1}{4}\mathbb{P}\left[\phi(\boldsymbol{x}')=1\right]} \leq \frac{\frac{1}{2}-\frac{1}{4}0}{1-\frac{1}{2}-\frac{1}{4}\frac{1}{2}} = \frac{4}{3}. \qquad \square$$

Proof for small input large output. Given $\phi(\boldsymbol{x},\boldsymbol{y})$, reduce $\phi \in$ ALL-MIN-SAT to DECIDE-ε-DP. The resulting circuit ψ will have 1 input bit but $n+1$ output bits.

Let $\psi(z) = (\boldsymbol{x}, (z \vee p_1) \wedge (\neg z \vee (p_2 \vee (p_3 \wedge p_4 \wedge \phi(\boldsymbol{x},\boldsymbol{r})))))$, with $p_1,\ldots,p_4,\boldsymbol{x},\boldsymbol{r}$ all chosen randomly. Then the circuit has the property:

- Choose and output some $\boldsymbol{x}$ and,

- If $z = 0$ return 1 w.p. $\frac{1}{2}$.
- If $z = 1$ return 1 w.p. $\frac{1}{2} + \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 1]$.

Claim 7.15. $\phi \in$ All-Min-Sat $\iff \ln(\frac{4}{3})$*-differential privacy holds.*

If $\phi \notin$ All-Min-Sat then for some $\boldsymbol{x}$ with $\mathbb{P}[\phi(\boldsymbol{x}) = 1] > \frac{1}{2}$, $\mathbb{P}[\phi(\boldsymbol{x}) = 0] < \frac{1}{2}$

$$\begin{aligned}
\frac{\mathbb{P}[\psi(0) = (\boldsymbol{x}, 0)]}{\mathbb{P}[\psi(1) = (\boldsymbol{x}, 0)]} &= \frac{\frac{1}{2}}{1 - \frac{1}{2} - \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4} - \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4}(1 - \mathbb{P}[\phi(\boldsymbol{x}) = 1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4}(\mathbb{P}[\phi(\boldsymbol{x}) = 0])} \\
&> \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4}\frac{1}{2}} = \frac{4}{3},
\end{aligned}$$

thus indicating $\ln(\frac{4}{3}$-differential privacy is violated.

If $\phi \in$ All-Min-Sat then for all $\boldsymbol{x}$ it is the case $\mathbb{P}[\phi(\boldsymbol{x}) = 1] \leq \frac{1}{2}$, $\mathbb{P}[\phi(\boldsymbol{x}) = 0] \geq \frac{1}{2}$. The ratios between adjacent inputs are shown to be less or equal to $\frac{4}{3}$.

$$\frac{\mathbb{P}[\psi(0) = (\boldsymbol{x}, 0)]}{\mathbb{P}[\psi(1) = (\boldsymbol{x}, 0)]} \leq \frac{4}{3}$$

$$\frac{\mathbb{P}[\psi(1) = (\boldsymbol{x}, 0)]}{\mathbb{P}[\psi(0) = (\boldsymbol{x}, 0)]} = \frac{1 - \frac{1}{2} - \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 1]}{\frac{1}{2}} = \frac{\frac{1}{4} + \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 0]}{\frac{1}{2}} \leq 1$$

$$\frac{\mathbb{P}[\psi(1) = (\boldsymbol{x}, 1)]}{\mathbb{P}[\psi(0) = (\boldsymbol{x}, 1)]} = \frac{\frac{1}{2} + \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 1]}{\frac{1}{2}} \leq \frac{\frac{1}{2} + \frac{1}{4}\frac{1}{2}}{\frac{1}{2}} < \frac{4}{3}$$

$$\frac{\mathbb{P}[\psi(0) = (\boldsymbol{x}, 1)]}{\mathbb{P}[\psi(1) = (\boldsymbol{x}, 1)]} = \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{4}\mathbb{P}[\phi(\boldsymbol{x}) = 1]} \leq 1. \qquad \square$$

Proof for small input small output.
Given $\phi(\boldsymbol{x})$, reduce $\phi \in$ Maj-Sat to Decide-ε-DP. The resulting circuit ψ will have 1 output bit and 1 input bits.

Let $\psi(z) = (p_1 \wedge z) \vee (\neg p_1 \wedge (z \oplus \phi(\boldsymbol{r})))$, where p_1 and $\boldsymbol{r}$ are chosen randomly. Then the circuit has the property:

- return z w.p. $\frac{1}{2}$,
- return $z \oplus \phi(\boldsymbol{r})$ w.p. $\frac{1}{2}$. (Output z, flipped proportionally to the number of accepting allocations to ϕ.)

Claim 7.16. $\phi \in \textsc{Maj-Sat} \iff \psi$ *is* $\ln(3)$*-differentially private*

Conducting a similar case analysis to the above, the probabilities behave as follows:

Output ↓	Input →	1	0	Max-Ratio
0	Maj	$> \frac{1}{4}$	$< \frac{3}{4}$	> 3
	Min	$\leq \frac{1}{4}$	$\geq \frac{3}{4}$	≤ 3
1	Maj	$< \frac{3}{4}$	$> \frac{1}{4}$	> 3
	Min	$\geq \frac{3}{4}$	$\leq \frac{1}{4}$	≤ 3

Then in the Maj cases the ratio is greater than 3, thus violating privacy ln(3)-differential privacy and for both Min case the ratio is bounded by 3 (maintaining privacy). □

7.3 The complexity of deciding approximate differential privacy

Theorem 7.9 shows that Decide-ε-DP is $\mathbf{coNP}^{\#\mathbf{P}}$-complete, in particular $\mathbf{coNP}^{\#\mathbf{P}}$-hard and since Decide-$\varepsilon$-DP is a special case of Decide-ε, δ-DP, this is also $\mathbf{coNP}^{\#\mathbf{P}}$-hard. Nevertheless the proof is based on particular values of ε and an alternative proof of hardness is provided (Theorem 7.18) based on δ (which applies even for $\varepsilon = 0$).

It is not clear whether deciding (ε, δ)-differential privacy can be done in $\mathbf{coNP}^{\#\mathbf{P}}$. Recall that in the case of ε-differential privacy it was enough to consider singleton events $\{\boldsymbol{o}\}$ where $\boldsymbol{o} \in \{0,1\}^\ell$, however in the definition of (ε, δ)-differential privacy requires quantifying over output events $E \subseteq \{0,1\}^\ell$. When considering circuits with one output bit ($\ell = 1$), then the event space essentially reduces to $E \in \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$ and so the same technique can be test again, checking only $\mathbb{P}[\psi(\boldsymbol{x}) = 1]$ vs $\mathbb{P}[\psi(\boldsymbol{x}') = 1]$ and similarly for $= 0$. Noting that $\emptyset$ and $\{0,1\}$ do not need to be checked as they have probability zero and one from any input respectively.

This can be extended to the case when the number of outputs bits is logarithmic $\ell \leq \log(|\psi|)$. To cater to this, rather than guessing a violating $E \in \{0,1\}^\ell$, consider a violating subset of events $E \subseteq \{0,1\}^\ell$. Given such an event E create a circuit ψ_E on ℓ inputs and a single output which indicates whether the input is in the event E. The size of this circuit is exponential in ℓ, thus polynomial in $|\psi|$. Composing $\psi_E \circ \psi$, check the conditions hold for this event E, with just one bit of output.

Claim 7.17. Decide-ε, δ-DP*, restricted to circuits* ψ *with* ℓ *bit outputs where* $\ell \leq \log(|\psi|)$*, is in* $\mathbf{coNP}^{\#\mathbf{P}}$ *(and hence* $\mathbf{coNP}^{\#\mathbf{P}}$*-complete).*

Claim 7.17 trivially extends to $\ell \leq c \cdot \log(|\psi|)$ for any *fixed* $c > 0$.

When ℓ is larger, one can verify "succinctly separable" events E, by quantifying over all circuits of size $p(|\psi|)$. However the majority of separation circuits $f : \{0,1\}^\ell \to \{0,1\}$, require at least approximately $\frac{2^\ell}{\ell}$ bits [RS42].

7.3.1 Direct proof that DECIDE-ε, δ-DP is $\mathbf{coNP^{\#P}}$-hard

The following proves that DECIDE-ε, δ-DP is $\mathbf{coNP^{\#P}}$-hard, even when there is just one output bit and for every ε.

Theorem 7.18. DECIDE-ε, δ-DP *is* $\mathbf{coNP^{\#P}}$*-hard.*

Hardness can be shown for DECIDE-ε, δ-DP by direct reduction from ALL-MIN-SAT, which would entail that $(\varepsilon, \frac{1}{2})$-differential privacy is $\mathbf{coNP^{\#P}}$-hard. To show hardness for a large range of δ, the problem ALL-MIN-SAT is also shown to be hard, where $\frac{1}{2}$ is generalised to an arbitrary fraction f. ALL-FRAC-f-SAT is then reduced to DECIDE-ε, δ-DP.

Generalising ALL-MIN-SAT

ALL-MIN-SAT generalises to ALL-FRAC-f-SAT by instead of requiring that the minority (less than half) of allocations to $\boldsymbol{y}$ give true, rather no more than a fraction f. Similarly E-MAJ-SAT generalises to E-FRAC-f-SAT. Hardness will be shown on E-FRAC-f-SAT using E-MAJ-SAT.

Definition 7.19. A formula $\phi(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x} \in \{0,1\}^n, \boldsymbol{y} \in \{0,1\}^m$ and $f \in [0,1] \cap \mathbb{Q}$ is ALL-FRAC-f-SAT if for every $\boldsymbol{x} \in \{0,1\}^n$

$$\frac{|\{\boldsymbol{y} \in \{0,1\}^m \ \} \ \phi(\boldsymbol{x}, \boldsymbol{y}) \text{ is true}|\,|}{2^m} \leq f. \qquad ◀$$

Remark. If $f = 0$, then this requires that for all $\boldsymbol{x}$, no input of $\boldsymbol{y}$ gives true, therefore requiring ϕ is unsatisfiable. For $f = 1$, is essentially no restriction and for $f = \frac{2^m - 1}{2^m}$ requires that ϕ is not a tautology. ALL-MIN-SAT is then when $f = \frac{1}{2}$. ◀

Definition 7.20. A formula ϕ is E-FRAC-f-SAT if it is not ALL-FRAC-f-SAT. ◀

This means a formula $\phi(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x} \in \{0,1\}^n, \boldsymbol{y} \in \{0,1\}^n$ and $f \in [0,1] \cap \mathbb{Q}$ is E-FRAC-f-SAT if there exists an allocation $\boldsymbol{x}$ that more than f fraction of allocations to $\boldsymbol{y}$ result in $\phi(\boldsymbol{x}, \boldsymbol{y})$ being true. That is there exists $\boldsymbol{x} \in \{0,1\}^n$ such that $\frac{|\{\boldsymbol{y} \in \{0,1\}^m \ \} \ \phi(\boldsymbol{x}, \boldsymbol{y}) \text{ is true}|\,|}{2^m} > f$.

To show All-Frac-f-Sat is $\mathbf{coNP}^{\#\mathbf{P}}$-hard it is shown that E-Frac-f-Sat is $\mathbf{NP}^{\#\mathbf{P}}$-hard, entailing Corollary 7.22.

Lemma 7.21. E-Frac-f-Sat *is* $\mathbf{NP}^{\#\mathbf{P}}$*-hard for* $f \in [\frac{1}{2^m}, \frac{2^m-1}{2^m})$.

Corollary 7.22. All-Frac-f-Sat *is* $\mathbf{coNP}^{\#\mathbf{P}}$*-hard for* $f \in [\frac{1}{2^m}, \frac{2^m-1}{2^m})$.

Definition 7.23. Let $\chi_{\frac{b}{2^k}}(z_1, \ldots, z_k))$ be a circuit on k bits, which is true for b of its 2^k inputs, but not true for $z_1 = \cdots = z_k = 1$ when $b < 2^k$. Given b, k such that $0 < \frac{b}{2^k} < 1$, create a formula, over $2k$ bits, which given two k-bit integers m, n, return whether $m \leq n$ (such a formula is of size polynomial in k). By fixing n to b, we have a circuit on m input bits which decides if $m \leq b$. If $z_1, \ldots, z_k$ is chosen probabilistically the circuit outputs 1 with probability $b2^k$. ◀

Proof of Lemma 7.21. The proof proceeds in three cases, depending on the possible representation of the fraction f.

Case 1 (For $f = \frac{1}{2^{k+1}}$)**.** Reduction from E-Maj-Sat, i.e. given a formula $\phi(\boldsymbol{x}, \boldsymbol{y})$ to E-Frac-$\frac{1}{2^{k+1}}$-Sat.

(Assume $\frac{1}{2^{k+1}}$ takes $O(k)$ bits.)

Define a formula $\phi'(\boldsymbol{x}, \boldsymbol{w})$, with $\boldsymbol{w} \in \{0,1\}^{m+k}$. Let $\boldsymbol{w} = (y_1, \ldots, y_m, z_1 \ldots z_k)$, where $\boldsymbol{y} = (y_1, \ldots, y_m)$.

$\phi'(\boldsymbol{x}, \boldsymbol{w}) = z_1 \wedge \cdots \wedge z_k \wedge \phi(\boldsymbol{x}, y_1, \ldots, y_m)$

For $\boldsymbol{x}$ fixed if g allocations to $y_1, \ldots, y_m$ satisfy ϕ then each of these satisfy ϕ' only when $z_1 = \cdots = z_k = 1$. All remaining times are unsatisfied.

Suppose $\frac{g}{2^m}$ of $\boldsymbol{y}$'s satisfy $\phi(\boldsymbol{x}, \boldsymbol{y})$ then $\frac{g}{2^m.2^k}$ $\boldsymbol{w}$'s satisfy $\phi'(\boldsymbol{x}, \boldsymbol{w})$.

That is we have:

$$\frac{g}{2^m \cdot 2^k} > \frac{1}{2^{k+1}} \iff \frac{g}{2^m} > \frac{1}{2}.$$

Case 2 (For $f = \frac{a}{2^{k+1}}$)**.** Reduce E-Maj-Sat, given a formula $\phi(\boldsymbol{x}, \boldsymbol{y})$ to E-Frac-$\frac{1}{2^{k+1}}$-Sat. Assume a odd (otherwise, half and take $\frac{a/2}{2^k}$) and greater than 1, else use Case 1.

Define a formula $\phi'(\boldsymbol{x}, \boldsymbol{w})$, with $\boldsymbol{w} \in \{0,1\}^{m+k}$. Let $\boldsymbol{w} = (y_1, \ldots, y_m, z_1 \ldots z_k)$, where $\boldsymbol{y} = (y_1, \ldots, y_m)$. Let $b = \frac{a-1}{2}$ (b is always between 1 and $2^k - 1$).

Then we let $\phi'(\boldsymbol{x}, \boldsymbol{w}) = (z_1 \wedge \cdots \wedge z_k \wedge \phi(\boldsymbol{x}, y_1, \ldots, y_m)) \vee \chi_{\frac{b}{2^k}}(z_1, \ldots, z_k)$.

That is the formula ϕ' is true whenever $z_1 = \cdots = z_k = 1$ and ϕ is true, or on the $\frac{b}{2^k}$ choices of $z_1 \ldots z_k$.

Suppose $\frac{g}{2^m}$ of $\boldsymbol{y}$'s satisfy $\phi(\boldsymbol{x},\boldsymbol{y})$ then $\frac{g}{2^m.2^k}+\frac{b}{2^k}=\frac{g}{2^m.2^k}+\frac{a-1}{2^{k+1}}$ $\boldsymbol{w}$'s satisfy $\phi'(\boldsymbol{x},\boldsymbol{w})$.

That is we have:

$$\frac{g}{2^m\,.\,2^k}+\frac{a-1}{2^{k+1}}>\frac{a}{2^{k+1}}\iff\frac{g}{2^m\,.\,2^k}>\frac{1}{2^{k+1}}\iff\frac{g}{2^m}>\frac{1}{2}$$

Case 3 (For $f=\frac{a}{b}$)**.** Let m be the number such that $\boldsymbol{y}\in\{0,1\}^m$, the number of $\boldsymbol{y}$ bits of the formula, or the number of "MAJ" bits. Let z be such that $\frac{z}{2^m}<\frac{a}{b}<\frac{z+1}{2^m}$. We reduce E-FRAC-$\frac{z}{2^m}$-SAT to E-FRAC-$\frac{a}{b}$-SAT, by simply taking ϕ unchanged.

Suppose $\frac{g}{2^m}$ of $\boldsymbol{y}$'s satisfy $\phi(\boldsymbol{x},\boldsymbol{y})$ then

$$\frac{g}{2^m}>\frac{z}{2^m}\iff\frac{g}{2^m}\geq\frac{z+1}{2^m}\iff\frac{g}{2^m}>\frac{a}{b}.\qquad\square$$

Main Proof of Theorem 7.18

Proof. Assume that an instance of ALL-FRAC-f-SAT is given, a formula $\phi(\boldsymbol{x},\boldsymbol{y})$ for $\boldsymbol{x}\in\{0,1\}^n,\boldsymbol{y}\in\{0,1\}^m$ and $f\in[0,1]$.

Define a circuit ψ, with inputs $\boldsymbol{x}\in\{0,1\}^{n+1}$, written as $(z,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)$; matching the inputs $\boldsymbol{x}$ and a single additional bit z. There are m probabilistic bits $\boldsymbol{r}\in\{0,1\}^m$, matching $\boldsymbol{y}$. There is one output bit $\boldsymbol{o}\in\{0,1\}^1$. The circuit ψ will behave like ϕ when $z=1$ and simply output 0 when $z=0$; i.e. $\boldsymbol{o}_1=z\wedge\phi(\boldsymbol{x}_1,\dots,\boldsymbol{x}_n,\boldsymbol{r}_1,\dots,\boldsymbol{r}_m)$.

Claim 7.24. *$\phi\in$ ALL-FRAC-f-SAT if and only if ψ is (ε,δ)-differentially private, for $\delta=f$ and any choice of ε (including zero).*

Direction: if $\phi\notin$ ALL-FRAC-f-SAT then not (ε,δ)-differentially private. Given $\phi\notin$ ALL-MIN-SAT then there exists $\boldsymbol{x}\in\{0,1\}^n$ such that $\phi(\boldsymbol{x},\boldsymbol{y})$ is on more than f portion of $\boldsymbol{y}\in\{0,1\}^m$. The differential privacy condition is then shown to be violated exactly using this $\boldsymbol{x}$, let $\boldsymbol{x}=(1,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)$ and $\boldsymbol{x'}=(0,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)$. Now consider the probability of the event $o_1=1$.

Then we have $\mathbb{P}\left[\psi(1,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)=1\right]>f$ and $\mathbb{P}\left[\psi(0,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)=1\right]=0$, violating differential privacy since,

$$\mathbb{P}\left[\psi(1,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)=1\right]-e^{\varepsilon}\mathbb{P}\left[\psi(0,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)=1\right]>f-0=\delta.$$

Direction: if $\phi\in$ALL-FRAC-f-SAT then (ε,δ)-differentially private. Since $\phi\in$ ALL-FRAC-f-SAT then for all $\boldsymbol{x}\in\{0,1\}^n$ we have $\phi(\boldsymbol{x},\boldsymbol{y})$ true for less or equal f proportion of the allocations to $\boldsymbol{y}\in\{0,1\}^m$. Equivalently the more than f of $\boldsymbol{y}\in\{0,1\}^m$ with $\phi(\boldsymbol{x},\boldsymbol{y})$ false.

Privacy is shown by considering all neighbouring inputs and all output event. The output events are $E \subseteq \{0,1\}^1$, giving $\emptyset, \{0\}, \{1\}, \{0,1\}$. The probability of "no output" ($\emptyset$) is zero for all inputs, so cannot violate differential privacy. Similarly the probability of "output anything" ($\{0,1\}$) is one for all inputs, so does cannot violate differential privacy. Thus arguments are necessary that events $\{0\}$ and $\{1\}$ do not violate differential privacy, for all neighbouring inputs.

Inputs take the form $\boldsymbol{x} = (z, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, and $\boldsymbol{x}, \boldsymbol{x'}$ can be neighbouring either with fixed $\boldsymbol{x}$ and differing z or, fixed z and $\boldsymbol{x}$ differing in one position; in each case it is shown that $\mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] \leq \delta$ and $\mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] \leq \delta$.

Let z be fixed to zero. Hence we have $\boldsymbol{x}, \boldsymbol{x'}$ with $\boldsymbol{x}$'s differing in one position. For $z = 0$ the circuit outputs zero in all cases, independently of $\boldsymbol{x}$, thus does not violate differential privacy since $\mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] = \mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right]$.

Let z be fixed to one. Hence we have $\boldsymbol{x}, \boldsymbol{x'}$ with $\boldsymbol{x}$'s differing in one position. Without loss of generality suppose the difference is x_j.

For the event $E = \{1\}$ we have the probability being $\leq f$ for each input; that ism regardless of $\boldsymbol{x}$ we have $\mathbb{P}\left[\psi(\boldsymbol{x}) = 1\right] \leq f$. So $\mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] \leq \mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] \leq f = \delta$ and $\mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] \leq \mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] \leq f = \delta$.

For the event $E = \{0\}$ we have the probability being $\geq 1 - f$ for each input; that is, regardless of $\boldsymbol{x}$ we have $\mathbb{P}\left[\psi(\boldsymbol{x}) = 0\right] \geq 1 - f$. So $\mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] \leq 1 - \mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] \leq f = \delta$ and $\mathbb{P}\left[\psi(\boldsymbol{x'}) = E\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] \leq 1 - \mathbb{P}\left[\psi(\boldsymbol{x}) = E\right] \leq f = \delta$.

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be fixed. Hence we have $\boldsymbol{x}, \boldsymbol{x'}$ with differing z. Without loss of generality, let $\boldsymbol{x}$ have $z = 1$, that is, $\boldsymbol{x} = (1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $\boldsymbol{x'}$ have $z = 0$, $\boldsymbol{x'} = (0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

For the event $\{1\}$, when $z = 1$, we have $\mathbb{P}\left[\psi(\boldsymbol{x}) = 1\right] \leq f$, but for $z = 0$ the circuit is always 0, thus $\mathbb{P}\left[\psi(\boldsymbol{x'}) = 1\right] = 0$.

Then $\mathbb{P}\left[\psi(\boldsymbol{x}) = 1\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x'}) = 1\right] = \mathbb{P}\left[\psi(\boldsymbol{x}) = 1\right] \leq f = \delta$ and $\mathbb{P}\left[\psi(\boldsymbol{x'}) = 1\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x}) = 1\right] \leq 0 \leq \delta$.

For the event $\{0\}$ we have then $\mathbb{P}\left[\psi(\boldsymbol{x}) = 0\right] \geq 1 - f$ and $\mathbb{P}\left[\psi(\boldsymbol{x'}) = 0\right] = 1$ Then $\mathbb{P}\left[\psi(\boldsymbol{x}) = 0\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x'}) = 0\right] \leq 1 - e^{\varepsilon} \leq 0 \leq \delta$ and $\mathbb{P}\left[\psi(\boldsymbol{x'}) = 0\right] - e^{\varepsilon}\mathbb{P}\left[\psi(\boldsymbol{x}) = 0\right] \leq 1 - \mathbb{P}\left[\psi(\boldsymbol{x}) = 0\right] \leq f = \delta$. $\square$

7.3.2 Decide-ε,δ-DP $\in \mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$

This section shows that, in the most general case, Decide-ε,δ-DP can be solved in $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$.

Theorem 7.25. Decide-ε,δ-DP *can be decided in* $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$.

Non-determinism will be used to choose inputs leading to a violating event, but, unlike in Section 7.2 it will not be used for finding a violating output event E, as an (explicit) description of such an event may be of super-polynomial length. It is useful to use a reformulation of approximate differential privacy, using a sum over potential individual outcomes:

Reformulation 7.26 (Pointwise differential privacy [Bar+16b]). A randomised circuit ψ is (ε,δ)-differentially private if and only if for all neighbouring $\boldsymbol{x},\boldsymbol{x}' \in X^n$ and for all $\boldsymbol{o} \in \{0,1\}^\ell$,

$$\sum_{\boldsymbol{o}\in\{0,1\}^\ell} \delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o}) \leq \delta$$

where $\delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o}) = \max\left(\mathbb{P}\left[\psi(\boldsymbol{x}) = \boldsymbol{o}\right] - e^\varepsilon \cdot \mathbb{P}\left[\psi(\boldsymbol{x}') = \boldsymbol{o}\right], 0\right).$ ◀

Proof of Theorem 7.25. Assume $e^\varepsilon = \alpha$ is given in binary, thus $\alpha = \frac{u}{2^v}$ for some integers u and v.

Define $\mathcal{M}$, a non-deterministic Turing Machine with access to a $\#\mathbf{P}$-oracle, and where each execution branch runs in polynomial time. On inputs a randomised circuit ψ and neighbouring $\boldsymbol{x},\boldsymbol{x}' \in X^n$ the number of accepting executions of $\mathcal{M}$ would be proportional to $\sum_{\boldsymbol{o}\in\{0,1\}^\ell} \delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o})$.

In more detail, on inputs ψ, $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\mathcal{M}$ chooses $\boldsymbol{o} \in \{0,1\}^\ell$ and an integer $C \in \{1,2,\ldots,2^{m+v}\}$ (this requires choosing $l+m+v$ bits). Through a call to the $\#\mathbf{P}$ oracle, $\mathcal{M}$ computes

$$a = \left|\{\boldsymbol{r} \in \{0,1\}^m : \psi(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{o}\}\right|, \quad \text{and} \quad b = \left|\{\boldsymbol{r} \in \{0,1\}^m : \psi(\boldsymbol{x}',\boldsymbol{r}) = \boldsymbol{o}\}\right|.$$

Finally, $\mathcal{M}$ accepts if $2^v \cdot a - u \cdot b \geq C$ and otherwise rejects.

Claim 7.27. *Give two inputs $\boldsymbol{x},\boldsymbol{x}' \in X^n$, $\mathcal{M}(\psi,\boldsymbol{x},\boldsymbol{x}')$ has exactly $2^v \cdot 2^m \sum_{\boldsymbol{o}\in\{0,1\}^\ell} \delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o})$ accepting executions.*

Proof. Let $\mathbb{1}\{X\}$ be the indicator function, which is one if the predicate X holds and zero otherwise.

$$
\begin{aligned}
&2^m 2^v \sum_{\boldsymbol{o}\in\{0,1\}^\ell} \delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o}) = \sum_{\boldsymbol{o}\in\{0,1\}^\ell} 2^v 2^m \max\left(\mathbb{P}\left[\psi(\boldsymbol{x}) = \boldsymbol{o}\right] - \alpha\mathbb{P}\left[\psi(\boldsymbol{x}') = \boldsymbol{o}\right], 0\right) \\
&= \sum_{\boldsymbol{o}\in\{0,1\}^\ell} 2^v 2^m \max\left(\frac{1}{2^m}\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{o}\} - \alpha\frac{1}{2^m}\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}',\boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \\
&= \sum_{\boldsymbol{o}\in\{0,1\}^\ell} \max\left(2^v\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{o}\} - 2^v\alpha\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}',\boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \\
&= \sum_{\boldsymbol{o}\in\{0,1\}^\ell} \max\left(2^v\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{o}\} - u\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}',\boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \\
&= \sum_{\boldsymbol{o}\in\{0,1\}^\ell} \sum_{C=1}^{2^{v+m}} \mathbb{1}\left\{\max\left(2^v\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x},\boldsymbol{r}) = \boldsymbol{o}\} - u\sum_{\boldsymbol{r}\in\{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}',\boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \geq C\right\} \\
&= \text{number of accepting executions in } \widehat{\mathcal{M}}. \qquad \blacksquare
\end{aligned}
$$

Now the $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ procedure for Decide-ε,δ-DP can be described, entailing Theorem 7.25. The procedure takes as input a randomised circuit ψ.

1. Non-deterministically choose neighbouring $\boldsymbol{x}$ and $\boldsymbol{x}' \in \{0,1\}^n$ (i.e. $2n$ bits)

2. Let $\mathcal{M}$ be the non-deterministic Turing Machine with access to a $\#\mathbf{P}$-oracle as described above. Create a machine $\widehat{\mathcal{M}}$ with no input that executes $\mathcal{M}$ on $\psi, \boldsymbol{x}, \boldsymbol{x}'$

3. Make an $\#\mathbf{P}^{\#\mathbf{P}}$ oracle call for the number of accepting executions $\widehat{\mathcal{M}}$ has.

4. Reject if the number of accepting executions is greater than $2^v \cdot 2^m \cdot \delta$ and otherwise accept.

By Claim 7.27, there is a choice $\boldsymbol{x}, \boldsymbol{x}'$ on which the procedure rejects if and only if ψ is not (ε,δ)-differentially private. $\square$

7.4 Inapproximability of the privacy parameters ε and δ

Given the difficulty of deciding if a circuit is differentially private, one might naturally consider whether approximating ε or δ could be efficient. This section shows these tasks are both **NP**-hard and **coNP**-hard.

The problem of distinguishing between (ε,δ), and (ε',δ')-differential privacy is **NP**-hard, by reduction from a problem called here Not-Constant which will be shown to be **NP**-hard. A Boolean formula is in Not-Constant if it is satisfiable and not also a tautology.

Lemma 7.28. NOT-CONSTANT *is* **NP**-*complete. (hence* CONSTANT *is* **coNP**-*complete).*

Proof. Clearly, NOT-CONSTANT $\in$ **NP**, the witness being a pair of satisfying and non-satisfying assignments. Reducing 3-SAT to NOT-CONSTANT. Given a Boolean formula ϕ over variables $x_1, \dots, x_n$ let $\phi'(x_1, \dots, x_n, x_{n+1}) = \phi(x_1, \dots, x_n) \wedge x_{n+1}$. Note that ϕ' is never a tautology as $\phi'(x_1, \dots, x_n, 0) = 0$. Furthermore, ϕ' is satisfiable if and only if ϕ is. $\square$

Randomised Response Revisited

In Section 7.2.2 randomised response is used in the pure differential privacy setting. Now consider the approximate differential privacy variant $RR_{\varepsilon,\delta} : \{0,1\} \to \{\top, \bot\} \times \{0,1\}$ defined as follows:

$$RR_{\varepsilon,\delta}(x) = \begin{cases} (\top, x) & \text{w.p. } \delta \\ (\bot, x) & \text{w.p. } (1-\delta)\frac{\alpha}{1+\alpha} \\ (\bot, \neg x) & \text{w.p. } (1-\delta)\frac{1}{1+\alpha} \end{cases} \quad \text{where} \quad \alpha = e^{\varepsilon}.$$

That is, with probability δ, $RR_{\varepsilon,\delta}(x)$ reveals x and otherwise it executes $RR_{\varepsilon}(x)$. The former is marked with "$\top$" and the latter with "$\bot$". This mechanism is equivalent to the one described in [MV16].

$RR_{\varepsilon,\delta}$ is minimally (ε, δ)-differentially private, in the sense that this is the smallest possible δ, and for this δ the smallest possible ε. Naturally ε could be reduced for larger δ.

Description of the circuit $RR_{\varepsilon,\delta}$

Assume a number between 0 and 1 is given that can be written in binary of the form $\frac{c}{2^m}$. Recall from Definition 7.23 there is a there is a circuit χ_c which is true $\frac{c}{2^m}$ of the time. Assume ε is such that $c = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$ can be written as a decimal in m bits and δ can be written as $\frac{d}{2^q}$ where d has q bits.

Specifically the circuit $RR_{\varepsilon,\delta}$ is defined so that $(o_1, o_2) = RR_{\varepsilon,\delta}(x)$, with $p_1, \dots, p_m$ and $p'_1, \dots, p'_q$ random bits. Let $o_1 = \chi_{\delta}(p'_1, \dots, p'_q)$ and $o_2 = (o_1 \wedge x) \vee (\neg o_1 \wedge (x \oplus \chi_c(p_1, \dots, p_m)))$. (Formally, output bit o_1 cannot be used to determine o_2, create an internal node for the value of o_1 and use this for both o_1 and o_2). $RR_{\varepsilon,\delta}$ is (ε, δ)-differentially private.

Definition 7.29. Let $0 \leq \varepsilon \leq \varepsilon'$, $0 \leq \delta \leq \delta' \leq 1$, with either $\varepsilon < \varepsilon'$ or $\delta < \delta'$. Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP asks, given a circuit ψ, guaranteed to be either (ε, δ)-differentially private, or (ε', δ')-differentially private. Determine whether ψ is (ε, δ)-differentially private or (ε', δ')-differentially private. ◀

Lemma 7.30. Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP *is* **NP**-*hard (and* **coNP**-*hard).*

Proof. Reducing Not-Constant to Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP. Given the Boolean formula $\phi(\boldsymbol{x})$ on n bits, create a randomised circuit ψ. The input to ψ consists of the n bits of $\boldsymbol{x}$ plus a single bit y. The circuit ψ has four output bits (o_1, o_2, o_3, o_4) such that $(o_1, o_2) = RR_{\varepsilon,\delta}(y)$ and $(o_3, o_4) = RR_{\varepsilon',\delta'}(\phi(\boldsymbol{x}))$.

Observe that $(o_1, o_2) = RR_{\varepsilon,\delta}(y)$ is always (ε, δ) differentially private. As for $(o_3, o_4) = RR_{\varepsilon',\delta'}(\phi(\boldsymbol{x}))$, if $\phi \in$ Not-Constant then there are neighbouring $\boldsymbol{x}, \boldsymbol{x}'$ such that $\phi(\boldsymbol{x}) \neq \phi(\boldsymbol{x}')$. In this case, $(o_3, o_4) = RR_{\varepsilon',\delta'}(\phi(\boldsymbol{x}))$ is (ε', δ')-differential privacy, and, because $(\varepsilon, \delta) < (\varepsilon', \delta')$, so is ψ . On the other hand, if $\phi \notin$ Not-Constant then $\phi(\boldsymbol{x})$ does not depend on $\boldsymbol{x}$ and hence (o_3, o_4) does not affect privacy, in which case we have that ψ is (ε, δ) differentially private.

The same argument also gives **coNP**-hardness. □

Notice that the above theorem holds when $\delta = \delta'$ and $\varepsilon < \varepsilon'$ (similarly, $\varepsilon = \varepsilon'$ and $\delta < \delta'$), which entails the following theorem:

Theorem 7.31. *Assuming* $\mathbf{P} \neq \mathbf{NP}$, *for any approximation error* $\gamma > 0$, *there does not exist a polynomial time approximation algorithm that given a randomised circuit* ψ *and* δ *some computes* $\hat{\varepsilon}$, *where* $|\hat{\varepsilon} - \varepsilon| \leq \gamma$ *and* ε *is the minimal such that* ψ *is* (ε, δ)*-differentially private within error* γ. *Similarly, given* ε, *no such* $\hat{\delta}$ *can be computed where* $\left|\hat{\delta} - \delta\right| \leq \gamma$ *and* δ *is minimal.*

Proof. Suppose there is a polynomial time algorithm:

- **Approximate-ε:** Then Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP (**coNP**-hard) is solved by approximating with $\gamma = \frac{\varepsilon' - \varepsilon}{3}$ to find $\hat{\varepsilon}$. Then Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP if and only if $\hat{\varepsilon} < \frac{\varepsilon' + \varepsilon}{2}$. Similarly the complement (**NP**-hard) is solved when $\hat{\varepsilon} > \frac{\varepsilon' + \varepsilon}{2}$.
- **Approximate-δ:** Then Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP (**coNP**-hard) is solved by approximating with $\gamma = \frac{\delta' - \delta}{3}$ to find $\hat{\delta}$. Then Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP if and only if $\hat{\delta} < \frac{\delta' + \delta}{2}$. Similarly the complement (**NP**-hard) is solved when $\hat{\delta} > \frac{\delta' + \delta}{2}$. □

Remark. The results also hold when approximating pure differential privacy, i.e. when $\delta = 0$. Moreover, the result applies when approximating within a given ratio $\rho > 1$ (e.g. in the case of approximating ε, to find $\hat{\varepsilon}$ such that $\frac{\hat{\varepsilon}}{\varepsilon} \leq \rho$). ◀

7.5 Conclusion

This chapter considers the complexity of verifying differential privacy in randomised circuits. Deciding ε-differential privacy in randomised circuits is $\mathbf{coNP}^{\#\mathbf{P}}$-complete and (ε, δ)-differential privacy is $\mathbf{coNP}^{\#\mathbf{P}}$-hard and in $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$. Both problems are positioned in between the polynomial hierarchy **PH** and **PSPACE**.

Nevertheless there is a gap to be closed for approximate differential privacy. The direct hardness proof for δ has just one output bit, to improve the lower bound, circuits with more output bits must be considered and possibly further dependency on ε could refine the lower bound.

Towards a conjecture; $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ would seem intuitive. **coNP** is used to resolve the input, the first #**P** resolves the the output, and the final #**P** resolves the probability for each input/output choice. It seems necessary that the output cannot be resolved without a #**P** oracle, because it requires choice of an object of exponential size. This is in contrast to ε-differential privacy, where the choice of output is small and so can be chosen with **coNP** and the choice of input and output can be resolved with one level of **coNP**. However there is no completeness result, and it is possible that some ingenious construction enables the output and probability choice can be combined into a single #**P** oracle, rather than using a #**P** oracle which itself requires a #**P** oracle.

Considering the practical implications for verification of "simple" programs, the results seem to point to a deficiency in available tools for model checking. The model checking toolkit includes well established SAT solvers for **NP** (and **coNP**) problems, solvers for further quantification in **PH**, solvers for #SAT (and hence for #**P** problems) (see e.g. [Aut19]). However it would seem that there are currently no solvers that are specialised for mixing the polynomial hierarchy **PH** and counting problems #**P**, in particular for problems in $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$.

Approximating the differential privacy parameters is shown to be **NP**-hard and **coNP**-hard, indicating polynomial approximation is not possible (unless **P** = **NP**). This is determined by the problem of distinguishing (ε, δ)-differential privacy from (ε', δ')-differential privacy where $(\varepsilon, \delta) < (\varepsilon', \delta')$, which is both **NP**-hard and **coNP**-hard. The full classification of the complexity of this problem is left as an open problem.

Chapter 8

Conclusions and Future Work

The thesis contributed to the algorithmic verification of differential privacy; the setting in which no further hints are provided to the verification procedure, and the procedure must be fully automated. This is studied in the context of two formal settings, namely *labelled Markov chains* and *randomised circuits*. The main interest is the feasibility and complexity of verification procedures for differential privacy on these models.

In the labelled Markov chain setting it is shown that most decision problems are undecidable when considered directly and exactly. Focusing first on δ, approximation is between **#P** and **PSPACE** on finite word models matching the results for the standard total variation distance, but as yet there is no complexity classification for the approximation on infinite word models. Towards feasible verification techniques, this thesis demonstrates the theory of bisimilarity distances on such models, showing these are sound upper bounds on δ (existing work shows sound upper bounds for ε). This thesis takes a computational approach and shows that the distances for δ can be computed in polynomial time with an **NP** oracle and a slightly weaker distance computed in polynomial time. In experiments it appears that these two distances coincide, but further techniques may be required to enable practical implementations. Figure 5.1 summarises the complexity results on δ and the relations between distances.

Turning to ε, the bisimilarity distance can be approximated in **PSPACE**, leaving further work to determine if it can be computed exactly, and to fully classify the complexity of approximation. The distance of direct interest is also studied, for which, the question of whether it is bounded is studied through the big-O problem; determining that the complexity is **coNP**-complete for unary models, and on models with bounded languages is decidable subject to a well-known conjecture from number theory. Whilst studied in the context of differential privacy here, the big-O problem, with its similarities to the analysis of algorithms, may have interest outside the field of privacy as a natural approach to whether two systems behave similarly asymptotically, or whether one dominates another in the long run.

In the setting of *randomised circuits*, the complexity of verifying pure differential is fully captured as $\mathbf{coNP}^{\#\mathbf{P}}$-complete; formalising the intuition that differential privacy is universal quantification followed by a condition on probabilities. For the case of approximate differential privacy the problem lies between $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$, and various restrictions on the shape of circuit are shown to make the problem $\mathbf{coNP}^{\#\mathbf{P}}$-complete. The work extends to the verification of any differentially private algorithm with an a priori bound on the length of the computation; providing a general purpose, if perhaps infeasible in practice, verification procedure for a large class of algorithms.

Turning to the problem of approximating the parameters, it is is shown that this cannot be done in polynomial time. This can be seen a significant blow; one may have expected that it could not be done exactly in polynomial time, but the relaxation to approximation may have admitted a speed up. However this result shows that from a complexity perspective, verification is a difficult task, even on restricted models.

Labelled Markov chains and randomised circuits differ in their representation of a probabilistic program. A labelled Markov chain must represent each reachable configuration of the state explicitly in its own state, but can have looping behaviour and can produce output during the computation. Circuits, on the other hand, express the configuration more succinctly, using a binary representation. However circuits are a model with finite state space, requiring that any loop must be unrolled. Due to this finiteness one can be sure of decidability, if necessary by complete exhaustion; the results highlight the extent to which this complete exhaustion is necessary. Many of the complexity classes shown in this thesis are super-polynomial (unless $\mathbf{P} = \mathbf{NP}$), indicating that each of the tasks considered do not have realistically efficient algorithms. In practice it is more important that the privacy of the mechanisms can be verified (or refuted), rather than being able to decide for any possible algorithm. Yet, the theoretical nature of this thesis, studying these problems from the computational complexity perspective, provides further the understanding of the difficulties involved in fully automated procedures for these tasks.

8.1 Open problems and future work

Whilst much has been resolved, this work has brought about new open questions. The following are concrete questions deserving of further investigation.

Estimators of δ As an estimate to δ, three bounds have been provided, bd_α, ld_α and lgd_α. The first, bd_α, can be considered superseded by ld_α, which can still be computed in polynomial time with an **NP** oracle. However, there is an open question as to the gap between ld_α and lgd_α which is computable in polynomial time. This leaves open Conjecture 5.27, that is, is $ld_\alpha = lgd_\alpha$? If not, is there another way to get a polynomial time algorithm for ld_α, or a polynomial time algorithm for an even better estimate of δ?

Extended models The models considered in Chapters 4 and 5 are fully probabilistic, that is, there is no non-determinism. As sketched in Section 5.7 there is scope to extend bisimilarity distances to models which support both probabilistic choice and non-deterministic choice and this could enable the verification of new classes of protocols and algorithms. Questions relating to the estimation of δ and the computation on such distances are of interest in this direction.

The big-O problem In Chapter 6, the big-O problem, and thus questions relating to ε have been considered. In full generality the problems are undecidable, even on labelled Markov chains, but in constrained classes the questions are decidable, even on weighted automata. This leaves open the boundary between decidability and undecidability in the big-O problem. In particular is there a relation to leaktight automata, in the case of probabilistic automata, as in the case of the VALUE-1 problem? In the cases where it is shown that the existence of a constant is undecidable, it is also undecidable to approximate the constant even under the promise that it does exist. It is open whether the constant, that is $\exp(\varepsilon)$, can be approximated in the cases where existence is decidable.

Verification of circuits The complexity of verifying ε-differential privacy in randomised circuits was fully settled to be $\mathbf{coNP}^{\#\mathbf{P}}$-complete. However there is a gap between $\mathbf{coNP}^{\#\mathbf{P}}$ or $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ for (ε, δ)-differential privacy; although certain restrictions on the shape of the formula show $\mathbf{coNP}^{\#\mathbf{P}}$-completeness. Concretely it is open whether the problem is, in full generality, $\mathbf{coNP}^{\#\mathbf{P}}$-complete or $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$-complete, or complete for some class in between? Note that well defined intermediate classes do exists, for instance $\mathbf{coNP}^{\#\mathbf{P}^{\mathbf{NP}}}$ sits between $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$, although it is not even known if $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ are different.

Models of privacy This thesis has conducted its analysis on both *pure* and *approximate* differential privacy. These are the standard versions of privacy which have received the most attention in the literature. However, new definitions of privacy are starting to gain traction which overcome some of the limitations of these models. One can consider the complexity of verifying these models, in particular Renyi-differential privacy [Mir17], which uses a different divergence function instead of the one based on the total variation distance as studied in this work. Concretely, one can study the complexity of computing or bounding Renyi-differential privacy in labelled Markov chains and randomised circuits.

Bibliography

[AACP11] Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. “On the Relation between Differential Privacy and Quantitative Information Flow”. In: *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011*. Vol. 6756. Lecture Notes in Computer Science. Springer, 2011, pp. 60–76. DOI: 10.1007/978-3-642-22012-8_4.

[Abo18] John M Abowd. “The US Census Bureau Adopts Differential Privacy”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2018, pp. 2867–2867.

[AH18] Aws Albarghouthi and Justin Hsu. “Synthesizing coupling proofs of differential privacy”. In: *Proc. ACM Program. Lang.* 2.POPL (2018), 58:1–58:30. DOI: 10.1145/3158146.

[App] Apple. “Apple Differential Privacy Technical Overview”. Online at: https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf Accessed 12/12/2019.

[Arr06] Michael Arrington. “AOL Proudly Releases Massive Amounts of Private Data”. In: *TechCrunch* (2006). Online at: https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/ Accessed 12/12/2019.

[Aut19] Automated Reasoning Group. “BeyondNP.org”. Online at: http://beyondnp.org/pages/solvers/ Accessed 12/12/2019. 2019.

[Bac+19] Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, Radu Mardare, Qiyi Tang, and Franck van Breugel. “Computing Probabilistic Bisimilarity Distances for Probabilistic Automata”. In: *30th International Conference on Concurrency Theory, CONCUR 2019*. 2019, 9:1–9:17. DOI: 10.4230/LIPIcs.CONCUR.2019.9.

[Bar+14] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, César Kunz, and Pierre-Yves Strub. “Proving Differential Privacy in Hoare Logic”. In: *IEEE 27th Computer Security Foundations Symposium, CSF 2014*. IEEE Computer Society, 2014, pp. 411–424. DOI: 10.1109/CSF.2014.36.

[Bar+15a] Gilles Barthe, Thomas Espitau, Benjamin Grégoire, Justin Hsu, Léo Stefanesco, and Pierre-Yves Strub. “Relational Reasoning via Probabilistic Coupling”. In: *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015*. Vol. 9450. Lecture Notes in Computer Science. Springer, 2015, pp. 387–401. DOI: 10.1007/978-3-662-48899-7_27.

[Bar+15b] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. “Higher-Order Approximate Relational Refinement Types for Mechanism Design and Differential Privacy”. In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015*. ACM, 2015, pp. 55–68. DOI: 10.1145/2676726.2677000.

[Bar+16a] Gilles Barthe, Noémie Fong, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. “Advanced Probabilistic Couplings for Differential Privacy”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 55–67. DOI: 10.1145/2976749.2978391.

[Bar+16b] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. “Proving Differential Privacy via Probabilistic Couplings”. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016*. ACM, 2016, pp. 749–758. DOI: 10.1145/2933575.2934554.

[Bar+20] Gilles Barthe, Rohit Chadha, Vishal Jagannath, A. Prasad Sistla, and Mahesh Viswanathan. “Deciding Differential Privacy for Programs with Finite Inputs and Outputs”. In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2020*. Arxiv pre-print: http://arxiv.org/abs/1910.04137. ACM, 2020, pp. 141–154. DOI: 10.1145/3373718.3394796.

[Bar12] Daniel Barth-Jones. “The’re-identification’of Governor William Weld’s medical information: a critical re-examination of health data identification risks and privacy protections, then and now”. In: *Then and Now (July 2012)* (2012).

[BBLM13] Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. “On-the-Fly Exact Computation of Bisimilarity Distances”. In: *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013*. Vol. 7795. Lecture Notes in Computer Science. Springer, 2013, pp. 1–15. DOI: 10.1007/978-3-642-36742-7_1.

[BBLM17] Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. “On-the-Fly Computation of Bisimilarity Distances”. In: *Logical Methods in Computer Science* 13.2 (2017). DOI: 10.23638/LMCS-13(2:13)2017.

[BGB09] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. “Formal certification of code-based cryptographic proofs”. In: *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009*. ACM, 2009, pp. 90–101. DOI: 10.1145/1480881.1480894.

[BGHP16] Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin C. Pierce. “Programming language techniques for differential privacy”. In: *SIGLOG News* 3.1 (2016), pp. 34–53. URL: https://dl.acm.org/citation.cfm?id=2893591.

[Bic+18] Benjamin Bichsel, Timon Gehr, Dana Drachsler-Cohen, Petar Tsankov, and Martin T. Vechev. “DP-Finder: Finding Differential Privacy Violations by Sampling and Optimization”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018*. ACM, 2018, pp. 508–524. DOI: 10.1145/3243734.3243863.

[Bil86] Patrick Billingsley. “Probability and Measure”. 2nd. John Wiley and Sons, 1986.

[BK08] Christel Baier and Joost-Pieter Katoen. “Principles of model checking”. MIT Press, 2008. ISBN: 978-0-262-02649-9.

[BKOB12] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. “Probabilistic relational reasoning for differential privacy”. In: *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012*. ACM, 2012, pp. 97–110. DOI: 10.1145/2103656.2103670.

[BKOB13] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. "Probabilistic Relational Reasoning for Differential Privacy". In: *ACM Trans. Program. Lang. Syst.* 35.3 (2013), 9:1–9:49. DOI: 10.1145/2492061.

[BO13] Gilles Barthe and Federico Olmedo. "Beyond Differential Privacy: Composition Theorems and Relational Logic for f-divergences between Probabilistic Programs". In: *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013.* Vol. 7966. Lecture Notes in Computer Science. Springer, 2013, pp. 49–60. DOI: 10.1007/978-3-642-39212-2_8.

[BPR05] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. "Betti number bounds, applications and algorithms". In: *Current trends in combinatorial and computational geometry: papers from the special program at MSRI* 52 (2005), pp. 87–97.

[Bre17] Franck van Breugel. "Probabilistic Bisimilarity Distances". In: *ACM SIGLOG News* 4.4 (Nov. 2017), pp. 33–51. DOI: 10.1145/3157831.3157837.

[BSW07] Franck van Breugel, Babita Sharma, and James Worrell. "Approximating a Behavioural Pseudometric Without Discount for Probabilistic Systems". In: *Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007.* Vol. 4423. Lecture Notes in Computer Science. Springer, 2007, pp. 123–137. DOI: 10.1007/978-3-540-71389-0_10.

[BSW08] Franck van Breugel, Babita Sharma, and James Worrell. "Approximating a Behavioural Pseudometric without Discount for Probabilistic Systems". In: *Logical Methods in Computer Science* 4.2 (2008). DOI: 10.2168/LMCS-4(2:2)2008.

[BW01] Franck van Breugel and James Worrell. "An Algorithm for Quantitative Verification of Probabilistic Transition Systems". In: *Concurrency Theory, 12th International Conference, CONCUR 2001.* Vol. 2154. Lecture Notes in Computer Science. Springer, 2001, pp. 336–350. DOI: 10.1007/3-540-44685-0_23.

[BW14] Franck van Breugel and James Worrell. "The Complexity of Computing a Bisimilarity Pseudometric on Probabilistic Automata". In: *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday.* Vol. 8464. Lecture Notes in Computer Science. Springer, 2014, pp. 191–213. DOI: 10.1007/978-3-319-06880-0_10.

[Can88] John F. Canny. "Some Algebraic and Geometric Computations in PSPACE". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988.* ACM, 1988, pp. 460–467. DOI: 10.1145/62212.62257.

[Car15] Jean Cardinal. "Computational Geometry Column 62". In: *SIGACT News* 46.4 (2015), pp. 69–78. DOI: 10.1145/2852040.2852053.

[CBW12] Di Chen, Franck van Breugel, and James Worrell. "On the Complexity of Computing Probabilistic Bisimilarity". In: *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012.* Vol. 7213. Lecture Notes in Computer Science. Springer, 2012, pp. 437–451. DOI: 10.1007/978-3-642-28729-9_29.

[CDM17] Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. "Approximate counting in SMT and value estimation for probabilistic programs". In: *Acta Inf.* 54.8 (2017), pp. 729–764. DOI: 10.1007/s00236-017-0297-2.

[CGPX14] Konstantinos Chatzikokolakis, Daniel Gebler, Catuscia Palamidessi, and Lili Xu. "Generalized Bisimulation Metrics". In: *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014*. Vol. 8704. Lecture Notes in Computer Science. Springer, 2014, pp. 32–46. DOI: 10.1007/978-3-662-44584-6_4.

[Cha88] David Chaum. "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability". In: *J. Cryptology* 1.1 (1988), pp. 65–75. DOI: 10.1007/BF00206326.

[Che+19] Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. "Distributed Differential Privacy via Shuffling". In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 245. URL: https://eprint.iacr.org/2019/245.

[Che08] Steve Cheng. "A Crash Course on the Lebesgue Integral and Measure Theory". 2008.

[Chr86] Marek Chrobak. "Finite Automata and Unary Languages". In: *Theor. Comput. Sci.* 47.3 (1986), pp. 149–158. DOI: 10.1016/0304-3975(86)90142-8.

[CK14] Taolue Chen and Stefan Kiefer. "On the total variation distance of labelled Markov chains". In: *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014*. ACM, 2014, 33:1–33:10. DOI: 10.1145/2603088.2603099.

[CKMP20] Dmitry Chistikov, Stefan Kiefer, Andrzej S Murawski, and David Purser. "The Big-O Problem for Labelled Markov Chains and Weighted Automata". Manuscript in preparation. 2020.

[CKV14a] Rohit Chadha, Dileep Kini, and Mahesh Viswanathan. "Decidable Problems for Unary PFAs". In: *Quantitative Evaluation of Systems - 11th International Conference, QEST 2014*. Vol. 8657. Lecture Notes in Computer Science. Springer, 2014, pp. 329–344. DOI: 10.1007/978-3-319-10696-0_26.

[CKV14b] Rohit Chadha, Dileep Kini, and Mahesh Viswanathan. "Quantitative Information Flow in Boolean Programs". In: *Principles of Security and Trust - Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014*. Vol. 8414. Lecture Notes in Computer Science. Springer, 2014, pp. 103–119. DOI: 10.1007/978-3-642-54792-8_6.

[CMP18] Dmitry Chistikov, Andrzej S. Murawski, and David Purser. "Bisimilarity Distances for Approximate Differential Privacy". In: *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018*. Vol. 11138. Lecture Notes in Computer Science. Full version with proofs can be found at https://arxiv.org/abs/1807.10015. Springer, 2018, pp. 194–210. DOI: 10.1007/978-3-030-01090-4_12.

[CMP19] Dmitry Chistikov, Andrzej S. Murawski, and David Purser. "Asymmetric Distances for Approximate Differential Privacy". In: *30th International Conference on Concurrency Theory, CONCUR 2019*. Vol. 140. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 10:1–10:17. DOI: 10.4230/LIPIcs.CONCUR.2019.10.

[Coh13] Henri Cohen. "A course in computational algebraic number theory". Vol. 138. Springer Science & Business Media, 2013.

[Col15] Thomas Colcombet. “Unambiguity in Automata Theory”. In: *Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015.* Vol. 9118. Lecture Notes in Computer Science. Springer, 2015, pp. 3–18. DOI: `10.1007/978-3-319-19225-3_1`.

[CP10] Konstantinos Chatzikokolakis and Catuscia Palamidessi. “Making random choices invisible to the scheduler”. In: *Inf. Comput.* 208.6 (2010), pp. 694–715. DOI: `10.1016/j.ic.2009.06.006`.

[DAn+13] Loris D’Antoni, Marco Gaboardi, Emilio Jesús Gallego Arias, Andreas Haeberlen, and Benjamin C. Pierce. “Sensitivity analysis using type-based constraints”. In: *Proceedings of the 1st annual workshop on Functional Programming Concepts in Domain-Specific Languages, FPCDSL@ICFP 2013.* ACM, 2013, pp. 43–50. DOI: `10.1145/2505351.2505353`.

[DD09] Yuxin Deng and Wenjie Du. “The Kantorovich Metric in Computer Science: A Brief Survey”. In: *Electr. Notes Theor. Comput. Sci.* 253.3 (2009), pp. 73–82. DOI: `10.1016/j.entcs.2009.10.006`.

[DGJP04] Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. “Metrics for labelled Markov processes”. In: *Theor. Comput. Sci.* 318.3 (2004), pp. 323–354. DOI: `10.1016/j.tcs.2003.09.013`.

[Dif17] Differential Privacy Team, Apple. “Learning with privacy at scale”. Online at: `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf` Accessed 12/12/2019. 2017.

[Din+18] Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. “Detecting Violations of Differential Privacy”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018.* ACM, 2018, pp. 475–489. DOI: `10.1145/3243734.3243818`.

[DJGP02] Josee Desharnais, Radha Jagadeesan, Vineet Gupta, and Prakash Panangaden. “The Metric Analogue of Weak Bisimulation for Probabilistic Processes”. In: *17th IEEE Symposium on Logic in Computer Science, LICS 2002.* IEEE Computer Society, 2002, pp. 413–422. DOI: `10.1109/LICS.2002.1029849`.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. “Calibrating Noise to Sensitivity in Private Data Analysis”. In: *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006.* Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 265–284. DOI: `10.1007/11681878_14`.

[DR14] Cynthia Dwork and Aaron Roth. “The Algorithmic Foundations of Differential Privacy”. In: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (2014), pp. 211–407. DOI: `10.1561/0400000042`.

[DR16] Cynthia Dwork and Guy N. Rothblum. “Concentrated Differential Privacy”. In: *CoRR* abs/1603.01887 (2016). URL: `http://arxiv.org/abs/1603.01887`.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. “Boosting and Differential Privacy”. In: *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010.* IEEE Computer Society, 2010, pp. 51–60. DOI: `10.1109/FOCS.2010.12`.

[Dwo+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. "Our Data, Ourselves: Privacy Via Distributed Noise Generation". In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Vol. 4004. Lecture Notes in Computer Science. Springer, 2006, pp. 486–503. DOI: 10.1007/11761679_29.

[Dwo+09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. "On the complexity of differentially private data release: efficient algorithms and hardness results". In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*. ACM, 2009, pp. 381–390. DOI: 10.1145/1536414.1536467.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014*. ACM, 2014, pp. 1054–1067. DOI: 10.1145/2660267.2660348.

[Erl+19] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. "Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*. SIAM, 2019, pp. 2468–2479. DOI: 10.1137/1.9781611975482.151.

[ESS15] Hamid Ebadi, David Sands, and Gerardo Schneider. "Differential Privacy: Now it's Getting Personal". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015*. ACM, 2015, pp. 69–81. DOI: 10.1145/2676726.2677005.

[EY10] Kousha Etessami and Mihalis Yannakakis. "On the Complexity of Nash Equilibria and Other Fixed Points". In: *SIAM J. Comput.* 39.6 (2010), pp. 2531–2597. DOI: 10.1137/080720826.

[FGO12] Nathanaël Fijalkow, Hugo Gimbert, and Youssouf Oualhadj. "Deciding the Value 1 Problem for Probabilistic Leaktight Automata". In: *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012*. IEEE Computer Society, 2012, pp. 295–304. DOI: 10.1109/LICS.2012.40.

[Fij17] Nathanaël Fijalkow. "Undecidability results for probabilistic automata". In: *SIGLOG News* 4.4 (2017), pp. 10–17. URL: https://dl.acm.org/citation.cfm?id=3157833.

[FJ14] Matthew Fredrikson and Somesh Jha. "Satisfiability modulo counting: a new approach for analyzing privacy properties". In: *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014*. ACM, 2014, 42:1–42:10. DOI: 10.1145/2603088.2603097.

[FRZ11] Peter Franek, Stefan Ratschan, and Piotr Zgliczynski. "Satisfiability of Systems of Equations of Real Analytic Functions Is Quasi-decidable". In: *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011*. Vol. 6907. Lecture Notes in Computer Science. Springer, 2011, pp. 315–326. DOI: 10.1007/978-3-642-22993-0_30.

[FS80] Shmuel Friedland and Hans Schneider. "The Growth of Powers of a Nonnegative Matrix". In: *SIAM J. Matrix Analysis Applications* 1.2 (1980), pp. 185–200. DOI: 10.1137/0601022.

[Gab+13] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. "Linear dependent types for differential privacy". In: *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2013*. ACM, 2013, pp. 357–370. DOI: `10.1145/2429069.2429113`.

[Gad+19] Andrea Gadotti, Florimond Houssiau, Luc Rocher, Benjamin Livshits, and Yves-Alexandre de Montjoye. "When the Signal is in the Noise: Exploiting Diffix's Sticky Noise". In: *28th USENIX Security Symposium, USENIX Security 2019*. USENIX Association, 2019, pp. 1081–1098. URL: `https://www.usenix.org/conference/usenixsecurity19/presentation/gadotti`.

[Gir12] Davide Giraudo. "Approximating a sigma-algebra by a generating algebra". In: *StackExchange* (2012). Online at: `https://math.stackexchange.com/questions/228998/approximating-a-sigma-algebra-by-a-generating-algebra` Accessed 17/12/2019.

[GJS90] Alessandro Giacalone, Chi-Chang Jou, and Scott A. Smolka. "Algebraic Reasoning for Probabilistic Concurrent Systems". In: *Programming concepts and methods: Proceedings of the IFIP Working Group 2.2, 2.3 Working Conference on Programming Concepts and Methods, 1990*. North-Holland, 1990, pp. 443–458. ISBN: 0-444-88545-5.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. "Geometric Algorithms and Combinatorial Optimization". Vol. 2. Algorithms and Combinatorics. Springer, 1988. DOI: `10.1007/978-3-642-97881-4`.

[GM18] Anna C. Gilbert and Audra McMillan. "Property Testing For Differential Privacy". In: *56th Annual Allerton Conference on Communication, Control, and Computing, 2018*. IEEE, 2018, pp. 249–258. DOI: `10.1109/ALLERTON.2018.8636068`.

[GNP20] Marco Gaboardi, Kobbi Nissim, and David Purser. "The Complexity of Verifying Loop-Free Programs as Differentially Private". In: *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020 (to appear)*. 2020. URL: `http://arxiv.org/abs/1911.03272`.

[GS02] Alison L Gibbs and Francis Edward Su. "On choosing and bounding probability metrics". In: *International statistical review* 70.3 (2002), pp. 419–435.

[GS64] Seymour Ginsburg and Edwin H Spanier. "Bounded ALGOL-like languages". In: *Transactions of the American Mathematical Society* 113.2 (1964), pp. 333–368.

[Gup+18] Neha Gupta, Henry Crosby, David Purser, Stephen A. Jarvis, and Weisi Guo. "Twitter Usage Across Industry: A Spatiotemporal Analysis". In: *Fourth IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2018*. 2018, pp. 64–71. DOI: `10.1109/BigDataService.2018.00018`.

[Hal74] Paul R Halmos. "Measure theory". Vol. 18. Springer, 1974.

[Hoa69] Charles Antony Richard Hoare. "An axiomatic basis for computer programming". In: *Communications of the ACM* 12.10 (1969), pp. 576–580.

[HPN11] Andreas Haeberlen, Benjamin C. Pierce, and Arjun Narayan. "Differential Privacy Under Fire". In: *20th USENIX Security Symposium 2011*. USENIX Association, 2011. URL: `http://static.usenix.org/events/sec11/tech/full_papers/Haeberlen.pdf`.

[Hsu+14] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C. Pierce, and Aaron Roth. “Differential Privacy: An Economic Method for Choosing Epsilon”. In: *IEEE 27th Computer Security Foundations Symposium, CSF 2014*. IEEE Computer Society, 2014, pp. 398–410. DOI: 10.1109/CSF.2014.35.

[HT03] Thanh Minh Hoang and Thomas Thierauf. “The complexity of the characteristic and the minimal polynomial”. In: *Theor. Comput. Sci.* 295 (2003), pp. 205–222. DOI: 10.1016/S0304-3975(02)00404-8.

[IRS76] Harry B. Hunt III, Daniel J. Rosenkrantz, and Thomas G. Szymanski. “On the Equivalence, Containment, and Covering Problems for the Regular and Context-Free Languages”. In: *J. Comput. Syst. Sci.* 12.2 (1976), pp. 222–268. DOI: 10.1016/S0022-0000(76)80038-4.

[JL91] Bengt Jonsson and Kim Guldstrand Larsen. “Specification and Refinement of Probabilistic Processes”. In: *Proceedings of the Sixth Annual Symposium on Logic in Computer Science, LICS 1991*. IEEE Computer Society, 1991, pp. 266–277. DOI: 10.1109/LICS.1991.151651.

[JP19] Petr Jancar and David Purser. “Structural liveness of Petri nets is ExpSpace-hard and decidable”. In: *Acta Inf.* 56.6 (2019), pp. 537–552. DOI: 10.1007/s00236-019-00338-6.

[Kan42] L. V. Kantorovich. “On the translocation of masses”. In: *Doklady Akademii Nauk SSSR* 37(7-8) (1942), pp. 227–229.

[Kie+13] Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. “On the Complexity of Equivalence and Minimisation for Q-weighted Automata”. In: *Logical Methods in Computer Science* 9.1 (2013). DOI: 10.2168/LMCS-9(1:8)2013.

[Kie18] Stefan Kiefer. “On Computing the Total Variation Distance of Hidden Markov Models”. In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 130:1–130:13. DOI: 10.4230/LIPIcs.ICALP.2018.130.

[KMT17] Markus Krötzsch, Tomás Masopust, and Michaël Thomazo. “Complexity of universality and related problems for partially ordered NFAs”. In: *Inf. Comput.* 255 (2017), pp. 177–192. DOI: 10.1016/j.ic.2017.06.004.

[Koz92] Dexter C. Kozen. “The Design and Analysis of Algorithms”. In: New York, NY: Springer New York, 1992. Chap. Integer Arithmetic in NC, pp. 160–165. DOI: 10.1007/978-1-4612-4400-4_30.

[KSM95] Sampath Kannan, Z. Sweedyk, and Stephen R. Mahaney. “Counting and Random Generation of Strings in Regular Languages”. In: *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 1995*. ACM/SIAM, 1995, pp. 551–557. URL: http://dl.acm.org/citation.cfm?id=313651.313803.

[Lad89] Richard E. Ladner. “Polynomial Space Counting Problems”. In: *SIAM J. Comput.* 18.6 (1989), pp. 1087–1097. DOI: 10.1137/0218073.

[LGM98] Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. “The Computational Complexity of Probabilistic Planning”. In: *J. Artif. Intell. Res.* 9 (1998), pp. 1–36. DOI: 10.1613/jair.505.

[LS89] Kim G Larsen and Arne Skou. “Bisimulation through probabilistic testing (preliminary report)”. In: *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM. 1989, pp. 344–352.

[LS91] Kim Guldstrand Larsen and Arne Skou. "Bisimulation through Probabilistic Testing". In: *Inf. Comput.* 94.1 (1991), pp. 1–28. DOI: 10.1016/0890-5401(91)90030-6.

[LSL17] Min Lyu, Dong Su, and Ninghui Li. "Understanding the Sparse Vector Technique for Differential Privacy". In: *PVLDB* 10.6 (2017), pp. 637–648. DOI: 10.14778/3055330.3055331.

[LWZ18] Depeng Liu, Bow-Yaw Wang, and Lijun Zhang. "Model Checking Differentially Private Properties". In: *Programming Languages and Systems - 16th Asian Symposium, APLAS 2018.* Vol. 11275. Lecture Notes in Computer Science. Springer, 2018, pp. 394–414. DOI: 10.1007/978-3-030-02768-1_21.

[Mac+08] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke, and Lars Vilhuber. "Privacy: Theory meets Practice on the Map". In: *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008.* IEEE Computer Society, 2008, pp. 277–286. DOI: 10.1109/ICDE.2008.4497436.

[Mar02] Andrew Martinez. "Efficient Computation of Regular Expressions from Unary NFAs". In: *Fourth International Workshop on Descriptional Complexity of Formal Systems - DCFS 2002.* Vol. Report No. 586. Department of Computer Science, The University of Western Ontario, Canada, 2002, pp. 174–187.

[McS09] Frank McSherry. "Privacy integrated queries: an extensible platform for privacy-preserving data analysis". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009.* ACM, 2009, pp. 19–30. DOI: 10.1145/1559845.1559850.

[Mei18] Sebastian Meiser. "Approximate and Probabilistic Differential Privacy Definitions". In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 277. URL: https://eprint.iacr.org/2018/277.

[Mil89] R. Milner. "Communication and Concurrency". Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989. ISBN: 0131149849.

[Mir12] Ilya Mironov. "On significance of the least significant bits for differential privacy". In: *The ACM Conference on Computer and Communications Security, CCS 2012.* ACM, 2012, pp. 650–661. DOI: 10.1145/2382196.2382264.

[Mir17] Ilya Mironov. "Rényi differential privacy". In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF).* IEEE. 2017, pp. 263–275.

[MV16] Jack Murtagh and Salil P. Vadhan. "The Complexity of Computing the Optimal Composition of Differential Privacy". In: *Theory of Cryptography - 13th International Conference, TCC 2016.* Vol. 9562. Lecture Notes in Computer Science. Springer, 2016, pp. 157–175. DOI: 10.1007/978-3-662-49096-9_7.

[MW96] Angus Macintyre and Alex J Wilkie. "On the decidability of the real exponential field". 1996.

[NS06] Arvind Narayanan and Vitaly Shmatikov. "How To Break Anonymity of the Netflix Prize Dataset". In: *CoRR* abs/cs/0610105 (2006). URL: http://arxiv.org/abs/cs/0610105.

[OW14] Joël Ouaknine and James Worrell. "Positivity Problems for Low-Order Linear Recurrence Sequences". In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014.* SIAM, 2014, pp. 366–379. DOI: 10.1137/1.9781611973402.27.

[Pan96] Victor Y Pan. “Optimal and nearly optimal algorithms for approximating polynomial zeros”. In: *Computers & Mathematics with Applications* 31.12 (1996), pp. 97–138.

[Pap+17] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. “Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data”. In: *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, 2017. URL: `https://openreview.net/forum?id=HkwoSDPgg`.

[Paz14] Azaria Paz. “Introduction to probabilistic automata”. Academic Press, 2014.

[PD08] Knot Pipatsrisawat and Adnan Darwiche. “A New Algorithm for Computing Upper Bounds for Functional E-MAJSAT”. Tech. rep. Tech. Rep. D–156, Automated Reasoning Group, Computer Science Department, UCLA, 2008.

[PF79] Nicholas Pippenger and Michael J. Fischer. “Relations Among Complexity Measures”. In: *J. ACM* 26.2 (1979), pp. 361–381. DOI: `10.1145/322123.322138`.

[Rav19] Vishal Jagannath Ravi. “Automated methods for checking differential privacy”. M.S. Thesis. University of Illinois at Urbana-Champaign, Illinois, USA, 2019. URL: `http://hdl.handle.net/2142/104913`.

[Ren92] James Renegar. “On the computational complexity and geometry of the first-order theory of the reals. Part I, II, III”. In: *Journal of symbolic computation* 13.3 (1992), pp. 255–352.

[Roy+10] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. “Airavat: Security and Privacy for MapReduce”. In: *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010*. USENIX Association, 2010, pp. 297–312. URL: `http://www.usenix.org/events/nsdi10/tech/full_papers/roy.pdf`.

[RP10] Jason Reed and Benjamin C. Pierce. “Distance makes the types grow stronger: a calculus for differential privacy”. In: *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010*. ACM, 2010, pp. 157–168. DOI: `10.1145/1863543.1863568`.

[RS42] John Riordan and Claude E Shannon. “The number of two-terminal series-parallel networks”. In: *Journal of Mathematics and Physics* 21.1-4 (1942), pp. 83–93.

[SA19] Calvin Smith and Aws Albarghouthi. “Synthesizing differentially private programs”. In: *PACMPL* 3.ICFP (2019), 94:1–94:29. DOI: `10.1145/3341698`.

[Sch61] Marcel Paul Schützenberger. “On the Definition of a Family of Automata”. In: *Information and Control* 4.2-3 (1961), pp. 245–270. DOI: `10.1016/S0019-9958(61)80020-X`.

[Sch86] Hans Schneider. “The influence of the marked reduced graph of a nonnegative matrix on the Jordan form and on related properties: A survey”. In: *Linear Algebra and its Applications* 84 (1986), pp. 161–189.

[Sch99] Alexander Schrijver. “Theory of linear and integer programming”. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999. ISBN: 978-0-471-98232-6.

[Ser13] Bruno Sericola. “Markov chains: theory and applications”. John Wiley & Sons, 2013.

[SJ05] Zdenek Sawa and Petr Jancar. “Behavioural Equivalences on Finite-State Systems are PTIME-hard”. In: *Computers and Artificial Intelligence* 24.5 (2005), pp. 513–528.

[SM73] Larry J. Stockmeyer and Albert R. Meyer. “Word Problems Requiring Exponential Time: Preliminary Report”. In: *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, 1973.* ACM, 1973, pp. 1–9. DOI: `10.1145/800125.804029`.

[Smi08] Adam D. Smith. “Efficient, Differentially Private Point Estimators”. In: *CoRR* abs/0809.4794 (2008). URL: `http://arxiv.org/abs/0809.4794`.

[Son85] Eduardo D. Sontag. “Real Addition and the Polynomial Hierarchy”. In: *Inf. Process. Lett.* 20.3 (1985), pp. 115–120. DOI: `10.1016/0020-0190(85)90076-6`.

[SS17] Marcus Schaefer and Daniel Stefankovic. “Fixed Points, Nash Equilibria, and the Existential Theory of the Reals”. In: *Theory Comput. Syst.* 60.2 (2017), pp. 172–193. DOI: `10.1007/s00224-015-9662-0`.

[SY13] Entong Shen and Ting Yu. “Mining frequent graph patterns with differential privacy”. In: *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013.* ACM, 2013, pp. 545–553. DOI: `10.1145/2487575.2487601`.

[Tar55] Alfred Tarski. “A lattice-theoretical fixpoint theorem and its applications”. In: *Pacific Journal of Mathematics* 5.2 (1955), pp. 285–309.

[TB16] Qiyi Tang and Franck van Breugel. “Computing Probabilistic Bisimilarity Distances via Policy Iteration”. In: *27th International Conference on Concurrency Theory, CONCUR 2016.* Vol. 59. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 22:1–22:15. DOI: `10.4230/LIPIcs.CONCUR.2016.22`.

[TB17] Qiyi Tang and Franck van Breugel. “Algorithms to Compute Probabilistic Bisimilarity Distances for Labelled Markov Chains”. In: *28th International Conference on Concurrency Theory, CONCUR 2017.* 2017, 27:1–27:16. DOI: `10.4230/LIPIcs.CONCUR.2017.27`.

[TB18] Qiyi Tang and Franck van Breugel. “Deciding Probabilistic Bisimilarity Distance One for Probabilistic Automata”. In: *29th International Conference on Concurrency Theory, CONCUR 2018.* 2018, 9:1–9:17. DOI: `10.4230/LIPIcs.CONCUR.2018.9`.

[TKD11] Michael Carl Tschantz, Dilsun Kirli Kaynar, and Anupam Datta. “Formal Verification of Differential Privacy for Interactive Systems (Extended Abstract)”. In: *Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics, MFPS 2011.* Vol. 276. Electronic Notes in Theoretical Computer Science. Elsevier, 2011, pp. 61–79. DOI: `10.1016/j.entcs.2011.09.015`.

[To09] Anthony Widjaja To. “Unary finite automata vs. arithmetic progressions”. In: *Inf. Process. Lett.* 109.17 (2009), pp. 1010–1014. DOI: `10.1016/j.ipl.2009.06.005`.

[Tod91] Seinosuke Toda. “PP is as Hard as the Polynomial-Time Hierarchy”. In: *SIAM J. Comput.* 20.5 (1991), pp. 865–877. DOI: `10.1137/0220053`.

[Tro14] J.K. Trotter. “Public NYC Taxicab Database Lets You See How Celebrities Tip”. In: *Gawker* (2014). Online at: `https://gawker.com/the-public-nyc-taxicab-database-that-accidentally-track-1646724546` Accessed 12/12/2019.

[Tur37] Alan Mathison Turing. “On computable numbers, with an application to the Entscheidungsproblem”. In: *Proceedings of the London mathematical society* 2.1 (1937), pp. 230–265.

[Tze92] Wen-Guey Tzeng. "A Polynomial-Time Algorithm for the Equivalence of Probabilistic Automata". In: *SIAM J. Comput.* 21.2 (1992), pp. 216–227. DOI: 10.1137/0221017.

[US 19] US Census Bureau. "OnTheMap project". Online at: https://onthemap.ces.census.gov Accessed 12/12/2019. 2019.

[Vad17] Salil P. Vadhan. "The Complexity of Differential Privacy". In: *Tutorials on the Foundations of Cryptography.* Ed. by Yehuda Lindell. Springer International Publishing, 2017, pp. 347–450. DOI: 10.1007/978-3-319-57048-8_7.

[VRG20] Elisabet Lobo Vesga, Alejandro Russo, and Marco Gaboardi. "A Programming Framework for Differential Privacy with Accuracy Concentration Bounds". In: *41st IEEE Symposium on Security and Privacy.* 2020. URL: http://arxiv.org/abs/1909.07918.

[Wag86] Klaus W. Wagner. "The Complexity of Combinatorial Problems with Succinct Input Representation". In: *Acta Inf.* 23.3 (1986), pp. 325–356. DOI: 10.1007/BF00289117.

[War65] Stanley L Warner. "Randomized response: A survey technique for eliminating evasive answer bias". In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69.

[XCL14] Lili Xu, Konstantinos Chatzikokolakis, and Huimin Lin. "Metrics for Differential Privacy in Concurrent Systems". In: *Formal Techniques for Distributed Objects, Components, and Systems. FORTE 2014.* Vol. 8461. Lecture Notes in Computer Science. Springer, 2014, pp. 199–215. DOI: 10.1007/978-3-662-43613-4_13.

[Xu15] Lili Xu. "Formal Verification of Differential Privacy in Concurrent Systems." PhD thesis. École Polytechnique, Palaiseau, France, 2015. URL: https://tel.archives-ouvertes.fr/tel-01384363.

[YT10] Hirotoshi Yasuoka and Tachio Terauchi. "Quantitative Information Flow - Verification Hardness and Possibilities". In: *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010.* IEEE Computer Society, 2010, pp. 15–27. DOI: 10.1109/CSF.2010.9.

[YT11] Hirotoshi Yasuoka and Tachio Terauchi. "On bounding problems of quantitative information flow". In: *Journal of Computer Security* 19.6 (2011), pp. 1029–1082. DOI: 10.3233/JCS-2011-0437.

[ZK17] Danfeng Zhang and Daniel Kifer. "LightDP: towards automating differential privacy proofs". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017.* ACM, 2017, pp. 888–901. URL: http://dl.acm.org/citation.cfm?id=3009884.