Université de Montréal

De computatione quantica

par
José Manuel Fernandez

Département d'Informatique et recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique et recherche opérationnelle

Décembre 2003

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée:

De computatione quantica

présentée par:

José Manuel Fernandez

a été évaluée par un jury composé des personnes suivantes:

M. François MAJOR,	président-rapporteur
M. Gilles BRASSARD,	directeur de recherche
M. Alain TAPP,	membre du jury
M. Raymond LAFLAMME,	examinateur externe
M. François MAJOR,	représentant du doyen de la FÉS

Thèse acceptée le: ....

RÉSUMÉ

Cette thèse traite du calcul quantique, dans ses aspects théoriques et expérimentaux. La théorie du calcul quantique est une généralisation de la théorie du calcul standard inspirée par les principes de la mécanique quantique. La découverte d'algorithmes quantiques efficaces pouvant résoudre des problèmes pour lesquels il ne semble pas y avoir d'algorithmes classiques performants remet en question la thèse forte de Church-Turing, qui énonce que tous les modèles de calcul sont essentiellement équivalents en ce qui concerne ce qu'ils peuvent calculer et avec quelle efficacité.

Dans la première partie de cette thèse, nous étudions la nature de cette différence. Nous introduisons tout premièrement un cadre général et simplifié pour caractériser l'essentiel d'une théorie quantique, en comparaison avec un théorie déterministe ou probabiliste. Nous avançons la thèse que les axiomes les plus fondamentaux sont ceux reliés à la mesure, et que c'est à partir d'eux que ces différences sont engendrées. Nous couvrons certaines variations sur ces théories et démontrons quelques relations structurelles intéressantes qui les concernent. Par exemple, les seuls modèles qui sont simultanément quantiques et probabilistes sont les modèles déterministes.

Nous utilisons ce méta-modèle pour ré-introduire de façon succincte et uniforme les modèles de calcul déterministes, probabilistes et quantiques de machine de Turing et de circuit. De plus, nous généralisons le modèle de circuit sur des demi-anneaux arbitraires. Nous réussissons ainsi à fournir une nouvelle classification de classes de complexité existantes en variant le choix du demi-anneaux et de norme vectorielle sur les espaces vectoriels d'états sur lesquelles elles sont définies. En particulier, les modèles déterministes, probabilistes et quantiques standards peuvent être caractérisés (avec la même norme) en définissant des circuits sur l'algèbre booléenne, les rationaux ou réels positifs et les rationaux ou réels, respectivement. Nous explorons aussi ce modèle avec d'autres demi-anneaux non-standards. Nous démontrons que les modèles basés sur les quaternions sont équivalent au calcul quantique. Finalement, nous renforçons cette «hiérarchie algébrique» en utilisant le formalisme des formules tensorielles pour construire une famille de problèmes complets (ou complets avec promesse) pour les classes de complexité correspondantes.

La deuxième partie concerne le calcul quantique expérimental par résonance magnétique nucléaire (RMN). Un important obstacle de cette démarche est l'incapacité d'initialiser correctement le registre de mémoire quantique, un problème relié à celui du rapport signal-bruit en spectroscopie par RMN. Une des techniques les plus prometteuses pour le résoudre est le refroidissement algorithmique, une généralisation des techniques de transfert de polarisation déjà utilisées par les spectroscopistes en RMN. Nous discutons les procédés adiabatiques traditionnelles ainsi que leurs limitations. Nous décrivons une variation sur cette technique, l'approche non-adiabatique, qui utilise l'environnement pour refroidir au delà de ces limites. En particulier, nous décrivons un nouvel algorithme efficace de refroidissement algorithmique qui pourrait atteindre des températures de spin de presque 0^o K à en RMN à l'état liquide avec des registres de taille déjà plus raisonnable (30–60 spins). Finalement, nous faisons part de la réalisation en laboratoire de la toute première expérience réussie de refroidissement algorithmique non-adiabatique. Ceci constitue, nous l'espérons, un premier pas vers le développement complet de cette technique prometteuse, avec des applications bien au delà du calcul quantique par RMN.

Mots clés : Calcul quantique, théorie du calcul, théorie de complexité du calcul, théorie de complexité du calcul quantique, calcul quantique expérimental, résonance magnétique nucléaire (RMN), transfert de polarisation, refroidissement algorithmique, refroidissement algorithmique non-adiabatique.

ABSTRACT

This thesis covers the topic of Quantum Computation, in both its theoretical and experimental aspects. The Theory of Quantum Computing is an extension of the standard Theory of Computation inspired by the principles of Quantum Mechanics. The discovery of efficient quantum algorithms for problems for which no efficient classical algorithms are known challenges the Strong Church-Turing Thesis, which states that all computational models are essentially equivalent in terms of what they can compute and how efficiently they can do so.

In Part I of this thesis, we study the nature of this difference. We first introduce a general and simplified framework for characterising the essence of a quantum theory, in contrast with a deterministic or a probabilistic theory. We put forth the thesis that the most fundamental axioms are those related to measurement, from which these differences emanate. We cover some variations on these theories, and show some interesting structural relationships between them. For example, the only models which can be simultaneously quantum and probabilistic are the deterministic ones.

We then use this meta-model to re-introduce in a succinct and uniform manner the deterministic, probabilistic and quantum versions of the Turing Machine and circuit computational models. Moreover, we generalise the circuit model on arbitrary semirings. We thus succeed in providing a new classification of existing complexity classes by varying the semirings and vector norms with which their vector space of states is defined. In particular, the standard models of deterministic, probabilistic, and quantum computing can be characterised (with the same norm) by defining circuits on the boolean algebra, the positive rational or reals, and the rational or reals, respectively. We further explore this model with other non-standard semirings. We show that quaternion based models are equivalent to quantum computation. Finally, we strengthen this "algebraic hierarchy" by using the tensor formula formalism to construct a family of complete (or promise complete) problems for the corresponding complexity classes.

Part II concerns experimental Quantum Computing by Nuclear Magnetic Resonance (NMR). An important obstacle in this approach is the inability to properly initialise the quantum register, which is related to the signal-to-noise problem in NMR spectroscopy. One of the most promising techniques to solve it is Algorithmic Cooling, a generalisation of the polarisation transfer techniques already used by NMR spectroscopists. We discuss the traditional adiabatic approaches and their limitations. We describe a variation on this technique, the non-adiabatic approach, which uses the environment in order to cool beyond these limits. In particular, we provide a new, efficient non-adiabatic cooling algorithm which could achieve near-zero spin temperatures in liquid-state NMR already with more reasonably sized registers (30–60 spins). Finally, we report on the successful laboratory realisation of the first ever non-adiabatic cooling experiment. This constitutes, we hope, the first step towards the fully fledged development of this promising technique, with applications far beyond NMR-based Quantum Computing.

Keywords: Quantum Computation, Theory of Computation, Computational Complexity Theory, Quantum Computational Complexity Theory, Experimental Quantum Computation, Nuclear Magnetic Resonance (NMR), Polarisation Transfer, Algorithmic Cooling, Non-Adiabatic Algorithmic Cooling.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviations

- **PD.** Probability distribution.
- **TM, PTM, QTM.** Turing Machine. Probabilistic, Quantum Turing Machine.
- **NMR.** Nuclear Magnetic Resonance.
- **QC.** Quantum Computing.
- **AC.** Algorithmic Cooling.

Symbols

- Equals by definition: $\triangleq$.
- Logarithms: $\log(\cdot)$ and $\mathrm{Ln}(\cdot)$, logarithms in base 2 and e, respectively.
- Dirac delta function: $\delta_i(j) = 1$ if $i = j$ and 0 otherwise.
- Kronecker delta: $\delta_{ij} = \delta_i(j)$
- Statistical relationship: $\mathsf{a}\overset{p}{\relbar\!\ominus\!\!\rightarrow}b$ means that the random variable a will take value b with probability $p = \Pr(\mathsf{a} = b)$. Also, $\boldsymbol{a}\overset{p}{\relbar\!\ominus\!\!\rightarrow}\boldsymbol{b}$ means that that some object in state $\boldsymbol{a}$ will evolve to state $\boldsymbol{b}$ with probability p, under some pre-defined statistical transformation.

Typographical Conventions

- Complex numbers: Greek letters, $\alpha, \beta, \ldots$.
- Quaternions: Greek letters with "hat" symbol, $\hat{\alpha}, \hat{\beta}, \ldots$.
- Conjugation:
  - For complex numbers: α^*
  - For quaternions: $\hat{\alpha}^*$
- State variables: $\boldsymbol{s}$.

- Measurement variables or random variables: v.
- Abstract spaces: $\mathcal{H}$.
- Vector spaces (modules): $\mathcal{V}(\mathbb{K})$, with $\mathcal{V}$ being the set of vectors with scalar multiplication defined over the field (or semi-ring, respectively) $\mathbb{K}$.
- Matrices: A
  - Matrix transposition: A^{t}
  - Matrix conjugate transposition:
    - $\mathsf{A}^{\ddagger}$, for quaternionic matrices
    - $\mathsf{A}^{\dagger}$, in general
  - Ket: $|\phi\rangle$, a column vector.
  - Bra: $\langle\phi|$, a row vector with $\langle\phi| \triangleq |\phi\rangle^{\dagger}$.
  - n-fold tensor product: $\mathsf{A}^{\otimes n} \triangleq \mathsf{A} \otimes \cdots \otimes \mathsf{A}$, n times.
  - Selection operators:
    - $(\mathsf{A})_{i,j}$ - the entry in the i-th row and j-th column of A.
    - $(\vec{v})_i$ - the i-th coordinate of $\vec{v}$.
    - $|\phi\rangle_i$ - the i-th coordinate of $|\phi\rangle$.
  - Trace: $\mathrm{Tr}(\mathsf{A}) \triangleq \sum_i (\mathsf{A})_{i,i}$, for a square matrix A,
  - Diagonal: $\mathrm{Diag}(\mathsf{A})$, the N-dimensional *vector* formed by selecting the N diagonal elements, in order, of an $N \times N$ matrix A.
  - Identity matrix: I when dimensionality is explicit, otherwise:
    - I_n identity matrix of order $2^n \times 2^n$
    - I_2, I_3, ... of order 2×2, 3×3, etc. (subindex a numeral)
- Inner product: $\langle\vec{x}, \vec{y}\rangle = \langle x|y\rangle$
- Linear operators: $\hat{L}$
- Superoperators: $\hat{\hat{S}}$
- $N \triangleq 2^n$, the dimensionality of a state space of n bits/qubits.

À tous ceux et celles qui n'ont pas pu mener jusqu'au bout ce combat ingrat, et dont l'exemple m'a obligé à me traîner jusqu'à la fin, malgré tous les obstacles et la tristesse de découvrir qu'en fin de compte, ça n'en valait même pas la peine :

À Vincent (père et fils), Frédéric, Sasha, Murielle...

ACKNOWLEDGEMENTS

First and foremost, the author wishes to thank his academic collaborators and co-authors, here in alphabetical order:

Martin Beaudry, Stéphane Beauregard, Gilles Brassard,
Markus Holzer, Raymond Laflamme, Seth Lloyd, Sasha Mikaelian, Tal Mor,
William Schneeberger, Vwani Rowchoudhury, and Yossi Weinstein.

Their participation in joint work, advice and support has been recognised separately in the introduction of Parts I and II of this thesis, along with that of others who have played a significant role in this work.

J'aimerais donc profiter de cette occasion pour proférer des remerciements plus personnels.

Tout d'abord, Gilles Brassard, mon directeur. Il m'a fourni le temps, la patience, l'oreille, la liberté, les contacts et le cash, nécessaires pour accomplir cette tâche. Mais c'est surtout comme ami et conseiller qu'il mérite ma gratitude, et le pourquoi, seuls lui et moi le saurons.

À ce même niveau, Lucie Burelle, mon autre patron, mérite d'être reconnue et remerciée. En plus de me rappeler constamment que c'était temps que je finisse, elle m'a octroyé le temps et la chance de me développer personnellement, ce qui m'a fourni la maturité nécessaire pour finir, ainsi qu'un havre dans lequel j'ai pu me réfugier des frustrations du doctorat pendant toutes ces années.

Aussi, Paul Dumais, ancien condisciple à l'Université de Montréal, mérite tout mon respect, car en plus d'avoir réussi à finir plus d'un an avant moi, a aussi développé le module QuCalc pour Mathematica. Sans les adaptations à ce module qu'il a accepté de réaliser pour mieux subvenir à mes besoins, il n'aurait pas été possible de développer NMRcalc et ainsi que les autres outils avec lesquels il nous a été possible d'analyser et de préparer nos expériences en résonance magnétique nucléaire. Un grand merci.

Aussi, je tiens à remercier Michel A., Michel J., Steve D. et Louis B., qui ont eu la vision d'embarquer avec nous dans cette aventure et ont été une grande source de motivation et d'appui à tous les niveaux.

Je voudrais également remercier le Très Révérend Père Dom Jacques Garneau et les moines bénédictins de l'abbaye de St-Benoît-du-Lac de m'avoir accueilli chez eux pendant une période cumulative de près de deux mois. Sans la quiétude que cette isolation m'a fournie, il n'aurait simplement pas été possible de trouver la concentration et la volonté pour finir cet ouvrage. Je tiens très particulièrement à remercier les pères Beaulieu et Salvas de m'avoir assisté dans la traduction et dans la recherche de termes scientifiques adéquats en latin, ainsi que les pères Carette et Leal Martínez pour leur hospitalité et accueil chaleureux.

Among my close friends, Laura and Drummond deserve special recognition for their invaluable encouragements and making sure that I remained sane in these last few months. Irina also helped, but to her I am especially thankful for showing me that indeed I might have something to teach, and that I would enjoy doing so, and thus indirectly providing an indispensable source of motivation to accomplish all of this and make it to the end.

Mes futurs collègues de l'École Polytechnique de Montréal (tout particulièrement Samuel Pierre pour ses encouragements et Pierre Robillard pour sa patience) ont sans le vouloir jouer un rôle fondamental dans cette dernière année : ils ont fourni un conditionnant positif sans lequel je me demande encore si j'aurais trouvé la motivation pour finir.

D'un autre côté, je dois aussi reconnaître et remercier mon épouse Sandra, qui après toutes ces années a fini par comprendre comment et pourquoi il fallait que ça prenne aussi longtemps. Mais aussi, parce qu'elle a été sans le vouloir la source de conditionnement négatif tout aussi nécessaire pour m'avoir poussé à terminer.

Et finalement, ma mère Anne-Marie qui pendant tout ce temps m'a soutenu à tous les niveaux (y compris la soupe). Mais aussi car je suis pleinement conscient de tous les sacrifices qu'elle a réalisés pour ce doctorat devienne une réalité, récemment, mais aussi il y a longtemps.

INTRODUCTION

About Computation...

Thus is the nature of the game in Theory of Computing and Complexity: We look at the different models which could be built, in principle, to see if they are equivalent. If they are the physics or engineering are of no concern, and can be "abstracted away"... If they are not, then we know what features are worth worrying about. As a result of playing this game, with many possible models, the two following principles of the Theory of Computing have been postulated:

- The Church-Turing (CT) Thesis: All models of computation that we can propose are either weaker or equally powerful to the so-called universal models of computations, of which the historically most significant are the Turing Machine and Church's Lambda calculus.
- The Strong Church-Turing Thesis: Of all models that we can propose, none is fundamentally more efficient than the above.

However, here we have the theory of quantum computing, which thumbs its nose at the Strong Church-Turing thesis. In appearance, it seems that the thing quantum does help computation as it seems to allow to compute things more efficiently with it than without. So, if we are willing to believe that such a beast as a quantum computer could be built, it seems that the Strong CT Thesis could go the way of the theory of the ether...

About Quantum Physics...

But what is this mysterious thing quantum? What is its essence? The term "quantum" is a Latin adverb meaning "as much as," and has the same root as "quantity." Historically, in Physics, it had to do with certain physical quantities being "quantised," or more precisely discretised. Energy, in particular, a continuous quantity in Classical Physics, is discretised and one speaks of "quanta of energy". Beyond the mere name, lies a new paradigm for the description of physical reality and its dynamics, with a new and axiomatic mathematical formalism describing it.

As a doctoral student with no Quantum Mechanics background, I was damned if I was going to learn the physics. So I had to exorcise the daemons of Physics out of quantum computing, before I could understand it, let alone do research in it. Demons such as "Observables," "Borel sets," "Hamiltonians," "infinite-dimensional space," and a menagerie of other weird animals, such as cats who cannot make up their minds, fermions that can't stand each other, bosons that do, etc, etc.

More to the point, Quantum Physics was introduced at the turn of the last century to help solve some of the mysteries physicists were facing then (photoelectric effect, etc.). However the mystery that I (and many others) face as a computer scientist is quite another. About this thing quantum which aids computation, we must ask the following: what in it is essential in the context of computation, and of these properties which are the ones at the origin of the demise of the strong CT thesis?

About the thing quantum which accelerates computation...

Fortunately, by the time I started my Ph.D. the exorcism was well under way. The quantum circuit abstraction was already the king of the hill. A simple model which under some quantum equivalent of the strong CT thesis (or rather, a patched version replacing it) is believed to be equivalent to all other models quantum. It is an important fact that not all of the axioms of Quantum Mechanics (as it is usually defined) are relevant in the context of computation. Thus, when we speak here of the thing quantum, we are already considering a subset of the properties of "quantumness" of physical reality. A lot of the "demoniacal" characteristics of Quantum Physics (space, time, energy, etc.) can and must be abstracted away, which is what the quantum circuit model achieves. Another non-negligible positive side-effect of adopting that model is that it also exorcises many of Computability Theory's own idiosyncrasies, with which a dose of the quantum thing becomes monstrous.[1]

A first impression of mine was that in quantum computing, quantumness had little to do with quantities being "quantised" (yech! what an awful phrase...). As a matter of fact, things *are* already quantised in computation: it is a fundamental tenet of the Theory of Computing (which supports both the weak and strong CT theses), that under reasonable assumptions [2] all quantities can be quantised, i.e. digitised, without loss of generality or

[1] Indeed, much like the effect of an after-midnight snack on a peaceful Gremlin, the result of feeding some quantum tape into an unsuspecting Turing Machine is not a pretty sight. The poor thing becomes so confused and agitated that it cannot figure out when to stop! Kids, do not try this at home, unless you want to spend the rest of your Ph.D. figuring out when or how it is going to halt and go back to its normal, peaceful self...

[2] More precisely, models for which a bounded amount of error exists. Errorless models such as analogue

power of computation. So, we can forget about looking at the Latin dictionary for more clues...

Having thus extracted the essential, which of the remaining properties of quantum computing are responsible to create that difference, to assist computation? The first suspect is its time-symmetry or reversibility, linked to the unitarity of the allowed dynamic transformations in Quantum Physics. However, that is clearly not it (it is a restriction!), as again reversibility was shown to be a requirement which does not restrict power, i.e. the strong CT still holds under it.

Quantum parallelism is another popular one. It has to do with the ability of a system or computational device to mysteriously "be" in several states at the same time. Or with cats which are not content with their seven lives, but also want to have a few deaths as well... More seriously, it can be modelled by replacing the set of configurations of the computational device (its states) by a richer set which includes suitable *linear combinations* of the elements of the old; thus, non-trivial linear combinations represent these "mysterious" parallel states. That could be it, but early on it was pointed out to me that no, that was not it in itself. This property is essential, yes, but not *a priori* sufficient.

In fact quantum parallelism unbridled can lead to models of computation which are unreasonably powerful (a phenomenon similar to that of analogue computing...). What provides that modulating restriction on the model is the inability to obtain total information on these new states of a computational device. This restriction is given by a measurement rule or principle, which in some sense says that the nice mathematical trick we have pulled by replacing the state set by a linear space is very nice and clever, but useless in the end: our observations are still bound to and based on the original state set, which we often refer to as the *computational basis* of the linear state space. This principle is, in some sense, the computational equivalent of the principle of quantisation in Physics. One can say that even though the linear state space is "continuous" (it does form a complete metric space), the quantities representing measurements on it are "quantised" in that only a finite (albeit exponentially large) number of "values" (the base states) can be observed. Thus, with this tenuous epistemological link, we renege our initial impression that there was nothing "quantum" about quantum computation...

Given this restriction, I was told, what is necessary for this thing quantum to pro-

computing are not covered by the strong CT thesis, but these models are considered unreasonable due to the impossibility of constructing devices with arbitrary precision and accuracy.

duce interesting results is the ability for these parallel existences to cancel each other in some cases; a phenomenon referred to as *interference.* At that time, it was already intuitively perceived that it was this property of quantumness which is essential, as without this property the known, interesting quantum algorithms would cease to work. This is not only true of computation, but of quantum mechanics at large also. As Feynmann, thinking about Physics rather than about computation, had already put it, "somehow or other, it is as if probabilities sometimes needed to go negative." How does one, however, carefully enunciate, let alone prove, such a principle?

Because these things are hard, it was not my intention to go about formalising such a principle. However, it happened, by accident. So, how does one go about it? "Go back to principles," the old adage says. And indeed we go back to the nature of the game in Computability and Complexity Theory, as we described right at the beginning: we define new models, we see if they are equivalent.

First, we will play this game with the intent of narrowing down what that thing quantum really is, from the computing point of view. We will build a "unified" model of computation or meta-model in which we can describe all the usual models (deterministic, probabilistic and quantum), and even some new ones. These models will all have the same structural properties of parallelism (linearity of the state space) and the same measurement principles. In this meta-model of computation, we have a single varying parameter which instantiates the different models: the algebraic structure on which the linear state space is defined. This structure will define the allowed coefficients in linear combinations of base states, coefficients which, due to their relationship with the measuring rules, are often called *probability amplitudes.* Since the only thing that changes is this underlying algebraic structure, we call this method the *algebraic approach.*

Using this method is a good idea, firstly because it provides a nice "big picture." Secondly, it provides us with tools for proving the equivalence of some of these models. Thus, we can build a "hierarchy" of models (or complexity classes) based on this parameter. Some levels collapse, some we do not know. More concretely, and coming back to our main question, on that hierarchy we will be able to draw a line, a frontier, beyond which things can possibly violate the strong CT thesis. In particular, quantum computing, as usually defined, is beyond that line, and classical and probabilistic on this side of it. This line, this separation, corresponds *precisely* to what we thought it was, this is, paraphrasing Feynmann, that ability for probability amplitudes to become negative, and hence with the possibility annulling each other.

Unfortunately, while we strongly believe that this line is "for real" and not just an illusion or a product of our own mathematical ineptitude, there is to this day no solid proof of its existence. Even though we can solve surprisingly hard problems within the quantum model, there is still no formal proof that these problems could not be solved under more traditional models[3]. With respect to this question, a third, albeit marginal, advantage of this meta-model is that it seems to bring us closer to resolving that question. In fact it gives a target to shoot at. Along with that line comes a complete problem for the quantum model, which we can try to show outside of traditional complexity classes. At the heart of the problem is the necessity to keep track of the signs of the amplitudes in order to correctly simulate the final probabilities of measurement. The apparent inability of classical probabilistic algorithms to do so provides some "evidence", as a complexity-theorist would put it, that there is a non-empty gap between the probabilistic and quantum models.

About computing with quantum things...

> *In theory, there is no difference between theory and practice.*
> *In practice, there is...*

Thus quoth the Computer Engineer to the Theoretical Computer Scientist[4]. To this the computer scientist can respond in a variety of ways. If he is able to do so, he will go back to the whiteboard and prove that these differences are insubstantial to the power of computation of the device being built. He then hits the engineer right back with the almighty CT thesis, in its appropriately patched, hard cover version, saying "No, there isn't." If he is generous enough, he might provide the Engineer with a proof of that statement, which she can hopefully make into a simulation method of the original theoretical model by the device.

Of course, if the computer scientist does not succeed to prove it (either because he is too stupid, or because it is simply not true), then he will create a new, revised theory, write it up and present it to the Engineer, nonchalantly saying "Now, there

[3] More precisely, the only exponential complexity separations between the classical and quantum models that have been shown are for the query complexity of certain computational tasks; no such separations have been exhibited in terms of absolute complexity measures (e.g. time or space).

[4] Or, for that matter, so can the physicist in the context of quantum computing.

isn't." Nodding her head, the Engineer will ask, "So what of the old theory? Does this mean that we cannot compute what we wanted to?" And so on...

This is exactly the situation that has arisen in the field of Experimental Quantum Computer Engineering. So far the most successful technique for implementing small-scale quantum computations is that employing the principles of Nuclear Magnetic Resonance. This method utilises the magnetic spins of a macroscopic sample to represent quantum bits (called *qubits*), and employs an NMR spectrometer to manipulate and read out this collective magnetisation. In theory, this method is a correct implementation of the quantum model. However, this is only so under conditions which cannot be achieved today and might never be easy to achieve. Thus in practice, and if we restrict ourselves to NMR, there is a difference between what we can do and what we would like to do, theoretically. More concretely, the two main deviations of NMR QC from the garden-variety quantum computing model are a) a different measurement rule, and b) the inability to initialise the computation with a pure state, or in other words the inability to reset the memory to a known state before starting the computation.

In this case, the first reaction of the theoretician corresponds to trying to show to the experimentalist that these shortcomings will not make a difference in the power of computation of his device, hence providing an extended version of the strong "quantum" CT thesis. So going back to the beginning once more, we, theoreticians, define revised theoretical model and try to show them equivalent or inequivalent to the previous models. So far, we have been unable to do so. In fact, even the time-tested technique of Divide and Conquer, a favourite of many Theoretical Computer Scientists, does not work here: we do not know how to show the equivalence of these models (or lack thereof) even if *only one* of these discrepancies is introduced.

Ultimately, since we cannot conclusively answer any of the questions of the Engineer/Experimentalist, what is a theoretician to do? One answer is to become an experimentalist ourselves and try to solve some of these problems at that level, and make some of these differences vanish. Of course, there are limits on how "wet" most theoreticians will get their hands...

To solve the problem of initialisation, one very interesting theoretical result of an "experimental" nature is the discovery of the technique of *algorithmic cooling.* In its original variant, it consists in performing a pre-computation preparation of the memory, such as the result a small portion of it is in a known state. This purified portion can then be used for the intended computation. This process is in fact a kind of compression algorithm, whose action can be viewed as lowering the entropy or associated "temperature" of the purified portion of the memory; hence the name. The price to pay is that the remainder of the memory cannot be used. Furthermore, the size of the purified region is limited by how mixed the states are initially, and is far too small to make a difference with current experimental technology. So, while the idea is nice in theory, it is not in practice.

What is ironic, is that that these techniques had long been known under the name of *polarisation transfer* by NMR spectroscopists. While some interesting theoretical results providing limits on the efficiency of these techniques were known, as far as we know true compression experiments had not been performed, nor had true scalable algorithms for arbitrary-sized memories been described.

A second improved variant was proposed which in principle could be made practical. It involves recycling the wasted part of the memory by interaction with the environment. This process, called *thermalisation*, allows those bits to go back to their initial state, and then to be used again to purify another region of the memory with the same compression algorithms as before. By combining compression and thermalisation, and under the right conditions, it is possible to purify the *whole memory.* We present here a new and more practical method of doing so, achieving satisfactory levels of purification, even with initial states such as those encountered with current technology. We also study the fundamental limits of such techniques, in thus extending those results known about the first variant.

Unfortunately, these methods require ideal conditions which are not easy to obtain in the lab. In particular, we must have that the purified part of the memory be completely isolated from the environment, or alternatively that the length of the thermalisation process be very short compared with the time it takes for the purified part to go back to its natural mixed state.

Wanting desperately to get my hands wet, the objective I set myself was to actually perform a proof-of-concept experiment demonstrating this method, or at least a simplified version thereof. The first step was to analyse which were the threshold conditions which had to be attained in order for the experiment to work, in particular which was the minimal gap between thermalisation times of the purified portion and the portion to be recycled. Once these minimal conditions were established, an experimental technique had to be found in order to achieve those conditions. We were fortunate to find and perfect a chemical laboratory technique that allowed us to manipulate just enough these thermalisation times. The experiment was successful (barely), thus accomplishing two important objectives: a) proving that algorithmic cooling can work even in the lab, and b) proving that if a mere theoretician can make it work, surely professional experimentalists can improve it to make it even more practical. So there!!!

Part I

De adhibenda re quantica ad accelerandam computationem

About the quantum thing which accelerates computation

What is indeed the nature of this thing quantum which appears to make computations using it go faster? Is it entanglement, is it state superposition, is it interference of computational paths? This question has puzzled researchers in Quantum Computing in its beginnings, but it is now generally understood that the key ingredient without which no speed up can occur is interference. In this first part of the thesis we have sought to make that intuition as formal as possible.

Chapter 1 introduces a mini-epistemology of deterministic, probabilistic and quantum theories. We make *tabula rasa* of all previous axioms and principles of these theories, in particular Quantum Mechanics, and painstakingly re-introduce only those characteristics of these theories which are relevant and non-redundant. The prize is a uniform mathematical model, based on vector spaces, within which we can uniformly describe the models of these theories, in particular computational models.

In Chapter 2, we review the standard models of classical and quantum computation, however, presenting them, in some cases, in a more generalised fashion. In particular, we develop a notion of circuit more general and formal than the one usually introduced, which will allow us to define circuits operating and computing with states defined on arbitrary algebraic structures. This will in turn allow us to characterise the most important complexity classes, both classical and quantum, within a unified framework, whereby variations in the underlying state structure plays a central role in generating the richness and variety of the complexity picture.

In the last two chapters of this part, we go beyond merely restating what was known in more formal terms. In Chapter 3, we complete the algebraic picture of complexity classes by considering the quaternionic numbers. We ask and answer the question of which complexity class(es) they generate. Finally, in chapter 4 we use the formalism of tensor formulæ to strengthen the abstract classification of Chapter 2 by providing a family of complete and promise-complete problems for the relevant classical and quantum complexity classes.

Credits and Acknowledgements

The work in Chapters 1 and 2 is mostly a straightforward formalisation and generalisation of previously introduced concepts. Few novel results are to be found there, except maybe for the proof that no models other than deterministic ones exist which are simultaneously quantum and probabilistic. This is joint work with Michel Boyer, who generalised an earlier result of the author. This whole question was born in discussions with David Poulin.

The result on quaternions in Chapter 3 is joint work with William Schneeberger, who should be credited with the main idea. The author was only responsible for working out the details...[5] The contents of this chapter are almost exactly the same as those of our joint article on this topic [FS03].

Finally, the notion of a strong characterisation through complete problems based on tensor formula was born in conversations with Markus Holzer while he was a postdoctoral fellow with Pierre McKenzie at the Université de Montréal. These ideas were painstakingly developed by Martin Beaudry, from the Université de Sherbrooke, in a long and slow process, whose results are gathered in a joint article [BFH02], on which Chapter 4 is largely based.

In addition to my co-authors, the following persons played a crucial role in discussions, to motivate and validate (not always agreeing...) the work of the author in these topics: Lance Fortnow, Michele Mosca, David Poulin and John Watrous.

[5] within which, one can say in the author's defence, the Devil lives...

CHAPTER 1

A THEORY OF THEORIES

"Vade retro Satanas!"

1.1 About Theories and Models

From a scientific point of view, a *theory* is the set of abstract principles, axioms, and hypotheses that have been formulated with the objective of describing a reality, and ultimately make predictions about it.[1] The axioms and rules of a theory allow us to construct *models* of reality. These models allow us in turn to describe, understand and, hopefully, accurately predict the behaviour of the particular sliver of reality that the theory strives to encompass.

One central dogma of modern Epistemology is that no theory is perfect nor can it be. Reality is just too big and complex for our feeble little minds. Independently of whether we want to accept this philosophical principle or not, the practical reality is that sometimes a theory of everything is just too cumbersome. Whether by necessity or by choice, our theories remain *abstract* and factor away those aspects of reality which we cannot grasp or we choose to ignore. For example, the *Principle of Abstraction* has become in Modern Science and Engineering a maxim indoctrinated into beginners and a universal methodology:

> *Ignore everything that will not affect your predictions.*

In the sciences and engineering, the models that a theory allow us to construct describe *systems*, which are the limited portions of reality which we wish to understand. The *state* of a system is the abstract object that describes it. From it depend the results of *measurements*, which his how information can be extracted about the system from

[1] The Oxford English Dictionary[OED89] gives the following definitions of *theory* in this sense:

a) a scheme or system of ideas or statements held as an explanation or account of a group of facts or phenomena;

b) a statement of what are held to be the general laws, principles, or causes of something known or observed.

and the Webster[Web03] as:

a) The general or abstract principles of a body of fact, a science, or an art.

without. Finally, the *dynamics* of the theory prescribe how the states will change and evolve with time. [2]

As unexpected results are obtained or as phenomena unexplainable under the current theories are observed, we formulate new theories or revise old ones. That has traditionally been the pattern in the Natural Sciences. However, in the Applied Sciences and Engineering it is often that some of the aspects that we had previously neglected or abstracted away become of interest. Thus, theories can become partially obsolete out of no fault of their own, but out of the fact that they no longer meet our changing scope of applications.

Furthermore, according to the relativism mentioned above, different fields of knowledge contribute complementary aspects of the understanding of reality. Physics and Computer Science have provided in the last thirty years one remarkable example of how these points of view can meet and enrich each other. The field of Quantum Computing is indeed one the children of this happy marriage.[3]

In the end, and for whichever reason, the fact is that there are many valid and worthy theories, each with their own utility and scope of application. Indeed, their variety and number make it useful to study them systematically, identifying their common features and classifying them according to their differences. In other words, to define an epistemological theory of theories. Not a theory of everything but one limited to physical theories and theories of computation, one that will suit our particular purpose.

The types of theories that will interest us in the context of Physics and Computation are *deterministic*, *probabilistic*, and *quantum* theories. But before we start enunciating what they are, a word about nomenclature. With the advent of Quantum Computing, the traditional Theory of Computation, i.e. the deterministic and probabilistic models of computing, have been re-dubbed the *classical* Theory of Computation. A similar phenomenon has occurred in Physics where nowadays the term "classical" is used for those theories which are neither relativistic nor quantum. However, there might be some ambiguity about whether probabilistic theories should be considered classical. Some scientific texts and historians of Science, especially older ones, still refer to Classical Physics as that of Newton and its successors, excluding Statistical Mechanics. In the rest of this document, we will abide with the more modern use in both Physics and the

[2] When we say that states evolve with "time," we do not necessarily mean *physical time*, a very slippery business... It is only under very specific circumstances that computational time can be equated to physical time.

[3] and so is Physics of Computation, the unjustly forgotten elder brother.

Theory of Computation.

1.2 Deterministic Theories

1.2.1 Concepts and Definitions

In classical deterministic theories, the notion of state unequivocally defines with absolute precision all of the attributes and properties of the system. From the knowledge of the state of a system, all predictions are deterministic and the answer to every possible question or result of every possible measurement at that moment is uniquely defined by the state object. Furthermore, the dynamics of a deterministic theory is such that given its initial state, the state of a system at any future moment is uniquely defined, and thus so is the outcome of all possible measurements in the future. These are in fact the two axioms of a deterministic theory.

Definition 1.1 (Deterministic Theory). The axioms of Deterministic Theory are the following:

Axiom 1 (Measurement Rule). The state of a system precisely and unequivocally defines the results of all possible measurements.

Axiom 2 (Dynamics). How systems evolve with time depends exclusively on the properties of these previous states of the system, whether measurement related or not.

For simplicity, and without loss of generality, we will assume throughout that the result of all measurements can be represented with a single variable defined over a unified outcome domain. In that case, we can then give the following formal definition.

Definition 1.2 (Deterministic Model; Deterministic Theory). A *deterministic model* is a 4-tuple $(\mathcal{S}, \mathcal{V}, f, \mathcal{D})$ comprised of the following elements:

- $\mathcal{S}$ is the state space,
- $\mathcal{V}$ is the set of results of all measurements,
- $f : \mathcal{S} \mapsto \mathcal{V}$ is the *measurement rule*, where $f(s)$ represents the outcome of measurement on state $s \in \mathcal{S}$, and
- $\mathcal{D} = \{D(t_1, t_2) : \mathcal{S} \mapsto \mathcal{S}\}$ is the family of evolution functions parametrised by valid times of observation $t_1 < t_2$, such that if the state of the system at t_1 is s_1, then the state of the system at time t_2 is $s_2 = D(t_1, t_2)(s_1)$.

To simplify our presentation, let us represent the outcome of a measurement at time t with a single variable $\mathsf{v}(t)$. Also[4], let $D_t : \mathcal{S} \mapsto \mathcal{S}$ represent the time evolution function from some fixed initial time t_0 to a later time $t > t_0$, i.e.

$$D_t = D(t_0, t) \tag{1.1}$$

A very important characteristic of deterministic models is that it is always possible to make a deterministic prediction of the outcome of measurement. In particular, given the initial state s_0 at t_0, it is possible to infer the value of $\mathsf{v}(t)$ as follows

$$\mathsf{v}(t) = v_t = f(D_t(s_0)) \tag{1.2}$$

However, what the dynamics *does not* define is a map between measurement outcomes v_0 and v_t at times t_0 and t respectively. In other words, given *only* knowledge of v_0, we cannot determine with certainty the value of v_t. This situation is described in Figure 1.1.

$$\begin{tikzcd}
s_0 \arrow[r, "D_t"] \arrow[d, "f"'] \arrow[dr, "f \circ D_t"] & s_t \arrow[d, "f"] \\
v_0 \arrow[r, "\times" marking] & v_t = f(D_t(s_0))
\end{tikzcd}$$

Figure 1.1: The relationship between states and measurement outcomes in deterministic models. The crossed-out arrow indicates that no deterministic relationship exists.

The non-existence of a deterministic time evolution map between measurement outcomes stems from the fact that there might be more than one state with the same outcome, or in other words that f is not a bijection. Suppose that we could only observe a system at one fixed moment in time. Then, all states s of the system with the same measurement outcome would appear *indistinguishable* to us. In fact, the measurement rules define an *equivalence relation* in the state space $\mathcal{S}$, where each equivalence class $[s] = \{s' \mid f(s') = f(s)\}$ represents all the states which are mutually indistinguishable.

In the absence of knowledge on the dynamics or because of the inability to re-measure at a later time, this indistinguishability is complete. That is why these equivalence

[4] While it is true that in most physical theories, the time evolution function $D(t_1, t_2)$ will only depend on the difference $\Delta(t) = t_2 - t_1$, this is not the case for all models that interest us, as we will see.

classes are sometimes called *partial information states* or also *macro-states* (also written as *M-states*), in contrast with the original states which are dubbed *total information states* or *micro-states* (or μ-states). This is equivalent to considering a model with a new state space $\mathcal{S}/f$, the quotient of $\mathcal{S}$ under the relation defined by the measurement rule f. The state space $\mathcal{S}/f$ is in fact isomorphic to the space of measurement values $\mathcal{V}$ adopted by the total measurement variable v, with each value v being associated uniquely with an M-state$[s]$ as follows

$$v \equiv [s] \iff f(s') = v \text{ for some } s' \in [s] \tag{1.3}$$

Because there is no deterministic map between outcome values, unfortunately such a model would not be deterministic, as there would be no deterministic map between M-states.

1.2.2 Examples of Deterministic Models

In the rest of this section, we will introduce and discuss some examples of deterministic models relevant to our purposes.

1.2.2.1 Turing Machines

The quintessential example of deterministic models in the Theory of Computation is the Turing Machine, an abstraction of computational devices which according to the Church-Turing Thesis embodies the essence of all imaginable and reasonable computation. It consists of a finite state machine or automaton, an infinite memory tape (with a beginning and no end) consisting of individual cells, and a read-write tape head controlled by the automaton that can move to any position on the tape, but only moving by one cell at each step. In one of its simplest formulation, the Turing Machine tape contains only binary input symbols in $\Sigma = \{0, 1\}$ and blank cells, represented with the symbol $\llcorner\!\lrcorner$. In other words, the tape alphabet is $\Gamma = \Sigma \cup \{\llcorner\!\lrcorner\}$. In that case, it can then be formally defined as follows:

Definition 1.3 (Turing Machine). A Turing Machine (TM) is a 4-tuple, (Q, q_0, q_f, δ) where

- Q is the set of internal states of the automaton.

- $q_0 \in Q$ is the *initial* state of the automaton.

- $q_f \in Q$ is the *final* or *halting* state of the automaton.

- δ, is the transition function describing the behaviour of the TM, where

$$\delta \, : \, Q - \{q_f\} \, \times \, \Gamma \longmapsto Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\} \, .$$

The image $\delta(q, b) = (q', b', d)$ indicates what the TM will do next when the automaton is in internal state $q \in Q, q \neq q_f$, and the symbol under the tape head is b. First, it will overwrite it with b'. Then, it will move the head in the direction indicated by d: right if $d = \mathtt{R}$, left if $d = \mathtt{L}$, and stationary if $d = \mathtt{L}$ and it is at the beginning of the tape. Finally, the internal state of the automaton will change to q'. Upon entering the final state q_f, the TM will *halt* and perform no further action.

States. At any given moment of the computation, the state of a TM, also referred to as its *configuration*, is defined by the state of its internal finite automaton, the position of the tape head, and the symbols written on its tape.[5] Configurations are typically represented by a string $[c_\mathrm{L}, q, c_\mathrm{R}]$, where q is a string representation of the internal state of the automaton, $c_\mathrm{L} \in \Gamma^*$ is the string of tape symbols to the left of tape head, and $c_\mathrm{R} \in \Gamma^*$ is the string of symbols under the head and to its right up to the last non-blank symbol. The configuration string is thus always finite.

Dynamics. The dynamics of this model is defined by the transition function δ and the fact that the TM will halt upon entering the final internal state q_f. It is therefore fully deterministic.

Measurement. According to the spirit of the model, what can be "seen" of a Turing Machine by an external observer are its tape contents, its tape head position and whether the machine has halted. More formally, that means if the TM is in configuration $[c_\mathrm{L}, q, c_\mathrm{R}]$, then the result of our observations can be represented as $[c_\mathrm{L}, h, c_\mathrm{R}]$, where h is a Boolean variable representing whether $q = q_f$.

[5] In the Theory of Computation, the term configuration is used to distinguish the state of the TM with the state of the internal automaton, a system within a system. Here we use the term *state* for the state of the entire TM.

However, nothing fundamental in the Theory of Computation prevents observation of all the components of the state of a TM, including the internal state. In fact, this allows for the principle of *simulation* in which a TM or another kind of computational device *mimics* the behaviour of a TM by observing and keeping track of its successive configurations. One of the consequences of the Universality Theorem for Turing Machines is that computability is not affected, nor is problem complexity (for most reasonable classes), by whether we adopt this white box model or the more abstract black box model just mentioned. Consequently, the results and predictions of the underlying theory are not affected whether we allow full or only partial observation of the TM configurations, or in other words on how we define macro-states as long as they include the contents of the tape.

1.2.2.2 Boolean Circuits

Boolean or *logical* circuits were invented as an abstraction of electronic circuits, today ubiquitous even within the computer on which this text is being written. They consist of elementary logical gates operating on Boolean variables which are "brought" to them by *wires* which interconnect the gates. These Boolean values (0 or 1) are idealised abstractions representing some physical property such as voltage. The gates operate on these values

Abstractly, a circuit can be defined as a graph with the following properties:

Definition 1.4. A circuit $C = (\mathcal{I}, \mathcal{O}, \mathcal{G}, \mathcal{W})$ is a special kind of *directed acyclic graph* $(\mathcal{V}, \mathcal{W})$ with vertices $\mathcal{V}$ and edges $\mathcal{W}$, where

- The set of vertices is partitioned into three components $\mathcal{V} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{G}$, where
  - $\mathcal{I}$ represents the input nodes, to which a given "input" value is assigned at initialisation.
  - $\mathcal{O}$ represents the output nodes, from which the "output" of the circuit will be determined.
  - $\mathcal{G}$ is the set of gates in the circuit, each transforming the values on its input connections onto values on its output connections according to a fixed rule.
- The set of edges $\mathcal{W} \subset \big(\mathcal{I} \cup \mathcal{O} \cup (\mathcal{G} \times \mathbb{N})\big) \times \big(\mathcal{I} \cup \mathcal{O} \cup (\mathcal{G} \times \mathbb{N})\big)$ of the graph, corresponds to "wires" linking input and output nodes with the (labelled) connections of the gates, with the following restrictions and semantics. If $w = (u, v) \in \mathcal{W}$ then:

- $u \in \mathcal{I}$ and $v = (g, j) \in \mathcal{G} \times \mathbb{N}$, indicates that the u-th input node is connected to the j-th input connection of gate $g \in \mathcal{G}$
- $v \in \mathcal{O}$ and $u = (g, i) \in \mathcal{G} \times \mathbb{N}$, indicates that the i-th output connection of g is connected to the v-th output node
- $u, v \in \mathcal{G} \times \mathbb{N}$, with $u = (g, i)$ and $v = (g', j)$ indicates that the i-th output connection of g is connected to the j-th input connection of gate g'.
- $u \in \mathcal{I}$ and $v \in \mathcal{O}$ indicates a direct connection from an input node to an output node, without going through any gates.
- the in-degree of any input connection or output node is 1 (no fan-in).

Intuitively, circuits are used as computing devices by setting the input nodes to the input of the computational task we want to solve. The values then "propagate" to the input connections of all connected gates through the wires of the circuits. These gates in turn act by "reading" all input connections, and "writing" the correct values onto the output connections, which are in turn propagated to the next gates or output nodes. The output of the circuit is defined when all output nodes have "received" a value, and consists of the combined values of all output nodes.

In particular, a Boolean circuit is one where all the gates g implement a fixed logical operation such as AND, OR or NOT, or more generally any map $g : \Sigma^k \mapsto \Sigma^l$, where k and l are the number of input and output connections of g, respectively.

As an example of a deterministic model, Boolean circuits have the following elements.

Measurement. The interface with the outside is defined by the output nodes. Their values are the observables of the system and their value is always uniquely defined.

States and Dynamics. If viewed as a physical system, the state of the circuit could be defined as the collection of Boolean values that *each* of the wires of the circuit has. However, such a description is not very useful as it represents the complete computation, from input to output, and from beginning to end. We would like to require the notion of state to correspond to a particular *moment* within the computation. However, in circuits the notion of time becomes somewhat elusive and unconventional.

There is no implicit notion of time as a motor of change in circuits: change in a circuit is effected by the gates. Whilst a notion of "before" and "after" can be defined within each gate, the fact is that the graph of wires interconnecting the gates does not necessarily define a unique total ordering of the gates themselves. As a consequence, in some cases saying that this gate was "before" this one can be meaningless.

What are these "moments" then, at which it is sensible to define the state of circuit? They can modelled as bipartite cuts in the circuit graph with certain particular properties.

Definition 1.5 (Temporal Cut of a Circuit). A *temporal cut* of a circuit C is a bipartite cut $(\mathcal{G}_\mathrm{L}, \mathcal{G}_\mathrm{R})$ of the circuit graph (i.e. $\mathcal{G}_\mathrm{L} \cap \mathcal{G}_\mathrm{R} = \varnothing$ and $\mathcal{G}_\mathrm{L} \cup \mathcal{G}_\mathrm{R} = \mathcal{V}$), where we call $\mathcal{G}_\mathrm{L}$ the "left" side and $\mathcal{G}_\mathrm{R}$ the "right" side, with the following properties:

i. All input nodes are on the left side, and all output nodes are on the right side, i.e. $\mathcal{I} \subset \mathcal{G}_\mathrm{L}$ and $\mathcal{O} \subset \mathcal{G}_\mathrm{R}$.

ii. All wires across the cut are directed from the left to the right, i.e. there are no $w = (u, v) \in \mathcal{W}$ such that $u \in \mathcal{G}_\mathrm{R}$ and $v \in \mathcal{G}_\mathrm{L}$, we

The *width* of a temporal cut is the number wires that cross the cut, i.e. $|\{w = (u, v) \mid u \in \mathcal{G}_\mathrm{L} \text{ and } v \in \mathcal{G}_\mathrm{R}\}|$. [6]

Let $\mathcal{T}_C$ be the set of all temporal cuts of a circuit C. We can define a partial order within $\mathcal{T}_C$ as follows

Definition 1.6. Let $t = (\mathcal{G}_\mathrm{L}, \mathcal{G}_\mathrm{R})$ and $t' = (\mathcal{G}'_\mathrm{L}, \mathcal{G}'_\mathrm{R})$ be two temporal cuts in $\mathcal{T}_C$. We say that $t < t'$ if

a) (*Immediate successor.*) $\mathcal{G}'_\mathrm{L} = \mathcal{G}_\mathrm{L} \cup \{u\}$, where $u \in \mathcal{G}_\mathrm{R}$, or

b) (*Transitivity.*) There exist, $t_1, \ldots, t_m \in \mathcal{T}_C$, $m \geq 1$, such that $t < t_1 < \cdots < t_m < t'$.

With this definition, we can properly identify the "initial" and "final" moments t_0 and t_f as the cuts cutting all and only the input wires ($\mathcal{G}_\mathrm{L} = \mathcal{I}$) and that cutting all and only the output wires ($\mathcal{G}_\mathrm{R} = \mathcal{O}$), respectively. In particular, we will have that $t_0 = \min(\mathcal{T}_C)$ and $t_f = \max(\mathcal{T}_C)$. We can thus view $\mathcal{T}_C$ as a computational equivalent of the space-time continuum for C. However, $\mathcal{T}_C$ is not totally ordered and therefore there is no unique "trajectory" of time, or in other words no well defined time "axis."

If we really insist on having a "proper," fully ordered notion of time, we can always choose one by making arbitrary choices or by incorporating into our model other elements of reality which would make that choice for us. For example, if we were to attribute

[6] Note that the situation is somewhat simplified if we consider reversible circuits or *gate arrays*, where all temporal cuts have the same width.

lengths to the wires then it would be possible to define a sensible temporal ordering of the gates in the graph, associated with the physical events of voltage change at each of the gates, where the initial moment corresponds to a simultaneous change of voltage in all the input nodes.[7]

Regardless, these temporal cuts can also be intuitively viewed as moments at which one of the many correct simulations of the circuit could have stopped. Thus, it is sensible to associate to each possible cut a *state* of the circuit at the corresponding moment.

Definition 1.7 (State of a Boolean Circuit). The *state* of a boolean circuit C at a temporal cut $t = (\mathcal{G}_\mathrm{L}, \mathcal{G}_\mathrm{R})$ of width m is the vector of boolean values $(b_1, b_2, \ldots, b_m)$, where for $1 \leq i \leq m$, $b_i \in \mathbb{B}$ is the wire value of the i-th wire $w = (u, v)$ crossing the cut (according to some arbitrary ordering of the wires).

In particular, the states associated with t_0 and t_f are called the *initial state* and the *final state*, respectively.

The dynamics of a Boolean circuit can be deterministically described in terms of its gates and its states as follows. Let $\boldsymbol{s}$ be the state at a cut t, and let cut t' be an immediate successor of t, i.e. $\mathcal{G}'_\mathrm{L} = \mathcal{G}_\mathrm{L} \cup \{g\}$, where $g \in \mathcal{G}$ is a gate of the circuit. The state $\boldsymbol{s'}$ at time t' is constructed by taking the entries corresponding to the input connections of g, and substituting them with the values of the output connections as defined by the truth table of g and the values at the input connections. In general, and by applying this method iteratively, it is thus possible to uniquely determine the state of the circuit at any moment t', given a description of the state of the circuit at any moment $t < t'$ that precedes it.

1.2.2.3 Classical Mechanics

In its Laplacian or Hamiltonian formulations, Classical Mechanics is also a deterministic theory.

States. It is sufficient to consider the position and momentum vectors for all particles in the system. Thus, states can be represented as the set of all 3-dimensional coordinates of the position and momentum vectors with respect to an initial frame.

[7] But that is precisely the kind of model that would violate our cherished Principle of Abstraction, as these lengths would in no way change the ultimate outcome of the computation, which is why we do not do this in Computer Science: we let the electrical engineers and physicists worry about such things.

Dynamics. The dynamics of such systems is represented by the Laplacian or Hamiltonian operators. From them, and given an initial state, it is always possible to uniquely determine the state of the system at a later time t. In other words, there exists a deterministic map D_t on the state space $\mathcal{S}$, $D_t : \mathcal{S} \mapsto \mathcal{S}$, such that $s_t = D_t(s)$ is the state of the state description of the system after time t has elapsed from the system being in state s. In this case, D_t only depends on the Hamiltonian/Laplacian of the system and the elapsed time t.

Measurement. Nothing fundamental in this theory prevents us from obtaining full information on the exact value of these coordinates. Measurements are, in principle, fully unrestricted. This would be tantamount to equating micro-states and macro-states. Under more reasonable circumstances, however, observations are restricted to macroscopic variables such as speed and position of the centre of mass, angular momentum, etc. In this case, macro-states are equivalent to the Cartesian product of the domain of these variables.

1.3 Probabilistic Theories

1.3.1 What Is a Probabilistic Theory?

In our objective to trim our meta-model of theories to the bare essentials, let us forget for a minute everything we know or might know about probabilistic models, and adopt a nescient approach. First, and by their name, we can assume that they somehow must involve probabilities, an otherwise abstract mathematical concept defined axiomatically. Secondly, we presume that they are distinct from deterministic theories. As a consequence, some element of determinism of the latter theories must go: either determinism of the dynamics or that of measurement.

But in fact, we cannot choose to abandon determinism of the dynamics without abandoning deterministic measurements also. To see this, consider a non-deterministic theory which retains determinism of measurements. Let $\mathcal{S}$ be state space of this theory, and let $\mathcal{V} = \{v_1, v_2, \ldots\}$ represent the sample space of measurements.[8] By assumption, there must exist a mapping $f : \mathcal{S} \mapsto \mathcal{V}$ defined by our measurement rule, such that

[8] For convenience, we will make the assumption in the rest of this document that the sample space is discretised. It is possible to make this discussion more general, but at the cost of introducing much heavier mathematical machinery than the author is willing or able to use.

knowledge of the initial state s_0 at time t_0 would in principle allow us to know the unique outcome of measurement at that moment, i.e. $f(s_0)$.

However, also by assumption, the evolution map N_t is not deterministic, which means that at time t, s_0 could have several possible images $\{s_1, s_2, \ldots\}$, or in other words $N_t(s)$ could be any non-singleton subset of $\mathcal{S}$. In that case, measurement at time t would not yield a unique answer, with any of the values in $\{f(s_1), f(s_2), \ldots\}$ being possible. In other words, we have that the determinism of measurement is not preserved by a non-deterministic dynamics. This situation is depicted in Figure 1.2.

$$\begin{tikzcd}
s_0 \arrow[r, dotted, "N_t"] \arrow[d, "f"'] & N_t(s_0) = \{s_1, s_2, \ldots\} \arrow[d, "\times" marking] \\
v_0 & v_t = ??
\end{tikzcd}$$

Figure 1.2: The relationship between states and measurement outcomes in probabilistic models where we have defined a deterministic measurement rule. If the dynamic mapping between states is non-deterministic (represented here by a dotted arrow), then there is not necessarily a unique outcome v_t at time t.

Hence, we are left with the other scenario in which we no longer have deterministic measurement rules. This important departure from deterministic theories means, among other things, that states no longer determine all the properties of the system. For example, if the same system can be "prepared" to be in the same state more than once, the outcome of measurement can be different each time. Equivalently, two copies of an identical system even if prepared and initialised in the same identical fashion will yield different outcome values when observed. However, one thing will remain constant for both instances of the system: the relative frequencies of each outcome value. This is how "probabilities" are involved in a probabilistic theory: a system observed under fixed conditions defines a *probability distribution* or (PD) on the sample space of the measurement variables.

The evolution of a system will change its characteristics, and in particular these probabilities will change. However, it is expected that these probabilities will depend on the initial probability distribution, and *only* on it. In other words, while the measurements are no longer deterministic, the dynamics between probability distributions of the outcomes still is.

Definition 1.8 (Probabilistic Theory). The axioms of Probabilistic Theory are the following

Axiom 1 (States). The states of a system are objects which assign a probability to each point of a given sample space, i.e. they can represented as probability distributions on the sample space.

Axiom 2 (Measurement Rule). A system which is observed under the same identical initial conditions, i.e. in the same state, will yield different results, but according to the same fixed probability distribution determined by the state.

Axiom 3 (Dynamics). The probability distribution of outcomes of a system after it has evolved under its natural dynamics depends exclusively on the probability distribution of its initial state.

Probability distributions are usually represented as probability vectors. When working with probability vectors, it is useful to use the l_1 norm instead of the usual Euclidean norm.

Definition 1.9 (l_1 norm). The l_1 norm over a discrete vector space $\mathcal{V}(\mathcal{K})$ is the map $\|\cdot\|_1 : \mathcal{V} \mapsto \mathcal{K}$ defined as

$$\|\vec{x}\|_1 \triangleq \sum_i |x_i|$$

where x_i are the coordinates of $\vec{x}$.

Like all norms, the l_1 norm is preserved under base changes and it also has an associated distance called *grid* or *Manhattan distance*, i.e. $d_1(\vec{x}, \vec{y}) \triangleq \|\vec{x} - \vec{y}\|_1$. The l_1 norm allows us to re-characterise probability vectors more succinctly.

Definition 1.10 (PD-space). The *probability distribution space* or *PD-space* of dimension N is the subset $\mathrm{PD}(N)$ of the N-dimensional real vector space $\mathbb{R}^N$ such that for every *probability vector* $\vec{p} \in \mathrm{PD}(N)$:

i. $p_i = (\vec{p})_i \geq 0, \forall i \in \{1, \dots, n\}$, and

ii. $\|\vec{p}\|_1 = 1$.

Furthermore, the PD-space $\mathrm{PD}(\mathcal{V})$ associated with a sample space $\mathcal{V}$ of cardinality N is simply $\mathrm{PD}(N)$.

Note that another common characterisation of PD(N) is as the simplex generated by the canonical basis $\{\vec{e}_1, \ldots, \vec{e}_N\}$ of $\mathbb{R}^n$. This suggests a natural way in which one can map an arbitrary random variable v with outcomes in $\mathcal{V}$ onto the corresponding probability vector (i.e. probability distribution). The map is defined by assigning to each sample point $v_i \in \mathcal{V}$ the canonical base vector $\vec{e}_i$. More generally, we have that

$$\mathsf{v} \longmapsto \vec{v} = \sum_i \Pr(\mathsf{v} = v_i)\vec{e}_i \tag{1.4}$$

Based on the axioms of Definition 1.8 and on the mathematical formalism just defined for PD-spaces (Definition 1.10), we can formulate probabilistic models as models in which the states *are* the probability distributions, and are thus represented by probability vectors.

Definition 1.11 (Probabilistic Model). A *probabilistic model* is a 2-tuple $(\mathcal{V}, \mathcal{P})$ where:

- $\mathcal{V}$ is the set of results of all measurements, i.e. the sample space.
- The state space of the model is PD($\mathcal{V}$), the PD-space of probability distributions over the sample space $\mathcal{V}$.
- $\mathcal{P} = \{P(t_1, t_2) : \mathrm{PD}(\mathcal{V}) \mapsto \mathrm{PD}(\mathcal{V})\}$ is the family of evolution functions parametrised by valid times of observation $t_1 < t_2$, such that if the state of the system at t_1 is $\vec{p}_1$, then the state of the system at time t_2 is

$$\vec{p}_2 = P(t_1, t_2)\,(\vec{p}) \tag{1.5}$$

In comparison with deterministic models, we let the total measurement variable $\mathsf{v}(t)$ become a *random variable*, whose domain $\mathcal{V}$ becomes the sample space. Its outcome will be distributed according to the probability distribution describing the state of the system.

$$(\vec{p})_i = \Pr(\mathsf{v} = v_i \mid \vec{p}) \tag{1.6}$$

While this measurement rule is intrinsically non-deterministic, it is possible that for some of the states the outcome *will be* deterministic. This states correspond to $\{0,1\}$-valued PD's $\vec{s}_i$, where

$$(\vec{s})_j = \Pr(\mathsf{v} = v_j \mid s_i) = \delta_i(j) \tag{1.7}$$

These states which are called the *deterministic* or *fundamental* states of the theory play in important role. In the probability vector formalism, they are of course represented by the canonical base vectors $\{\vec{e_i}\}$. By the same token, any deterministic models can be represented as probabilistic models, as is illustrated in Figure 1.3. Finally, we will also adopt the same notation shorthand of Equation 1.1, by defining the evolution operator P_t for some fixed initial t_0 and all $t > t_0$ as

$$P_t = P[t_0, t] \tag{1.8}$$

$$\begin{tikzcd}
\vec{p}_0 \arrow[r, "P_t"] \arrow[d, "\circ" marking, "(\vec{p})_i"'] \arrow[dr, "\circ" marking] & \vec{p}_t \arrow[d, "\circ" marking, "(\vec{p})_j"] \\
v_i \arrow[r, "\circ\times" marking] & v_j
\end{tikzcd}$$

(a) Probabilistic Model

$$\begin{tikzcd}
\vec{s}_0 \arrow[r, "D_t"] \arrow[d, "\circ" marking, "\delta_{f(s_0)}(i)"'] \arrow[dr] & \vec{s}_t \arrow[d, "\circ" marking, "\delta_{f(s_t)}(j)"] \\
v_i \arrow[r, "\times" marking] & v_j
\end{tikzcd}$$

(b) Deterministic Model

Figure 1.3: In probabilistic models (a), the "states" or PD's allow us to make predictions both in the "present' (time t_0) and in the "future" (time t). These predictions are symbolised as "$\overset{p}{\circ\!\!\rightarrow}$ v", indicating that the value v will be obtained with probability p. Deterministic models (b), can also be viewed as a special case of probabilistic model where to each state s we associate a $\{0,1\}$-valued PD $\vec{s} = \delta_{f(s)}$. In neither case, can we assume that in general there exist a probabilistic (or deterministic) mapping that will determine the probability that we obtain v_j at time t if we have obtained v_i at time t_0, which is indicated by the crossed-out arrows.

1.3.2 The "Hidden Variable" Interpretation

Notwithstanding this ability to make probabilistic predictions, it remains that in probabilistic models the states of a system no longer describe unequivocally all of its properties. This erosion of the concept of state is somewhat unsatisfying, as the theory seems no longer "complete". In fact the idea that a scientific theory is inherently incomplete is abhorred by many. Indeed, one might ask what the source of this uncertainty is and whether it is "real" or just a mathematical artifact.

The fact that a model exhibits probabilistic behaviour might be attributed to the *model* being incomplete and not the theory itself. The theory is complete because what is really happening is that the states of our model are but a partial and imperfect reflection of reality, a reality which can be described in terms of some *hidden variables*, which had not been considered, and which in fact might not even be directly observable. Nonetheless, they do affect the outcome of measurement and they will also affect the dynamics of the system. The new extended model is deterministic and the skeptics are happy: randomness is nothing but an illusion...

More precisely, the explanation behind this uncertainty comes in the form of a deterministic *hidden variable model*, whose measurement outcomes coincide with those of the original probabilistic model. Under full knowledge of the hidden variables, the outcome is deterministic. However, and precisely because of the fact that they are "hidden", we assume under this interpretation that they are distributed according to some fixed *a priori* distribution. The probability distribution on the measurement outcome is induced by this *a priori* distribution and the measurement rule of the hidden variable model, through the law of conditional probabilities.

Again, let us assume for simplicity and without loss of generality, that such hidden variables can be represented with a single holistic hidden variable or *hidden state*, whose domain $\mathcal{R}$ contains all possible (combined) values of these hidden variables. Furthermore, given the time evolution function D_t on $\mathcal{R}$, let D'_t be its equivalent defined onto the PD-space of $\mathcal{R}$, i.e.

$$\begin{aligned} D_t : \mathcal{R} &\longmapsto \mathcal{R} \\ \boldsymbol{r}_i &\longmapsto D_t(\boldsymbol{r}_i) = \boldsymbol{r}_j \end{aligned} \quad \Longrightarrow \quad \begin{aligned} D'_t : \mathrm{PD}(\mathcal{R}) &\longmapsto \mathrm{PD}(\mathcal{R}) \\ \vec{e_i} &\longmapsto D'_t(\vec{e_i}) = \vec{e_j} \end{aligned} \tag{1.9}$$

which is extended by linearity to the general case

$$D'_t(\vec{R}) \triangleq \sum_{\substack{i,j \\ j \,\mid\, r_j = D(r_i)}} (R)_j \, \vec{e_i} \tag{1.10}$$

Note that D'_t is in fact only a reshuffling of the base vectors of $\mathrm{PD}(\mathcal{R})$. Similarly, we can define a generalised measurement rule $f' : \mathrm{PD}(R) \mapsto \mathrm{PD}(\mathcal{V})$ applied to PD's on $\mathcal{R}$. In the "deterministic case" where $\vec{R}$ is a $\{0,1\}$-valued distribution equal to 1 at $\boldsymbol{r}$, then its image under f' will also be a $\{0,1\}$-valued distribution equal to 1 at $v = f(\boldsymbol{r})$. Let R be the random variable distributed according to $\vec{R}$, and v be the measurement outcome random variable. Then, by applying the law of conditional probabilities we obtain the general case as follows:

$$\begin{aligned}(f'(\vec{R}))_i \triangleq \;& \Pr(\mathsf{v} = v_i \mid \mathsf{R}) \\ = \;& \sum_{\boldsymbol{r} \in \mathcal{R}} \Pr(\mathsf{v} = v_i \mid \mathsf{R} = \boldsymbol{r}) \Pr(\mathsf{R} = \boldsymbol{r}) \\ = \;& \sum_{\boldsymbol{r} \in \mathcal{R}} \delta_{f(\boldsymbol{r})}(v_i) \Pr(\mathsf{R} = \boldsymbol{r}) \\ = \;& \sum_{\boldsymbol{r} \in \mathcal{R} \mid f(\boldsymbol{r}) = v_i} \Pr(\mathsf{R} = \boldsymbol{r}) \\ = \;& \sum_{\boldsymbol{r}_j \in \mathcal{R} \mid f(\boldsymbol{r}_j) = v_i} (\vec{R})_j \end{aligned} \tag{1.11}$$

This generalised measurement rule is the basis for justifying the "existence" of randomness in our probabilistic model. The idea is that given an *a priori* distribution, the rule f' will generate a PD on measurement outcomes, which will mimic that of the original model, and moreover, will do so even when we substitute the original PD with its image under the generalised time evolution D'_t. More formally, we have the following definition.

Definition 1.12 (Hidden Variable Interpretation). Let $\mathcal{M} = (\mathcal{V}, \mathcal{P})$ be a probabilistic model. We say that a deterministic model $\mathcal{H} = (\mathcal{R}, \mathcal{V}, f, \mathcal{D})$ is a *hidden variable interpretation* for $\mathcal{M}$ if for any arbitrary valid observation times $t > t_0$, there is for every possible initial PD $\vec{p_0}$ an *a priori* distribution $\vec{R_0} \in \mathrm{PD}(R)$ of the hidden state space $\mathcal{R}$ such that

Initial consistency. $\vec{R_0}$ is "consistent" with $\vec{p_0}$ in the following sense

$$f'(\vec{R_0}) = \vec{p_0} \tag{1.12}$$

Dynamics consistency. The hidden PD at time t, $\vec{R_t} = D'_t(\vec{R_0})$ is "consistent' with the state at time t, $\vec{p_t} = P_t(\vec{p_0})$, i.e.

$$f'(D'_t(\vec{R_0})) = \vec{p_t} \tag{1.13}$$

From this definition, it would seem as if the designers of hidden variable models would have to come up with an *a priori* distribution every time. Beyond the fact that this seems to add an unnecessary degree of complication, it is somewhat unsatisfactory.

We have introduced hidden variable interpretations precisely to be able to "derandomise" probabilistic models, and now we still have to invent probability distributions at every step. The following lemma addresses that concern.

Lemma 1.13. *For every PD in the sample space $\vec{p} \in \mathrm{PD}(\mathcal{V})$, there exists a PD on the hidden state space $\vec{R}_p \in \mathrm{PD}(\mathcal{R})$ such that $\vec{R}_p$ is consistent with $\vec{p}$, i.e. such that $f'(\vec{R}_p) = \vec{p}$.*

Proof. As we saw in Section 1.2, the original measurement rule f defines equivalence classes $[v_i] = \{\boldsymbol{r} \mid f(\boldsymbol{r}) = v_i\}$, on the state space $\mathcal{R}$, with each equivalence class corresponding to those hidden states yielding the same result.

Suppose $\vec{p}$ is a base vector of $\mathrm{PD}(\mathcal{V})$, i.e. $\vec{p} = \vec{e_i}$. Then, we will assign to it the distribution $\vec{R}_i$ which is uniform on the equivalence class $[v_i]$, i.e.

$$\Pr(\mathsf{R}_i = \boldsymbol{r}) = \begin{cases} \frac{1}{|[v_i]|} & \text{if } \boldsymbol{r} \in [v_i] \\ 0 & \text{otherwise} \end{cases} \tag{1.14}$$

We are in fact defining a map $g : \mathrm{PD}(\mathcal{V}) \mapsto \mathrm{PD}(\mathcal{R})$ such that the image of this base vector is

$$g(\vec{e_i}) = \vec{R}_i = \sum_{r_j \in [v_i]} \frac{1}{|\,[v_i]\,|}\, \vec{e_j} \tag{1.15}$$

In general, we will have for an arbitrary PD $\vec{p}$

$$g(\vec{p}) = \sum_i (\vec{p})_i\, \vec{R}_i \tag{1.16}$$

It now suffices to prove that $f' \circ g = \mathrm{id}$. For any $\vec{p} \in \mathrm{PD}(\mathcal{V})$, and all indices i of $\mathcal{V}$, we have that

$$\begin{aligned}
\big(f'(g(\vec{p}))\big)_i &= \sum_{r_j \in [v_i]} \big(g(\vec{p})\big)_j && \text{by Equation 1.11} \\
&= \sum_{r_j \in [v_i]} \Big(\sum_{v_k \in \mathcal{V}} (\vec{p})_k\, \vec{R}_k\Big)_j && \text{by Equation 1.16} \\
&= \sum_{r_j \in [v_i]} \Big(\sum_{v_k \in \mathcal{V}} (\vec{p})_k \,\big(\sum_{r_l \in [v_k]} \frac{1}{|\,[v_k]\,|}\, \vec{e_l}\big)\Big)_j && \text{by Equation 1.15}
\end{aligned}$$

but since the operator $(\cdot)_j$ selects coefficients of terms where $\vec{e_l} = \vec{e_j}$, we have

$$\begin{aligned}
&= \sum_{r_j \in [v_i]} \left(\sum_{v_k \in \mathcal{V}} (\vec{p})_k \frac{1}{|[v_k]|} \right) \\
&= (\vec{p})_i \left(\sum_{r_j \in [v_i]} \frac{1}{|[v_i]|} \right) = (\vec{p})_i
\end{aligned}$$

and thus $f'(g(\vec{p})) = \vec{p}$. □

As a consequence of this lemma, we see that finding a specific *a priori* PD for each initial PD $\vec{p_0}$ of the probabilistic model is not critical to the existence of a hidden variable interpretation: one always exists and can be obtained from the map g defined in the proof of Lemma 1.13. In particular, if we fix the map g as the universal way of constructing *a priori* distributions, then the initial consistency condition can be ignored and the dynamics condition can be rewritten as

$$\forall t > t_0, \quad P_t = f' \circ D'_t \circ g \tag{1.17}$$

Thus, the difficulty lies in finding evolution operators D_t such that the second condition, which is the critical one, is met. This is illustrated in Figure 1.4.

$$\begin{tikzcd}
\vec{R}_0 \arrow[r, "D'_t"] \arrow[d, "f'"'] \arrow[dr, dotted, "P_t \circ f' = f' \circ D'_t" description] & \vec{R}_t \arrow[d, "f'"] \\
\vec{p}_0 \arrow[r, "P_t"] & \vec{p}_t
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
\vec{R}_0 \arrow[r, "D'_t"] & \vec{R}_t \arrow[d, "f'"] \\
\vec{p}_0 \arrow[u, "g"] \arrow[r, "P_t"'] \arrow[r, bend left, "P_t = f' \circ D'_t \circ g"] & \vec{p}_t
\end{tikzcd}$$

(b)

Figure 1.4: The conditions of consistency of a hidden variable interpretation. Figure (a) represents the conditions of initial and dynamics consistency in the original definition (Def. 1.12). Succinctly put, both conditions are met if this diagram commutes along the dotted diagonal arrow. Figure (b) shows the situation when the prior $\vec{R_0}$ is obtained by the general method from Lemma 1.13. In that case, the dynamics consistency condition is met when both paths from $\vec{p_0}$ to $\vec{p_t}$, i.e. along the original probabilistic model and along the hidden variable model, commute.

In summary, hidden variable interpretations allow for a possible answer to the question of origin of uncertainty in probabilistic models: it is just a consequence of the fact that we are missing some information. If indeed all probabilistic models had hidden variable interpretations, this would provide a satisfying and definite answer: randomness is only an ilusion created by our lack of information of a "larger" deterministic reality. We will address in Section 1.4 the question of which probabilistic models have such hidden variable interpretations. We will now discuss some significant examples of probabilistic models.

1.3.3 Examples of Probabilistic Models

1.3.3.1 Markov Chains

Markov chains are the work horses of stochastic modelling. They are used in a wide variety of domains such as Telecommunications, Operations Research, Population Theory, Finance, etc. A Markov chain model consists of a discrete set of fundamental "states" that we can observe the system to be in. It is not a deterministic model, however, and the state space of the system being modelled is composed of probability distributions over these fundamental states. In our formalism, the fundamental states represent the sample space $\mathcal{V}$, and the state space is then $\mathrm{PD}(\mathcal{V})$.

In these models time is discretised into individual time steps. The defining characteristic of a Markov chain is that the state of a system depends only on its immediate predecessor in time. This is also called the Markovian property of stochastic processes and it is said of Markov chains that they are "memoryless". The term "chain" in fact refers to the collection of random variables $\{\mathsf{v}_0, \mathsf{v}_1, \ldots, \mathsf{v}_t, \ldots\}$, where v_t represents the outcome of measurement at time t. The Markovian property is thus expressed as:

$$\Pr(\mathsf{v}_t = v \mid \mathsf{v}_0 = v_{i_0}, \ldots, \mathsf{v}_{t-1} = v_{i_{t-1}}) = \Pr(\mathsf{v}_t = v \mid \mathsf{v}_{t-1} = v_{i_{t-1}}) \tag{1.18}$$

If furthermore, we assume that the there exist transition probabilities between the fundamental states of the system, which are time-independent and independent of the previous states, the system is said to be of *first order.*

$$\begin{aligned} P(i,j) &\triangleq \Pr(\mathsf{v}_t = v_j \mid \mathsf{v}_{t-1} = v_i) \\ &= \Pr(v_i \overset{1\text{ step}}{\longmapsto} v_j) \\ &= \Pr(\vec{e_i} \longmapsto \vec{e_j}) \end{aligned} \tag{1.19}$$

These transition probabilities define in fact the images of the base vectors $\vec{e_i}$ of the PD-space PD($\mathcal{V}$)

$$P(\vec{e_i}) = \sum_j \Pr(\vec{e_i} \longmapsto \vec{e_j})\, \vec{e_j} = \sum_j P(i,j)\, \vec{e_j} \tag{1.20}$$

and by linearity of the PD-space, also those of any PD $\vec{p}$

$$P(\vec{p}) = \sum_i (\vec{p})_i\, P(\vec{e_i}) = \sum_{i,j} (\vec{p})_i P(i,j) \vec{e_j} \tag{1.21}$$

which can be represented in terms of a matrix product

$$= P \cdot \vec{p}, \text{ with } P_{ij} \triangleq P(j,i) \tag{1.22}$$

Furthermore, The operator describing the dynamics after t steps of evolution is then simply $P_t = P^t$.

The class of matrices which can represent valid linear dynamics of this kind is called the *stochastic matrices.* Their entries are in the interval $[0,1]$ and their columns add up to 1 (the i-th column is the probability vector representing the image of $\vec{e_i}$).

1.3.3.2 Probabilistic Turing Machines

A Probabilistic Turing Machines (PTM) is exactly like a deterministic TM, except that the internal control is a probabilistic finite automaton. A probabilistic finite automaton (PFA) is a special kind of non-deterministic automaton, which has a fixed probability assigned to each of the possible state transitions.

Definition 1.14 (Probabilistic Turing Machine). A *probabilistic Turing machine* is a 4-tuple (Q, q_0, q_f, π) where

- Q is the set of states of the automaton
- $q_0 \in Q$ and $q_f \in Q$ are the initial and final (accepting) states of the automaton
- π is a *probabilistic transition function*

$$\pi : (Q \times \Gamma) \times (Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}) \longmapsto [0,1]$$

with the property that for all $q \in Q$, $b \in \Gamma$

$$\sum_{\substack{q' \in Q \\ b' \in \Gamma \\ d' \in \{\mathtt{L},\mathtt{R}\}}} \pi(q, b, q', b', d') = 1$$

The evolution of a non-deterministic Turing machine can be represented as a computation tree. It is a tree of configurations, where the root corresponds to the initial configuration, and the children of a node are the configurations which can reached from it in one step. The i-th level thus corresponds to the configurations of the machine after exactly i steps.

In order to construct a probabilistic model such as those defined in Definition 1.11 for PTM's we would like to assign probabilities to the nodes in the tree, normalised for each level, such that the value of a node at the i-th level corresponds to the probability that the PTM reached that configuration in i steps. The transition probability π of the PFA induces the following transition probability function for configurations. Let C represent the space of configurations. Let $c, c' \in C$ be two configurations, and let $a, b \in \Gamma$ and $u, v \in \Gamma^*$, then

$$P(c, c') = \begin{cases} \pi(q, b, q', b', \mathtt{L}) & \text{if } c = [ua, q, bv] \text{ and } c' = [u, q', ab'v] \quad \textit{(head moves left)} \\ \pi(q, b, q', b', \mathtt{R}) & \text{if } c = [u, q, abv] \text{ and } c' = [ub', q', bv] \quad \textit{(head moves right)} \\ \pi(q, b, q', b', \mathtt{L}) & \text{if } c = [q, bv] \text{ and } c' = [q', b'v] \quad \textit{(head hits left end)} \\ 1 & \text{if } c = c' = [u, q_f, v] \quad \textit{(final state)} \\ 0 & \text{otherwise} \end{cases} \tag{1.23}$$

In a traditional non-deterministic computation tree it is possible for the same configuration to be present more than once in the same level of the tree, each instance corresponding to a different computation path. Because we are assuming that the "choices" of the PFA at step i are independent from previous choices at steps $j < i$, we can define the probability of a computation path as the product of each of the configuration transition probabilities along the path, starting from the root. However, in order for probabilities to be well defined for the configurations themselves, it is necessary to "merge" all instances of the same configuration within the same level. We thus do not have a probabilistic computation *tree*, but instead a directed acyclic graph (DAG) with a single root (the initial configuration), where edges are labelled with the (non-zero) transition probabilities from Equation 1.19. Having done that, the following equation defines proper probabilities for a configuration c at step i in a computation DAG rooted at configuration c_0.

$$P^{(i)}(c) = \begin{cases} \delta_{c_0}(c) & \text{if } i = 0 \\ \sum_{c' \in \mathcal{C}} P(c', c) P^{(i-1)}(c') & \text{if } i \geq 1 \end{cases} \tag{1.24}$$

Note the above sum is not restricted to "parent" configurations c', because by definition of $P(c', c) = 0$ if c cannot be preceded by c' (Equation 1.23). This is useful because it makes the above definition valid independently of the topology of the computation DAG, which could be different for each initial configuration.

We are now ready to construct our probabilistic model. The sample space is the space of configurations $\mathcal{C}$ and the state space is PD($\mathcal{C}$). If we represent PD's as probability vectors, we can obtain from Equation 1.24 an expression for the image PD of the initial configuration c_0 as follows:

$$P_t(\vec{c_0}) = \sum_{c \in \mathcal{C}} P^{(t)}(c)\, \vec{c} \tag{1.25}$$

where $\vec{c_0}$ and $\vec{c}$ are the $\{0, 1\}$-valued PD's associated with configurations c_0 and c. This defines the dynamics map in PD($\mathcal{C}$) for all of its base vectors, and hence by linearity it is defined on all PD's in PD-space.

Only one little problem. A technicality which have readily swept under the rug is the fact that the sums in the above equations are infinite... This is worrisome for two main reasons. First, one has to start worrying about whether these sums converge. In principle, that is not a problem, as long as the Turing machine halts, and because the set of non-zero probabilities in any such PD that will be reached by a PTM is finite. Last but not least, it is quite inconvenient to manipulate or describe potentially infinite state objects, specially with the aggravation we are essentially describing a finite object (the probabilistic Finite Automaton). That is in part why probabilistic Turing machines are not usually modelled in this way.

1.3.3.3 Probabilistic Circuits

Probabilistic circuits are the natural probabilistic extension of Boolean circuit. The most traditional description of probabilistic circuits is one where we extend the standard set of universal deterministic gates with additional elementary *probabilistic gates.*

Definition 1.15 (Probabilistic Circuit). A *probabilistic circuit* is a circuit $C = (\mathcal{I}, \mathcal{O}, \mathcal{G}, \mathcal{W})$ where

1. Each input node $i \in \mathcal{I}$ is associated a Boolean value $x_i \in \Sigma$.

2. Each gate $g \in \mathcal{G}$ has an associated transition probability map $\pi_g : \Sigma^k \times \Sigma^l \mapsto [0, 1]$

$$\pi_g\big((x_1, \ldots, x_k), (y_1, \ldots, y_\ell)\big) = \Pr\big((x_1, \ldots, x_k) \xmapsto{g} (y_1, \ldots, y_\ell)\big) \tag{1.26}$$

   with the property that

$$\sum_{y_1,\ldots,y_\ell} \pi_g\big((x_1, \ldots, x_k), (y_1, \ldots, y_\ell)\big) = 1$$

   where k and ℓ are the number of input and output connections of g, respectively.

States. In probabilistic circuits, even if we think of the wires as “carrying” boolean values, the nature of the gates is such that we may observe different values on the wires of the circuits every time we “reset” the circuit, and this, even if we initialise all the input nodes in the same fashion every time.[9]

Unlike in Boolean circuits, we cannot soundly assign definite boolean values to the wires. Instead, one could try to describe the state cuts by assigning to each wire a probability value. Unfortunately, this is equivalent to describing the value of each wire with an independent random variable, and in general these variables will *not* be independent. In particular, consider the 2-way FAN-OUT gate, whose two output connections are by definition equal to the value of its single input connection. Let w_0 be the random variable representing the input connection, distributed according to probabilities p and q $(p + q = 1)$ of being 0 or 1. Let also w be the random variable representing the values of both output connections. By definition, of FAN-OUT, w will be distributed with probabilities p and q of obtaining 00 and 11 respectively, and probability 0 of obtaining 01 or 10. If we define (partial) random variables w_1 and w_2 for the outcome probabilities of the first and second output connection, i.e. the marginals of w, we will find that $\mathsf{w}_0 \equiv \mathsf{w}_1 \equiv \mathsf{w}_2$. However, the Cartesian product of these marginals will not describe the correct global statistics for both wires, i.e. $\mathsf{w} \neq \mathsf{w}_1 \times \mathsf{w}_2$.

[9] This situation would be an example of the scenario we discussed in Section 1.3.1 of a probabilistic model with a deterministic measurement rule.

Therefore, in order to correctly and fully predict the output statistics of all wires, locally and globally, we must consider the following notion of state.

Definition 1.16 (State of a Probabilistic Circuit). Let $t = (\mathcal{G}_{\mathrm{L}}, \mathcal{G}_{\mathrm{R}})$ be a temporal cut of a probabilistic circuit C of width m, and let $\mathcal{W}_t = \{w(u,v) \mid u \in \mathcal{G}_{\mathrm{L}}, v \in \mathcal{G}_{\mathrm{R}}\}$ be the set of wires across the t.

A *state* of a probabilistic circuit C at t is any PD $\vec{p}_t$ with domain Σ^m, where

Measurement. Furthermore, suppose that the wires traversing the cut t are labelled according to some conventional order (e.g. from "top" to "bottom"), i.e. $\mathcal{W}_t = \{w_1, w_2, \ldots, w_m\}$. To each wire w_i, we associate a corresponding random variable w_i. Then, if the state of the circuit at t is described by $\vec{p}_t$, we can formulate a "holistic" measurement rule as follows

$$\Pr(\mathsf{w}_1 = b_1, \mathsf{w}_2 = b_2, \ldots, \mathsf{w}_m = b_m) = (\vec{p}_t)_b \tag{1.27}$$

where the binary string $b_1, b_2, \ldots, b_m$ is the b-th string in Σ^m in lexicographical order.

Dynamics Finally, we can describe the dynamics on states by using the transition probabilities defined by the gates as follows. Let $\vec{p}$ be the state at t, where t' is an immediate successor to t, with gate g being the extra gate being added to the left side of t', i.e. $\{g\} = \mathcal{G}'_{\mathrm{L}} - \mathcal{G}_{\mathrm{L}}$. Let g have k input connections and ℓ output connections, and t and t' have widths $k+m$ and $\ell+m$ respectively. Without loss of generality, let $\mathcal{W} = \{w_1, \ldots, w_{k+m}\}$ and $\mathcal{W}' = \{w'_1, \ldots, w'_{\ell+m}\}$, the set of wires across t and t' respectively, be ordered such that g is at the "top", i.e. such that wires $w_1, \ldots, w_k$ are going into the input connections of g ($w_i = (\cdot, g)$, for $1 \leq i \leq k$) and wires $w'_1, \ldots, w'_\ell$ are leaving the output connections of g ($w'_j = (g, \cdot)$, for $1 \leq j \leq \ell$). Then the state of the circuit at t', represented by $\vec{p}'$, is given by

$$\begin{aligned}(\vec{p}')_{y_1,\ldots,y_\ell,y_{\ell+1},\ldots,y_{\ell+m}} &\triangleq \Pr(\mathsf{w}'_1 = y_1, \ldots, \mathsf{w}'_\ell = y_\ell,\ \mathsf{w}'_{\ell+1} = y_{\ell+1}, \ldots, \mathsf{w}'_{\ell+m} = y_{\ell+m}) \\ &= \sum_{x_1,\ldots,x_k} \pi_g\left((x_1,\ldots,x_k),(y_1,\ldots,y_\ell)\right) \cdot (\vec{p})_{x_1,\ldots,x_k,y_{\ell+1},\ldots,y_{\ell+m}} \end{aligned} \tag{1.28}$$

1.3.3.4 Non-Linear Probabilistic Models

We can describe non-linear maps on a discrete PD-space PD(N) as a vector of functions $f = (f_1, f_2, \ldots, f_N)$, with each component a function $f_i : \mathbb{R}^N \mapsto [0,1]$ taking as argument the components of an arbitrary PD $\vec{p}$ and mapping it into the corresponding coordinate of the image of $\vec{p}$. In other words, we have that

$$\begin{aligned} f : \mathrm{PD}(N) &\longmapsto \mathrm{PD}(N) \\ \vec{p} = \begin{pmatrix} p_1 \\ \vdots \\ p_N \end{pmatrix} &\longmapsto \begin{pmatrix} f_1(p_1, \ldots, p_N) \\ \vdots \\ f_N(p_1, \ldots, p_N) \end{pmatrix} \end{aligned} \tag{1.29}$$

of course with the condition that

$$\forall \vec{p}, \quad \sum_i f_i(p_1, \ldots, p_N) = 1 \tag{1.30}$$

Consider the following very simple example: $f_1 = p_1^2 + p_2^2$ and $f_2 = 2p_1p_2$. Such model are not hard to construct, as we can select any functions from $[0,1]^N \mapsto [0,1]$ for the first $N-1$ coordinates, the last one being chosen to match the condition of Equation 1.30.

Viewed as stochastic processes, these models would still have the Markovian property; they are sometimes referred to as *higher order* Markovian processes. Unfortunately, the author does not know any good natural examples of such systems. One reason why it is hard to find such examples is that, as we will see later, such models cannot be easily derandomised, as they do not have well defined transition probabilities.

From a Theory of Computation point of view, it would be interesting to try to define what an abstract classical non-linear computational model would look like and what its relative power would be. Unlike the non-linear quantum case, which we will discuss in Section 1.5.2.2, there is no shortage of non-linear behaviour in the Classical world[10].

1.4 A Taxonomy of Classical Theories

The hidden variable interpretations address the question of uncertainty by suggesting that it is an illusion. The opposite operational approach would be to answer "we don't care!" and simply carry on using probabilistic and accept the fact that the theory might

[10] As any electronics expert would tell us, non-linearity is the enemy!

be incomplete.

Beyond philosophical questions of validity of one or the other interpretation, the truth is that both of these interpretations provide a rich language within which to describe and discuss probabilistic systems, as we have seen in the various examples of Section 1.3.3. However, the natural question to ask is, are both approaches equivalent and equally valid? Or more precisely, under what circumstances can they both be used, in which circumstances they cannot, and whether they encompass all reasonable probabilistic models. Having defined models as formal objects allows us to regroup them in classes, akin to complexity classes, for each of the flavours of theories introduced so far. The questions above can then be formulated as complexity-like questions of class inclusions and exclusions.

Definition 1.17 (Classical Theories). Let the following theories be formally defined as the class of models with the following properties.

Deterministic Theory. D, the class of deterministic models.

Hidden Variable Theory. HV, the class of models with hidden variable interpretations.

Linear and Non-Linear Probabilistic Theories. LP, the class of probabilistic models whose PD-space dynamics is linear, and its complement NLP = I − P.

Probabilistic Theory. P, the class of all probabilistic models.

It is easy to see that any deterministic model $\mathcal{D} = (\mathcal{S}, \mathcal{V}, f, \mathcal{D})$ has a unique associated (formal) probabilistic model $(\mathrm{PD}(\mathcal{S}), \mathcal{D}')$, where $\mathcal{D}'$ are the formal dynamics map defined in Equation 1.10. The "outcomes" of this probabilistic model are states of the original model, whose ultimate outcome in $\mathcal{V}$ is defined by f, or in other words the measurement rule of this probabilistic model is defined by f', defined in Equation 1.11. Thus, Deterministic Theory can sensibly be considered a special case of Probabilistic Theory. We therefore have the following inclusion chains, trivially obtained from the definitions above:

$$\mathrm{D} \subseteq \left\{ \begin{array}{c} LP \\ HV \end{array} \right\} \subseteq \mathrm{P} \supseteq \mathrm{NLP} \tag{1.31}$$

In fact, the main result of this section is the following theorem which significantly clarifies the picture given in Equation 1.31.

Theorem 1.18. *Let $\mathcal{M} = (\mathcal{V}, \mathcal{P})$ be a probabilistic model. Then, the following statements are all equivalent:*

i. *$P_t = P(t_0, t)$ is linear, for all $t > t_0$*

ii. *$\mathcal{M}$ has well defined transition probabilities* $\Pr(v_i \overset{t}{\mapsto} v_j)$*, for all $t > t_0$.*

iii. *$\mathcal{M}$ has a hidden variable interpretation.*

Proof.

i $\Longrightarrow$ ii. If P_t is linear, then the images of the base vectors $\vec{e_i}$ of $\mathrm{PD}(\mathcal{V})$ can be written as

$$P_t(\vec{e_i}) = \sum_j P_t(j,i)\, \vec{e_j} \tag{1.32}$$

As defined, these coefficients $P_t(j,i)$ depend only on the map P_t. But on the other hand, by the semantics given to base vectors of $\mathrm{PD}(\mathcal{V})$ in Equation 1.4, the definition of P_t in (1.8), and the notion of PD's as state in Equation 1.6, we have that the coefficients of $P_t(\vec{e_i})$ are

$$\begin{aligned}(P_t(\vec{e_i}))_j &= \Pr(\mathsf{v}_t = v_j \mid \mathsf{v}_0 = v_i)\\ &= \Pr(v_i \xmapsto{t} v_j)\end{aligned} \tag{1.33}$$

And, thus we have that the coefficients $P_t(j,i)$ are precisely the sought after transition probabilities.

ii $\Longrightarrow$ iii. This is the most important and complicated part of the proof. Intuitively, the idea is the following. The hidden variable model $\mathcal{H}$ that we will construct will take a "deterministic" view of the given probabilistic model $\mathcal{M} = (\mathcal{V}, \mathcal{P})$, similar to that depicted Figure 1.2. This is, $\mathcal{H}$ pretends that the underlying state space is indeed $\mathcal{V}$ (and not that of PD's defined over it, $\mathrm{PD}(\mathcal{V})$). As discussed in Section 1.3.1, this does not allow us in general to make deterministic predictions of the future. However, since for any given time interval $[t_0, t]$, $\mathcal{M}$ has well-defined transition probabilities $P_t(j,i) = \Pr(\boldsymbol{v}_i \overset{t}{\mapsto} \boldsymbol{v}_j)$, then $\mathcal{H}$ can "simulate" the probabilistic dynamics P_t by "choosing" one of the possible new "states" $\boldsymbol{v}_j \in \mathcal{V}$ according to the corresponding transition probability $P_t(j,i)$, where

$\boldsymbol{v}_i$ is the current "state" at time t_0. Then, the probability that the corresponding result v_j would have been obtained by the (probabilistic) measurement rule of $\mathcal{P}$, will be identical to the probability of choosing $\boldsymbol{v}_j$ as the new state.

The is that, unfortunately, since $\mathcal{H}$ is a deterministic model, it cannot really "choose" a $\boldsymbol{v}_j$ with the right probability. Instead, it will decide which of the possible states to transition to based on its *hidden variables*, i.e. those components of its state space not representing the "states" $\boldsymbol{v} \in \mathcal{V}$ of the simulated model $\mathcal{M}$.

More formally, we have that $\mathcal{H} = (\mathcal{R} \times \mathcal{V}, \mathcal{V}, f, \mathcal{D})$, where $\mathcal{R}$ is the *hidden portion* of $\mathcal{H}$'s state space, which is used to make the "choices" of transitions. For example, a simple choice model for $\mathcal{H}$ is the following. Let $\mathcal{R} = [0, 1]$ and let the measurement rule f simply ignore the hidden portion, i.e. $f(r, \boldsymbol{v}) = \boldsymbol{v}$. For a time interval $[t_0, t]$ the dynamics D_t is defined as follows

$$D_t(r, \boldsymbol{v}_i) = (r, \boldsymbol{v}')$$

where $r \in [0, 1]$ and $\boldsymbol{v}'$ depends on $\boldsymbol{v}_i$ and r as follows

$$\boldsymbol{v}'(i, r) = \boldsymbol{v}_j, \quad \text{if} \sum_{i=1}^{j-1} P_t(j, i) \leq r \leq \sum_{i=1}^{j} P_t(j, i) \tag{1.34}$$

In other words, r is used to select which is the next state $\boldsymbol{v}_j$ by comparing it with the cumulative density function for the transition probabilities $P_t(\cdot, i)$ from v_i.

Note that in this case, $\mathrm{PD}(\mathcal{R})$ is isomorphic with $\mathrm{PD}([0, 1]) \times \mathrm{PD}(\mathcal{V})$. Let U represent the uniformly distributed random variable over the $[0, 1]$ interval. Then, a generalisation of the function g from the proof of Lemma 1.13 for continuous state spaces will provide an *a priori* distribution $\vec{R}_0 = g(\vec{p}_0) = (\mathsf{U}, \vec{p}_0)$ which will trivially meet the initial consistency criterion of Definition 1.12 for the initial state $\vec{p}_0$ of $\mathcal{M}$. It is possible to verify, but we will not provide the details here, that the dynamics consistency criterion of Equation 1.13 is also met, i.e. that $f'(D'(\mathsf{U}, \vec{p}_0)) = \vec{p}_t$, where $\vec{p}_t = P_t(\vec{p}_0$ is the state of $\mathcal{M}$ at time t.

Finally, because transition probabilities are well defined for all intervals, the choices made by $\mathcal{H}$ for evolution in a subsequent time interval $[t, t']$, (with $t_0 < t < t'$), for example, can be made independently of those for the interval $[t_0, t]$. In other words, we can apply the reasoning above iteratively for each new interval, as long as we provide the hidden model $\mathcal{H}$ is provided with a "fresh supply" of values r's, independently and uniformly distributed over $[0, 1]$.

iii $\Longrightarrow$ i. Let $\mathcal{H} = (\mathcal{R}, \mathcal{V}, f, \mathcal{D})$ be a hidden variable interpretation for $\mathcal{M} = (\mathcal{V}, \mathcal{P})$. Then by the dynamics consistency condition as expressed in Equation 1.17, we have that for all $t > t_0$, $P_t = f' \circ D'_t \circ g$. The map f' is a linear homomorphism between the PD($\mathcal{V}$) and PD($\mathcal{R}$), since it was defined that way in Equation 1.11. So is map $g : \text{PD}(\mathcal{V}) \mapsto \text{PD}(\mathcal{R})$, as can be seen from its definition in Equation 1.16. Finally, the formal map D'_t : between PD's in PD($\mathcal{R}$) is only a permutation of the base vectors (Equation 1.10, and is also linear. Since linear homomorphisms are closed under composition, we have that P_t must also be linear. □

Corollary 1.19. LP = HV

As a consequence, we can now somewhat simplify the inclusion chain of Equation 1.31

$$\text{D} \subseteq \text{LP} \subseteq \text{P} \supseteq \text{NLP} \tag{1.35}$$

In some sense, the fact that HV $\subseteq$ LP is not surprising, since hidden variable interpretations are based on the constructions of marginals by the law of conditional probabilities, which is an inherently linear operation; this linearity is then forced through to the dynamics of the probabilistic model.

A much more surprising and interesting consequence, in our opinion, is that non-linear probabilistic models *cannot have hidden variable interpretations.* These non-linear probabilistic models somehow challenge the traditional view that the *intrinsic* or PD-based approach to probabilistic model and the hidden variable interpretation approach are equivalent. They are so, but only for linear models. As we have seen in Section 1.3.3.4, one can find reasonable abstract examples of such models, but precisely because they have no hidden variable interpretations, it is hard to see what they would correspond to, or in other words, what kind of reality they model. The author has approached some experts in the field of simulation and stochastic processes[VA03], and came up empty handed in his search for a system that would need to be modelled by such non-linear probabilistic models. These non-linear models seem to be the odd child that nobody has heard of or plainly ignores.

Another "odd child" within the family of scientific theories is Quantum Theory. Like probabilistic theories, given a state description it is possible to make probabilistic predictions on the outcome of any (allowed) measurement. Nonetheless, there is no direct correspondence between these probabilities as the system evolves with time, as we shall see.

1.5 Quantum Theories

1.5.1 Exorcising the Daemons

As discussed in the Introduction, rather than introducing quantum theory with the traditional axioms of Quantum Theory such as presented in a typical textbook on Quantum Mechanics, we will take here a minimalist approach with the goal of arriving to a lean and mean formulation of Quantum Theory.

According to the axiomatic definition given in Section 1.3.1, states in Probability Theory are probability yielding objects. Based on this, the measurement rule simply states that these probabilities indeed correspond to probabilities of observing outcomes of measurement. The state-defining Axiom 1 of Definition 1.8 imposes a restriction on the state space: the values must be probabilities, i.e. they must be non-negative and add up to 1 over the sample space. However, these PD's are essentially functions from the sample space to the $[0, 1]$ interval. As any good mathematician will know, representing such functions[11] as vectors is nothing special. The fact that PD's are represented as vectors in Euclidean space is a mathematical convenience, *and nothing more!* The theory does not gain (nor lose) anything by this choice. To see this, simply consider that if $\vec{p_1}$ and $\vec{p_2}$ are probability distributions, then so is $a\vec{p_1} + b\vec{p_2}$ for any value of $a, b \in [0, 1]$ such that and $a + b = 1$. In other words, PD's are closed under convex linear combinations.

We have made this trivial fact excruciatingly clear to underline the following: the fact that the state space has a linear structure is not an additional axiom, but rather a consequence of the conditions imposed on the state space by the measurement rule.

The situation is similar in Quantum Theory, incredible as it may sound to some. States in a quantum model, are objects defining *probability amplitudes* or just *amplitudes*, which, we will assume for now are complex numbers[12]. Furthermore, the meaning of these amplitudes is given by the measurement rule of quantum theories, which states that the probabilities of observation are related to the squared modulus of these probability amplitudes.

Axiom 1 (States). The states of a system are objects which assign to each point of the

[11] Strictly speaking, when talking of infinite dimensional functions, one must talk of bounded functions in the case of a discrete domain and *integrable* functions on continuous domains.

[12] Because the physicists say so... We will revise this assumption in later chapters.

sample space a probability amplitude.

Axiom 2 (Measurement Rule). A system which is observed under the same identical initial conditions, i.e. in the same state, will yield non-deterministic results. However, the probability of observing a given result will be the squared modulus of the corresponding probability amplitude.

For now, these are the only two axioms which we will "accept". As was the case for probabilistic theories, these two axioms have the following immediate implications:

1. The domain of such amplitude-returning functions must be equal to or at least include the sample space of measurements.
2. Amplitudes must have modulus less than or equal to 1, in order for the squared modulus (i.e. probabilities) to be less than or equal to 1.
3. The sum of squared moduli over the sample space must be equal to 1.

Furthermore, we also have that if $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ are two objects with these properties, then so will $\alpha\mathbf{\Phi}_1 + \beta\mathbf{\Phi}_2$, as long as $|\alpha|^2 + |\beta|^2$. In general, a linear combination $\sum_i \alpha_i \mathbf{\Phi}_i$ of such objects will also obey these properties if $\sum_i |\alpha_i|^2 = 1$. If we allow quantum theories to have *deterministic states*, corresponding to states which return with certainty a given measurement outcome, we thence have that, as before, we do not gain nor lose anything by representing quantum states as vectors (with the above restrictions). Well, this is almost the truth modulo a small technicality, the global phase factor, which we will address shortly.

At this point, let the physicists in the readership (if any) hold their breath no longer: we are switching to Dirac's ket notation[13]. Let us also introduce the l_2 norm, to make our discussion clearer.

Definition 1.20 (l_2 norm). The l_2 norm (also called *Euclidean norm*) over a discrete vector space $\mathcal{V}(\mathcal{K})$ is the map $\|\cdot\|_2 : \mathcal{V} \mapsto \mathcal{K}$ defined as

$$\| \vec{x} \|_2 \triangleq \sqrt{\sum_i |x_i|^2}$$

where x_i are the coordinates of $\vec{x}$.

[13] For those of you who are not familiar with it, here is the one-liner definition: $|v\rangle$ is a column vector, and $\langle v|$ is the same vector in row form and with each entry conjugated, i.e. $\langle v| \triangleq |v\rangle^\dagger$. That's it, you're done. Go forth, and continue reading without fear!

Of course, the l_2 norm is also preserved under base changes. Its associated distance is the usual Euclidean distance, which is why by default it is not subscripted, i.e. $\|\cdot\| = \|\cdot\|_2$. However, one important property of the l_2 norm, unparallelled in the other norms such as l_1, is its relationship with the vectorial inner product

$$\langle \vec{x}, \vec{x} \rangle = \| \vec{x} \|^2 \tag{1.36}$$

Properly equipped now, let us examine the question of the global phase factor. In complex vector spaces there will be more than one vector meeting the above conditions which will return a probability 1 of observing a given result v_i: they are all co-linear, separated only by a multiplicative constant $e^{i\theta}$, the so-called arbitrary global phase factor. The fact that more than one vector represents the same state is mathematically inconvenient, and thus two solutions are possible to remove this arbitrariness:

a) Represent quantum states as equivalence classes of these vectors, according to the following equivalence relation:

$$\mathbf{\Phi} \equiv \mathbf{\Phi'} \iff \exists\, \theta \text{ s.t. } |\Phi\rangle = e^{i\theta}|\Phi'\rangle \tag{1.37}$$

b) Think of quantum states as 1-dimensional subspaces or *rays* of the complex vector space. For this, we can represent states with the rank 1 projectors $|\Phi\rangle\langle\Phi|$ onto those subspaces, where $|\Phi\rangle$ is any unit vector in l_2 norm in the corresponding subspace.

In either case, it will be "safe" to use a vector space to represent quantum states, in the sense that we are not introducing unnecessary artifacts into the theory. Even though slightly more inelegant, we will follow the first approach (it is far more common) to represent the state space as follows:

Definition 1.21 (PA-space). The (complex) *probability amplitude space* or *PA-space* of dimension N is the set $\mathrm{PA}(N)$ of unit vectors in the N-dimensional complex vector space up to a global phase factor, or more precisely the set of equivalence classes

$$\left[\, |\Phi\rangle \,\right] = \left\{\, |\Phi'\rangle \text{ s.t. } |\Phi'\rangle = e^{i\theta}|\Phi\rangle, \theta \in \mathbb{R}\,, \|\, \Phi' \,\| = 1 \right\}$$

Moreover, the PA-space $\mathrm{PA}(\mathcal{V})$ associated with a a sample space $\mathcal{V}$ is simply $\mathrm{PA}(|\,\mathcal{V}\,|)$.

When this is unambiguous, and as notation shorthand we will forthwith skip the equivalence class symbol $[\cdot]$ and represent states with just the ket symbol $|\cdot\rangle$.

Notice that so far we have purposely specified nothing about the dynamics of Quantum Theory. The only condition imposed on the dynamics by the Axioms 1 and 2 is that it must preserve the PA-space, which is exactly equivalent to preserving l_2 norm. In probabilistic theories, there was no axiom of dynamics other than the implied conditions of preserving PD-space. In the case of Quantum Theory, and for reason that we will let the physicist explain, we will accept one more independent axiom.

Axiom 3 (Dynamics). The dynamics is a linear homomorphism on the state space.

In summary, we have reduced Quantum Theory to only two basic assumptions: i) the state space has a structure which supports a probability amplitude based measurement rule (Axioms 1 and 2), and ii) the dynamics on this state space is linear (Axiom 3). It is important to note that while Axioms 1 and 2 and interdependent, Axiom 3 is completely independent. In particular, we have chosen to introduce them in that order to illustrate that linearity of the state space is not a consequence of Axiom 3, an easy mistake to make, but of Axioms 1 and 2. In fact, it is mathematically sound (even though possibly not from a physical point of view) to think of non-linear quantum theories, as we will discuss shortly.

As a result of this discussion, we have a minimal set of axioms and a suitable mathematical formalism to describe quantum models as follows.

Definition 1.22 (Quantum Model). A *quantum model* is a 2-tuple $(\mathcal{V},\mathcal{U})$ where:

- $\mathcal{V}$ is the set of results of all measurements, i.e. the sample space.
- The state space of the model is $\mathrm{PA}(\mathcal{V})$, the PA-space of probability amplitude distributions over the sample space $\mathcal{V}$.
- v_Φ is the random variable representing measurement outcomes of systems in state $|\Phi\rangle$. The probability of observing result $v_i \in \mathcal{V}$ in that case is given by

$$\Pr(\mathsf{v}_\Phi = v_i) = |\langle\Phi|e_i\rangle|^2 \tag{1.38}$$

- $\mathcal{U} = \{U(t_1,t_2) : \mathrm{PA}(\mathcal{V}) \mapsto \mathrm{PA}(\mathcal{V})\}$ is the family of evolution linear operators parametrised by valid times of observation $t_1 < t_2$, such that if the state of the system at t_1 is $|\Phi_1\rangle$, then the state of the system at time t_2 is

$$|\Phi_2\rangle = U(t_1, t_2)|\Phi_1\rangle. \tag{1.39}$$

1.5.2 Examples of Quantum Models

1.5.2.1 Quantum Mechanics

Quantum Mechanics is often introduced in terms of fundamental rules or postulates. Different text vary in their presentation , but the following presentation (based on projective measurements) is typical of an introductory Quantum Mechanics textbook.

State Space. States are unit vectors of a complex Hilbert space H.

Composition Rule. The combined state space of two systems having state space H_1 and H_2 is their tensor product $H_1 \otimes H_2$.

Dynamics. The time evolution operators are obtained by solving Schrödinger's equation. For time-invariant Hamiltonians, and for initial time $t = 0$, these operators (also called *propagators*) will take the form $U(t) = e^{\mathrm{i}Ht}$ where H is the Hamiltonian operator of the system. Since H is hermitian, the matrices $U(t)$ above are necessarily unitary.

Measurement Rule. Observable quantities are represented by hermitian operators called *observables*. The sample space of such an observable O is the set of its eigenvalues and the expected result of measurement on a state $|\Phi\rangle$ is given by $\langle\Phi|O|\Phi\rangle$.

How do these postulates compare to the abstract model of quantum theories we have presented above?

The state space rule is in form identical in content to our Axiom 1. Even though obscured by the introduction of observables, the measurement rule is not really different from our own Axiom 2 either. Thanks to the spectral decomposition theorem, observables can be decomposed into a sum of projection operators onto orthogonal subspaces, i.e. $O = \sum_i \lambda_i P_i$, where $P_i^2 = P$ are the projectors for the eigenspace corresponding to the eigenvalue λ_i. These subspaces are the subset of states which will yield the corresponding eigenvalue with probability 1. With respect to this mathematical formulation, we have adopted the following simplifications:

- Only measurements "along" the canonical base vectors are allowed, or more precisely projective measurements onto the corresponding 1-dimensional subspaces.

- We do not care what the numerical value of measurement is, and therefore the notions of eigenvalue and observable as the weighted sum of the corresponding projector loose meaning.

None of these simplifications weaken Quantum Theory, in that all other "observables" can be obtained within our model if:

- We introduce partial measurements within the model. Partial measurements can be viewed as random variables whose domain is not restricted to singleton events, but to any arbitrary event of the event space of that sample space.

- We precede measurements with the appropriate unitary transformations.

In fact, the measurement postulate can even be presented in a more general form by using POVM or superoperator-based measurements. The truth, however, is that all of these presentations are equivalent, and we chose the simplest possible one.

In all fairness, one of the reasons it is sensible for us to apply Occam's Razor principle here is because we are dealing with discrete and even finite sample spaces. For physicist, the notion of observables and measurement operators becomes truly useful when having to deal with physical space and time and thus with random variables on continuous domains. In Quantum Information Theory and Communication Complexity, where one qubit more or less makes a difference, the separate study of generalised measurements is relevant and necessary. But as we are, in this our endeavour, mostly interested in quantum computation, we can afford to do these simplifications.

As can be seen, the dynamics of the system is typically described in terms of the Hamiltonian or the corresponding propagators. Is this equivalent to our assumption of linearity in Axiom 3? The following celebrated theorem gives the answer.

Theorem 1.23 (Wiener). *The only linear transformations which are inner-product preserving are the unitary ($U^\dagger = U^{-1}$) and the anti-unitary ($U^\dagger = -U^{-1}$) transformations.*

Like many important theorems, the proof is surprising simple, yet its consequences are profound. It means that the fact that propagators in Quantum Theory are unitary is not an axiom either. In particular, it is not a consequence of the structure of the

Hamiltonians or any other assumption about them either. It is a consequence of a lower-level assumption than that, the linearity of the dynamics. While this might sound like heresy to some, it is important for us to take this view because in our abstract approach, such things as Hamiltonians and the Schrödinger's Equation do not necessarily make sense (what is the Hamiltonian of an abstract quantum circuit?). Unitarity does have an important consequence, however, relevant to both Quantum Mechanics and Quantum Computation: reversibility or time-symmetry.

We did not discuss the composition rule in our presentation because we do not feel that it is an axiom either: it is an indirect consequence of the measurement rule. When we bring two systems together, each with its own sample space $\mathcal{V}_1$ and $\mathcal{V}_2$, the sample space of the combined system must be their Cartesian product $\mathcal{V}_1 \times \mathcal{V}_2$. If anything should be taken as an axiom or questioned, it is this simple fact. From this fact, and given the association between results and canonical base vectors (Equation 1.4), it becomes necessary that the basis of the combined state spaces be the Cartesian product of the basis of each of the original state spaces, and that naturally defines the tensor product space. In other words, there is nothing special about the tensor product composition rule. It adds nothing to the theory.

It is also important to underline that in our presentation state space structure and measurement (Axioms 1 and 2) form an indivisible logical unit: the measurement rule dictates the form of the state space, and at the same time the measurement rule is defined in terms of state objects. Because of this, one probably should combine and present them as one single axiom, but we chose not do so for clarity.

1.5.2.2 Non-Linear Quantum Theories

What would happen if we got rid of Axiom 3? The dynamics would still have to be l_2 norm preserving but might not be linear. The possibility of a non-linear Quantum Mechanics has been studied and explored by some theoretical physicists. There is even some speculation about the fact that non-linearity might actually happen at scales smaller than that which we can observe today. The fact is that all experimental evidence to date suggests that linearity holds, with very high levels of accuracy. In any case, discussing such a possibility is way beyond the scope of this document and the ability of the author.

Nonetheless, one interesting realisation is that non-linear quantum computation models might have significantly more power than the standard (linear) Quantum Computing models. In particular, Abrams and Lloyd [AL98] have shown that a non-linear Quantum Computing model with gates exhibiting even "small amounts" of non-linearity, would allow the resolution in polynomial-time of both NP-complete problems and problems poly-time reducible to #P functions (i.e. problems in $\mathrm{P}^{\#\mathrm{P}}$), something which is widely believed that the standard linear Quantum Computing models cannot do.

1.5.2.3 Quantum Turing Machines

Quantum Turing Machines (QTM) are the original quantum computing model. They were first introduced by Deutsch [Deu85], and they are discussed at length by Bernstein and Vazirani in [BV97]. The following definition is akin to that of Probabilistic TM's given in Definition 1.14 and is inspired on that of [BV97].

Definition 1.24 (Quantum Turing Machine). A *quantum Turing machine* is a 4-tuple (Q, q_0, q_f, v) where

- Q is the set of states of the automaton
- $q_0 \in Q$ and $q_f \in Q$ are the initial and final (accepting) states of the automaton
- π is an *amplitude transition function*

$$v : (Q \times \Gamma) \times (Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}\}) \longmapsto \mathbb{C}$$

with the property that for all $q \in Q$, $b \in \Gamma$

$$\sum_{\substack{q' \in Q \\ b' \in \Gamma \\ d' \in \{\mathtt{L},\mathtt{R}\}}} |v(q, b, q', b', d')|^2 = 1$$

Dynamics. Bernstein and Vazirani describe both probabilistic and quantum TM's in terms of holistic infinite but countably dimensional linear operators $\hat{P}$ or $\hat{U}$, which are stochastic and unitary, respectively. Very much like the *transition table* $\hat{M}$ sometimes used to describe the dynamics of deterministic machine, these operators can be viewed as infinite dimensional matrices, whose columns and rows are indexed by configurations of the machine. In particular, Equation 1.23 properly defines the operator P given the transition function π in the case of PTM's. In the case of a deterministic TM, the transition table can be defined in the same fashion and takes the form of a 0-1 stochastic matrix.

For quantum TM's we could similarly define the "entries" $(U)_{c,c'}$ of the operator U by replacing π with υ in Equation 1.23. Note however, that if we define U in this way, it will not be unitary, because we have two inherently non-reversible transformation: a) when the machine hits the left-end of the tape, and b) when the machine reaches the halting state and "loops" in it by staying in that state with probability amplitude 1. The standard way to deal with this (as in [BV97]) is to remove the two point of contention by a) considering a two-way tape, and b) requiring that the machine be *well behaved*, which is equivalent to saying that it halts after the same number of steps for all inputs of the same size.

We thus have that U is defined as follows

$$U(c,c') = \begin{cases} \upsilon(q,b,q',c,\mathtt{L}) & \text{if } c = [ua,q,bv] \text{ and } c' = [u,q',acv] \quad \textit{(head moves left)} \\ \upsilon(q,b,q',c,\mathtt{R}) & \text{if } c = [u,q,abv] \text{ and } c' = [uc,q',bv] \quad \textit{(head moves right)} \\ 0 & \text{otherwise} \end{cases} \tag{1.40}$$

Measurement. Accordingly, we can assign to each configuration c an amplitude after i steps (as long as $i \leq t$, where $t = t(n)$ is the (uniform) number of steps on inputs of length n) similarly as we did for PTM's in Equation 1.24

$$U^{(i)}(c) = \begin{cases} \delta_{c_0}(c) & \text{if } i = 0 \\ \sum\limits_{c' \in C} U(c',c)U^{(i-1)}(c') & \text{if } i \geq 1 \end{cases} \tag{1.41}$$

Thus, at each time step i, we can define a measurement outcome probability, by applying the quantum measurement rule

$$\Pr(\text{QTM } M \overset{\text{measure}}{\longrightarrow} c \text{ after } i \text{ steps}) = |U^{(i)}(c)|^2 \tag{1.42}$$

1.5.2.4 Quantum Circuits

Quantum Circuits were also introduced by Deutsch [Deu89]. They can be introduced in a very similar fashion as probabilistic circuits. However, since all transformations must be l_2 norm preserving, they must also be reversible, and in particular so must all gates. As a result all gates must have the same number of input and output connections. Furthermore, all circuits must have the same number of input and output nodes, which will also coincide with the width of all temporal cuts. Circuits with such a topology are often referred to as *gate arrays.*

Definition 1.25 (Quantum Circuit). A *quantum circuit* is a circuit $C = (\mathcal{I}, \mathcal{O}, \mathcal{G}, \mathcal{W})$ where

1. Each input node $i \in \mathcal{I}$ is associated a Boolean value $x_i \in \Sigma$.
2. There is an identical number of input and output nodes, i.e. $|\mathcal{I}| = |\mathcal{O}|$ called its *width.*
3. Each gate $g \in \mathcal{G}$ has an associated transition amplitude map $v_g : \Sigma^k \times \Sigma^k \mapsto \mathbb{C}$ with the property that

   $$\sum_{y_1,\ldots,y_k} |v_g((x_1, \ldots, x_k), (y_1, \ldots, y_k))|^2 = 1$$

   where k is the number of input and output connections of g.

States. As with probabilistic circuits, we must associate the states of a quantum circuit with its temporal cuts.

Definition 1.26 (State of a Quantum Circuit). Let C be a quantum circuit of width n and let $t = (\mathcal{G}_{\mathrm{L}}, \mathcal{G}_{\mathrm{R}})$ be one of its temporal cuts.

A *state* of a quantum circuit C at t is a vector $|\phi\rangle$ in the PA-space $\mathrm{PA}(2^n)$, where each canonical vector $|e_i\rangle$ has been associated with the i-wire across the cut t, according to some fixed ordering (normally top-to-bottom).

Measurement. Similarly as for probabilistic circuits, we can use the quantum measurement rule to define joint probabilities of outcomes for the wires of a temporal cut. As before, let w_i, $1 \leq i \leq n$, be the random variable associated with measuring the i-th wire across the cut (counting from the "top" of the circuit). If the state of the circuit at t is described by $|\phi_t\rangle$, then we have that

$$\Pr(\mathsf{w}_1 = b_1, \mathsf{w}_2 = b_2, \ldots, \mathsf{w}_m = b_m) = |\langle b|\phi\rangle|^2 \tag{1.43}$$

where $b = b_1, b_2, \ldots, b_m$ and $|b\rangle = |b_1\rangle \cdots |b_n\rangle$.

Dynamics. The dynamics of a quantum circuit can be defined similarly to that of a probabilistic circuit. Let $|\phi\rangle$ be the state before a gate g is applied and $|\phi'\rangle$ be the state after g, but before any other gates have been applied. Unlike for probabilistic gates, g must have the same number of input and output connections k; also all cuts will have the same width m. As before, assuming w.l.o.g. that g applies to the first k wires in the cut, we have that

$$\begin{aligned} |\phi'\rangle_{y_1,\ldots,y_k,y_{k+1},\ldots,y_m} &\triangleq \Pr(\mathrm{w}'_1 = y_1, \ldots, \mathrm{w}'_k = y_k,\ \mathrm{w}'_{k+1} = y_{k+1}, \ldots, \mathrm{w}'_m = y_m) \\ &= \sum_{x_1,\ldots,x_k} v_g\left((x_1,\ldots,x_k),(y_1,\ldots,y_k)\right) \cdot |\phi\rangle_{x_1,\ldots,x_k,y_{k+1},\ldots,y_m} \end{aligned} \tag{1.44}$$

where $|\phi\rangle_x$ represents the i-th coordinate of $|\phi\rangle$, when $x = x_1 \ldots x_m$ is the i-th string in Σ^m in lexicographical order.

However, the dynamics of gates is not usually defined with amplitude transition functions in this way, but rather in terms of unitary matrices (this time of finite dimensions, which is much tidier than in the case of QTM's). The "out-of-context" action of a k-ary gate g , i.e. the action of g when it is the only gate in a circuit, is described with a matrix U_g of dimension $2^k \times 2^k$ whose columns are the images $U|e_i\rangle$ corresponding to the i-th canonical vector of $\mathrm{PA}(2^k)$; in other words $(U_g)_{i,j} = v_g(x,y)$, where x and y are the j-th and i-th strings in lexicographical order. Similarly, we can associate to any circuit C of width n, a unitary matrix U_C of order $2^n \times 2^n$ describing its action on from the input nodes to the output nodes. We will describe in more detail in Chapters 3 and 4 how these circuit matrices can be constructed and succinctly represented.

1.6 A Unified View of Theories

The goal of this chapter was two-fold: to provide a simplified yet not less general account of Quantum Theory in order to try to understand its essence, and also provide a "big picture", epistemologically speaking, within which we could place Quantum Theory. In order to accomplish the latter we present a unifying view of the theories we have seen so far, based on the common mathematical formalism of vector spaces. This formalism will allow us to compare and classify these theories and their variations, thus completing the taxonomy introduced in Section 1.4.

1.6.1 The Vectorial Representation

With the help of vectorial representations the three theories that we have discussed can be brought together under the same light.

In probabilistic models, the canonical base vectors are identified with points in the sample space of measurement variables (Equation 1.4), and the state space is constituted by l_1 norm unit vectors with non-negative coefficients, i.e. the simplex generated by the canonical base vectors which we called PD-space (Definition 1.10). Using Dirac's notation to represent PD's, we can succinctly re-write the two axioms of probability theory as follows:

Definition 1.27 (Probabilistic Theory). Probabilistic Theory is re-defined with the following axioms:

Axiom 1 (States). States are represented as vectors $|p\rangle$ in $\mathrm{PD}(N) \subset \mathbb{R}^N(\mathbb{R})$, i.e.

$$\| p \|_1 = 1 \text{ and } |p\rangle_i = \langle p|e_i\rangle \geq 0 \tag{1.45}$$

Axiom 2 (Measurement Rule). Given a state $|p\rangle$, the probability of obtaining $v_i \in \mathcal{V}$ is given by

$$\Pr(\mathsf{v}_p = v_i) = \langle p|e_i\rangle \tag{1.46}$$

Axiom 3 (Dynamics). The dynamics of the model is defined by any family of PD-space preserving functions $\mathcal{P}$ such that $P(t_1, t_2) : \mathrm{PD}(\mathcal{V}) \mapsto \mathrm{PD}(\mathcal{V})$.

In this case, the maps P_t are non-negativity and l_1 norm preserving. If they are linear (which is not a requirement), then they can be represented by stochastic matrices, as discussed in Section 1.3.3.1, and we can re-write the dynamics equation for probabilistic models (1.5) for states $|p_0\rangle$ and $|p_t\rangle$ at times t_0 and t, respectively, as

$$|p_t\rangle = P_t|p_0\rangle \tag{1.47}$$

Even though this might not be practical or technologically viable, nothing fundamental in a deterministic theory prevents all properties of a system from being ultimately measured. Under this assumption, which we believe quite reasonable (i.e. that $\mathcal{V} = \mathcal{S}$), one can view deterministic models as special case of probabilistic theories, as discussed in Section 1.4. A formal vectorial representation of such models is possible, in which the canonical base vectors are identified with the actual states of the deterministic (Equation 1.7). In this case the state space is restricted to precisely these vectors, and the dynamics must preserve this set.

Definition 1.28 (Deterministic Theory). Is defined with the following axioms:

Axiom 1 (States). States are represented as vectors $|p\rangle$ in *D-space*, i.e. the following subset of $\mathbb{R}^N(\mathbb{R})$

$$\| p \|_1 = 1 \text{ and } |p\rangle_i = \langle p|e_i\rangle \in \{0, 1\} \tag{1.48}$$

Axiom 2 (Measurement Rule). Given a state $|p\rangle$, the probability of obtaining $v_i \in \mathcal{V}$ is given by

$$\Pr(\mathsf{v}_p = v_i) = \langle p|e_i\rangle \tag{1.49}$$

Axiom 3 (Dynamics). The dynamics of the model is defined by any family of D-space preserving functions.

Compared with the probabilistic case, Axiom 1 has the extra constraint on the inner product that it must not just be non-negative but also $\{0, 1\}$-valued. Axiom 2 is identical and so is Axiom 3 if we replace "PD-space" with "D-space".

Finally, we have Quantum Theory whose axioms can be re-formulated in similar terms:

Definition 1.29 (Quantum Theory). Is defined with the following axioms:

Axiom 1 (States). States are represented as vectors $|p\rangle$ in $\mathrm{PA}(N) \subset \mathbb{C}^N(\mathbb{C})$, i.e.

$$\| p \| = 1 \tag{1.50}$$

Axiom 2 (Measurement Rule). Given a state $|\phi\rangle$, the probability of obtaining $v_i \in \mathcal{V}$ is given by

$$\Pr(\mathsf{v}_\phi = v_i) = |\langle\phi|e_i\rangle|^2 \tag{1.51}$$

Axiom 3 (Dynamics). The dynamics of the model is defined by any family of PA-space preserving linear homomorphisms $\mathcal{U} = \{U(t_1, t_2) : \mathrm{PA}(\mathcal{V}) \mapsto \mathrm{PA}(\mathcal{V})\}$.

Let us now summarise our main conclusions:

- There is nothing special about vector spaces. Using this formalism is nothing but a mathematical convenience. It adds nothing nor does it detract anything from the theories.
- The difference between these theories are those induced by the measurement rules. These variations induce a different subset of the vector space as the valid state space.
- The only incontrovertible requirement of the dynamics is that it must be closed within the state space. In the case of Quantum Theory, there is the additional axiomatic requirement that the transformations be linear, but this is not a consequence nor is it the cause, in our view, of the linear structure of the state space.

Table 1.1 summarises the axioms of these theories when represented with as vector space.

	Deterministic	Probabilistic	Quantum
Sample space	$\mathcal{V} = \{v_1, v_2, \dots, v_N\}$		
State space	D-space $\boldsymbol{d} = \sum_{i=1}^{N} d_i \vec{e_i}$ s.t. • $\parallel \boldsymbol{s} \parallel_1 = 1$ • $d_i = \langle \boldsymbol{d} \vert e_i \rangle \in \{0, 1\}$	PD-space $\boldsymbol{p} = \sum_{i=1}^{N} p_i \vec{e_i}$ s.t. • $\parallel \boldsymbol{p} \parallel_1 = 1$ • $p_i = \langle \boldsymbol{s} \vert e_i \rangle \geq 0$	PA-space $\boldsymbol{\Phi} = \sum_{i=1}^{N} \alpha_i \vec{e_i}$ s.t. • $\parallel \boldsymbol{\Phi} \parallel_2 = 1$ • $\alpha_i = \langle \boldsymbol{s} \vert e_i \rangle \in \mathbb{C}$
Measurement $\Pr(\mathsf{v} = v_i)$	$\langle d \vert e_i \rangle$	$\langle p \vert e_i \rangle$	$\vert \langle \Phi \vert e_i \rangle \vert^2$
Dynamics	Any map $d\colon \{e_1, ...\} \mapsto \{e_1, ...\}$ $\vec{e_i} \longmapsto \vec{e}_{d(i)} = D\vec{e_i}$	Any map $P : \mathrm{PD}(N) \mapsto \mathrm{PD}(N)$ $\vec{p} \mapsto P(\vec{p})$	Any *linear* map $U : \mathrm{PA}(N) \mapsto \mathrm{PA}(N)$ $\vert \Phi \rangle \mapsto U \vert \Phi \rangle$

Table 1.1: The Deterministic, Probabilistic and Quantum Theories described in terms of the same vectorial representation.

1.6.2 Variations on the Dynamics

In our discourse so far, the axioms of dynamics come last and are added only if they are needed. The measurement rule dictates the structure of the state space, which in turn dictates any restrictions on the dynamics.

In the deterministic case, the state space (D-space) is comprised exclusively of the canonical base vectors and in principle we can think of the dynamics as a map on indexes, i.e. $D : [1..n] \mapsto [1..n]$, such that $\vec{e}_i \overset{D}{\longmapsto} \vec{e}_{D(i)}$. However, we can also "pretend", without changing the theory, that the mapping is linear on the rest of the vector space. This has the advantage for us that we can represent D as an $N \times N$ matrix containing a single 1 in every column, and zeroes everywhere else, i.e. $\{0, 1\}$-valued stochastic matrices. The image $\vec{e}_{D(i)}$ of $\vec{e}_i$ can thus be re-written as $D\vec{e_i}$.

In the probabilistic and quantum case, the only incontrovertible requirements of the dynamics is that they are closed within the PD- and PA-state, respectively. In the former, the necessary and sufficient conditions are preservation of the l_1 norm and of the non-negativity of coefficients. In the quantum case, l_2 norm preservation is sufficient. These conditions are not axioms *per se*, as they are a consequence of the previous axioms. If however, we make it a requirement that transformations be linear, we are restricting the set of valid transformations to stochastic matrices in the probabilistic case, and to unitary and anti-unitary matrices in the quantum case. At this point, we will brush aside anti-unitary matrices. The reasoning behind it, albeit quite "weak", is that the extra minus sign is tantamount (in most cases) to a global phase factor[14].

What about reversibility? Reversibility has many faces [Bra01]. Logical reversibility of a transformation means that it is possible to uniquely determine the input of the transformation given its images. In the Theory of Reversible Computation, one also talks of *physical reversibility* as the property of a computation to be performed exclusively with logically reversible elementary operations, and of *thermodynamic reversibility* as that of performing such computation without dissipating heat.

The reason behind discussing reversibility is because we are interested in studying its implications as a possible restriction or requirement on the dynamics of a theory. As such, the notion that is relevant to us is that of logical reversibility, which simply means that the dynamics map is one-to-one. It is tantamount to the possibility of determining exactly the states in the past, if we have full knowledge of the state in the present. This is a stronger notion than plain time reversibility, which is just the possibility that the dynamics map can be applied in both directions of time.

In the deterministic case the notion of logical reversibility comes quite naturally. Let

[14] This justification is mildly unsatisfactory, because these phase factors are introduced even in the density matrix representation and might, for example, affect the outcome of computation if we consider controlled anti-unitary operations. There is evidence, however, that adding these transformations do not change the model of computation[Bra03].

$d : [1..N] \mapsto [1..N]$ be the index map representing the base change $\vec{e_i} \mapsto \vec{e}_{d(i)}$. The corresponding 0-1 stochastic matrix is D, whose only non-zero entries are $D_{i,d(i)}$. While D is stochastic, there could be more than one non-zero entry in the j-th row, indicating that the $\vec{e_j}$ is the image of two different inputs. Requiring that d is one-to-one is equivalent to requiring that D be doubly stochastic. However, we have the following simple and trivial fact, which characterises what we get in this case:

Lemma 1.30. *The only 0-1 matrices which are also doubly stochastic are the permutation matrices.*

It is possible to construct examples of reversible non-linear transformations in both the probabilistic and quantum cases. Very little else can be said about them, as they have not been well studied.

As we discussed in Section 1.5.2.1, the linearity axiom of Quantum Theory implies unitarity. Unitary matrices are inherently logically reversible, as their inverses always exist and is in fact their conjugate transpose. Thus, the inverse image of any state $|\Phi\rangle = U|\Psi\rangle$ is equal to $U^{-1}|\Phi\rangle = U^{\dagger}|\Phi\rangle$.

This is not the case for stochastic matrices, as they do not necessarily have multiplicative inverses. The existence of well-defined transition probabilities allow us to write the dynamics equation of such models as a forward inference rule:

$$\begin{aligned}(\vec{p_t})_j = \Pr(\mathsf{v}_t = v_j) &= \sum_i \Pr(\mathsf{v}_0 = v_i)\,\Pr(v_i \xmapsto{P_t} v_j) \\ &= \sum_i (\vec{p_0})_i\, P_t(j,i) \qquad (1.52)\end{aligned}$$

which is sound since the sum along columns is $\sum_i \Pr(v_i \mapsto v_j) = 1$. In the case of doubly stochastic matrices, we can also write a backward inference rule as follows

$$\begin{aligned}\Pr(\mathsf{v'}_0 = v_i) &= \sum_j \Pr(\mathsf{v}_t = v_j)\,\Pr(v_i \xmapsto{P_t} v_j) \\ &= \sum_j (\vec{p_t})_j\, P_t(j,i) \qquad (1.53)\end{aligned}$$

These equations define proper probabilities because the sum along the rows $\sum_j \Pr(v_i \mapsto v_j)$ is also equal to one. We can thus talk of a well defined "reverse" probabilistic map $P_t{}^{\mathsf{t}}$. Let $\vec{p'}$ be the corresponding PD which is given by the expression $\vec{p'} = P_t{}^{\mathsf{t}}\vec{p_t}$. This PD represents predictions about the past, in the absence of knowledge about the current outcome of measurement, but with knowledge of the current distribution $\vec{p_t}$. Unfortunately, doubly stochastic matrices do not always have inverses either, and this PD does not correspond to the state of the system at time t_0. More precisely, if $\vec{p_t} = P_t\vec{p_0}$, then $\vec{p'}$ bears no relation with $\vec{p_0}$. Furthermore, it is not even true that $P_t\vec{p'} = \vec{p}$. With doubly stochastic matrices, one can say that the direction of time can be "reversed", but this does not brings us back to the "past" in the way we would have expected. In fact, the only time when this is possible is when we have $\vec{p'} = \vec{p_0}$, $\forall\vec{p}$, which is only the case for the permutation matrices, as they are the only doubly stochastic matrices whose inverses are also doubly stochastic.

Another special case of interest is that of symmetric doubly stochastic matrices. In their case, Equations 1.52 and 1.53 are the same, which means that time evolution is the same in the future and past directions. These are often called *time reversible* Markov chains, but the term *time symmetric* is more accurate and more in tune with the concept of reversibility as described above. In terms of matrix representation of the dynamics it simply means that $P_t = {P_t}^{\mathrm{t}}$. The equivalent concept for the deterministic case are permutations of order 2, which are composed exclusively of disjoint transpositions, i.e. nilpotent permutations Π s.t. $\Pi^2 = \mathrm{id}$. Similarly, in the quantum case the "time-symmetric" unitary transformations will be those for which $U = U^\dagger$, which corresponds exactly to the nilpotent unitary transformations with $U^2 = \mathrm{id}$ (e.g. Walsh-Hadamard, CNOT, 180-degree rotations, etc.).

		Deterministic	Probabilistic	Quantum
Unrestricted		$\{0,1\}$ stochastic	l_1 norm & ≥ 0 preserving	l_2 norm preserving
Linear	Unrestricted		Stochastic	(Anti)-Unitary
	Reversible	Permutations	Doubly stochastic	
	Logically rev.		Permutations	
	Time symmetric	Nilpotent permutations	Symmetric doubly stochastic	Nilpotent unitary

Table 1.2: A comparative table of dynamics in the deterministic, probabilistic and quantum theories.

1.6.3 Quantum vs. Classical Models

Let us now try to place Quantum Theory within the taxonomy of classical theories which we described in Section 1.4.

The first realisation is that quantum theories are not necessarily probabilistic in the sense defined. Consider the standard (linear) quantum theory. The PA-vectors representing states naturally define PD-vectors for the measurement of outcomes through the measurement rule of Equation 1.38. This can be defined as a map $M : \mathrm{PA}(N) \mapsto \mathrm{PD}(N)$ expressed as follows

$$(\vec{p_\Phi})_i \xmapsto{M} ||\Phi\rangle_i|^2 = |\langle\Phi|e_i\rangle|^2 \tag{1.54}$$

or equivalently

$$\vec{p_\Phi} = M(|\Phi\rangle) = \mathrm{Diag}\,(\,|\Phi\rangle\langle\Phi|\,) \tag{1.55}$$

where Diag$(\cdot)$ represents the vector constructed with the diagonal of a matrix. We can also give the following expression for the PD of a future state as

$$\begin{aligned} |\Phi\rangle &\longmapsto U|\Phi\rangle \\ \vec{p_\Phi} &\longmapsto \mathrm{Diag}(U|\Phi\rangle\langle\Phi|U^\dagger) \end{aligned} \tag{1.56}$$

The problem is that in general the expression $\mathrm{Diag}(U|\Phi\rangle\langle\Phi|U^\dagger)$ cannot be expressed in terms of the diagonal of $|\Phi\rangle\langle\Phi|$ alone, and thus there is no map $P : \mathrm{PD}(N) \mapsto \mathrm{PD}(N)$ s.t. $M(U|\Phi\rangle) = P(M(|\Phi\rangle))$. This situation is illustrated in Figure 1.5.

While not all quantum theories are probabilistic, our first question, then, is whether it is possible for a quantum theory to simultaneously be a probabilistic theory. Conversely, we can ask whether a probabilistic theory can be made quantum. This is not obviously true. Even though the map M is not one-to-one, it would be always possible to construct a pre-image of a given PD-vector by taking its component-wise square roots. However, there is no prescription for generating a proper dynamics on these PA-vectors. For linear maps, one might be tempted to consider the entry-wise square root of the original probabilistic dynamics matrix P_t as the quantum dynamics transformation. The images of the base vectors in PA-space under this map (i.e. the columns of this square root matrix) would indeed be unit vectors in l_2 norm, however since these columns are not orthogonal

$$\begin{tikzcd}
\vert\Phi_0\rangle \arrow[r, "U_t"] \arrow[d, "M"'] \arrow[dr, "M \circ U_t"] & \vert\Phi_t\rangle = U\vert\Phi_0\rangle \arrow[d, "M"] \\
\vec{p}_0 \arrow[r, "\times" marking] & \vec{p}_t = \mathrm{Diag}(U\vert\Phi_0\rangle\langle\Phi_0\vert U^\dagger)
\end{tikzcd}$$

Figure 1.5: The relationship between states and PD's of outcomes in quantum models. The map M associates with each quantum states the probability distribution of measurement in the canonical basis. The crossed-out arrow indicates that in general no direct deterministic map exists between the PD's of an initial state and that of its image under a unitary transformation. In other words, this diagram does not commute.

the resulting matrix would not be unitary, and consequently would not preserve l_2 norm on non-basis vectors. Thus, the resulting PA-dynamics would not be linear.

The following theorem gives a definite answer to both of these questions. It was proven by Boyer [Boy03], based on an initial partial result restricted to linear transformations discovered by the author.

Theorem 1.31. *The only unitary transformations U for which there exists a map P s.t. $P \circ M = M \circ U$ are the permutation matrices up to phase changes. More precisely, matrices of the form $U = D\Pi$, where Π is a permutation matrix, and $D = \sum_j e^{i\theta_j}\vert j\rangle\langle j\vert$ is a (diagonal) phase change matrix.*

In pictorial terms, what this theorem says is that the only transformation that will make the diagram in Figure 1.5 commute are the permutation matrices, up to phase factor. It is important to note, though, that given such a $U = D\Pi$, the corresponding map P will be the the underlying permutation Π. This means that for all intents and purposes we can ignore these phase factors, as they will not affect the probabilities of outcomes.

1.6.4 An Algebraic Twist to the Vectorial Representation

In Section 1.6.1 we introduced a representation of theories inspired on the minimal restrictions that the measurement rule imposed on a generic vector space. In the case of classical theories, the state spaces are restricted sets of the Euclidean vector space, while in the quantum case we use complex vector spaces. However, it is also possible to take the opposite approach and describe some of these theories within a uniform measurement rule and dynamics axioms. The difference between them is then established by changing the actual vector space.

Consider the Boolean Algebra $(\mathbb{B}, \vee, \wedge)$. While vector spaces are usually defined in terms of fields, it is also possible to verify that the $\mathbb{B}^N(\mathbb{B})$ has all the relevant vector space properties. We can define all required vectorial operations on it, including inner product. On this vector space we have the correct notion of orthogonality (two strings are orthogonal if they are distinct everywhere). Furthermore, the l_2 and l_1 norms coincide — the norm of a vector is simply the hamming weight of the corresponding binary string— and the only unit vectors are the canonical base vectors. We can thus view the D-space as the unit ball in $\mathbb{B}^N(\mathbb{B})$ and define the measurement rule and dynamics in terms of the l_2 norm: the model definition is still the same. In other words, deterministic models can be viewed as quantum models with a PA-space defined on a Boolean vector space.

On the other hand, suppose that we had a PA-space restricted to vectors of real non-negative amplitudes. The set $\mathbb{R}_+ = [0, \infty)$ of non-negative real numbers is not a field either, but as in the case of the Booleans, $\mathbb{R}_+^N(\mathbb{R}_+)$ can be considered a vector space with all the relevant properties. As with regular quantum theories, the map $M : \mathrm{PA}(N) \mapsto \mathrm{PD}(N)$ defines the PD of measurement outcomes associated with a given state. The difference now is that M is necessarily one-to-one and onto. The following lemma characterises exactly the kind of dynamics that we can have in this case.

Lemma 1.32. *The only linear transformations which preserve l_2 norm and non-negativity are the permutation matrices.*

Proof. A direct proof can be obtained by using elementary linear algebra techniques. However, we find it more interesting and instructive to use Theorem 1.31. Because M^{-1} is well defined (take the positive square root of the corresponding probability), the map $M \circ U \circ M^{-1}$ describes the dynamics of PD's, i.e. $\vec{p_t} = M(U(M^{-1}(\vec{p_0})))$. By Theorem 1.31, this imposes a restriction on U to be a permutation matrix with phase changes $e^{i\theta_i}$. But since U must preserve non-negativity, it is just a regular permutation matrix, with no phase changes. □

Having run into this special kind of probabilistic models twice now, and due to its importance in later chapters, we will give it a name.

Definition 1.33 (Extrinsic Probabilistic Model; Extrinsic Theory). A probabilistic model $(\mathcal{V}, \mathcal{P})$ is said to be *extrinsic* if all $P(t_1, t_2) \in \mathcal{P}$ are permutation matrices.

Extrinsic Theory is the subclass of Probabilistic Theory encompassing extrinsic probabilistic models and is represented by the symbol X.

Theorem 1.31 and Lemma 1.32 each provide an alternate characterisation for extrinsic models. The name "extrinsic" is motivated by computational models. The fact that the dynamics is perfectly deterministic means that randomness must be introduced in the system by initialising it in a non-deterministic state. This is in contrast with *intrinsic* probabilistic models, where the dynamics is allowed to be probabilistic (e.g. stochastic matrices) and can "generate" randomness, even when starting with a deterministic initial state.

Thus, we have that deterministic theories, extrinsic probabilistic theories, and quantum theories can all be represented in the PA-space formalism, each being generated with its implicit restrictions by changing the vector space on which the PA-space is defined. This is summarised in Table 1.3. While this approach is mathematically convenient and will allow us to give answers to some interesting questions regarding Complexity Theory in the following chapters, we wish to stress that we believe it is misleading and epistemologically incorrect to think of these theories as being defined by the underlying algebra of these vector spaces.

	Deterministic	Extrinsic	Quantum
Sample space	$\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$		
State space	$\lvert\Phi\rangle \in \ldots$		
	$\mathbb{B}^N(\mathbb{B})$	$\mathbb{R}_+^N(\mathbb{R}_+)$	$\mathbb{C}^N(\mathbb{C})$
	s.t. $\lVert \Phi \rVert_2 = 1$		
Measurement	$\Pr(\mathrm{v} = v_i) = \lvert\langle\Phi\rvert e_i\rangle\rvert^2$		
Dynamics	Permutations		Unitary

Table 1.3: A common algebraic picture for deterministic, extrinsic probabilistic, and quantum models. The measurement rule and the unit vector (in l_2 norm) requirement are identical, with all other specific restrictions of each model are embedded in the choice of the vector space.

1.6.5 The Big Picture: A Complete Taxonomy of Theories

In this chapter, we have introduced a generic abstract look at various types of scientific theories by addressing the following questions, in this order:

1. What can be measured? The answer to that question determines the sample space.[15]

2. How can it be measured or how are the results of measurements defined? This is the measurement rule, which in turn implies requirements on the states and defines the structure of the state space.

3. How do the systems change? In other words, what is the dynamics on the state space.

This approach has allowed us to come up with minimalist, non-superfluous characterisations of the various theories. In particular, from the usual postulates of Quantum Theory we have retained only those which we believe fundamental, or at least for the purposes of the Theory of Computation. We have thus completed the exorcism announced in the introduction...

Deo gratia...

But we have gone further, and produced several interesting byproducts.

1. We have found that linear (or stochastic) probabilistic models have two interesting characterisations: they have hidden variable interpretations and they have well defined transition probabilities (Theorem 1.18).

2. We have introduced a common mathematical representation for all of these theories based on vector spaces, which allows us to understand and compare the differences between them (Table 1.1).

3. We have introduced and discussed a rich variety of models of sub-theories in which the dynamics is restricted beyond the requirements imposed by the measurement rule (Table 1.2).

[15] In our case, we assumed a discrete sample space throughout, for simplicity of argument and because that will be sufficient for our purposes. However, the conclusions presented here still apply to other types such as continuous sample spaces.

4. We have introduced an "algebraic" meta-theory based on vector spaces and the l_2 norm, which characterises deterministic, extrinsic probabilistic, and quantum theories as instances of the same meta-model, each obtained with a different algebra on which the vector space is defined: $\mathbb{B}$, $\mathbb{R}_+$, and $\mathbb{C}$, respectively. (Table 1.3).

5. As a result, we have also found two characterisations of probabilistic extrinsic probabilistic models: they are the only (non-trivial) logically reversible probabilistic models (Lemma 1.30) and they are a special case of quantum theories with non-negative amplitudes.

In terms of classification of the main theories and their variations, the following inclusion chain summarises what we know.

$$\begin{rcases} \mathrm{Q} \supseteq \quad \mathrm{X} \subseteq \mathrm{DS} \\ \mathrm{D} \end{rcases} \subseteq \mathrm{LP} = \mathrm{HV} \subseteq \mathrm{P} \supseteq \mathrm{NLP} \tag{1.57}$$

$$\mathrm{Q} \cap \mathrm{P} = \mathrm{X} \tag{1.58}$$

where we have introduced the symbols DS and Q for probabilistic theories with doubly stochastic dynamics and for quantum theories, respectively. This taxonomy of theories is represented in Figure 1.6.

It is important to note that while Figure 1.6 looks like a complexity class Venn diagram, it is not. These generic and abstract epistemological conclusions do not necessarily translate verbatim into the language of complexity classes whose underlying computational devices are modelled according to these theories. Studying how these theories generate sound notions of computation and well defined complexity classes, as well as the relationship between them, is the topic of the next chapter.

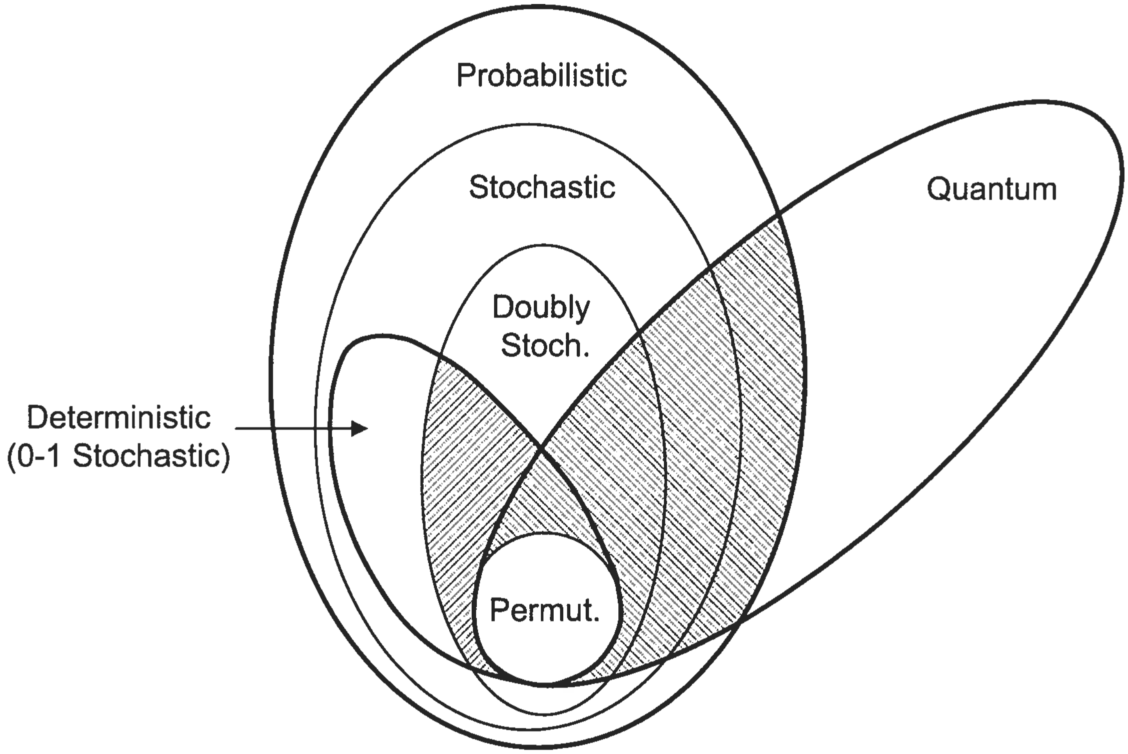


Figure 1.6: A Venn diagram representing a complete taxonomy of theories including deterministic, quantum, and the various varieties of probabilistic theories. The shaded areas of the diagram indicate that no models exist with those properties.

CHAPTER 2

CLASSICAL AND QUANTUM THEORIES OF COMPUTATION

In Chapter 1, we introduced and discussed various types of models within the Deterministic, Probabilistic and Quantum theories. While we arrived at some interesting abstract conclusions, these are not necessarily directly applicable to the corresponding computational models. In this chapter, we will re-describe the models of Classical and Quantum Theory of Computation in order to create a unified view of the corresponding complexity classes.

We start by quickly introducing the traditional models of the classical Theory of Computation, followed by those of Quantum Computation. We then re-introduce classical computation with "quantum-like" formulations, discuss its implications and describe a common complexity picture for all of these computational models.

2.1 Deterministic Computation

2.1.1 Turing Machines

Turing Machines were introduced in Section 1.2.2.1 as an example of a deterministic model. Their states, more commonly referred to as configurations, were described and so was their dynamics. Without loss of generality, we assumed that the sample space of "measuring" a TM would include a description of the state of its internal finite control (a deterministic finite automaton). However, no semantics was given to Turing Machines as computation devices.

Turing Machines such as defined in Definition 1.3 are typically described as computation devices in the following sense:

1. The initial state of the TM is $[q_0, x]$, where x is a binary string representing the logical input to a problem which we intend or pretend that the TM will solve.

2. Upon halting, the TM will be in the configuration $[F_{\mathrm{L}}, q_f, F_{\mathrm{R}}]$. The *output* of the computation on input x, represented as $M(x)$ is the (possibly empty) string $F_{\mathrm{L}}(x)$ to the left of the tape head.

3. If the TM does not halt, then the output of the TM on input x, $M(x)$ is not defined.

With this semantics, the following formal definitions can be given of a TM performing computational tasks.

Definition 2.1. Let L be a language in Σ^*. A TM M is said to *decide* L if

$$\begin{aligned} &\forall x \in L,\ M(x) = \text{“1”} \\ &\forall x \notin L,\ M(x) = \text{“0”} \end{aligned} \tag{2.1}$$

Definition 2.2. Let $\mathcal{F} = \{f_n \mid n = 1, 2, \ldots\}$ be a family of functions s.t. $f_n : \Sigma^n \mapsto \Sigma^*$. We say that a TM M *computes* $\mathcal{F}$ if

$$\forall n \in \mathbb{N}, \forall x \in \Sigma^n, \quad M(x) = f_n(x) \tag{2.2}$$

2.1.2 Boolean Circuits

We briefly discussed Boolean circuits in Section 1.2.2.2 as an example of deterministic model. Formally, they are a special type of circuits defined in Definition 1.4.

Definition 2.3 (Boolean Circuit). A *Boolean circuit* is a circuit $C = (\mathcal{I}, \mathcal{O}, \mathcal{G}, \mathcal{W})$ where

1. To each input node $i \in \mathcal{I}$ is associated a Boolean value $x_i \in \Sigma$.

2. Each gate $g \in \mathcal{G}$ has an associated map $g : \Sigma^k \mapsto \Sigma^l$, where k and l is the number of input and output connections of g, respectively.

3. To each wire $w = (u, v) \in \mathcal{W}$ is associated a Boolean value by the formal function $\mathrm{val}(\cdot)$ defined recursively as follows:

$$\mathrm{val}(w) = \begin{cases} x_u & \text{if } u \in \mathcal{I} \\ \Big(g\big(\mathrm{val}(w_1), \ldots, \mathrm{val}(w_l)\big)\Big)_j & \text{if } u = (g, j) \text{ and } w_i = (u', (g, i)) \end{cases} \tag{2.3}$$

Given an input string $x = (x_1, x_2 \ldots x_{|\mathcal{I}|})$, we can define the output $C(x)$ of a Boolean circuit C as the concatenation of the values of wires going to output nodes, i.e.

$$C(x) = \big(\mathrm{val}(w_{i_1}), \ldots, \mathrm{val}(w_{i_{|\mathcal{O}|}})\big) \tag{2.4}$$

where w_{i_1} is the unique wire going to the first output node, w_{i_2} to the second one, and so on.

With this notion of circuit output, Boolean circuits can be given a semantics with which they can be used as computational devices, similar to that of Turing Machines.

Definition 2.4. A family of (single output) Boolean circuits $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ is said to decide a language L if for all inputs $x \in \Sigma^n$,

$$\begin{aligned} x \in L &\implies C_n(x) = (1) \\ x \notin L &\implies C_n(x) = (0) \end{aligned} \tag{2.5}$$

This can be generalised to functions $f : \Sigma^n \mapsto \Sigma^m$ for circuits with n input and m output nodes.

Definition 2.5. A family of Boolean circuits $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ is said to compute a family of functions $\mathcal{F}$ if for every input $x \in \Sigma^n$,

$$C_n(x) = f_n(x) \tag{2.6}$$

As is the case for TM's, the computational complexity of such functions (or that of deciding languages) can be expressed in terms of circuit resources. The complexity measures of relevance are:

Circuit Size. The total number of gates in the circuit, i.e. $|\mathcal{G}|$.

Circuit Depth. The maximum number of gates between any input and output wire, i.e. the diameter of the graph.

Circuit Width. The maximum width of any temporal cut of the circuit, as given in Definition 1.5.

Boolean Circuits are typically comprised of the logical gates `AND`, `OR` and `NOT`, or alternatively just `NAND` gates. This is because any arbitrary boolean gate can be replaced with such universal sets of gates.

Lemma 2.6. *An arbitrary Boolean gate g of k input nodes and l output nodes can be replaced with a circuit C of size $O(2^{k+l})$ containing only gates in the set* $\{\texttt{AND}, \texttt{OR}, \texttt{NOT}\}$*,* *or equivalently by a circuit C' of size $O(2^{k+l})$ containing only* NAND *gates, such that*

$$\forall x \in \Sigma^k, \quad g(x) = C(x) = C'(x) \tag{2.7}$$

2.1.3 Deterministic Complexity Classes

While Computability Theory studies which languages can be decided and which functions can be computed, the Theory of Complexity concerns itself with how efficiently this can be done, when it can be done. One of the interpretations or consequences of the Strong Church-Turing Thesis is that the only resources that are of concern, expressed in the Turing Machine formalism, are

1. the total time of computation, defined as the number of steps used by the TM to arrive to its halting state, and

2. the total space used by the TM, corresponding to the number of cells to the left of the tape head's rightmost position throughout the computation.

Based on these resources, complexity classes can be defined to regroup languages or functions according to the difficulty of deciding or computing them, respectively. For example, the deterministic complexity classes representing tractable problems are defined as follows.

Definition 2.7. A TM M is said to run in *polynomial time*, for short just said to be *poly-time*, if there exists an n_0 and a $k \geq 1$ such that for all input x of length $n \geq n_0$, M halts in less than n^k steps.

Definition 2.8. A language L is in the class P if there exists a poly-time TM that decides L.

A family function $\mathcal{F}$ is in the class FP if there exists a TM that computes $\mathcal{F}$.

While these classes have traditionally been described in the Turing Machine model, there is a natural relationship between the computational complexity in the circuit and TM models, expressed by the following theorems, which allows us to re-formulate these classes in the Boolean circuit model also.

Theorem 2.9. *A TM M running in time $t(n)$ and space $s(n)$, where $t(n) > n < s(n)$, can be simulated by a circuit family C_M with circuit depth $O(t(n))$ and width $O(s(n))$.*

Since for any TM we have that $s(n) \leq t(n)$, this theorem is most often quoted (e.g. in [Sip96]) as follows:

Corollary 2.10. *A TM running in time $t(n)$ can be simulated by a circuit family of circuit size $O(t^2(n))$.*

Proof. The idea of the proof of this well known theorem is to construct a circuit of width $O(s(n))$ whose input will consist of a binary description of the initial configuration of the TM. The circuit is layered so that each level of the circuit computes on the "output wires" of that layer the configuration of the TM after one step. Because the TM will not use at any time more than $s(n)$ cells of tape and because the finite automaton has a constant number of states, configurations can be represented by $O(s(n))$ boolean wire values, and each layer will involve at most $O(s(n))$ gates, with a layer depth bounded by a fixed constant (depending only on the alphabet size and the size of the automaton). Similarly, because the TM will halt within $t(n)$ steps, at most $t(n)$ such layers will be needed. □

The converse of this theorem is also well known. The idea behind it is based on a circuit-based version of the Universality Theorem for Turing Machines. Consider a suitable encoding scheme for describing Boolean circuits, where each description contains, for example, an ordered list of elementary gates, representing $\mathcal{G}$, together with an adjacency list of the circuit graph representing the gate wiring $\mathcal{W}$.

Theorem 2.11. *There exists a Turing Machine U, called a* Universal Circuit Evaluator, *which on input a binary string $x \in \Sigma^n$ and a description of a circuit C_n of n inputs*

1. *will output the correct value output by C_n on x. This is,*

$$\forall x \in \Sigma^n,\ U(x) = M([C_n, x])$$

2. *will run in time $O(s(n))$ and use space $O(w(n))$, where $s(n)$ and $w(n)$ are the size and width of C_n, respectively.*

Proof. In order to simulate the circuit, the Turing Machine first of all "serialises" the circuit by finding a topological sort of the circuit graph. The outcome of this topological sort is an ordering of the gates from the input to the output, with the property that the input wires of the i-th gate are all leaving from gates with ordinal less than i. This ordering thus defines a safe order in which the TM can evaluate each of the gates. At each step, the TM will need to keep track of at most $w(n)$ wires, and because each gate can be evaluated with constant steps and space, the overall simulation will take time and space $O(s(n))$ and $O(w(n))$. □

One significant difference between computational complexity expressed in terms of circuits and in terms of TM's is that in the former there is no well defined notion of *algorithm.* In a circuit family $\mathcal{C}$ solving a particular problem, it is principle possible for the circuits C_n to be radically different from each other for different input sizes. Furthermore, it is even possible that there exists no general prescription or algorithm for generating a circuit description C_n given only the input size n, or at least no efficient way to do so.

This gives rise to the notion of *non-uniform* complexity classes, which capture the fact that in order to decide languages in those classes one might benefit from a little bit of "extra help" for any given input size. These classes can be expressed in terms of circuits, or equivalently in terms of TM having access to special *advice strings.* The most common such class is the following:

Definition 2.12. A language L is in the non-uniform complexity class P/poly if there exists a family of advice strings $\mathcal{A} = \{A_n \mid n \in \mathbb{N}\}$ and a poly-time Turing Machine M such that for all n and all inputs $x \in \Sigma^n$,

$$\begin{aligned} x \in L &\implies M([x, A_n]) = 1 \\ x \notin L &\implies M([x, A_n]) = 0 \end{aligned} \tag{2.8}$$

The alternative approach to circuit-based complexity is to require the circuit family $\mathcal{C}$ to be "uniform" in the sense that their circuit description can be computed and generated by an algorithm, potentially with bounded resources. Typically, the circuit-generating algorithm is characterised by a resource-bounded TM which on input a number $n \in \mathbb{N}$, represented in unary, will generate a circuit description for C_n.

Definition 2.13. A Turing Machine M is said to uniformly generate a circuit family $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ if on input the unary representation of $n \in \mathbb{N}$, M outputs a circuit description of C_n.

Based on this definition, we can have the following alternate, circuit-based formulations of the deterministic classes from Definition 2.8.

Definition 2.14. A language L is in the class P if there exists a poly-time TM that uniformly generates a circuit family $\mathcal{C}$ that decides L.

A function family $\mathcal{F}$ is in the class FP if there is a poly-time TM that uniformly generates a circuit family $\mathcal{C}$ that computes $\mathcal{F}$.

Notice that we did not put any bounds on the resources of the circuits in $\mathcal{C}$, as these are automatically implied in the poly-time bound of the generating TM: a poly-time TM can generate a description of a circuit of at most polynomial size. Thus, by Corollary 2.10 and Theorem 2.11, it is immediate that these definitions of P and FP are equivalent to those of Definition 2.8.

2.2 Probabilistic Computation

2.2.1 Probabilistic, Coin-Flip and Randomised Turing Machines

In Section 1.3.3.2 we introduced Probabilistic Turing Machines as an example of a model in a probabilistic theory. Probabilistic TM's are initialised in the same way as deterministic ones. However, the notion of a PTM computing or deciding is a bit more elusive than for a DTM, but not too much. Because the state of the PTM after t steps is a probability distribution over configurations, one cannot determine with certainty whether the machine has halted at that point, i.e. whether the deterministic automaton has reached the single halting state q_f. Let $\mathcal{C}_{\text{H}} = \{c \in \mathcal{C} \mid c = [C_{\text{L}}, q_f, C_{\text{R}}]\}$ be the subset of halting configurations, then we can define the *probability* that the machine has halted after t steps as:

$$\Pr(\text{TM has halted}) = \sum_{c \in \mathcal{C}_{\text{H}}} P^{(t)}(c) \tag{2.9}$$

where $P^{(t)}(\cdot)$ is the probability function for configurations described in Equation 1.24. As defined in Equation 1.23, the probability transition function is such that once the PTM enters a halting state, it will remain in that state with probability 1. Thus, the probability of halting will increase monotonically as a function of the number of steps. However, this probability might never reach 1.

On the other hand, even if the PTM has halted with probability 1, its output might not be uniquely defined, as the different configurations $[C_{\text{L}}, q_f, C_{\text{R}}]$ tape in the sum of

Equation 2.9 might have different strings C_{L} to the left of the tape head. We can, however, assign to each of these possible answers C_{L} a probability of "being" the output according to the probability distribution $P^{(t)}(\cdot)$ from Equation 1.24. Interestingly, note that these probabilities of output at time t are well defined, even if the probability of halting at t is not 1, i.e. even if the PTM has not "completely halted" on all its computation paths. The output of a PTM M can thus be viewed as a random variable $\mathsf{M}^{(t)}(x)$, which represents the contents of the tape to the left of the tape head after t steps and having being initialised with $[q_0, x]$. With this notion of probability of answers it is possible to have a probabilistic equivalent of Definitions 2.1 and 2.2. [1]

Definition 2.15. Let L be a language in Σ^*. A PTM M is said to *decide* L in time $t(n)$, with *accuracy bounds* A and B, $A, B \subseteq [0,1]$, if for every input size $n \in \mathbb{N}$, and input $x \in \Sigma^n$, we have that $t = t(n)$ and

$$\begin{aligned} x \in L &\implies \Pr(\mathsf{M}^{(t)}(x) = \text{“1”}) \in A \\ x \notin L &\implies \Pr(\mathsf{M}^{(t)}(x) = \text{“0”}) \in B \end{aligned} \tag{2.10}$$

Definition 2.16. Let $\mathcal{F} = \{f_n \mid n \in \mathbb{N}\}$ be a family of functions s.t. $f_n : \Sigma^i \mapsto \Sigma^*$. We say that a PTM M *computes* $\mathcal{F}$ in time $t(n)$, with accuracy bound $A \subseteq [0,1]$, if

$$\forall n \in \mathbb{N},\, \forall x \in \Sigma^n, t = t(n), \quad \Pr\big(\mathsf{M}^{(t)}(x) = f_n(x)\big) \in A \tag{2.11}$$

The parametrisation using the sets A and B allows us to define uniform probabilistic classes within the same language. Their meaning is as follows. A represents the set of "acceptable" probabilities values (where the different values correspond to different inputs) for the event that the PTM outputs the correct answer when the input x *is* in the given language L. Similarly, B represents the set of "acceptable" probability values in the opposite case, i.e. when the input is *not* in the given language L. Note that for the standard definitions of probabilistic (and quantum) complexity classes, the sets A and B will be intervals, normally containing 1 (i.e. the possibility of being always right on a given input is not discarded).

[1] In principle, it would have been more natural to restrict the output random variable $\mathsf{M}^{(t)}(x)$ to only be defined on those computation paths where the PTM has halted after t steps, thus eliminating the "freak" occurrence of the right answer being on the left portion of the tape before the PTM has finished its computation. However, this distinction is not strictly necessary as for most cases that will interest us (i.e. complexity classes) we can always construct an equivalent PTM M' which will (almost) never have the right answer on the left of the tape until the end.

Traditionally, however, Turing Machines have been introduced in a less general fashion as we have introduced them here. TM's are after all essential "digital" objects, and it was only natural that the range of transition probabilities also be "discretised," in particular by restricting them to binary choices, each with equal probability 1/2, i.e. "coin flip."

Definition 2.17. A *coin-flip Turing Machine* (CFTM) is a probabilistic Turing Machine (Definition 1.14), with the added restriction that the probabilistic transition function π can only take values 0, 1 or 1/2, i.e. for all $q, q' \in Q$, $b, b' \in \Gamma$ and $d \in \{\mathtt{L}, \mathtt{R}\}$

$$\pi(q, b, q', b', d') \in \{0, \frac{1}{2}, 1\}$$

Because the coin-flip TM's are a special case of PTM's, our definition of language acceptance for PTM's (Definition 2.15) is still applicable to them as is.

Besides the fact that it is always "messy" to deal with arbitrary real numbers (i.e. transition probabilities), the reason, we suspect, that the original probabilistic Turing Machine model is that of coin-flip TM's is due to the fact that they are essentially equivalent to the more generic kind, in the sense given by the following theorem.

Theorem 2.18. *For every probabilistic TM M running in time $t = O(\text{poly}(n))$, there exists a coin-flip TM M' running in time $t' = O(\text{poly}(n))$, which has almost the same output statistics. More precisely, for all input sizes $n \in \mathbb{N}$, inputs $x \in \Sigma^n$, and possible answers $y \in \Sigma^*$, we have that:*

$$\left|\Pr(\mathsf{M}^{(t)}(x) = y) - \Pr(\mathsf{M}'^{(t')}(x) = y)\right| < \frac{1}{2^n} \tag{2.12}$$

Proof. The general idea is that the CFTM M' will simulate the PTM M by making the same choices and trying to jump to the same type of configuration with equal or almost equal probabilities. Whenever M does a deterministic transition (i.e. $\pi(\cdot) = 0$ or 1), then M' can immitate M without introducing error in the final output statistics.

In general, however, this will not be the case. Let then $p_1, p_2, \ldots, p_m$ be the probability values for the next transition. The CFTM can simulate sampling according to that distribution by performing d "coin-flips" and choosing the first possibility with probability $\lfloor p_1 2^d \rfloor / 2^d$, the second with probability $\lfloor p_2 2^d \rfloor / 2^d$, etc. A relatively simple probabilistic argument will show that the probabilities of outcomes will not differ significantly if d is large enough, but more importantly that these differences will decrease exponentially in d.

Let us choose d to be polynomial, e.g. $d = n^a$, then the error statistics introduced with one instance will be less than $1/2^{n^a}$. Let M run in $t(n) = n^b$ steps. Then, we have that the probability that the CFTM M' commits an error in any of these transitions is bounded by

$$\begin{aligned}\Pr(M' \text{ chooses wrong transition at least once}) &\leq 1 - \left(1 - \frac{1}{2^{n^a}}\right)^{n^b} \\ &< 1 - \frac{1}{2^n}\end{aligned}$$

where we make the last inequality hold (for sufficiently large n) by choosing a sufficiently large d, i.e. $a = b$ for $b > 1$ or $a = 2$ otherwise. In either case, the total running time of M' will be at most n^{2b} and thus still polynomial. □

An apparent paradox of Theorem 2.18 is the following. If we construct a PTM with arbitrary transition probabilities, including for example uncomputable real numbers (e.g. Chaitin's Ω), then such a machine could solve uncomputable problems such as the Halting Problem. By applying the above theorem, we could thus also build a coin-flip TM that solves the same problems with arbitrary high probability. This seems unreasonable for such a simplified model. The catch is that the corresponding transition probabilities, while restricted to powers of 2, are still uncomputable. Thus the resulting coin-flip TM would not have a computable description interpretable by a Universal Turing Machine. Because of this fact, they are in some sense "unimplementable," which is why they are not usually discarded, not being a reasonable model of computation. In the forthcoming, we only consider TM's with computable transition probabilities, which thus assures that they can all be interpreted/simulated by the same Universal Turing Machine.

In chapter 1, PTM's were introduced as an instance of probabilistic models defined in Definition 1.8. Under that view, the states of a PTM are not its configurations, but the PD's over them. In other words, the set of configurations $\mathcal{C}$ is not the state space but the sample space of the model. This defines a deterministic dynamics (Equation 1.25) on the state space $\mathrm{PD}(\mathcal{C})$, and the probabilistic "computation rules" of Definitions 2.15 and 2.16, which are naturally obtained as specialisation of the abstract measurement rule of probabilistic models.

This view is a bit artificial and uncommon in the context of computing[2], which is why

[2] Translation: it is quite unnatural to computer scientists...

PTM's are more often described with $\mathcal{C}$ as the state space and a probabilistic dynamics based on a transition probability function such as that of Equation 1.23. In appearance, identifying states with configurations seems to yield a deterministic measurement rule, and hence deterministic computation rules such as those of Definitions 2.1 and 2.2. However, as we discussed in Section 1.3.1, the measurement rule is not completely deterministic, as we cannot make deterministic predictions about the outcome of measurement in the future. This situation is depicted in Figure 2.1.

$$\begin{tikzcd}
\vec{c}_t \arrow[r, "P^{(1)}"] \arrow[d, "\{\vec{c}_t\}_i"', "\circ" description] \arrow[dr, "\circ" description] & \vec{c}_{t+1} \arrow[d, "\{\vec{c}_{t+1}\}_j", "\circ" description] \\
\text{Left}(c_i) & \text{Left}(c_j)
\end{tikzcd}$$

(a) States as PD's

$$\begin{tikzcd}
c_t \arrow[r, "P(c_t, c_{t+1})", "\circ" description] \arrow[d] \arrow[dr, "\circ" description] & c_{t+1} \arrow[d] \\
\text{Left}(c_t) & \text{Left}(c_{t+1})
\end{tikzcd}$$

(b) States as configurations

Figure 2.1: (a) In the PD-state based representation of PTM's, the "output" at time t (the left portion of the tape) is assigned a probability according to the PD of configurations at that moment $\vec{c}_t$. The one-step deterministic dynamics on PD($\mathcal{C}$) defined by $P^{(1)}$ can be used to make a probabilistic prediction for (i.e. assign a probability to) the output at the next step. (b) If the space of configurations $\mathcal{C}$ is viewed as the state space of the model, then the measurement rule can be represented by the deterministic function Left($\cdot$) defined on $\mathcal{C}$, which returns the string on left portion of the tape. The dynamics is probabilistic and the configuration c_t at time t will transition to the next configuration c_{t+1} according to the transition probability prescribed by $P(c_t, c_{t+1})$. A deterministic prediction for the output at the next step is not possible, but a probabilistic one is.

Both of these representations are equally valid. Nonetheless, the configuration-based representation allows for an alternate interpretation which is less awkward and also often used to introduce probabilistic computation: a deterministic TM which has access to a a special "random" tape and computes as follows.

Definition 2.19 (Randomised Turing Machine). A *randomised Turing Machine* (RTM) M is a special kind of DTM that has an extra read-only, read-once tape called the *random input* tape. The transition function of M takes the form

$$\delta \,:\, Q - \{q_f\} \times \Gamma \times \Sigma \longmapsto Q \times \Gamma \times \{\text{L}, \text{R}\}\,. \tag{2.13}$$

The image $\delta(q, b, r) = (q', b', d)$ indicates that when in state q and having read symbols b

and r under its regular tape and random tape heads respectively, M will write symbol b' on the regular tape, move the regular head in the direction of d, move the random tape head to the right (always) and enter state q'.

A randomised TM is initialised similarly as a DTM except that its random tape is assumed to be infinite and contains no blank symbols. Their output is also defined as the left portion of the regular tape when the machine has finally halted. The output, however, will not only depend on the input x on the regular tape, but also on the portion of its random tape that was examined; it is represented by the function $M(x, r)$ where r represents the examined portion of the random tape.

Strictly speaking, randomised TM's are a deterministic model and we cannot speak of "probabilities" of an RTM returning a value "$M(x)$" as we do for probabilistic TM's (generic or coin-flip). Instead, we consider the *frequency counts* of an RTM returning an output, where we count the number of times that machine will return a given output as we run through all possible random tape inputs.

Definition 2.20. The *frequency count* function of a randomised TM M running in time $t = t(n)$ is the function $\nu_M : \Sigma^n \times \Sigma^* \mapsto [0, 1]$ defined as

$$\nu_M(x, y) \triangleq \frac{|\{r \in \Sigma^t \mid M(x, r) = y\}|}{|\{r \in \Sigma^t\}|} \tag{2.14}$$

With this notion defined, the criteria for randomised TM's to decide or to compute are defined as follows.

Definition 2.21. Let L be a language in Σ^*. An RTM M is said to *decide* L, with accuracy bounds A and B, $A, B \subseteq [0, 1]$, if

$$\begin{aligned} &\forall x \in L, \nu_M(x, \text{“0”}) \in A \\ &\forall x \notin L, \nu_M(x, \text{“1”}) \in B \end{aligned} \tag{2.15}$$

Definition 2.22. Let $\mathcal{F} = \{f_n \mid n \in \mathbb{N}\}$ be a family of functions s.t. $f_n : \Sigma^i \mapsto \Sigma^*$. We say that a PTM M *computes* $\mathcal{F}$, with accuracy bound $D \subseteq [0, 1]$, if

$$\forall n \in \mathbb{N}, \forall x \in \Sigma^n, \quad \nu_M\big(x, f(x)\big) \in D \tag{2.16}$$

This model of computation is nothing more than a hidden variable interpretation (Definition 1.12) for probabilistic TM's. The contents of the random tape are indeed the hidden variables of this model. The hidden state space is the state of configurations $[C_{\mathrm{L}}, q, C_{\mathrm{R}}][R_{\mathrm{L}}, q]$, where R_{L} represents the contents of the random tape to the left of that tape head.

These randomised TM's in fact constitute a "derandomisation" of PTM's in the sense that probabilities have been substituted with the relative frequency counts. Indeed, assigning the same weight to each string r of length t in the random tape is equivalent to assigning a uniform *a priori* distribution on the hidden portion of the state space, and in that sense these frequency counts can be viewed as "probabilities." In other words, any randomised TM can be so transformed into a PTM, and furthermore since the uniform distribution is the Cartesian product of the $\{1/2, 1/2\}$ distribution on each of the t cells of the random tape, we obtain the following equivalence.

Theorem 2.23. *Any randomised RTM M' can be simulated by a CFTM M' with the exact same output statistics and the same running time.* □

The converse statement is also true, and is in fact a computational (and much simpler) equivalent to the proof that probabilistic models having well defined transition probabilities had hidden variable interpretations (Theorem 1.18).

Theorem 2.24. *Any CFTM M can be simulated by an RTM M' with the same exact output statistics and running time.*

Proof. M' mimics M exactly in all of its deterministic transitions. In addition, every time M makes a coin-flip transition, M' "chooses" which is going to be the next configuration by considering the value of the random tape cell under its head. □

2.2.2 Probabilistic, Coin-Flip and Randomised Circuits

Very much like probabilistic Turing Machines, probabilistic circuits can be defined and described in a variety of ways. In Section 1.3.3.3, we introduced generic probabilistic circuits using the abstract circuit model of Definition 1.4 and generic probabilistic gates.

Let $n = |\mathcal{I}|$ be the number of inputs nodes in C, which have been assigned ordered labels $[1..m]$ (e.g. from "top" to "bottom"), and let $t_0 = \min(\mathcal{T}_C)$ be the initial moment of C (*vide supra* our discussion of "time" in abstract circuits in Section 1.3.3.3). Given a deterministic input $x = x_1 \dots x_n$, we usually initialise the circuit with a "deterministic" initial state, $\vec{p}_0$, where $(\vec{p}_0)_j = 1$, iff x is the j-th string in Σ^n in lexicographical order.

Initialised in such a way, the output of a probabilistic circuit C is naturally defined by the final state $\vec{p}_f$ at the final moment $t_f = \max(\mathcal{T}_C)$ of the circuit. Since $\vec{p}_f$ will depend on $\vec{p}_0$, which in turns depends on x, we can identify the (probabilistic) output of C with a random variable $\mathsf{C}(x)$ depending on x, with domain on Σ^m (where $m = |\,\mathcal{O}\,|$ is the number of output nodes of C), which is distributed according to the PD $\vec{p}_f$. With this notion of probabilistic output, we can give probabilistic circuits (generic or coin-flip) a sound semantics as computational devices.

Definition 2.25. A family of probabilistic circuits $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ is said to decide a language L, with accuracy bounds A and B, $A, B \subseteq [0, 1]$, if for every input size $n \in \mathbb{N}$, and input $x \in \Sigma^n$,

$$\begin{aligned} x \in L &\implies \Pr(C_n(x) = 1) \in A \\ x \notin L &\implies \Pr(C_n(x) = 0) \in B \end{aligned} \tag{2.17}$$

Definition 2.26. A family of probabilistic circuits $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ is said to compute a family of functions $\mathcal{F} = \{f_n \,|\, n \in \mathbb{N}\}$, with accuracy bound $D \subseteq [0, 1]$, if for every input $x \in \Sigma^n$,

$$\Pr(\, C_n(x) = f_n(x)\,) \in D \tag{2.18}$$

To the best of our knowledge, non-uniform probabilistic classes have not been defined or studied so far, and we will not concern ourselves with them. On the other hand, uniform probabilistic classes have been well studied and we will review them in Section 2.2.4. In our model of generalised probabilistic circuits, we advance the following notion of uniformity.

Definition 2.27. A family of probabilistic circuits $\mathcal{C}$ is said to be uniformly generated by a (deterministic) Turing Machine M if

i. There is a finite set of types of probabilistic gates which represents all gates in all circuits in $\mathcal{C}$.

ii. On input 1^n, M will output a description of the graph of C_n and a table of finite encodings of the transition probabilities for each gate g in C_n.

An important consequence of this somewhat restrictive model[3] in terms of complexity is that when we consider families of probabilistic circuits with n input nodes, the number

[3] The most general model we can think of is one where the set of elementary gates is not finite, but

of input and output nodes for each of the elementary probabilistic gates, whichever they may be, can be considered constants which do not vary with n.

Note also that any of the usual deterministic gates (AND, OR, CNOT, etc.) can be viewed as a special case of probabilistic gates with a $\{0,1\}$-valued transition probabilities.

The second and more traditional way to introduce probabilistic circuits is to define them in term of *coin-flip* gates. The coin-flip gate is the simplest possible probabilistic gate, which irrespective of its input will produce a 0 or a 1 with equal probability on its single output connection.

Definition 2.28 (Coin-flip Probabilistic Circuit). A *coin-flip* circuit C is a probabilistic circuit whose only probabilistic circuit is the single-input/single-output probabilistic gate CF, with transition probabilities

$$\pi_{\mathtt{CF}}(\,\cdot\,,0) = \pi_{\mathtt{CF}}(\,\cdot\,,1) = 1/2 \tag{2.19}$$

The following lemma addresses the question of the significance of this restriction, and tells us that under most reasonable circumstances it is sufficient to consider only coin-flip gates, as we will see later in Section 2.2.4.

Lemma 2.29. *Any probabilistic gate g of k inputs and l outputs can be simulated by a coin-flip probabilistic circuit C of size $O(2^{k+l}\log 1/\epsilon)$, such that the output statistics of g and C differ by at most ϵ. More precisely, for all inputs $x = (x_1, \ldots, x_k) \in \Sigma^k$ and possible outputs $y = (y_1, \ldots, y_l) \in \Sigma^l$, we have that*

$$\left|\pi_g(x,y) - \Pr(\mathrm{C}(x) = y)\right| < \epsilon \tag{2.20}$$

As we did in Section 2.2.1 with the introduction of randomised TM's, it is possible to derandomise probabilistic circuits with the introduction of a hidden variable interpretation.

Definition 2.30 (Randomised Circuit). A *randomised circuit* is a special kind of deterministic Boolean circuit represented as $C = (\mathcal{I}, \mathcal{O}, \mathcal{G}, \mathcal{W})$, where the set of input nodes $\mathcal{I} = \mathcal{I}_x \cup \mathcal{I}_r$ has two separate components:

- $\mathcal{I}_x$ the input nodes *per se*, which are initialised with the input values $x_1, \ldots, x_n$.

involves only computable and finitely encodable transition probabilities. We choose the above model for simplicity, as we do not believe the complexity classes we will be studying will be affected by the restrictions introduced in Definition 2.27.

- $\mathcal{I}_r$ the random input nodes, with are initialised with "random" value $r_1, \ldots, r_l$.

The computation semantics of randomised circuits are based on the relative frequency counts of random inputs which cause the circuit to produce the right answer, i.e.

Definition 2.31. The *frequency count* function of a randomised circuit C with n inputs nodes, ℓ random input nodes and m output nodes, is the function $\nu_C : \Sigma^n \times \Sigma^* \mapsto [0,1]$ defined as

$$\nu_C(x,y) \triangleq \begin{cases} \dfrac{|\{r \in \Sigma^\ell \mid C(x,r) = y\}|}{|\{r \in \Sigma^\ell\}|} & \text{, if } y \in \Sigma^m \\ 0 & \text{, otherwise} \end{cases} \tag{2.21}$$

Definition 2.32. A family of randomised circuits $\mathcal{C} = \{C_n | n \in \mathbb{N}\}$ using $\ell = \ell(n)$ random inputs is said to decide a language L, with accuracy bounds A and B, $A, B \subseteq [0,1]$, if for every input size $n \in \mathbb{N}$, and input $x \in \Sigma^n$,

$$\begin{aligned} x \in L &\implies \nu_{C_n}(x,1) \in A \\ x \notin L &\implies \nu_{C_n}(x,0) \in B \end{aligned} \tag{2.22}$$

Definition 2.33. A family of probabilistic circuits $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ using $\ell = \ell(n)$ random inputs is said to compute a family of functions $\mathcal{F} = \{f_n \mid n \in \mathbb{N}\}$ with accuracy bound $D \subseteq [0,1]$ if for every input $x \in \Sigma^n$,

$$\nu_{C_n}(x, f_n(x)) \in D \tag{2.23}$$

The idea behind viewing such circuits as hidden variable interpretations of probabilistic circuits is the following. Whenever a probabilistic gate g in the original circuit would make a probabilistic choice on one of the possible output vectors, the randomised circuit simulates that gate by choosing one of these vectors by looking at some of the random inputs. This is made precise by the following theorem.

Theorem 2.34. *For every family of probabilistic, poly-size circuits $\mathcal{C} = \{C_n\}$ there exists a family of poly-size randomised circuits $\mathcal{C}' = \{C'_n\}$ using at most $l = n^c$ random inputs that will have almost the same output statistics. More precisely for all inputs $x \in \Sigma^n$, and possible answers $y \in \Sigma^m$, we have that:*

$$\left|\Pr(\mathsf{C}_n(x) = y) - \nu_{C_n}(x,y)\right| < \frac{1}{2^n} \tag{2.24}$$

Proof. Consider first the special case where the only probabilistic gates in the circuits C_n are coin-flip gates. Then, the gates can be "simulated" by simply using (only once) one of the random inputs as the output of a coin flip gate. If we assume that each random input r_i is equidistributed to be 0 or 1, we will then obtain in Eqns. (2.22) and (2.23) the exact same output statistics as those of the original circuit, i.e. those from Eqns. (2.17) and (2.18). Conversely, a probabilistic circuit not having access to equidistributed random inputs, can generate them by simply using ancillary inputs and pre-processing them with a coin-flip gate.

In the general case, however, the circuit in $\mathcal{C}$ will use a finite set of elementary probabilistic gates $\{g_1, \ldots, g_c\}$, each with a constant number of inputs and outputs. By Lemma 2.29, each of these gates can be approximated within ϵ by circuits of size $O(\log 1/\epsilon)$. To simulate C_n, we assemble a coin-flip probabilistic circuit C'_n which simulates each elementary probabilistic gate g in C_n with error bound $\epsilon < 1/2^n$. Each sub-circuit for the simulation of each elementary gate g will involve at most $O(n)$ coin-flip and deterministic gates and thus C'_n will still have poly-size. We have thus reduced the general case to the special case above. □

On the other hand, the relationship between coin-flip and randomised circuits is obvious.

Theorem 2.35. *Any coin-flip circuit C of n inputs, of size s and containing m CF gates, can be simulated exactly by a randomised circuit C' of size $s - m$ with n input nodes and m random input nodes, where m is the number of CF gates in C.*

Conversely, any randomised circuit C' with n inputs and m random inputs, of size s', can be simulated exactly by a coin-flip circuit C of size $s' + m$.

In both cases we have that for all inputs $x \in \Sigma^n$ and possible outputs $y \in \Sigma*$

$$\Pr(\mathsf{C}(x) = y) = \upsilon_C(x, y) \tag{2.25}$$

Proof. To simulate the outcome of each CF gate, C' uses up one random input node. Conversely, to simulate a random input, C simply uses a single CF gate. The rest of the proof is simple accounting. □

As a result of this equivalence and Theorem 2.34 we have the following equivalence (with approximation) between probabilistic and coin-flip circuits, which in fact shows the following very important universality of the coin-flip gate CF for probabilistic circuits.

Theorem 2.36. *For every family of poly-size probabilistic circuits $\mathcal{C} = \{C_n\}$ using an arbitrary set of probabilistic gates (described by stochastic matrices), there exist a family of poly-size coin-flip circuits $\mathcal{C}' = \{C'_n\}$ such that for all inputs $x \in \Sigma^n$ and outputs in $y \in \Sigma^*$*

$$\Big|\Pr\big(\mathsf{C}_n(x) = y\big) - \Pr\big(\mathsf{C}'_n(x) = y\big)\Big| < \frac{1}{2^n} \tag{2.26}$$

□

2.2.3 Equivalence of Probabilistic Models of Computation

So far, Theorems 2.18 and 2.34, for TM's and circuits respectively, show us that we can go from probabilistic to randomised models at the cost of introducing some differences in the output statistics, albeit quite small. There is also a "lateral" equivalence relationship between randomised TM's and randomised circuits. As before, this relationship is qualified by the fact that we are talking about poly-time TM's and poly-size families of circuits.

Theorem 2.37. *For every randomised TM M running in time n^c there exists a poly-sized family of randomised circuits $C_n(x, r)$ using $\ell = \ell(n) \leq n^c$ random input nodes with the same exact output statistics.*

Conversely, for every family of randomised circuits $\mathcal{C} = \{C_n\}$, using $\ell = \ell(n)$ random inputs, and *that is uniformly generated in time n^d, there exists a randomised TM M running in time $O(n^d)$ with the same output statistics.*

More precisely, we have that for both cases and all input sizes $n \in \mathbb{N}$, inputs $x \in \Sigma^n$ and possible answers $y \in \Sigma^*$

$$\nu_M(x, y) = \nu_{C_n}(x, y) \tag{2.27}$$

Proof. The proof of this theorem is based on a simple adaptation of the deterministic equivalences for TM and circuits described in Theorems 2.9 and 2.11. Structurally, both M and the circuits in $\mathcal{C}$ are deterministic and the proof of these theorems carry with minor modifications.

In order to simulate M, construct a deterministic circuit based on its transition table, whose random inputs will correspond to the random tape cell values. Since M examines at most n^c such cells, the simulating circuit will have at most a total of $n + n^c$ input nodes (including random inputs), and consequently will have at most polynomial size.

Conversely, let M_C be the poly-time TM generating the circuit $C_n \in C$ on input the string 1^n. We construct the machine M which on input x will:

1. Run M_C to obtain a description of C_n
2. Run the Universal Circuit Evaluator TM on $C_n(x, r)$, where r is a binary string of size l constructed by reading the first ℓ random tape input cells.
3. Output the value $C_n(x, r)$.

By definition M_C will run in time n^d, and by Theorem 2.11 the Circuit Evaluator will run in time $O(n^d)$ because C_n is at most of size n^d. Therefore the whole simulation is in $O(n^d)$. □

By using this last theorem and the equivalences RTM vs. CFTM and randomised vs. coin-flip circuits (Theorems 2.23, 2.24 and 2.35), we directly obtain the following lateral equivalence between coin-flip models.

Corollary 2.38. *For every coin-flip TM M running in time $t = O(\text{poly}(n))$ there exists a poly-sized family of coin-flip circuits $C_n(x, r)$ using $\ell = \ell(n) = O(n^c)$ random input nodes with the same exact output statistics.*

Conversely, for every family of randomised circuits $C = \{C_n\}$, using $\ell = \ell(n)$ random inputs, and *which is uniformly generated in poly-time, there exists a randomised TM M running in time $t = O(\text{poly}(n))$ with the same output statistics.*

More precisely, we have that for both cases and all input sizes $n \in \mathbb{N}$, inputs $x \in \Sigma^n$ and possible answers $y \in \Sigma^*$

$$\Pr(\mathsf{M}(x) = y) = \Pr(\mathsf{C}_n(x) = y) \tag{2.28}$$

□

Furthermore, by combining this last result with the equivalence (with approximation) of PTM's and CFTM's (Theorem 2.18), and that between probabilistic and randomised circuits (Theorem 2.34), we have the following equivalence (with approximation) between probabilistic TM's and circuits.

Corollary 2.39. *For every probabilistic TM M running in time $t = O(\text{poly}(n))$ there exists a poly-sized family of probabilistic circuits $C_n(x, r)$ using $\ell = \ell(n) = O(n^c)$ random input that approximates M with exponentially small error.*

Conversely, for every family of randomised circuits $C = \{C_n\}$, using $\ell = \ell(n)$ random inputs, and *uniformly generated in poly-time, there exists a probabilistic TM M running in time $t = O(\text{poly}(n))$ which approximates C with exponentially small error.*

More precisely, in both cases and all input sizes $n \in \mathbb{N}$, inputs $x \in \Sigma^n$ and possible answers $y \in \Sigma^*$ we have that

$$\left|\Pr\left(\mathsf{M}^{(t)}(x) = y\right) - \Pr\left(\mathsf{C}_n(x) = y\right)\right| < \frac{1}{2^n} \tag{2.29}$$

□

2.2.4 Probabilistic Complexity Classes

"Luck is not a factor!"
- Klingon Proverb

The various definitions of computation for probabilistic models introduced in Section 2.2 involve accuracy bounds, which define, in essence, how lucky we have to be in order to obtain a correct answer. This "luck" will either influence the choices made by a probabilistic gate or automaton, or will influence the choice of random bits in a randomised TM or circuit.

The discovery in 1976 [Rab76, SS77] that probabilistic algorithms existed for the problem of primality testing prompted a revision of the Strong Church-Turing thesis. Here was an example of a natural problem which could apparently not be solved deterministically in polynomial time [4], but for which there existed a probabilistic poly-time algorithm which in addition had a reasonable chance of success with every input. Accordingly, the notion of computational tractability was revised to include such probabilistic algorithms, *as long as the probability of success was sufficiently high*, or in other words as long as we do not have to rely on a lot of good luck to obtain the correct answer.

This relaxation of the Strong Church-Turing is reasonable in practice for several reasons. First, randomness exists in nature and in principle it should not be technologically too hard to harness it and bring it in to our computational devices. In particular, the randomised models above have a built-in mechanism for doing so: the random inputs. Secondly, if the probability of success is sufficiently large, it is possible to *amplify* it

[4] It was only relatively recently (2002) that it was shown that, indeed, there are deterministic poly-time algorithms for primality testing [AKS02].

significantly by doing a reasonable amount of repetitions of the same computation. It is thus possible, even within the the context of polynomially bounded resources, to obtain a probability of success which is as close to 1 as we desire, even though never *exactly* 1.

These notions are made formal by the introduction of probabilistic complexity classes.

Definition 2.40. A language L is in the class BPP if there exists a $k > 0$ and a probabilistic poly-time TM M that decides L with accuracy bounds $A = B = [1/2 + 1/n^k, 1]$.

The Chernoff bound is a technical result in Probability Theory which bounds the variance of the majority or sum of several independent instances of the same random variable. It can be used to show that if we repeat the execution of M a polynomial number of times and make the final decision by considering the most frequent answer (accept or reject), then the resulting error probability will be bound by a constant, independent of n. Thus, we can have an alternate characterisation of BPP by replacing the accuracy bounds in Definition 2.40 with $A = B = [2/3, 1]$ or even with $A = B = [1 - 1/2^k, 1]$, for any constant $k > 0$.

Suppose that a probabilistic algorithm had the ability to detect with certainty membership in a language, but not the opposite. This is the case for the language COMPOSITE and Rabin's probabilistic algorithm for primality testing. If $x \in$ COMPOSITE, i.e. if x is a composite integer, then the algorithm will produce with high probability a verifiable proof or "certificate" that x has factors (but not the factors themselves!) and accept x. On the other hand, if x is prime then no factors can be found and the probabilistic primality testing algorithm will "correctly" reject x in all cases. This kind of one-sided error algorithms inspires the class RP which can be defined as follows.

Definition 2.41. A language L is in the class RP if there exists a probabilistic poly-time TM M that decides L with accuracy bounds $A = [1/2, 1]$ and $B = \{1\}$.

It is precisely this kind of symmetry breaking that motivates the fact that the definitions of Sections 2.2.1 and 2.2.2 of BPP have two accuracy bounds A and B, one for strings within the language and one for those without. When they are both equal, or both similarly bounded, it is more natural and usual to talk of the *probability of error* of the PTM.

This one-sidedness is not completely unusual, however, and it is an important structural property of certain complexity classes. In particular, the non-deterministic class NP can be viewed as an instance of a probabilistic class which shows this characteristic.

Theorem 2.42. *A language is in the class* NP *iff there exists a poly-time PTM M that decides L with accuracy bounds $A = (0, 1]$ and $B = [1]$.*

Proof. This proof is most easily described in terms of a randomised TM, which is without loss generality as we showed in Section 2.2.1. Let M be the RTM accepting L with $A = (0, 1]$ and $B = [1]$. Then M can be viewed as a non-deterministic machine if we consider the random tape as a witness tape. Since $A = (0, 1]$ there must exist at least one random tape input r s.t. M will accept an x in L, and because $B = [1]$ no $x \notin L$ will be incorrectly accepted. Conversely, a non-deterministic TM M' with a witness tape accepting L can be thought of as a randomised TM, by considering the witness tape as a random input. The existence of a witness guarantees that at $0 \notin A$ and the infallibility of M' on inputs outside of L guarantees that no "random" input will make it yield the wrong answer in this case, and thus $B = \{1\}$. □

Consider the following "dummy" probabilistic algorithm:

1. On input x, flip a fair coin (i.e. look at k random tape cells, perform k coin-flip gates, etc.)
2. If the outcome is "tails" then accept x, otherwise reject x.

Obviously, this oblivious "algorithm" has, for all input x, a positive probability of being right. In fact, half of the time it will give the right answer...

We would expect any "useful" probabilistic algorithm to do at least better than this yardstick algorithm. The class of languages which can be solved by such probabilistic algorithms, for which we only require that they do strictly better than just choosing the answer at random (even if admitedly not very much better) is captured by the following definition.

Definition 2.43. A language L is in the class PP if there exists a probabilistic poly-time TM M that decides L with accuracy bounds $A = B = (1/2, 1]$.

An important difference with the class BPP is that the accuracy cannot be amplified efficiently. Indeed, the accuracy is bounded away from 1/2, but could be so only by very small, even exponentially small amount, e.g. $a = 1/2 + 1/2^n$. One cannot amplify this exponentially small advantage to any desired constant by using common techniques based on Chernoff's bound without an exponential number of repetitions.

This is in part why we informally call PP a "lucky" class. Even though the "amount of luck" required to get the correct answer is strictly less than that needed by the coin flipping "algorithm," a disproportionate amount of luck is still necessary. For example if we were to use the majority method, in order to obtain significant confidence in the correctness of the final answer one might need an exponential number of trials of the algorithm. Hence, problems in PP are not considered to be tractable.

Another important characteristic of the class PP is that the it can be defined with any fixed cut point other than 1/2. This is, we can substitute the condition in Definition 2.43 with $A = (\gamma, 1)$ and $B = (1 - \gamma, 1]$, for any constant γ in the open interval $(0, 1)$.

2.3 Quantum Computation

2.3.1 Quantum Turing Machines

We briefly discussed Quantum Turing Machines in Section 1.5.2.3, as a generalisation of probabilistic TM's. The main disadvantages of QTM's concerns halting.

In a deterministic TM given a fixed input x there is no ambiguity about when the machine halts (if it halts a all) and about its output then. In a probabilistic TM, we resolved that ambiguity by adopting the convention that once a PTM arrives to the (unique) halting configuration, it then transitions with probability 1 to that same configuration (Equation 1.23). With this convention, the probability of halting increases monotonically with the number of steps, and thus we can sensibly define a probability of the PTM outputting any given answer at any given time. In particular, we can speak of the probability that a PTM has accepted input x at a given finite time t even if some paths in the computation tree have not terminated. Thus, we can give proper computation semantics to PTM's (Definitions 2.15 and 2.16).

Unfortunately the same technique cannot be applied verbatim to QTM's, *vide supra*, because the trick above is inherently non-reversible, and thus the resulting dynamics is not unitary.

For this reason, we normally only consider QTM's with a 2-way infinite tape, which are *well behaved* in that they always halt on the same number of steps $t(n)$ for all inputs of the same size. Thus, at the final halting time t, we have that the QTM is an a superposition of configurations $\sum_i \alpha_i[u_i, q_f, v_i]$, with the output defined in terms of u_i, the string of symbols from the leftmost non-blank symbol to the position tape head. The probability of the QTM outputting u_i is defined in terms of the corresponding amplitude as $|\alpha_i|^2$. Thus, we can have a well defined random variable $\mathsf{M}(x)$ for a QTM M, which represents the output statistics of M on input x after $t(|x|)$ steps of computation. With this variable, we can give QTM's the same computation semantics as PTM's.

Definition 2.44. Let L be a language in Σ^*. A QTM M is said to *decide* L in time $t(n)$, with accuracy bounds A and B, $A, B \subseteq [0,1]$, if for every input size $n \in \mathbb{N}$, and input $x \in \Sigma^n$, we have that M always halts in exactly $t = t(n)$ steps and

$$\begin{aligned} x \in L &\implies \Pr(\mathsf{M}(x) = \text{“1”}) \in A \\ x \notin L &\implies \Pr(\mathsf{M}(x) = \text{“0”}) \in B \end{aligned} \tag{2.30}$$

Definition 2.45. Let $\mathcal{F} = \{f_n \mid n \in \mathbb{N}\}$ be a family of functions s.t. $f_n : \Sigma^n \mapsto \Sigma^*$. We say that a QTM M *computes* $\mathcal{F}$, with accuracy bound $D \subseteq [0,1]$, if for all input sizes $n \in \mathbb{N}$, and all inputs $x \in \Sigma^n$, M halts in $t(n)$ steps and

$$\Pr\big(\mathsf{M}(x) = f_n(x)\big) \in D \tag{2.31}$$

It is important to note that for polynomial time QTM's the "well-behaved" requirement is not unduly restrictive. Informally, a "naughty" poly-time QTM can in principle be simulated by a poly-time well behaved one[5]. Secondly, the requirement that the QTM uses a 2-way infinite tape does not affect the power of computation for poly-time machines either, as can be shown by straightforward adaptation to the quantum case of these facts for deterministic TM's.

It is also worthy to note two very important differences between PTM's and QTM's. First of all, there is no single "universal" quantum transition, like the coin-flip transition for PTM's. Thus we cannot easily define "coin-flip" QTM's or some similar simplified model. However, Adleman, DeMarrais, and Huang [ADH97] have shown that a universal QTM exists which uses only transition amplitudes in $\{0, \pm 3/5, \pm 4/5, 1\}$.

More importantly, we cannot derandomise quantum Turing Machines like we did for PTM's. This is, we cannot view QTM's as regular deterministic machines having

[5] We say informally, because the notion of simulating a naughty QTM is itself quite slippery...

access to a "quantum" tape. Indeed, even if the quantum tape cells where in some non-factorisable superposition, a randomised TM accessing tape cell-by-cell would necessarily "destroy" any non-locality between the tape cells. Thus, we can view this impossibility as a computational consequence of John Bell's celebrated theorem which states that there are quantum models for which no *local* hidden variable interpretation exists.

2.3.2 Quantum Circuits

We have already defined and briefly discussed quantum circuits in Section 1.5.2.4. They were introduced by Deutsch [Deu89] and studied in detail later by Yao [Yao93], who showed that poly-size quantum circuits can suitably approximate QTM's. Yao's result provide the quantum version of the "lateral" equivalence (up to approximation) between poly-time probabilistic TM's and poly-time probabilistic circuits (Corollary 2.39).

Furthermore, several universality results (which we will review in more detail in Section 3.1.2) exist for quantum circuits. In its most recent and simplified version [Shi02, Aha03], we have that any unitary transformation can be suitably approximated with a circuit consisting only of `TOFFOLI` and `HADAMARD` gates. Because the `TOFFOLI` gate is universal for reversible deterministic circuits, we can view these circuits as the quantum equivalent of the coin-flip probabilistic circuit.

Definition 2.46. A quantum T-F circuit is one which is exclusively comprised of the `TOFFOLI` and `HADAMARD` gates, represented with the following unitary matrices

$$\mathsf{TOFFOLI} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \quad ; \quad \mathsf{HADAMARD} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \qquad (2.32)$$

We can rephrase the result of [Shi02, Aha03] in terms similar to those of Theorem 2.36 as follows:

Theorem 2.47. *For every family of poly-size quantum circuits $\mathcal{C} = \{C_n\}$ using an* *arbitrary set of unitary gates, there exists a family of poly-size quantum T-F circuits $C' = \{C'_n\}$ such that for all inputs $x \in \Sigma^n$ and outputs in $y \in \Sigma^*$*

$$\Big|\Pr(\mathsf{C}_n(x) = y) - \Pr(\mathsf{C'}_n(x) = y)\Big| < \frac{1}{2^n} \tag{2.33}$$

□

Similarly to QTM's, quantum circuits cannot be locally derandomised. In other words, a model similar to randomised circuits having instead access to qubit-carrying wires could not accurately simulate quantum circuits for the same reasons as for QTM's.

Quantum circuits can be initialised with classical values in a similar fashion as probabilistic circuits. However, the fact that the circuit must be reversible imposes two differences with traditional probabilistic circuits in how we give quantum circuits a computation semantics. First of all, because the number of outputs is equal to the number of inputs, and not everything that we will want to compute is a permutation, we will have to designate some of the output nodes as carrying the result of the desired computation; those other output nodes which do not carry the desired answer are often referred to as *garbage (qu)bits.* Secondly, it might be the case that in order to compute some function on inputs of size n, some additional input nodes might be required, which must be initialised to some fixed classical value.[6]

As a result, we must make the following adjustments before we define a sound computation semantics. Given an input size $n \in \mathbb{N}$, let C be a circuit of width w, with $w \geq n$. Given a "logical" input x of size n, i.e. $x = x_1, \ldots, x_n$, then the "physical" input to the circuit will be the classical input $x' = x0^{w-n}$, i.e. the string x appropriately padded with 0's to total length w. In ket notation, we have that $|x'\rangle = |x\rangle \otimes |0\rangle^{\otimes w-n}$. By initialising, the circuit with $|x'\rangle$, we are implicitly designating the first n input nodes as "logical" input nodes, and the bottom $w-n$ input nodes as ancillæ, set to 0. Then, if $|\phi(x)\rangle$ is the final state of C when so initialised, we can define the associated holistic random variable $\mathsf{C}(x)$ with domain on Σ^w, with statistics defined as per Equation 1.43.

Definition 2.48. Let $\mathcal{C} = \{C_n \,|\, n \in \mathbb{N}\}$ be a family of quantum circuits of width w input nodes, $w = w(n) \geq n$. We say that $\mathcal{C}$ decides a language L, with accuracy bound A and

[6] These concepts were already well established within the field of Reversible Computation, and have little to do with "quantumness."

B, $A, B \subseteq [0,1]$, if for every input size $n \in \mathbb{N}$, and input $x \in \Sigma^n$,

$$\begin{aligned} x \in L &\implies \Pr(\mathsf{C}_n(x') = \text{“}1\ldots\text{”}) \in A \\ x \notin L &\implies \Pr(\mathsf{C}_n(x') = \text{“}0\ldots\text{”}) \in B \end{aligned} \tag{2.34}$$

Definition 2.49. Let $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ be a family of quantum circuits of width w input nodes, $w = w(n) \geq n$. We say that $\mathcal{C}$ computes a (uniform) family of functions $\mathcal{F} = \{f_n \mid f_n : \Sigma^n \mapsto \Sigma^m, m = m(n), \forall n \in \mathbb{N}\}$, with accuracy bound $D \subseteq [0,1]$, if for every input $x \in \Sigma^n$,

$$\sum_{\substack{y_1,\ldots,y_w \\ f(x)=y_1,\ldots,y_m}} \Pr\big(\mathsf{C}_n(x') = y_1 \ldots y_w\big) \in D \tag{2.35}$$

2.3.3 Quantum Complexity Classes

Similarly to probabilistic computing, non-uniform quantum circuit complexity has not been well studied, and is probably of limited interest.

On the other hand, even though they are “quantum” objects, quantum circuits can be described classically (e.g. by writing out the unitary matrix that describes them). Thus, we can think of quantum algorithms as uniformly generated families of quantum circuits. This approach, has many advantages over the QTM-based description of Quantum Complexity. Furthermore, for the poly-time/poly-size based complexity classes that interest us, the QTM vs. quantum circuit equivalence of [Yao93] along with Theorem 2.47 allows us to characterise all of the following classes in terms of uniformly generated families of quantum T-F circuits.

It is important to note that other circuit initialisation models are possible, such as that in which all input nodes are set to 0. This is the so called *input non-uniform* model, in which the deterministic TM generating quantum circuit descriptions is allowed to generate a different circuit C_x for every input x on its tape. This model is in fact equivalent to the one presented above in the context of polynomially bounded computations.

The notion of tractability applies to Quantum Computing as to classical computing models. First, we do not consider tractable problems for which the resources required to solve them increase super-polynomially. Second, we do not expect our quantum devices to be unreasonably lucky, much the same as for classical probabilistic computing devices. This gives rise to the the first quantum complexity class [BB92], which embodies problems which can tractably be solved with a quantum computer.

Definition 2.50. A language L is in the class BQP if there exists a $k > 0$ and a family of quantum (T-F) circuits $\mathcal{C}$ uniformly generated in deterministic poly-time that decides L with accuracy bound $A = B = [1/2 + 1/n^k, 1]$.

Note that like its probabilistic cousin BPP the class BQP can also be amplified [BBBV97] to arbitrary accuracy parameters $A = B = [2/3, 1]$ or $A = B = [1 - 1/2^k, 1]$.

The non-deterministic class NP, however, has two different analogues in the quantum world. If we view languages in NP as having poly-size witnesses, then the quantum analogue is the class QMA (sometimes also refered to as QNP), which is the class of problems having poly-size quantum witnesses verifiable in quantum poly-time. On the other hand if we view NP as an instance of a one-sided "lucky" probabilistic class, such as in Theorem 2.42, then its quantum analogue is the class NQP.

Definition 2.51. A language L is in the class NQP if there exists a family of quantum (T-F) circuits $\mathcal{C}$ uniformly generated in deterministic poly-time that decides L with accuracy bounds $A = (0, 1]$ and $B = \{1\}$.

It is an open problem whether the classes QMA and NQP classes are equivalent or not.

Finally, one can also define a quantum analogue of the two-sided "lucky" class PP as follows.

Definition 2.52. A language L is in the class QPP there exists a family of quantum (T-F) circuits $\mathcal{C}$ uniformly generated in deterministic poly-time that decides L with accuracy bounds $A = B = (1/2, 1]$.

However, the following theorem has been known by complexity theorists for a while [For02]. It has an elementary proof (reproduced in [BFH02]) once the necessary tools from the Theory of Counting Complexity have been introduced (see [For97] for a good survey on this topic).

Theorem 2.53. QPP = PP

2.4 A Unified Algebraic View of Complexity Classes

In this section, we build upon the ideas introduced in Section 1.6 and our previous discussion of the various circuits models in this chapter to provide a unified picture of complexity classes.

While it is not the usual way to do so, it is possible to formulate all of the above polytime-based probabilistic classes in terms of uniformly generated probabilistic circuits, even deterministic classes, as we have already done in Section 2.2.4. On the other hand, by applying the vectorial model for PD's (the PD-space introduced in Section 1.6.1, we can represent the states of a probabilistic circuits as unit vectors in l_1 norm within $\mathbb{R}_+^n(\mathbb{R})$. Similarly, we can consider states of boolean circuits, as unit vectors within the vector space $\mathbb{B}^n(\mathbb{B})$. In both cases, the outcome probabilities are defined by the same measurement rule (Equation 1.46).

In the quantum case, we have seen how to formulate quantum classes in terms of quantum circuits, whose states can be represented as vectors in PA-space, i.e. complex unit vectors in l_2 norm. In that case, however, the measurement rule (Equation 1.51) is different, which induces different restrictions on the dynamics, and thus on the circuit gates.

It is possible to generalise these notions of circuit by using the *semiring* algebraic structure.

Definition 2.54 (Semiring). A *semiring*[Gol92, KS86] is a tuple $(\mathbb{S}, +, \cdot)$ with $\{0, 1\} \subseteq \mathbb{S}$ and two binary operations $+, \cdot : \mathbb{S} \times \mathbb{S} \mapsto \mathbb{S}$ (sum and product), such that $(\mathbb{S}, +, 0)$ is a commutative monoid[7], $(\mathbb{S}, \cdot, 1)$ is a monoid, and multiplication distributes over sum, i.e.,

$$a \cdot (b + c) = a \cdot b + a \cdot c \quad \text{and} \quad (a + b) \cdot c = a \cdot c + b \cdot c,$$

for every a, b, and c in $\mathbb{S}$, and $0 \cdot a = a \cdot 0 = 0$ for every a in $\mathbb{S}$.

A semiring $\mathbb{S}$ is *commutative* if and only if $a \cdot b = b \cdot a$ for every a and b.

With these properties, we can soundly define *modules* $\mathbb{S}^n(\mathbb{S})$ as the sound generalisation of the concept of vector space [8] for the following semirings which will be of interest to us.

- The Boolean Algebra $(\mathbb{B}, \vee, \wedge)$.
- The commutative semiring of positive rational $(\mathbb{Q}_+, +, \cdot)$ and positive real $(\mathbb{R}_+, +, \cdot)$ numbers.

[7] A *monoid* is a group where not all elements have inverses.

[8] Normally, vector spaces are only defined in terms of fields.

- The fields of rational $(\mathbb{Q}, +, \cdot)$, real $(\mathbb{R}, +, \cdot)$ and complex numbers $(\mathbb{C}, +, \cdot)$.
- The skew field (or division algebra) of quaternions $(\mathbb{H}, +, \cdot)$.

Our definition of circuit (Definition 1.4) was merely topological, in that it only defined the properties that a graph must have to be called a circuit. The notion of state of a circuit, as defined for probabilistic or quantum circuits (Definitions 1.16 and 1.26), is in fact identical. Furthermore, the definition of dynamics on such states is structurally the same in both cases (Equations 1.28 and 1.44). Thus it is justified to advance the following generic definitions of circuit.

Definition 2.55 (Algebraic Circuit). An *algebraic circuit* over a semiring $\mathbb{S}$ is a circuit $C = (\mathcal{I}, \mathcal{O}, \mathcal{G}, \mathcal{W})$ with the following semantics:

Initialisation. The input nodes $i \in \mathcal{I}$ are initialised with individual values $|x_i\rangle \in \mathbb{S}^2(\mathbb{S})$.

Dynamics. Each gate $g \in \mathcal{G}$ performs a map $\gamma_g : \mathbb{S}^k \mapsto \mathbb{S}^\ell$ from its k input connections onto its ℓ output connections.

Furthermore, we say that an algebraic circuit

- is *linear*, if $\forall g \in \mathcal{G}$, γ_g is linear,
- is *probabilistic*, if it is linear, $\forall i \in \mathcal{I}$, $\| x_i \|_1 = 1$, and all gates preserve l_1 norm (i.e. $\forall g \in \mathcal{G}$, γ_g is stochastic),
- is *quantum*, if it is linear, $\forall i \in \mathcal{I}$, $\| x_i \|_2 = 1$, and all gates preserve l_2 norm (i.e. $\forall g \in \mathcal{G}$, γ_g is (anti)-unitary),
- is *reversible* if $\forall g \in \mathcal{G}$, γ_g is a bijection (C is necessarily a gate array), and
- is *deterministically initialised* if $\forall i \in \mathcal{I}$, $|x_i\rangle \in \{|0\rangle, |1\rangle\}$.

Let us now discuss generalisations of the computation semantics of probabilistic and quantum circuits (Definitions 2.25 and 2.48, respectively) for the corresponding types of algebraic circuits. Note that these definitions are structurally the same, as all that they require is a measurement rule assigning probabilities to the final state of the circuit.

So far we have only considered a computation semantics for deterministically initialised circuits. This might seem unnecessarily restrictive, because it does not necessarily allow the algebraic circuits to take full advantage of the rich algebraic structure of the underlying semiring. However, If we relax the initialisation requirement to allow for arbitrary input vectors $|x_i\rangle$, then the computational model become unnecessarily powerful, because, for example, some of those numbers involved might uncomputable or require large amounts of computation. The solution lies to this problem is the same as that for circuit uniformisation: we force the input values (possibly non-deterministic) to also be uniformly generated in polynomial time.

Definition 2.56. Let $\mathcal{C} = \{C_n\}$ be a family of algebraic circuits over some semiring $\mathbb{S}$, with C_n having $k = k(n)$ input nodes and $\ell = \ell(n)$ output nodes.

We say that a language L is *decided with arbitrary inputs* with accuracy bounds A and B by $\mathcal{C}$ if there exists a deterministic poly-time $M_{\mathcal{C}}$ which on input $x \in \Sigma^n$

1. generates a description of C_n, *and*

2. generates a description of the k 2-dimensional input vectors $|x_1\rangle, \ldots, |x_k\rangle$, such that

$$\begin{aligned} x \in L &\implies \sum_{y_2,\ldots,y_\ell} \Pr\left(\mathsf{C}_n(|x\rangle) = 1y_2\ldots y_\ell\right) \in A \\ x \notin L &\implies \sum_{y_2,\ldots,y_\ell} \Pr\left(\mathsf{C}_n(|x\rangle) = 0y_2\ldots y_\ell\right) \in B \end{aligned} \tag{2.36}$$

where $|x\rangle = |x_1\rangle \otimes \cdots \otimes |x_k\rangle$ and C_n is the random variable associated with the final state of C.

2.4.1 Characterisations Based on Probabilistic Algebraic Circuits

In the case of probabilistic algebraic circuits, we can only define a computation semantics such as that of 2.25 on non-negative semirings[9], because otherwise the measurement rule (as defined) would yield negative outcome probabilities. Thus restricted, it is sound to introduce the following notation for characterising complexity classes.

Definition 2.57. Let $\mathbb{S}$ be a non-negative semiring. We denote by $\mathcal{P}^*_{\mathbb{S}}(A, B)$, $A, B \subseteq [0, 1]$, the class of all languages decided with accuracy bounds A and B by families of *deterministically initialised* probabilistic algebraic circuits on $\mathbb{S}$, uniformly generated in poly-time.

[9] By this we simply mean that $\mathbb{S} \subseteq \mathbb{R}_+$.

However, this is an overly restrictive definition (as we will see in some cases), because it does not necessarily allow the algebraic circuits to take full advantage of the rich algebraic structure of the underlying semiring. However, if we relax the initialisation requirement to allow for arbitrary input vectors $|x_i\rangle$, then the corresponding computational models become unnecessarily powerful, for example, because these inputs could contain hard (or impossible) to compute "advice." The solution is to force the input values (possibly non-deterministic) to also be uniformly generated in polynomial time.

Definition 2.58. Let $\mathbb{S}$ be a non-negative semiring. We denote by $\mathcal{P}_{\mathbb{S}}(A, B)$, $A, B \subseteq [0, 1]$, the class of all languages decided with accuracy bounds A and B by families of polytime uniformly generated probabilistic algebraic circuits on $\mathbb{S}$, initialised with *arbitrary inputs*

The non-negative semirings that we consider are $\mathbb{B}$, $\mathbb{Q}_+$ and $\mathbb{R}_+$. For these we have the following structural relationships.

Theorem 2.59. *For all A and B in $[0, 1]$ we have that:*

i. $\mathcal{P}^*_{\mathbb{B}}(A, B) = \mathcal{P}_{\mathbb{B}}(A, B)$

ii. $\mathcal{P}_{\mathbb{Q}_+}(A, B) = \mathcal{P}^*_{\mathbb{Q}_+}(A, B) \subseteq \mathcal{P}^*_{\mathbb{R}_+}(A, B) = \mathcal{P}_{\mathbb{R}_+}(A, B)$

□

Proof. The proof of the first statement is immediate from the definitions. So are the inclusions, $\mathcal{P}^*_{\mathbb{Q}_+}(A, B) \subseteq \mathcal{P}_{\mathbb{Q}_+}(A, B)$ and $\mathcal{P}^*_{\mathbb{R}_+}(A, B) \subseteq \mathcal{P}_{\mathbb{R}_+}(A, B)$.

The converse of these inclusions is a simple generalisation of the proof of equivalence of randomised and coin-flip circuits (Theorem 2.35). Suppose that the i-th input is initialised with $|x_i\rangle = a_i|0\rangle + b_i|1\rangle$, with $a_i, b_i \in \mathbb{S}$. Then, given a description of the inputs $|x_i\rangle$, the generating TM can also generate a description of a gate which transforms the vector $|0\rangle$ into $|x_i\rangle$. Thus, a description of a deterministically initialised circuit in the same model can also be generated.

Finally, the inclusion $\mathcal{P}_{\mathbb{Q}_+}(A, B) \subseteq \mathcal{P}_{\mathbb{R}_+}(A, B)$ is trivially derived from the fact that $\mathbb{Q} \subset \mathbb{R}$. □

Note that in general the contrapositive of the inclusion in *ii* is not true for all A and B. However, for those classes which do not require the accuracies to be exponentially close to some fixed value, we can use Theorem 2.36 to show equality of both of these models.

In particular, with this notation we can easily classify the following classical complexity classes:

Theorem 2.60.

$$\begin{aligned}
\mathrm{P} &= \mathcal{P}^*_{\mathbb{B}}(\{1\},\{1\}) \\
\mathrm{BPP} &= \mathcal{P}^*_{\mathbb{Q}_+}([2/3,1],[2/3,1]) = \mathcal{P}_{\mathbb{R}_+}([2/3,1],[2/3,1]) \\
\mathrm{RP} &= \mathcal{P}^*_{\mathbb{Q}_+}((1/2,1],\{1\}) \\
\mathrm{NP} &= \mathcal{P}^*_{\mathbb{Q}_+}((0,1],\{1\}) \\
\mathrm{PP} &= \mathcal{P}^*_{\mathbb{Q}_+}((1/2,1],(1/2,1])
\end{aligned}$$

□

2.4.2 Characterisations Based on Reversible Circuits

We now turn to a special case of algebraic probabilistic circuits: probabilistic circuits that are also reversible. Since these circuits are by definition linear, this means that all gate transformations γ_g are represented by permutations matrices. This kind of dynamics we also called *extrinsic* in Chapter 1, because the probabilistic behavious is "outside" of the circuit. For this particular case, we introduce a similar notation as before for both the case of deterministic and arbitrary inputs.

Definition 2.61. Let $\mathbb{S}$ be a non-negative semiring. We denote as $\mathcal{XP}_{\mathbb{S}}(A,B)$, the class of all languages decided with accuracy bounds A and B, $A, B \subseteq [0,1]$, by families of reversible, probabilistic algebraic circuits on $\mathbb{S}$ uniformly generated in poly-time.

Similarly, $\mathcal{XP}^*_{\mathbb{S}}(A,B)$ the class of languages accepted by such circuits under the additional restriction that they be *deterministically initialised.*

As before we have the following structural relationships:

Theorem 2.62. *For all A and B, $A, B \subset [0,1]$ we have that:*

$$\mathcal{XP}_{\mathbb{B}}(A,B) = \mathcal{XP}^*_{\mathbb{B}}(A,B) = \mathcal{XP}^*_{\mathbb{Q}_+}(A,B) = \mathcal{XP}^*_{\mathbb{R}_+}(A,B) = \mathcal{P}^*_{\mathbb{B}}(A,B) \tag{2.37}$$

$$\subseteq \left\{ \begin{array}{ccc} \mathcal{XP}_{\mathbb{Q}_+}(A,B) & \subseteq & \mathcal{P}_{\mathbb{Q}_+}(A,B) \\ \cap\text{I} & & \cap\text{I} \\ \mathcal{XP}_{\mathbb{R}_+}(A,B) & \subseteq & \mathcal{P}_{\mathbb{R}_+}(A,B) \end{array} \right. \tag{2.38}$$

Proof. The first equality for $\mathbb{B}$ is trivial from the definition. The equalities $\mathcal{XP}^*_{\mathbb{S}}(A,B) = \mathcal{XP}^*_{\mathbb{B}}(A,B)$, for any $\mathbb{S}$, stem from the inability of deterministically initialised circuits to generate any state vectors other than deterministic ones with only permutation gates. The equality with $\mathcal{P}^*_{\mathbb{B}}(A,B)$ is derived from the fact any deterministic circuit can be converted into a reversible one (with deterministically initialised ancillæ) with at most a linear size overhead. All other inclusions are by definition. □

As a result of this characterisation we can also classify the standard classes as follows.

Theorem 2.63.

$$\begin{aligned} \text{P} &= \mathcal{XP}^*_{\mathbb{B}}(\{1\},\{1\}) = \mathcal{XP}^*_{\mathbb{Q}_+}([2/3,1],[2/3,1]) \\ \text{BPP} &= \mathcal{XP}_{\mathbb{Q}_+}([2/3,1],[2/3,1]) \\ \text{RP} &= \mathcal{XP}_{\mathbb{Q}_+}((1/2,1],\{1\}) \\ \text{NP} &= \mathcal{XP}_{\mathbb{Q}_+}((0,1],\{1\}) \\ \text{PP} &= \mathcal{XP}_{\mathbb{Q}_+}((1/2,1],(1/2,1]) \end{aligned}$$

□

2.4.3 Characterisations Based on Quantum Circuits

Unlike in the case of probabilistic circuits, we can define a PA-space on any of the semirings we have introduced above, non-negative or not. Thus, we can also use a similar notation to represent classes accepted by quantum circuits on arbitrary semirings, by generalising the computation semantics for "traditional" quantum circuits, i.e. quantum circuits on $\mathbb{C}$.

Definition 2.64. We denote as $\mathcal{Q}^*_{\mathbb{S}}(A,B)$, $A, B \subseteq [0,1]$, the class of all languages decided by families of deterministically initialised, quantum algebraic circuits on $\mathbb{S}$, uniformly generated in poly-time with accuracy bounds A and B.

Definition 2.65. We denote as $\mathcal{Q}_{\mathbb{S}}(A,B)$, $A, B \subseteq [0,1]$, the class of all languages decided with arbitrary inputs with accuracy bounds A and B by families of quantum circuits on $\mathbb{S}$.

In this case, several interesting structural relationships exist.

Theorem 2.66. *For any semiring $\mathbb{S}$, and $A, B \subseteq [0,1]$ we have*

$$\mathcal{Q}^*_{\mathbb{S}}(A,B) \cap \mathcal{P}^*_{\mathbb{S}}(A,B) = \mathcal{XP}^*_{\mathbb{B}}(A,B)$$

In other words, if we are restricted to deterministically initialised models, models which are both quantum and probabilistic are computationally equivalent to deterministic ones.

Proof. This proof is an application of Theorem 1.31, which tells us that the only dynamics possible in this case are matrices of the form $U = D\Pi$, where Π is a permutation and $D = \sum_j e^{\mathrm{i}\theta_j}|j\rangle\langle j|$ is a (diagonal) phase change matrix. In fact, from a computational point of view we can completely ignore these phase factors, because they will never change under permutations, and at the end of circuit evaluation no information can be extracted from them because of the measurement rule. Thus the whole process can be simulated by replacing U with Π, with no error. □

As a consequence of Lemma 1.32, we also directly obtain the following relationship

Corollary 2.67. *For any non-negative semiring $\mathbb{S}_+$ and $A, B \subseteq [0,1]$ we have*

$$\mathcal{Q}_{\mathbb{S}}(A,B) = \mathcal{XP}_{\mathbb{B}}(A,B)$$

□

Finally, it is also possible for quantum circuits to efficiently and reversibly generate "random" inputs, in a very similar fashion as for probabilistic circuits (*vide supra* the proof of Theorem 2.59)

Theorem 2.68. *For all semirings $\mathbb{S}$ and $A, B \subseteq [0,1]$ we have*

$$\mathcal{Q}^*_{\mathbb{S}}(A,B) = \mathcal{Q}_{\mathbb{S}}(A,B)$$

Finally, as we did for probabilistic classes, we can also use this notation to re-caracterise the following quantum complexity classes.

Theorem 2.69.

$$\begin{aligned}
\mathrm{EQP} &= \mathcal{Q}^*_{\mathbb{B}}(\{1\},\{1\}) \\
\mathrm{BQP} &= \mathcal{Q}^*_{\mathbb{Q}}([2/3,1],[2/3,1]) \\
\mathrm{NQP} &= \mathcal{Q}^*_{\mathbb{Q}}((0,1],\{1\}) \\
\mathrm{(Q)PP} &= \mathcal{Q}^*_{\mathbb{Q}}((1/2,1],(1/2,1])
\end{aligned}$$

□

2.4.4 Characterisations Based on Extrinsic Quantum Circuits

In Section 2.4.2 we decided to consider reversible circuit as probabilistic circuits. However, while the "probabilistic" attribute of an algebraic circuit is a property having to with its computation semantics, reversibility is purely a structural property. Thus, we can equally think of reversible "quantum" circuits, whose computation semantics is derived from l_2-based measurement rule. On the other hand, all quantum circuits must be reversible because of the linearity requirement (Theorem 1.23), and this would seem to be a shallow endeavour. In fact, what we are interested in is in studying the subset of quantum computing models where the dynamics is deterministic, but the initial states might not. In other words, the quantum equivalent of randomised circuits, which we will call *extrinsic* in that the "quantumness" will be external to the dynamics.

Definition 2.70. We denote by $\mathcal{XQ}_{\mathbb{S}}(A,B)$, $A, B \subseteq [0,1]$, the class of languages decided with arbitrary inputs with accuracy bounds A and B by families of quantum circuits whose gates all implement permutations (on the PA-space).

We do not bother to define a deterministically initialised version $\mathcal{XQ}^*_{\mathbb{S}}(\cdot,\cdot)$ of this notation because it is trivially equivalent to $\mathcal{XP}^*_{\mathbb{S}}(\cdot,\cdot)$. Other than this simple relationship we have that:

Theorem 2.71. *For all $A, B \subseteq [0,1]$*

$$\begin{aligned}
\mathcal{XQ}_{\mathbb{B}}(A,B) &= \mathcal{XP}_{\mathbb{B}}(A,B) \\
\mathcal{XQ}_{\mathbb{R}_+}(A,B) &= \mathcal{XP}_{\mathbb{R}_+}(A,B)
\end{aligned}$$

Proof. The first identity stems from the fact that the l_1 and the l_2 norm coincide on the booleans $\mathbb{B}$. They do not coincide in the case of the positive reals, however. Nonetheless, there is an isomorphism between l_1 unit vectors in $\mathbb{R}^n_+(\mathbb{R}_+)$ and l_2 unit vectors on the same vector space: simply take the square root of each component. Thus, since initial states are always product states, it is possible to efficiently convert them from one form (PD-vector) to the other (PA-vector) and *vice-versa.* □

Of particular note is the class $\mathcal{XQ}_{\mathbb{R}}(A,B)$, for $A = B = [2/3, 1]$. In essence, this class is identical to BQP except that we only allow quantum gates at the beginning of the circuit. Since the `HADAMARD` gate is quantum universal with the `TOFFOLI` gate, we can think of these extrinsic quantum circuits as having their quantumness externalised; they are deterministic circuits with access to "quantum coins" in the state $1/\sqrt{2}|0\rangle - 1/\sqrt{2}|1\rangle$ (the minus sign is necessary, because otherwise we would have the class BPP). Furthermore, we can easily remove the unsightly square roots by considering these coins two at a time. We thus have deterministic circuits with "negative probability coins."

Equivalently, we can think of this model as quantum circuit where we are only allowed to use quantum gates at the first level of the circuit. Despite this transition, note that this kind of circuit is capable of generating entanglement. We therefore think that it is very unlikely equivalent to classical computing, despite the fact that structurally differ in only a single minus sign...

As far as we know, however, it is an interesting open question (probably not even that hard to answer) whether this model is equivalent to the full blown quantum computing model. We conjecture that it is.

Conjecture 2.72. *$\mathcal{XQ}_{\mathbb{R}}(A,B)$=$\mathcal{Q}_{\mathbb{R}}(A,B)$*

In the next chapter we will take these algebraic generalisations further and prove computational equivalences regarding the quaternion algebra also. In the last chapter of this part, we will then reinforce the characterisations given in this section by providing complete and promise complete problems for the classes discussed here.

CHAPTER 3

REAL AND QUATERNIONIC COMPUTING

In Chapter 1 we introduced a common description of some of the classical theories and quantum theories using the mathematical formalisms of PA-spaces. In Chapter 2, we used that formalism to show and illustrate the fact that the most important complexity classes can also be expressed within a unified meta-model. For example, the fundamental classes of deterministic and probabilistic computing can be fully characterised by PA-based models over the Boolean and positive real vector spaces, respectively. On the other hand, we have described Quantum Computing in terms of the PA-space over complex vector spaces. We have done so for historical reasons, and following the lead of Quantum Mechanics. One question that then arises naturally is which other algebras we can define reasonable and sound computational models with, and how do their power of computation relate to that of the other PA-based models that we have described. This chapter will explore the question of what kind of computational models can be defined, which involve other finitely generated algebras over which a PA-space can be properly defined.

It was first shown that restricting ourselves to real amplitudes does not diminish the power of quantum computing [BV97], and further, that in fact rational amplitudes are sufficient [ADH97]. Both these results were proven in the Quantum Turing Machine model, and the respective proofs are quite technical. Direct proofs of the first result for the quantum circuit model stem from the fact that several sets of gates universal for quantum computing have been found [Kit97, BMP+00, Shi02, RG02], which involve only real coefficients. We will redefine and review the results known for computing PA-spaces based on real vector spaces in Section 3.1, also providing a new generic and structural proof of the equivalence of this model to standard complex quantum computing.This proof can then be easily adapted to the quaternionic case, as we will show in Section 3.2.

3.1 Real Computing

3.1.1 Definitions

Intuitively, the real computing model is defined as a restricted version of quantum computing, where all amplitudes in the state vectors are required to be real numbers.

Conjugation is equivalent to the identity operation and bras are simply transposed kets. Similarly the matrix dagger operator (†) can be replaced with the matrix transpose operator (^{t}).

In this case, we must replace unitary transformations with orthonormal transformations, as these are the only inner-product preserving operations on this inner-product space. One could conceive a model in which the state vectors always have real amplitudes, but in which arbitrary unitary transformations (on the complex Hilbert space) are allowed, as long as the end result is still a real amplitude vector. It is elementary to show that orthonormal transformations are the only ones that have this property, and hence this model is as general as can be, given the fact that we insist that the amplitudes be real.

Rebits and States

In quantum computing and quantum information theory, we define the *qubit* as the most elementary information-containing system. Abstractly, the state of a qubit can be described by a 2-dimensional state vector

$$|\Phi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ s.t. } \| \Phi \|_2 = \sqrt{|\alpha|^2 + |\beta|^2} = 1 \tag{3.1}$$

where $|0\rangle$ and $|1\rangle$ are the two canonical basis vectors for such a 2-dimensional space. Two vectors $|\Phi\rangle$ and $|\Phi'\rangle$ are said to represent the same qubit value if they are in the same 1-dimensional ray. In other words,

$$\Phi \equiv \Phi' \iff |\Phi\rangle = e^{i\theta}|\Phi'\rangle, \text{where } \theta \in [0, 2\pi). \tag{3.2}$$

Definition 3.1 (Rebit). The corresponding concept in real computing is called a *rebit.* As in Equation 3.1, its state can also be described by a 2-dimensional vector on the real Hilbert space

$$|\Phi\rangle = a|0\rangle + b|1\rangle, \text{ s.t. } \| \Phi \|_2 = \sqrt{a^2 + b^2} = 1 \tag{3.3}$$

In this case, the arbitrary phase factor can only be $+1$ or -1, and the rebit equivalence relation which replaces Equation 3.2 is

$$\Phi \equiv \Phi' \iff |\Phi\rangle = e^{\mathrm{i}\theta}|\Phi'\rangle, \text{where } \theta \in \{0, \pi\} \tag{3.4}$$

$$\iff |\Phi\rangle = \pm|\Phi'\rangle \tag{3.5}$$

Similarly as for qubits, single rebit states do have a nice geometrical interpretation: they are isomorphic to the circumference, having $|0\rangle$ and $|1\rangle$ at opposite extremes. One way to see this is to consider the locus of points in the Bloch sphere for which $e^{\mathrm{i}\theta} = 1$, or in other words, those with no circular polarisation. Unfortunately, there is no such nice geometric representation of an arbitrary n-qubit state, and we believe the same is true for n-rebit states.

The computational basis vectors for a rebit are still $|0\rangle$ and $|1\rangle$, and for arbitrary n-rebit systems they can also be represented as n-bit strings. The measurement rule in defining the probabilities of obtaining the corresponding bit string as a result is still the same

$$\Pr(|\Phi\rangle \mapsto \text{“}\mathbf{b}\text{”}) = \langle\Phi|b\rangle^2 \tag{3.6}$$

One physical interpretation that can be given for rebits or rebit systems is that of a system of photons, where we use the polarisation in the usual manner to carry the information. However, these photons are restricted to having zero circular polarisation, and being operated upon with propagators which never introduce circular polarisation, i.e. orthonormal operators. The computational basis measurements are still simple polarisation measurements in the vertical-horizontal basis.

Real Circuits and Real Computational Complexity

Real circuits are nothing but a particular example of algebraic circuits, as described in Definition 2.55, for the field of real numbers $\mathbb{R}$. Using the same notation as in Section 2.4.1, we denote the classes of languages accepted by real circuits with $\mathcal{Q}_{\mathbb{R}}(\cdot,\cdot)$, and $\mathcal{Q}^*_{\mathbb{R}}(\cdot,\cdot)$ for the deterministically initialised circuits.

3.1.2 Previously Known Results

From a Complexity Theory point of view, the first question that arises naturally is how does this real computing model compare with the quantum computing one. In other words, can the problems which are efficiently solved by a quantum algorithm also be solved by an efficient real algorithm?

For the Quantum Turing Machine model, the answer was previously known to be "Yes". Even though, it is not explicitly stated as such, the following theorem is traditionally attributed to Bernstein and Vazirani, as it can be easily deduced from the results in [BV97].

Theorem 3.2 (Bernstein, Vazirani). *Any Quantum Turing Machine can be approximated sufficiently well by another, whose transition matrix only contains computable real numbers of the form* $\pm\cos(kR)$ *and* $\sin(kR)$*, where* k *is an integer and*

$$R = \sum_{i=1}^{\infty} 1/2^{2^i}.$$

The need for having such transcendental amplitudes was eventually removed. By using transcendental number theory techniques, Adleman, Demarrais, and Huang showed in [ADH97], that, in fact, only a few rational amplitudes were required, in particular only the set $\{0, \pm1, \pm3/5, \pm4/5\}$.

It is important to note that Theorem 3.2 does not apply directly to circuits, or at least not in a completely trivial manner. The constructions in the proof are relatively elaborate and rely heavily on techniques of Turing Machine engineering. Nonetheless, quantum circuits were shown to be equivalent to Quantum Turing Machines by A.C.-C. Yao in [Yao93]. In principle, the construction of that proof could be used to show that quantum circuits do not require states with complex amplitudes to achieve the same power as any complex-valued circuit or QTM.

However, the celebrated universality result of Barenco, Bennett, Cleve, DiVicenzo, Margolus, Shor, Sleator, Smolin, and Weinfurter [BBC+95] provides a direct proof of that fact, as they show that CNOT and arbitrary 1-qubit rotations —which contain only real coefficients— form a universal set of gates for quantum circuits. More recent results have produced ever smaller sets of universal gates (this is just a sample list):

- TOFFOLI, HADAMARD, and $\pi/4$-rotation, by Kitaev [Kit97] in 1997.
- CNOT, HADAMARD, $\pi/8$-rotation by Boykin, Mor, Pulver, Roychowdhury, and Vatan [BMP+00] in 2000.
- TOFFOLI and HADAMARD, by Shih [Shi02] in 2002, with a simpler proof by Aharonov [Aha03] in 2003.

- Controlled θ-rotations, by Rudolph and Grover [RG02] in 2002

The motivation behind these results was to come up with the simplest possible gates, given the fact that quantum states in nature can and will have arbitrary complex amplitudes, and thus, so will their unitary propagators. The fact that the simpler sets involve only real numbers was *a priori* just a "desirable side-effect." Our motivation, however, is completely different. We play a different game: suppose that all we had were these mysterious "rebits," unable to enter complex amplitudes. What could we do then? Because of this motivation, our proof will have a different flavour. In fact, the proof is completely general in that it works with any universal set of gates. In particular, it will work with gates which have arbitrary complex transition amplitudes. In other words, in proving the following, more general theorem, we will completely ignore the above results. That will allow us to recycle its proof later on in Section 3.2.

Theorem 3.3. *Any n-qubit quantum circuit constructed with gates of degree d or less (possibly including non-standard complex coefficients gates) can be exactly simulated with an $n+1$ rebit circuit with the same number of gates of degree at most $d+1$.*

3.1.3 A New Proof of Equivalence

3.1.3.1 The Underlying Group Theory

The idea behind the proof is to make use of the fact that the group $\mathrm{SU}(N)$ can be embedded into the group $\mathrm{SO}(2N)$. We provide an explicit embedding h.[1] While this mapping is not unique, what is special about it is that it has all the necessary properties for us to define a sound simulation algorithm based on it. This mapping is defined as follows. Given an arbitrary unitary transformation U, its image $O = h(U)$ is

$$U \overset{h}{\mapsto} O = h(U) \triangleq \left(\begin{array}{c|c} \mathrm{Re}(U) & \mathrm{Im}(U) \\ \hline -\mathrm{Im}(U) & \mathrm{Re}(U) \end{array}\right) \tag{3.7}$$

where the Re and Im operators return the real and imaginary parts of a complex number, respectively, and applied to complex matrices, return the matrix composed of the real

[1] Independently, Aharonov [Aha03] has also used this mapping recently to provide a simple proof that TOFFOLI and HADAMARD are universal.

and imaginary parts of each entry. Note also, that if we define the following formal tensor

$$\mathcal{T} \triangleq \begin{pmatrix} \mathrm{Re} & \mathrm{Im} \\ -\mathrm{Im} & \mathrm{Re} \end{pmatrix} \tag{3.8}$$

we can express the definition of h more simply as

$$U \overset{h}{\mapsto} O = h(U) = \mathcal{T} \otimes U \tag{3.9}$$

The first fundamental property that this mapping must have for us to use it effectively in a simulation is the following.

Theorem 3.4. *Let G_N represent the image of* SU(N) *under h. Then h is a proper group isomorphism between* SU(N) *and G_N, and G_N is a subgroup of* SO(N).

Proof. It is easy to see that any matrix in G_N, which will have the form of Equation 3.7, will have a unique inverse image, and hence that h is an injective mapping. The following lemma is sufficient to show that h is a group homomorphism.

Lemma 3.5. *Let A and B be any two arbitrary $N \times N$ matrices, then $h(AB) = h(A)h(B)$.*

Proof. The first step is to obtain a simple matrix multiplication rule for matrices, using the operators Re and Im. For arbitrary complex numbers α and β, we have that

$$\begin{aligned} \mathrm{Re}(\alpha\beta) &= \mathrm{Re}(\alpha)\,\mathrm{Re}(\beta) - \mathrm{Im}(\alpha)\,\mathrm{Im}(\beta) \\ \mathrm{Im}(\alpha\beta) &= \mathrm{Re}(\alpha)\,\mathrm{Im}(\beta) + \mathrm{Im}(\alpha)\,\mathrm{Re}(\beta) \end{aligned} \tag{3.10}$$

Since these rules hold for the products of all of their entries, it is then easy to see that this same multiplication rule will also hold for complex matrices. In other words, we can substitute α and β in Equation 3.10 with any two arbitrary complex matrices A and B which are multipliable, to get

$$\begin{aligned} \mathrm{Re}(AB) &= \mathrm{Re}(A)\,\mathrm{Re}(B) - \mathrm{Im}(A)\,\mathrm{Im}(B) \\ \mathrm{Im}(AB) &= \mathrm{Re}(A)\,\mathrm{Im}(B) + \mathrm{Im}(A)\,\mathrm{Re}(B) \end{aligned} \tag{3.11}$$

We are now equipped to verify our claim

$$\begin{aligned}
h(A)h(B) &= (\mathcal{T}\otimes A)(\mathcal{T}\otimes B) \\
&= \left(\begin{array}{c|c} \operatorname{Re}(A) & \operatorname{Im}(A) \\ \hline -\operatorname{Im}(A) & \operatorname{Re}(A) \end{array}\right) \left(\begin{array}{c|c} \operatorname{Re}(B) & \operatorname{Im}(B) \\ \hline -\operatorname{Im}(B) & \operatorname{Re}(B) \end{array}\right) \\
&= \left(\begin{array}{c|c} \operatorname{Re}(A)\operatorname{Re}(B) - \operatorname{Im}(A)\operatorname{Im}(B) & \operatorname{Re}(A)\operatorname{Im}(B) + \operatorname{Im}(A)\operatorname{Re}(B) \\ \hline -\operatorname{Im}(A)\operatorname{Re}(B) - \operatorname{Re}(A)\operatorname{Im}(B) & -\operatorname{Im}(A)\operatorname{Im}(B) + \operatorname{Re}(A)\operatorname{Re}(B) \end{array}\right) \\
&= \left(\begin{array}{c|c} \operatorname{Re}(AB) & \operatorname{Im}(AB) \\ \hline -\operatorname{Im}(AB) & \operatorname{Re}(AB) \end{array}\right) \\
&= \mathcal{T}\otimes AB = h(AB) \qquad (3.12)
\end{aligned}$$

□

Finally, we want to show that $G_N \subset \mathrm{SO}(2N)$. This is equivalent to showing that all the images $O = h(U)$ are orthonormal, i.e. that $O^{\mathrm{t}} = O^{-1}$. Since by Lemma 3.5 h is a group homomorphism, it maps inverse elements into inverse elements, i.e. $h(U^{-1}) = h(U)^{-1}$. Since U is unitary, we have that

$$O^{-1} = h(U)^{-1} = h(U^{-1}) = h(U^{\dagger}) \qquad (3.13)$$

while the following lemma will give us an expression for O^{t}.

Lemma 3.6. *Let A be an arbitrary $N \times N$ complex matrix, then $h(A^{\dagger}) = h(A)^{\mathrm{t}}$.*

Proof. By definition of h and by transposition rules of block matrices, we have

$$\begin{aligned}
h(A)^{\mathrm{t}} &= (\mathcal{T}\otimes A)^{\mathrm{t}} \\
&= \left(\begin{array}{c|c} \operatorname{Re}(A) & \operatorname{Im}(A) \\ \hline -\operatorname{Im}(A) & \operatorname{Re}(A) \end{array}\right)^{\mathrm{t}} \\
&= \left(\begin{array}{c|c} \operatorname{Re}(A)^{\mathrm{t}} & -\operatorname{Im}(A)^{\mathrm{t}} \\ \hline \operatorname{Im}(A)^{\mathrm{t}} & \operatorname{Re}(A)^{\mathrm{t}} \end{array}\right) \\
&= \left(\begin{array}{c|c} \operatorname{Re}(A^{\dagger}) & \operatorname{Im}(A^{\dagger}) \\ \hline -\operatorname{Im}(A^{\dagger}) & \operatorname{Re}(A^{\dagger}) \end{array}\right) \\
&= \mathcal{T}\otimes A^{\dagger} = h(A^{\dagger}) \qquad (3.14)
\end{aligned}$$

where we also used the following generic matrix identities

$$\begin{aligned}\mathrm{Re}(A^\dagger) &= \mathrm{Re}(A)^{\mathrm{t}} \\ \mathrm{Im}(A^\dagger) &= -\mathrm{Im}(A)^{\mathrm{t}}.\end{aligned} \tag{3.15}$$

□

In particular, we have that $O^{\mathrm{t}} = h(U)^{\mathrm{t}} = h(U^\dagger) = h(U^{-1}) = O^{-1}$, and we are done proving Theorem 3.4. □

The fact that h is a group isomorphism is important, because it implies that G_N is preserved under "serial" circuit construction. In other words, it means that if we have real circuits that simulate the quantum circuits with operators U and V, then we can simulate a quantum circuit with operator UV by simply putting both real circuits together. This suggests a way in which to decompose the problem of simulating a generic quantum circuit, i.e. by constructing the real circuit one level at a time.

3.1.3.2 The Simulation Algorithm

Let C be a generic n-qubit quantum circuit with operator U_C, composed of s elementary gates. The simulation algorithm will consist of the following steps:

Step 1. Serialise the given circuit by finding an ordering of its gates, so that they can be evaluated in that order, one by one. In other words, find a total order of the circuit gates, such that $U_C = U^{(s)}U^{(s-1)}\dots U^{(2)}U^{(1)}$.

Step 2. For each gate $g \in \{1, \dots, s\}$ in the above ordering, replace the n-ary operation $U^{(g)}$, corresponding to the g-th gate, with an adequate real circuit $O^{(g)}$ simulating it.

Step 3. Construct the overall real circuit C' by concatenating the circuits for each level g, in the same order as defined in Step 1. This is, if O_C is the operator for C', then let $O_C = O^{(s)}\dots O^{(2)}O^{(1)}$.

Step 4. Write a description of the real circuit C' and of its input state and ask the real computing "oracle" to provide the result of a measurement on its final state.

Step 5. Perform the classical post-processing on the result of the measurement and provide a classical answer.

The algorithm, as described so far, is not completely defined. In what follows, we will derive, one by one, the missing details.

First, the total order in Step 1 can be obtained by doing a topological sort of the circuit's directed graph. This can be done efficiently in time polynomial in the size of the circuit[2]. The effects of Step 1 on C are depicted in Figure 3.1.

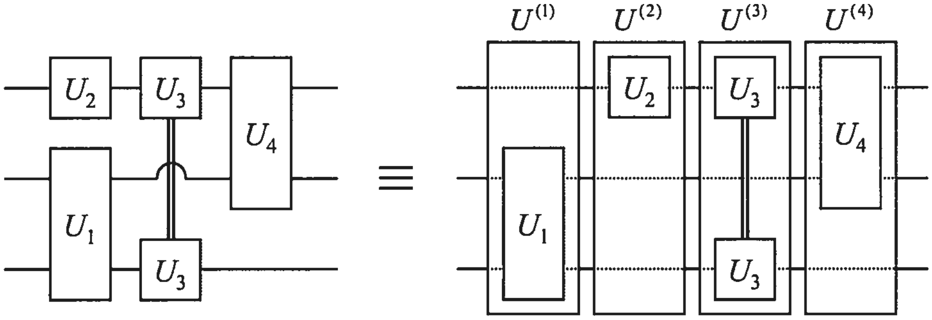


Figure 3.1: Serialisation of the quantum circuit C in Step 1.

3.1.3.3 Constructing the Real Circuit

In principle, each of the elementary quantum gates g is described by a unitary operator defined on the d-qubit complex Hilbert space. We can assume without loss of generality that these gates are described in the input to the simulation algorithm as $2^d \times 2^d$ matrices[3], which we denote with subscripted capitals. Thus, the g-th gate has associated to it a d-ary gate operator U_g (with typically $d = 1, 2$).

However, in the context of a circuit the operator fully describing the action of gate g is an N-ary operator acting on all n qubits, which depends not only on U_g but also on the positions of the wires on which g acts. We denote this operators with superscripted capitals. Thus, after serialisation of the circuit C in Step 1, these operators $U^{(g)}$ will correspond to the g-th level of the serialised version of C.

In general, the g-th gate will be a d-ary gate operating on wires with indices $j_1 < j_2 < \cdots < j_d$, not necessarily contiguous, with the associated circuit operator $U^{(g)}$, which will depend on $j_1, \ldots, j_d$. For example, in the case of a 2-qubit gate g operating on the j-th and k-th wires, $1 \leq j < k \leq n$, $U^{(g)}$ can be expressed in terms of its elementary gate

[2] These orderings, because of the fact that they can be found efficiently, are the base of the "strong" equivalence of the circuit and the Turing Machine models of computation.

[3] In fact, what we are given are finite-precision *approximations* of these matrices.

U_g as follows

$$\begin{aligned} U^{(g)}(j,k) &= S_{1,j}\, S_{2,k}\, (U_g \otimes I_{n-2})\, S_{2,k}\, S_{1,j} \\ &\triangleq S_g\, (U_g \otimes I_{n-2})\, S_g \end{aligned} \tag{3.16}$$

where I_m is the identity operator for m qubits, $S_{i,j}$ is the n-qubit swap operator acting on wires i and j, and $S_g \triangleq S_{j,k}$ is a shorthand for describing the necessary swap operator for the g-th gate. The logic behind Equation 3.16 is explained graphically in Figure 3.2. Note however, that this conversion using swap gates is not itself part of the simulation, but only a mathematical convenience to be used later. These swaps gates will not be included in the final real circuit C' and do not represent a computational overhead.

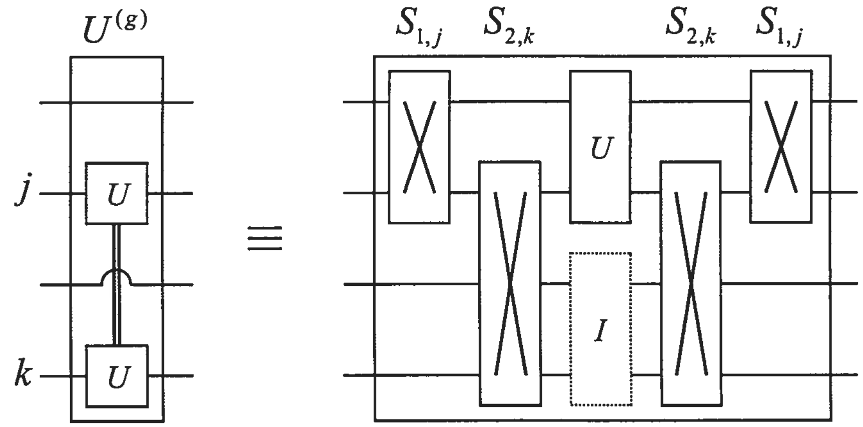


Figure 3.2: Obtaining an expression for the N-ary circuit operator $U^{(g)}$.

As for Step 2, the isomorphism h readily suggests a method for substituting each of the s levels of the original quantum circuit C. Let $H^d_{\mathbb{C}}$ be the 2^d-dimensional complex Hilbert space on which U_g acts, and let $H^d_{\mathbb{R}}$ be the 2^{d+1}-dimensional real Hilbert space on which its image $O_g = h(U_g)$ acts. If g is a d-qubit gate, then O_g operates on $d+1$ rebits. We thus have an extra wire, and it is not *a priori* clear how to map the original d quantum wires with these $d+1$ real wires. To resolve this ambiguity, we need to define how we associate the base vectors of $H^d_{\mathbb{C}}$ with those of $H^d_{\mathbb{R}}$.

We use the columns of the tensor $\mathcal{T}$ defining h in Equation 3.9, to define the following mappings between $H^d_{\mathbb{C}}$ and $H^d_{\mathbb{R}}$. Let $|\Phi\rangle$ be an arbitrary state vector in $H^d_{\mathbb{C}}$, and let

$\mathcal{T} = [\mathcal{T}_0|\mathcal{T}_1]$,

$$|\Phi\rangle \xmapsto{h_0} |\Phi_0\rangle \triangleq \mathcal{T}_0 \otimes |\Phi\rangle = \begin{pmatrix} \text{Re} \\ -\text{Im} \end{pmatrix} \otimes |\Phi\rangle \tag{3.17}$$

$$\xmapsto{h_1} |\Phi_1\rangle \triangleq \mathcal{T}_1 \otimes |\Phi\rangle = \begin{pmatrix} \text{Im} \\ \text{Re} \end{pmatrix} \otimes |\Phi\rangle \tag{3.18}$$

Note that the images $|\Phi_0\rangle$ and $|\Phi_1\rangle$ are mutually orthogonal in $H^d_{\mathbb{R}}$. In addition, both h_0 and h_1 are proper linear homomorphisms, as can be easily verified given the distributivity of the tensor product with matrix addition.

The base vectors $|b\rangle$ of $H^d_{\mathbb{C}}$ are column vectors with all zero entries, except with a 1 at the integer value j of $\mathbf{b}$; i.e. $\langle j|b\rangle = 1$, and $\langle k|b\rangle = 0, k \neq j$. Thus, it is easy to see what these basis vectors are mapped to:

$$|b\rangle \quad \mapsto \quad |b_0\rangle = |0\rangle \otimes |b\rangle \tag{3.19}$$

$$\mapsto \quad |b_1\rangle = |1\rangle \otimes |b\rangle \tag{3.20}$$

These homomorphisms define the semantics to give to each of the $d+1$ real wires on which O_g acts, as is shown in Figure 3.3. When the original quantum gate takes $|b\rangle$ as input, the corresponding real gate O_g has two possible base vectors $|b_0\rangle$ or $|b_1\rangle$ as inputs. This corresponds to having an extra wire at the top of the gate with value $|0\rangle$ or $|1\rangle$ respectively, and the base state $|b\rangle$ in the bottom d wires. Finally, note that since U_g is represented as a matrix of constant dimension, then O_g is also a small matrix, which can be computed from U_g and written down in constant time.

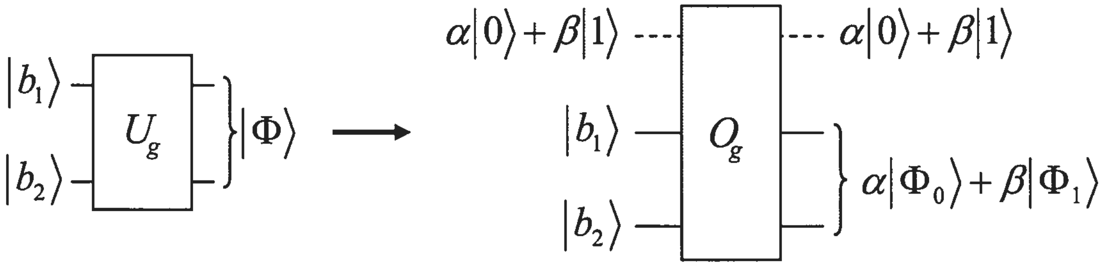

$|b_1\rangle$
$|b_2\rangle$
U_g
$|\Phi\rangle$
$\alpha|0\rangle+\beta|1\rangle$
$\alpha|0\rangle+\beta|1\rangle$
$|b_1\rangle$
$|b_2\rangle$
O_g
$\alpha|\Phi_0\rangle+\beta|\Phi_1\rangle$

Figure 3.3: Simulation of an individual elementary binary quantum gate, by a tertiary real gate. Note that in general, for non classical inputs, the final state of O_g cannot be factored like in the example shown.

Even though we have defined how to simulate "out-of-context" d-ary elementary quantum gates, we have not yet explained how to simulate them in their corresponding positions in the circuit C. In other words, we still have to describe how to simulate the N-ary operators $U^{(g)}$. Again, the isomorphism h comes to the rescue: we will simulate $U^{(g)}$ by finding an $(n+1)$-rebit circuit that computes its image $O^{(g)} = h(U^{(g)})$ under h. Unfortunately, we cannot simply construct this circuit from the matrix definition of $h(U^{(g)})$, because it is a huge matrix and that would require exponential time. However, $U^{(g)}$ is a very simple N-ary operator: it is after all just a d-ary gate, which has a succinct description given by Equation 3.16. Since it involves at most d qubits, then the circuit $O^{(g)}$ only needs to involve those same wires and one other extra rebit.

At this point, we have to make a further apparently arbitrary choice, i.e. which one of the $n-d$ other available wires will play the role of the "top" rebit for the O_g gate? In other words, where shall we place the extra wire required for implementing $O^{(g)}$? The answer comes from the homomorphisms h_0 and h_1 in Equations (3.17) and (3.18), respectively. They are also automatically defined on the state space $H_{\mathbb{C}}^N$ of the whole circuit, and hence they generate the same wire semantics as for isolated d-ary gates: the extra wire must be at the top of the circuit, as is shown in Figure 3.4. Similarly, as in Equation 3.16, we have for the case where $d=2$, an expression for $O^{(g)}$ in terms of O_g.

$$\begin{aligned} O^{(g)}(j,k) &= S_{2,j+1}\, S_{3,k+1}\, (O_g \otimes I_{n-2})\, S_{3,k+1}\, S_{2,j+1} \\ &\triangleq S'_g\, (O_g \otimes I_{n-2})\, S'_g \end{aligned} \tag{3.21}$$

where again we define S'_g for convenience, and j and k are the indices of the wires on which gate g acts on the original circuit C.

We now have a simple and well defined scheme for constructing the desired simulating circuit C'. In Step 3, we will construct C' by concatenating the real circuits for the N-ary operators $O^{(g)}$. One important characteristic of this scheme is that we are *reusing* the extra wire needed for each gate, each time using the same top wire. This is illustrated in Figure 3.5. Even though they act on the whole space $H_{\mathbb{C}}^N$, the $O^{(g)}$ operators are simply $(d+1)$-ary gates put in context, and they can be described in a succinct manner requiring only a constant number of symbols. Therefore, the overall size of the description for C' will be linear in the size of the initial description of C which was given as input.

What is remarkable about this scheme, is that despite its simplicity, it gives precisely

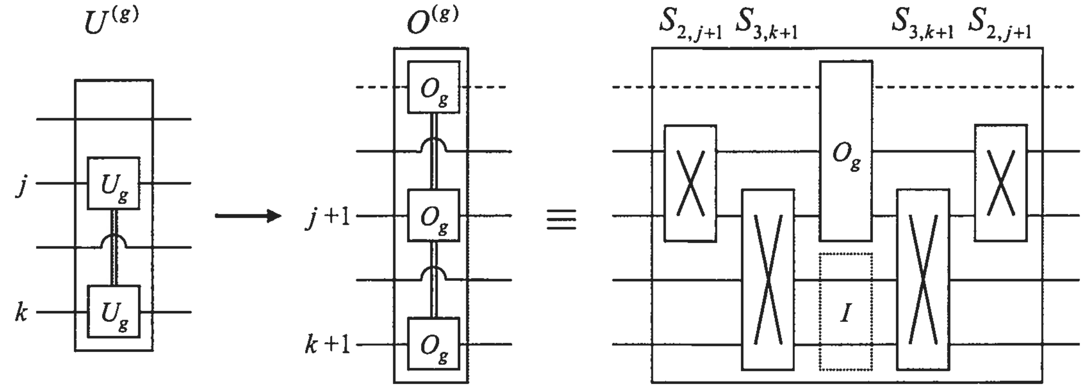


Figure 3.4: Obtaining an expression for the $(N+1)$-ary circuit $O^{(g)}$.

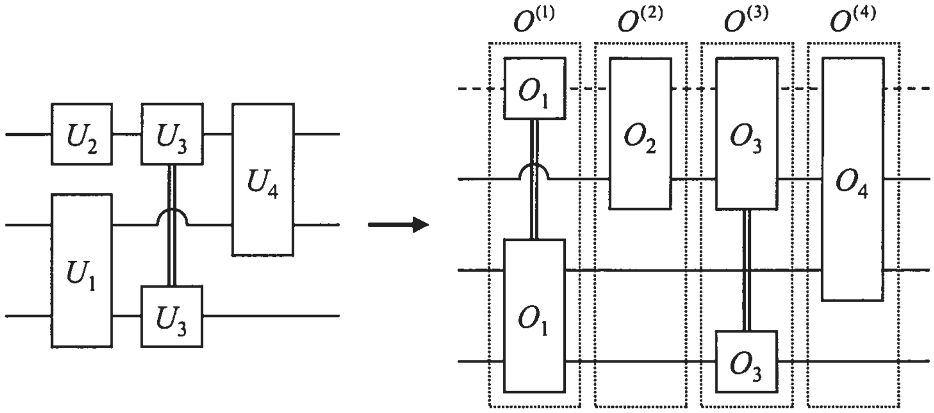


Figure 3.5: Simulation of a quantum circuit by a real circuit.

what we wanted, this is, that the final operator O_C be in some sense as similar as possible to the operator U_C of the original circuit. In fact, we have the following third nice property of our simulation.

Lemma 3.7. *The inverse image of O_C is precisely U_C, i.e. $O_C = h(U_C)$.*

Proof. Because of the serialisation of Step 1, we have that $U_C = U^{(s)} \dots U^{(2)} U^{(1)}$. We use this and the group isomorphism properties of h from Lemma 3.5 to obtain the following expression for its image

$$\begin{aligned}
h(U_C) &= h(U^{(s)} \ldots U^{(1)}) \\
&= h(U^{(s)}) \ldots h(U^{(1)}) \\
&= \prod_{\substack{i=0, \\ g=s-i}}^{s-1} h(U^{(g)})
\end{aligned}$$

We can now use the expression of Equation 3.16 to substitute for $U^{(g)}$,

$$\begin{aligned}
&= \prod h(\, S_g \, (U_g \otimes I_{n-2}) \, S_g) \\
&= \prod h(S_g) \; h(U_g \otimes I_{n-2}) \; h(S_g)
\end{aligned}$$

Since S_g is composed only of 0's and 1's, we have that $\mathrm{Re}(S_g) = S_g$ and $\mathrm{Im}(S_g) = 0$. Furthermore, we have that $S'_g = I_1 \otimes S_g$ from their definition in Equations (3.16) and (3.21), and thus,

$$\begin{aligned}
&= \prod (I_1 \otimes S_g) \; h(U_g \otimes I_{n-2}) \; (I_1 \otimes S_g) \\
&= \prod S'_g \, h(U_g \otimes I_{n-2}) \; S'_g
\end{aligned}$$

However, the tensor product is just a formal operation, and its associativity property holds even with a tensor of operators like $\mathcal{T}$. Hence, we have

$$\begin{aligned}
&= \prod S'_g \, [\mathcal{T} \otimes (U_g \otimes I_{n-2})] \; S'_g \\
&= \prod S'_g \, [\,(\mathcal{T} \otimes U_g) \otimes I_{n-2}\,] \; S'_g \\
&= \prod S'_g \, [h(U_g) \otimes I_{n-2}\,] \; S'_g \\
&= \prod S'_g \, (O_g \otimes I_{n-2}) \; S'_g
\end{aligned}$$

which with the padding expression of $O^{(g)}$ in Equation 3.21 finally gives

$$= \prod_{\substack{i=0, \\ g=s-i}}^{s-1} O^{(g)} = O_C \, . \qquad (3.22)$$

□

3.1.3.4 Circuit Initialisation and Measurement

Having described how to construct the real circuit C' from the original circuit C, we still have to address the issue of how to initialise C' in Step 4, and furthermore of how to interpret and use its measurements to simulate the initial quantum algorithm in Step 5.

Let $|\Psi\rangle$ represent the initial state given to C, and let $|\Phi\rangle$ be its image under U_C, i.e. the final state of the circuit before measurement. If we think back of the two homomorphisms h_0 and h_1 from $H_{\mathbb{C}}^N$ to $H_{\mathbb{C}}^N$, induced by h, we have two logical choices for initialising the corresponding real circuit O_C, the states $|\Psi_0\rangle$ and $|\Psi_1\rangle$. Which should we choose, and in either case what will the output look like? The answer to the latter question is given by the following lemma.

Lemma 3.8. *The images of $|\Psi_0\rangle$ and $|\Psi_1\rangle$ in the real circuit C' are*

$$\begin{aligned} O_C|\Psi_0\rangle &= \mathcal{T}_0 \otimes |\Phi\rangle = |\Phi_0\rangle && (3.23)\\ O_C|\Psi_1\rangle &= \mathcal{T}_1 \otimes |\Phi\rangle = |\Phi_1\rangle && (3.24)\end{aligned}$$

Proof. As in the proof of Lemma 3.5, all we require are the matrix multiplication rules of Equation 3.11

$$\begin{aligned} O_C|\Psi_0\rangle &= (\mathcal{T} \otimes U_C)(\mathcal{T}_0 \otimes |\Psi_0\rangle) \\ &= \left(\begin{array}{c|c} \mathrm{Re}(U_C) & \mathrm{Im}(U_C) \\ \hline -\mathrm{Im}(U_C) & \mathrm{Re}(U_C) \end{array}\right) \left(\begin{array}{c} \mathrm{Re}(|\Psi_0\rangle) \\ \hline \mathrm{Im}(|\Psi_0\rangle) \end{array}\right) \\ &= \left(\begin{array}{c} \mathrm{Re}(U_C)\,\mathrm{Re}(|\Psi_0\rangle) + \mathrm{Im}(U_C)\,\mathrm{Im}(|\Psi_0\rangle) \\ \hline -\mathrm{Im}(U_C)\,\mathrm{Re}(|\Psi_0\rangle) + \mathrm{Re}(U_C)\,\mathrm{Im}(|\Psi_0\rangle) \end{array}\right) \\ &= \left(\begin{array}{c} \mathrm{Re}(U_C|\Psi_0\rangle) \\ \hline -\mathrm{Im}(U_C|\Psi_0\rangle) \end{array}\right) \\ &= \mathcal{T}_0 \otimes (U_C|\Psi_0\rangle) \\ &= \mathcal{T}_0 \otimes |\Phi\rangle = |\Phi_0\rangle && (3.25)\end{aligned}$$

With the same method, we can obtain a similar expression for Φ_1, i.e.

$$\begin{aligned} O_C|\Psi_1\rangle &= (\mathcal{T} \otimes U_C)(\mathcal{T}_1 \otimes |\Psi_1\rangle) \\ &= \left(\begin{array}{c|c} \mathrm{Re}(U_C) & \mathrm{Im}(U_C) \\ \hline -\mathrm{Im}(U_C) & \mathrm{Re}(U_C) \end{array}\right) \left(\begin{array}{c} \mathrm{Im}(|\Psi_1\rangle) \\ \hline \mathrm{Re}(|\Psi_1\rangle) \end{array}\right) = \dots \\ &= \mathcal{T}_1 \otimes |\Phi\rangle = |\Phi_1\rangle \end{aligned} \tag{3.26}$$

□

Let us assume for a moment —and in fact, this is without loss of generality— that the original circuit was to be initialised with some base vector $|x\rangle$, with a final state $|\Phi\rangle = U|x\rangle$. Again, there are two possible choices for initialising the corresponding real circuit, namely $|x_0\rangle = |0\rangle|x\rangle$ and $|x_1\rangle = |1\rangle|x\rangle$. What would then be the output of the simulated circuit in either case? In the very special case that $|\Phi\rangle$ is also a base vector, then we would have $|\Phi_0\rangle = |0\rangle|\Phi\rangle$ and $|\Phi_1\rangle = |1\rangle|\Phi\rangle$, and thus, in either case, the bottom n-wires would contain the right answer and we can ignore the top wire. But when $|\Phi\rangle$ is some arbitrary pure state, neither purely real nor purely imaginary, we cannot give such a nice semantic to the top wire. In particular, it might be entangled with the rest of the wires, and hence we cannot factor the final state.

Nonetheless, what is surprising is that if we *trace out* the top wire, in all cases we will get the same statistics and furthermore that we will obtain the right statistics, i.e. the same as if we had used the original quantum circuit C. More formally, we have

Lemma 3.9. *Let $|\Phi\rangle$ be an arbitrary n-qubit pure state, and let $\rho_0 = \mathrm{Tr}_1|\Phi_0\rangle\langle\Phi_0|$ and $\rho_1 = \mathrm{Tr}_1|\Phi_1\rangle\langle\Phi_1|$ represent the partial traces obtained by tracing out (i.e. forgetting about) the top wire. Then we have that*

$$\rho_0 = \rho_1, \tag{3.27}$$

$$\mathrm{Diag}\,(\rho_0) = \mathrm{Diag}\,(\rho_1) = \mathrm{Diag}\,(|\Phi\rangle\langle\Phi|). \tag{3.28}$$

Proof. The partial trace of the first wire of an arbitrary density operator given in block matrix form

$$\rho = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$$

is given by,

$$\begin{aligned}\mathrm{Tr}_1(\rho) &= [I_n|0]\,\rho\,[I_n|0]^\dagger + [0|I_n]\,\rho\,[0|I_n]^\dagger \\ &= A + D \end{aligned} \tag{3.29}$$

In particular, we have that

$$|\Phi_0\rangle\langle\Phi_0| = (\mathcal{T}_0 \otimes |\Phi\rangle)\ (\mathcal{T}_0 \otimes |\Phi\rangle)^{\mathrm{t}}$$

which by applying transposition rules for block matrices and Equation 3.11 gives

$$\begin{aligned} &= \left(\begin{array}{c} \mathrm{Re}(|\Phi\rangle) \\ \hline -\mathrm{Im}(|\Phi\rangle) \end{array}\right) \left(\begin{array}{c|c} \mathrm{Re}(\langle\Phi|) & \mathrm{Im}(\langle\Phi|) \end{array}\right) \\ &= \left(\begin{array}{c|c} \mathrm{Re}(|\Phi\rangle)\,\mathrm{Re}(\langle\Phi|) & \mathrm{Re}(|\Phi\rangle)\,\mathrm{Im}(\langle\Phi|) \\ \hline -\mathrm{Im}(|\Phi\rangle)\,\mathrm{Re}(\langle\Phi|) & -\mathrm{Im}(|\Phi\rangle)\,\mathrm{Im}(\langle\Phi|) \end{array}\right) \end{aligned} \tag{3.30}$$

and similarly for $|\Phi_1\rangle$,

$$\begin{aligned} |\Phi_1\rangle\langle\Phi_1| &= (\mathcal{T}_1 \otimes |\Phi\rangle)\ (\mathcal{T}_1 \otimes \langle\Phi|)^{\mathrm{t}} \\ &= \left(\begin{array}{c|c} -\mathrm{Im}(|\Phi\rangle)\,\mathrm{Im}(\langle\Phi|) & \mathrm{Im}(|\Phi\rangle)\,\mathrm{Re}(\langle\Phi|) \\ \hline -\mathrm{Re}(|\Phi\rangle)\,\mathrm{Im}(\langle\Phi|) & \mathrm{Re}(|\Phi\rangle)\,\mathrm{Re}(\langle\Phi|) \end{array}\right) \end{aligned} \tag{3.31}$$

By symmetry, we thus have the same expression for both partial traces

$$\begin{aligned} \rho_0 = \rho_1 &= \mathrm{Re}(|\Phi\rangle)\,\mathrm{Re}(\langle\Phi|) - \mathrm{Im}(|\Phi\rangle)\,\mathrm{Im}(\langle\Phi|) \\ &= \mathrm{Re}\left(|\Phi\rangle\langle\Phi|\right) \end{aligned} \tag{3.32}$$

Since $|\Phi\rangle\langle\Phi|$ is hermitian, its diagonal entries are all real, and therefore it has the same diagonal entries as ρ_0 and ρ_1. □

In other words, combining this with Lemma 3.8, we arrive to the conclusion that it does not matter what we set as the initial value of the top wire, $|0\rangle$ or $|1\rangle$. Furthermore, it is easy to verify that any 1-rebit state will do, whether pure or even totally mixed, as long as it is unentangled and uncorrelated with the bottom wires.

3.1.4 Further Considerations and Consequences

3.1.4.1 Complexity of simulation

In general, if we initially have a d-qubit gate, the new gate will be a $(d+1)$-rebit gate. However, if U_g contains only real entries, then $O_g = I \otimes U_g$, which means that in this particular case the top rebit need not be involved, and therefore the new gate is the same as the original. If the whole quantum circuit we are given is constructed with such real gates, then we are in luck and we do not require the extra rebit at all. In the general complex case, however, the circuit width is at most one more than that of the original circuit.

However, one non-negligible consequence of our simulation is that any parallelism that the original circuit may have had is lost after we serialise the circuit in Step 1 of the simulation algorithm. While it might be still possible to parallelise parts of the real circuit C' (e.g. where we had real gates in the C), in the worst case, if all gates in C require complex amplitudes, then the top wire is always used and the circuit depth for C' is equal to its gate count s. This is a consequence of our decision to reuse the same wire as the "top wire" for each gate. However, it is possible to reduce this depth increase at the cost of using several "top wires" and re-combining them towards the end of the circuit. This will result in only a $O(\log s)$ increase in circuit depth.

Finally, as we have mentioned before, the overall classical pre- and post-processing requires little computational effort. Converting a description for the original circuit C into C' requires time linear in the size of the circuit description, i.e. $O(s)$. Post-processing will be exactly the same as for the original quantum algorithm, since the statistics of measuring the bottom wires of C' (or any subset thereof) will be exactly the same as those of measuring the wires of C, as per Lemma 3.9.

3.1.4.2 Universality

We knew already, from the previous results mentioned in Section 3.1.2, that it is possible to express any quantum circuit in terms of real gates only. If we had not known already that fact, we could have presumed that quantum circuits would be described and given to us in terms some universal set of gates containing at least one non-real, complex gate. In that case, Theorem 3.3 would provide a proof that a real universal set could be constructed, simply by replacing any non-real gates by its image under h.

One advantage of this technique is that it does this conversion with very limited overhead in terms of width, requiring 1 extra rebit *for the whole circuit*, and not an extra rebit for every substituted gate, as might have been expected. In addition to its usefulness in Section 3.2, this is one of the reason that we believe that this particular version of the equivalence theorem is interesting of its own, when compared to previously known results. In particular, the fact that it provides a much tighter bound on simulation resources needed, might prove useful in the study of lower quantum complexity classes and possibly in quantum information theory.

3.1.4.3 Interpretation

With Lemma 3.9, we are left with a curious paradox: while we require an extra rebit to perform the simulation, we do not care about its initial or its final value. In particular, it can be anything, even the maximally mixed state. So, what is this rebit doing?

Let H_0 and H_1 be the orthogonal subspaces, each of dimension N, spanned by the $|b_0\rangle$ and $|b_1\rangle$ base vectors of Equations 3.17 and 3.18, respectively. If a state $|\Phi\rangle$ has only real amplitudes then $|\Phi_0\rangle \in H_0$ and $|\Phi_1\rangle \in H_1$. For a generic $|\Phi\rangle$, however, $|\Phi_0\rangle$ and $|\Phi_1\rangle$ are not contained in either subspace, but in the space spanned by both, i.e. the complete rebit space $H_{\mathbb{R}}^{n}$. In that case, the top rebit will not be just $|0\rangle$ or $|1\rangle$ but some superposition thereof.

In other words, it somehow keeps track of the phase (angle) of the representation of $|\Phi\rangle$ in rebit space with respect to these subspaces. The CNOT gate (or any other real gate) does not change this phase factor. However, as arbitrary gates with complex transition amplitudes affect this phase factor, their effect is simulated by "recording" this change in the top rebit. How we initialise the top rebit gives an arbitrary initial phase to the representation of $|\Phi\rangle$, but as we saw, this initial phase does not affect statistics of the bottom wires, and thus can be set to any value. However, how this phase has been changed by previous complex gates will affect the bottom rebits in subsequent complex gates, in a similar fashion as the *phase kickback* phenomenon in many quantum algorithms[4]. That is why that top rebit is needed.

[4] With the noticeable difference that phase kickback would not work if the top qubit were maximally mixed...

3.2 Quaternionic Computing

This section closely mimics Section 3.1. First we define what we mean by quaternionic computing, making sure that it is a sensible model. We then prove an equivalence theorem with quantum computing, by using the same techniques as those of Theorem 3.3.

3.2.1 Definitions

3.2.1.1 Quaternions

Quaternions were invented by the Irish mathematician William Rowan Hamilton in 1843, as a generalisation of complex numbers. They form a non-commutative, associative division algebra. A quaternion is defined as

$$\hat{\alpha} = a_0 + a_1\mathrm{i} + a_2\mathrm{j} + a_3\mathrm{k} \tag{3.33}$$

where the coefficients a are real numbers and i, j, and k obey the equations

$$\mathrm{ii} = \mathrm{jj} = \mathrm{kk} = \mathrm{ijk} = -1 \tag{3.34}$$

Multiplication of quaternions is defined by formally multiplying two expressions from Equation 3.33, and recombining the cross terms by using Equation 3.34. It is very important to note that while all non-zero quaternions have multiplicative inverses they are *not commutative* [5]. Thus, they form what is called a *division algebra*, sometimes also called a *skew field.*

The quaternion conjugation operation is defined as follows

$$\hat{\alpha}^\star = a_0 - a_1\mathrm{i} - a_2\mathrm{j} - a_3\mathrm{k} \tag{3.35}$$

where for clarity, we represent with the (non-standard) symbol (*) in order to distinguish it from complex conjugation represented with (*). With this conjugation rule, we can define the modulus of a quaternion as

$$|\hat{\alpha}| = \sqrt{\hat{\alpha}\hat{\alpha}^\star} = \sqrt{a_0^2 + a_1^2 + a_2^2 + a_3^2} \tag{3.36}$$

[5] While the square roots of -1 are anti-commutative, e.g. $\mathrm{ij} = -\mathrm{ji}$, this is not true in general, i.e. $\hat{\alpha}\hat{\beta} \neq -\hat{\beta}\hat{\alpha}$.

Furthermore, the usual vector inner product has the required properties (i.e. it is norm defining), and a proper Hilbert space can be defined on any quaternionic linear space.

It is also possible to *complexify* the quaternions, this is, to represent them in terms of complex numbers only. Let $\hat{\alpha}$ be an arbitrary quaternion, then we define its *complex* and *weird* parts as

$$\begin{aligned} \mathrm{Co}(\hat{\alpha}) &\triangleq a_0 + a_1\mathrm{i} && (3.37) \\ \mathrm{Wd}(\hat{\alpha}) &\triangleq a_2 + a_3\mathrm{i}. && (3.38) \end{aligned}$$

We can then decompose $\hat{\alpha}$ in its complex and weird part as follows:

$$\begin{aligned} \hat{\alpha} &= a_0 + a_1\mathrm{i} + a_2\mathrm{j} + a_3\mathrm{k} \\ &= (a_0 + a_1\mathrm{i}) + (a_2 + a_3\mathrm{i})\mathrm{j} \\ &= \mathrm{Co}(\hat{\alpha}) + \mathrm{Wd}(\hat{\alpha})\mathrm{j} && (3.39) \end{aligned}$$

This equation allows us to derive multiplication rules, similar to those of Equation 3.10

$$\begin{aligned} \mathrm{Co}(\hat{\alpha}\hat{\beta}) &= \mathrm{Co}(\hat{\alpha})\,\mathrm{Co}(\hat{\beta}) - \mathrm{Wd}(\hat{\alpha})\,\mathrm{Wd}^*(\hat{\beta}) \\ \mathrm{Wd}(\hat{\alpha}\hat{\beta}) &= \mathrm{Co}(\hat{\alpha})\,\mathrm{Wd}(\hat{\beta}) + \mathrm{Wd}(\hat{\alpha})\,\mathrm{Co}^*(\hat{\beta}) && (3.40) \end{aligned}$$

where we define $\mathrm{Co}^*(\hat{\alpha}) \triangleq [\mathrm{Co}(\hat{\alpha})]^*$, and similarly for the weird part $\mathrm{Wd}^*(\hat{\alpha}) \triangleq [\mathrm{Wd}(\hat{\alpha})]^*$. It is interesting to note how the non-commutativity of quaternions is made apparent by the fact that neither identity in Equation 3.40 is symmetric with respect to $\hat{\alpha}$ and $\hat{\beta}$, unlike their equivalent for complex numbers (Equation 3.10), because in general $\mathrm{Co}^*(\hat{\alpha}) \neq \mathrm{Co}(\hat{\alpha})$ and $\mathrm{Wd}^*(\hat{\alpha}) \neq \mathrm{Wd}(\hat{\alpha})$. We can also rewrite Equation 3.36 for the modulus as

$$|\hat{\alpha}| = \sqrt{|\mathrm{Co}(\hat{\alpha})|^2 + |\mathrm{Wd}(\hat{\alpha})|^2} \qquad (3.41)$$

which is very similar to the modulus definition for complex numbers. Finally, we have the following useful identities

$$\begin{aligned} \mathrm{Co}(\hat{\alpha}^*) &= \mathrm{Co}^*(\hat{\alpha}) \\ \mathrm{Wd}(\hat{\alpha}^*) &= -\,\mathrm{Wd}(\hat{\alpha}) && (3.42) \end{aligned}$$

3.2.1.2 Quaterbits

Similarly as in quantum information theory, we can define the quaternionic equivalent to the qubit, as the most elementary quaternionic information system, the *quaterbit.* [6]

Definition 3.10 (Quaterbit). A *quaterbit* is a 2-level system with quaternionic amplitudes. It can be represented by a unit vector $|\Phi\rangle$ in a 2-dimensional quaternionic Hilbert space, i.e.

$$|\Phi\rangle = \hat{\alpha}|0\rangle + \hat{\beta}|1\rangle, \text{ s.t. } \| \Phi \|_2 = \sqrt{|\hat{\alpha}|^2 + |\hat{\beta}|^2} \tag{3.43}$$

up to an arbitrary *quaternionic* phase factor. Indeed, we have that

$$\Phi \equiv \Phi' \iff |\Phi\rangle = \hat{\eta}|\Phi'\rangle, \text{where } |\hat{\eta}| = 1. \tag{3.44}$$

The canonical values of the quaterbit correspond to the canonical basis $|0\rangle$ and $|1\rangle$ of that vector space, and are given the same semantics just as before. Similarly, we can define n-quaterbit states, with the same canonical basis as for rebits and qubits. With this definition, the measurement rule in Equation 3.6 is still sound and we adopt it axiomatically.

Quaternions are often used in computer graphics to represent rotations of the 3D Euclidean space. However, contrary to rebits or qubits, we have not found a nice geometric interpretation for the state space of even a single quaterbit.

3.2.1.3 Quaternionic Circuits

For the sake of clarity, let us distinguish the conjugate transpose operation for quaternion and complex matrices by representing them differently with the $(^\ddagger)$ and $(^\dagger)$ symbols, respectively. As before, the only linear transformations Q that preserve l_2 norm on this vector space are the *quaternionic unitary transformations*, which have the same property $Q^\ddagger = Q^{-1}$ as complex unitary transformations. They form the so-called *symplectic* group which is represented as $\mathrm{Sp}(N)$.

Thus armed with linear, inner-product preserving operations, we can in principle apply our generic algebraic circuit definition of Definition 2.55 for the quaternionic case. A *quaternionic circuit* is simply an algebraic circuit over $\mathbb{H}$.

Unfortunately, we cannot apply the same definition of computation semantics as

[6] The name "quits" has also been suggested [Pou02] and abandoned...

before (Definition 2.56). The reason is simple and quite surprising: the output of the circuit is not uniquely defined!

To see this, consider the following property of the matricial tensor product, i.e. the distributivity of the tensor product with the regular matrix product.

$$(\mathsf{A} \otimes \mathsf{B}) \cdot (\mathsf{C} \otimes \mathsf{D}) = (\mathsf{A} \cdot \mathsf{C}) \otimes (\mathsf{B} \cdot \mathsf{D}) \tag{3.45}$$

where $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$ are arbitrary matrices. This equation is in general true for any *commutative* semiring and for non-commutative semirings only if C and D are 0-1 matrices.

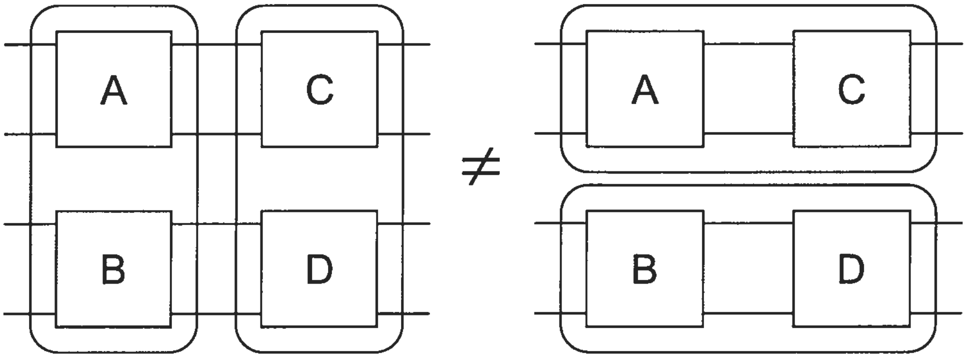


Figure 3.6: Effects of quaternionic non-commutativity on quaternionic circuits. The operator for the circuit on the left is obtained by combining the operators "vertically", by taking the tensor product first; this corresponds to the operator on the left side of Equation 3.45. The operator for the right circuit is obtained by combining them "horizontally" first, and gives the expression on the right hand side of the same equation.

Suppose now that the matrices $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$ correspond to the gate transformations in the circuit depicted in Figure 3.6. Then, the fact that Equation 3.45 does not hold means that the two different ways shown there of combining the gates will yield different operator for the circuit. Furthermore, if both cases even if we initialise in both cases with the same input, we will obtain different output statistics.

In Section 1.2.2.2, we defined the states of a circuit in terms of *temporal cuts* in the circuit graph (Definition 1.5). We then talked of the "space-time continuum" of the circuit as the set of temporal cuts, on which the circuit topology defines a partial order. Each topological sort of the circuit graph is one of the many possible total orderings of the set of cuts, or in other words a *chain* in the poset (partially-ordered set) of cuts. In more physical terms, each of these chains or total orders corresponds to a possible path in the space-time continuum of the circuit. When Equation 3.45 holds, we are guaranteed that the overal over each and all of these paths *will be the same.*

In case of the quaternionic circuis, we can expect each of these paths to give a different answer. Which of these many paths (for a poly-size circuit, there are exponentially many of them) is the "correct" one? Which one is somehow privileged by nature? Which one should we choose to be the "computational output" of the circuit? The fact is that we do not know how to resolve this ambiguity, and without it it is not completely clear what "the" model of quaternionic computing should consist of.

Nonetheless, we were able to obtain the following result, which in essence tells us that all of these paths somehow have the same computational power and can be independently simulated in an efficient fashion by a standard quantum computer. This theorem is the main result of this chapter, and its proof is very heavily inspired from that of Theorem 3.3.

Theorem 3.11. *Let $\hat{C}$ be any n-quaterbit circuit of size s, composed of gates of xdegree at most d. Let $\sigma = \{t_0, \ldots, t_s \mid t_0 < t_1 < \ldots < t_s\}$ represent any chain (or path) in the poset of temporal cuts of $\hat{C}$. Let Q_σ represent the operator of the circuit $\hat{C}$ when the gates are combined one-by-one following the ordering in σ, i.e.*

$$Q_\sigma = Q^{(s)} Q^{(s-1)} \ldots Q^{(2)} Q^{(1)},$$

where $Q^(i)$ is the (in-context) operator corresponding to the i-th gate in σ.

Then, there exists a quantum circuit of $n+1$ qubits, employing the same number of gates, each of degree at most $d+1$, that exactly *simulates the operator Q_σ.*

3.2.2 Proof of Main Theorem

3.2.2.1 More Group Theory

As before, the proof is based on the (lesser known) fact that $\mathrm{Sp}(N)$ can be embedded into $\mathrm{SU}(2N)$. We provide a mapping from one to the other, which is very similar to the one from $\mathrm{SU}(N)$ to $\mathrm{SO}(2N)$.

The mapping $\hat{h}$ from $\mathrm{Sp}(N)$ to $\mathrm{SU}(2N)$ is defined similarly to the one from $\mathrm{SU}(N)$ to $\mathrm{SO}(2N)$ given in Equation 3.7

$$Q \overset{\hat{h}}{\mapsto} U = \hat{h}(Q) \triangleq \left(\begin{array}{r|l} \mathrm{Co}(Q) & \mathrm{Wd}(Q) \\ \hline -\mathrm{Wd}^*(Q) & \mathrm{Co}^*(Q) \end{array} \right) \tag{3.46}$$

or equivalently in its tensor form, as in Equation 3.8

$$
\begin{aligned}
&= \begin{pmatrix} \mathrm{Co} & \mathrm{Wd} \\ -\mathrm{Wd}^* & \mathrm{Co}^* \end{pmatrix} \otimes Q \\
&\triangleq \hat{T} \otimes Q
\end{aligned}
\tag{3.47}
$$

At this point, what we need to show is that this $\hat{h}$ is also a group isomorphism, in other words the equivalent of Theorem 3.4.

Theorem 3.12. *Let $\hat{G}_N$ represent the image of* $\mathrm{Sp}(N)$ *under $\hat{h}$. Then $\hat{h}$ is a proper group isomorphism between* $\mathrm{Sp}(N)$ *and $\hat{G}_N$, and $\hat{G}_N$ is a subgroup of* $\mathrm{SU}(N)$*.*

Thanks to the tensor formalism, we do not need to construct the proof in full detail, as we did for Theorem 3.4. The only thing we need to show are equivalent statements to those of Lemmas 3.5 and 3.6.

Lemma 3.13. *Let A and B be any two arbitrary $N \times N$ quaternion matrices, then $\hat{h}(AB) = \hat{h}(A)\hat{h}(B)$.*

Proof. As before, it is simple to verify that the quaternion multiplication rules in Equation 3.40 also generalise to any multipliable quaternionic matrices A and B. Thus we have that

$$
\begin{aligned}
\hat{h}(A)\hat{h}(B) &= (\hat{T} \otimes A)(\hat{T} \otimes B) \\
&= \left(\begin{array}{c|c} \mathrm{Co}(A) & \mathrm{Wd}(A) \\ \hline -\mathrm{Wd}^*(A) & \mathrm{Co}^*(A) \end{array}\right) \left(\begin{array}{c|c} \mathrm{Co}(B) & \mathrm{Wd}(B) \\ \hline -\mathrm{Wd}^*(B) & \mathrm{Co}^*(B) \end{array}\right) \\
&= \left(\begin{array}{c|c} \mathrm{Co}(A)\,\mathrm{Co}(B) - \mathrm{Wd}(A)\,\mathrm{Wd}^*(B) & \mathrm{Co}(A)\,\mathrm{Wd}(B) + \mathrm{Wd}(A)\,\mathrm{Co}^*(B) \\ \hline -\mathrm{Wd}^*(A)\,\mathrm{Co}(B) - \mathrm{Co}^*(A)\,\mathrm{Wd}^*(B) & -\mathrm{Wd}^*(A)\,\mathrm{Wd}(B) + \mathrm{Co}^*(A)\,\mathrm{Co}^*(B) \end{array}\right) \\
&= \begin{pmatrix} \mathrm{Co} & \mathrm{Wd} \\ -\mathrm{Wd}^* & \mathrm{Co}^* \end{pmatrix} \otimes AB = \hat{T} \otimes AB = \hat{h}(AB)
\end{aligned}
\tag{3.48}
$$

□

Lemma 3.14. *Let A be an arbitrary $N \times N$ quaternion matrix, then $\hat{h}(A^{\ddagger}) = \hat{h}(A)^{\dagger}$.*

Proof. Similarly to the proof of Lemma 3.6, we require the following matrix identities, which are easily verified

$$\begin{aligned} \mathrm{Co}(A^{\ddagger}) &= \quad \mathrm{Co}(A)^{\dagger} \\ \mathrm{Co}^{*}(A^{\ddagger}) &= \quad \mathrm{Co}^{*}(A)^{\dagger} \\ \mathrm{Wd}(A^{\ddagger}) &= -\,\mathrm{Wd}^{*}(A)^{\dagger}. \end{aligned} \tag{3.49}$$

We then have that

$$\begin{aligned} \hat{h}(A)^{\dagger} &= \left(\begin{array}{c|c} \mathrm{Co}(A) & \mathrm{Wd}(A) \\ \hline -\,\mathrm{Wd}^{*}(A) & \mathrm{Co}^{*}(A) \end{array}\right)^{\dagger} \\ &= \left(\begin{array}{c|c} \mathrm{Co}(A)^{\dagger} & -\,\mathrm{Wd}^{*}(A)^{\dagger} \\ \hline \mathrm{Wd}(A)^{\dagger} & \mathrm{Co}^{*}(A)^{\dagger} \end{array}\right) \\ &= \left(\begin{array}{c|c} \mathrm{Co}(A^{\ddagger}) & \mathrm{Wd}(A^{\ddagger}) \\ \hline -\,\mathrm{Wd}^{*}(A^{\ddagger}) & \mathrm{Co}^{*}(A^{\ddagger}) \end{array}\right) \\ &= \hat{T} \otimes A^{\ddagger} = h(A^{\ddagger}) \end{aligned} \tag{3.50}$$

□

3.2.2.2 The Simulation Algorithm

Let $\hat{C}$ be a quaternionic circuit composed of s elementary gates of at most d quaterbits, and let Q be its quaternion linear operator. Then the quantum simulation algorithm for $\hat{C}$ will be very similar to that described in Section 3.1.3.

Step 1 Serialise the given circuit $\hat{C}$ according to σ, i.e. such that $Q_\sigma = Q^{(s)}Q^{(s-1)}\dots Q^{(2)}Q^{(1)}$.

Step 2 For each gate $g \in \{1, \dots, s\}$ in the ordering defined by σ, let $g_1 < \dots < g_d$ be the wires on which the d-ary gate Q_g acts. Replace $Q^{(g)}$ with $U^{(g)}$ the appropriately padded $(n+1)$-qubit operator for the quantum gate $U_g = \hat{h}(Q_g)$ acting on wires $g_1 + 1 < \dots < g_d + 1$ and the top qubit wire.

Step 3 Construct the overall real circuit C by concatenating the circuits for each level g, in the same order as defined in Step 1. That is, if U is the operator for C, then let $U = U^{(s)}\dots U^{(2)}U^{(1)}$.

Step 4 Write a description of the quantum circuit C and of its (classical) input state and ask the quantum computing "oracle" to provide the result of a measurement on its final state.

Step 5 Perform exactly the classical post-processing on the result as the original quaternionic algorithm.

The construction of the circuit as described in Section 3.1.3 is purely formal, and does not depend at all on the actual gates and operators. In particular, other than circuit operator algebra, the proof of Lemma 3.7 only required that h be a group isomorphism, fact which we have already established for $\hat{h}$. Thus we can claim the following equivalent lemma.

Lemma 3.15. *The inverse image of U is precisely Q, i.e. $U = \hat{h}(Q)$.*

3.2.2.3 Initialisation and Measurement

We can maintain the same semantics for $|\Phi_0\rangle$ and $|\Phi_1\rangle$, such as defined in Equations (3.17) and (3.18), by the using the columns $\hat{\mathcal{T}}_0$ and $\hat{\mathcal{T}}_1$ of the new tensor $\hat{\mathcal{T}} = [\hat{\mathcal{T}}_0|\hat{\mathcal{T}}_1]$,

$$|\Phi\rangle \overset{\hat{h}_0}{\mapsto} |\Phi_0\rangle \triangleq \hat{\mathcal{T}}_0 \otimes |\Phi\rangle = \begin{pmatrix} \mathrm{Co} \\ -\mathrm{Wd}^* \end{pmatrix} \otimes |\Phi\rangle \tag{3.51}$$

$$\overset{\hat{h}_1}{\mapsto} |\Phi_1\rangle \triangleq \hat{\mathcal{T}}_1 \otimes |\Phi\rangle = \begin{pmatrix} \mathrm{Wd} \\ \mathrm{Co}^* \end{pmatrix} \otimes |\Phi\rangle \tag{3.52}$$

With these definitions, we have the same base cases for setting the top wire, thanks to the following lemma, equivalent to Lemma 3.8.

Lemma 3.16. *Let $|\Psi\rangle$ be any n-quaterbit state, then we have that the images of $|\Psi_0\rangle$ and $|\Psi_1\rangle$ in the quantum circuit C are*

$$U|\Psi_0\rangle = \hat{\mathcal{T}}_0 \otimes |\Phi\rangle = |\Phi_0\rangle \tag{3.53}$$

$$U|\Psi_1\rangle = \hat{\mathcal{T}}_1 \otimes |\Phi\rangle = |\Phi_1\rangle \tag{3.54}$$

Proof. With the quaternion matrix multiplication rules obtained from Equation 3.40, we have

$$\begin{aligned} U|\Psi_0\rangle &= (\mathcal{T}\otimes Q)(\mathcal{T}_0\otimes|\Psi_0\rangle) \\ &= \left(\begin{array}{r|l} \mathrm{Co}(Q) & \mathrm{Wd}(Q) \\ \hline -\mathrm{Wd}^*(Q) & \mathrm{Co}^*(Q)\end{array}\right)\left(\begin{array}{r}\mathrm{Co}(|\Psi_0\rangle)\\ \hline -\mathrm{Wd}^*(|\Psi_0\rangle)\end{array}\right) \\ &= \left(\begin{array}{r}\mathrm{Co}(Q)\,\mathrm{Co}(|\Psi_0\rangle) - \mathrm{Wd}(Q)\,\mathrm{Wd}^*(|\Psi_0\rangle)\\ \hline -\mathrm{Wd}^*(Q)\,\mathrm{Co}(|\Psi_0\rangle) - \mathrm{Co}^*(Q)\,\mathrm{Wd}^*(|\Psi_0\rangle)\end{array}\right) \\ &= \left(\begin{array}{r}\mathrm{Co}(Q|\Psi_0\rangle)\\ \hline -\mathrm{Wd}^*(Q|\Psi_0\rangle)\end{array}\right) \\ &= \mathcal{T}_0\otimes(Q|\Psi_0\rangle) \\ &= \mathcal{T}_0\otimes|\Phi\rangle = |\Phi_0\rangle \end{aligned} \tag{3.55}$$

And similarly for Φ_1, i.e.

$$\begin{aligned} U|\Psi_1\rangle &= (\mathcal{T}\otimes Q)(\mathcal{T}_1\otimes|\Psi_1\rangle) \\ &= \left(\begin{array}{r|l} \mathrm{Co}(Q) & \mathrm{Wd}(Q) \\ \hline -\mathrm{Wd}^*(Q) & \mathrm{Co}^*(Q)\end{array}\right)\left(\begin{array}{c}\mathrm{Wd}(|\Psi_1\rangle)\\ \hline \mathrm{Co}^*(|\Psi_1\rangle)\end{array}\right) = \ldots \\ &= \mathcal{T}_1\otimes|\Phi\rangle = |\Phi_1\rangle \end{aligned} \tag{3.56}$$

□

Finally, we need to show that as before we can initialise with any qubit value in the top wire, ignore it at measurement, and still get the same statistics as we would have with the original quaternionic circuit. For that, we have to show that the equivalent of Lemma 3.9 is still true.

Lemma 3.17. *Let $|\Phi\rangle$ be an arbitrary n-quaterbit state, $|\Phi_0\rangle$ and $|\Phi_1\rangle$ its images under $\hat{h}_0$ and $\hat{h}_1$, and ρ_0 and ρ_1 be their respective partial traces when the first qubit wire is traced out. Then,*

$$\mathrm{Diag}\,(\rho_0) = \mathrm{Diag}\,(\rho_1) = \mathrm{Diag}\,(|\Phi\rangle\langle\Phi|). \tag{3.57}$$

Proof. The expressions for the non-reduced density operators are given by

$$\begin{aligned}
|\Phi_0\rangle\langle\Phi_0| &= (\hat{\mathcal{T}}_0 \otimes |\Phi\rangle)\,(\hat{\mathcal{T}}_0 \otimes \langle\Phi|)^\dagger \\
&= \left(\begin{array}{c} \mathrm{Co}(|\Phi\rangle) \\ \hline -\,\mathrm{Wd}^*(|\Phi\rangle) \end{array}\right) \left(\begin{array}{c|c} \mathrm{Co}(\langle\Phi|) & \mathrm{Wd}(\langle\Phi|) \end{array}\right) \\
&= \left(\begin{array}{c|c} \mathrm{Co}(|\Phi\rangle)\,\mathrm{Co}(\langle\Phi|) & \mathrm{Co}(|\Phi\rangle)\,\mathrm{Wd}(\langle\Phi|) \\ \hline -\,\mathrm{Wd}^*(|\Phi\rangle)\,\mathrm{Co}(\langle\Phi|) & -\,\mathrm{Wd}^*(|\Phi\rangle)\,\mathrm{Wd}(\langle\Phi|) \end{array}\right)
\end{aligned} \tag{3.58}$$

and similarly,

$$\begin{aligned}
|\Phi_1\rangle\langle\Phi_1| &= (\hat{\mathcal{T}}_1 \otimes |\Phi\rangle)\,(\hat{\mathcal{T}}_1 \otimes \langle\Phi|)^\dagger \\
&= \left(\begin{array}{c} \mathrm{Wd}(|\Phi\rangle) \\ \hline \mathrm{Co}^*(|\Phi\rangle) \end{array}\right) \left(\begin{array}{c|c} -\,\mathrm{Wd}^*(\langle\Phi|) & \mathrm{Co}^*(\langle\Phi|) \end{array}\right) \\
&= \left(\begin{array}{c|c} -\,\mathrm{Wd}(|\Phi\rangle)\,\mathrm{Wd}^*(\langle\Phi|) & \mathrm{Wd}(|\Phi\rangle)\,\mathrm{Co}^*(\langle\Phi|) \\ \hline -\,\mathrm{Co}^*(|\Phi\rangle)\,\mathrm{Wd}^*(\langle\Phi|) & \mathrm{Co}^*(|\Phi\rangle)\,\mathrm{Co}^*(\langle\Phi|) \end{array}\right)
\end{aligned} \tag{3.59}$$

As before, the reduced density operators are the sum of block matrices in the diagonal, which, unlike in Lemma 3.9, are not the same in both cases. However, the i-th entry in the diagonal is given by

$$\begin{aligned}
\langle i|\rho_0|i\rangle &= \langle i|\ [\mathrm{Co}(|\Phi\rangle)\,\mathrm{Co}(\langle\Phi|) - \mathrm{Wd}^*(|\Phi\rangle)\,\mathrm{Wd}(\langle\Phi|)]\ |i\rangle \\
&= \langle i|\,\mathrm{Co}(|\Phi\rangle)\ \mathrm{Co}(\langle\Phi|)|i\rangle - \langle i|\,\mathrm{Wd}^*(|\Phi\rangle)\ \mathrm{Wd}(\langle\Phi|)|i\rangle \\
&= \mathrm{Co}(\Phi_i)\,\mathrm{Co}(\Phi_i^\star) - \mathrm{Wd}^*(\Phi_i)\,\mathrm{Wd}(\Phi_i^\star) \\
&= \mathrm{Co}(\Phi_i)\,\mathrm{Co}^*(\Phi_i) + \mathrm{Wd}^*(\Phi_i)\,\mathrm{Wd}(\Phi_i) \\
&= |\mathrm{Co}(\Phi_i)|^2 + |\mathrm{Wd}(\Phi_i)|^2 = |\Phi_i|^2
\end{aligned} \tag{3.60}$$

where Φ_i is the i-th coordinate of $|\Phi\rangle$, and we use the properties of Co and Wd in Equation 3.42. We also have,

$$\begin{aligned}\langle i|\rho_1|i\rangle &= \langle i|\ [-\,\mathrm{Wd}(|\Phi\rangle)\,\mathrm{Wd}^*(\langle\Phi|)\ +\mathrm{Co}^*(|\Phi\rangle)\,\mathrm{Co}^*(\langle\Phi|)]\ |i\rangle \\ &= -\langle i|\,\mathrm{Wd}(|\Phi\rangle)\ \ \mathrm{Wd}^*(\langle\Phi|)|i\rangle + \langle i|\,\mathrm{Co}^*(|\Phi\rangle)\ \ \mathrm{Co}^*(\langle\Phi|)|i\rangle \\ &= -\,\mathrm{Wd}(\Phi_i)\,\mathrm{Wd}^*(\Phi_i^\star) + \mathrm{Co}^*(\Phi_i)\,\mathrm{Co}^*(\Phi_i^\star) \\ &= \mathrm{Wd}(\Phi_i^\star)\,\mathrm{Wd}^*(\Phi_i^\star) + \mathrm{Co}(\Phi_i^\star)\,\mathrm{Co}^*(\Phi_i^\star) \\ &= |\mathrm{Wd}(\Phi_i^\star)|^2 + |\mathrm{Co}(\Phi_i^\star)|^2 = |\Phi_i^\star|^2 = |\Phi_i|^2 \end{aligned} \tag{3.61}$$

□

3.2.3 Considerations and Consequences

3.2.3.1 Complexity of Simulation

In terms of simulation resources, the situation is similar to that of real computing. Circuit width is increased by only one, but circuit depth can be equal to the circuit size in the worst case.

For circuit size, however, we have to make a slight distinction. While the number of $(d+1)$-ary gates in the new circuit will be the same as the number of d-ary gates in the original circuit, one might not be satisfied with this type of gate count complexity for the quantum circuit, given that we do not know d and that we have very small universal gates for quantum circuits. In general, if we suppose that the original circuit given to us is constructed with some set of universal gates, then the simulation will depend on d, the number of quaterbits in the largest gate in the universal set. In particular, if $d > 3$ we might require to decompose such a gate Q_g into a set of elementary 3-, 2- or 1-qubit gates, universal for quantum computing.

We can assume wlog that we are given a full description of Q_g in terms of its $2^d \times 2^d$ quaternion matrix. We can then use the generic method for decomposing the matrix for the image quantum operator $U_g = \hat{h}(Q_g)$ into our set of elementary gates. Since U_g is a $2^{d+1} \times 2^{d+1}$ matrix this might require $O(2^d)$ time, and furthermore up to 2^{d+1} elementary gates might be required to decompose of Q_g.

If a "nice" universal set is being used where d is a small constant, then this decomposition will occur in $O(1)$ time and will produce $O(1)$ extra gates. Hence, we have that the total gate count is not *exactly* n, but is still in $O(n)$. The circuit depth which could already be as large as s, could be increased further by gate decomposition, but again, only by a constant factor.

While we have not gone through the exercise of looking for a finite universal set of elementary gates, that would be computationally universal for the symplectic group, we believe that one exists. Even without the luxury of a finite universal set, it would be in principle sensible to define a computational model using quaternionic gates, as long as the description of all circuits (and their gates) can is of limited size and can be uniformly generated. In fact, our results do not need the existence of a universal set; they just would make the computing model more "realistic."

On the other hand, let us also consider a variety of quaternionic circuits which includes gates of arbitrary degree —since we cannot show a "nice" universal set with constant degree gates, let us do so for the sake of completeness. In that case, if the circuit description has size polynomial in n, then the description of Q_g must also be of polynomial size, and this puts an upper bound on d, i.e. $d = O(\log n)$. Thus, in the worst, case, we can have that each Q_g will require $2^{d+1} = O(n)$ elementary quantum gates, all in series, with a resulting $O(n)$ depth and size overhead *for each* gate. Computing these decompositions would take time at most $O(n)$ per gate. We summarise these results in Table 3.1.

	Quaternionic circuit	Quantum circuit
width	n	$n+1$
size	s	$s2^{s+1}$
depth	t	$t2^{d+1}$

Table 3.1: The overall resources needed to simulate a quaternionic circuit built with d-ary gates, with a quantum circuit built with 2-ary gates.

We stress the fact that this is a worst case scenario due to the fact that we cannot bound d by a constant, as we have not yet shown any universal set of quaternionic gates. If we did, then $d = O(1)$, and the results would be the same, up to a constant, as those for Theorem 3.3.

3.2.3.2 Interpretation

Because of the similarity of the constructions of Theorems 3.3 and 3.11, we can give them similar interpretations. More concretely, if we label the basis of the $2N$-dimensional complex Hilbert space as $|b_c\rangle = \hat{h}_0(|b\rangle)$ and $|b_w\rangle = \hat{h}_1(|b\rangle)$, and order them accordingly, we can give the same semantics to the extra wire required by the simulation. This is, the extra qubit is at the top of the simulating circuit, and in a similar way as before keeps track of the "phase" information between both orthogonal subspaces of the complex Hilbert space spanned by the $|b_c\rangle$ and $|b_w\rangle$ base vectors. In this case, however, this information requires the full "power" of a qubit, and not just a rebit. This is due to the fact that the phase information is defined by a unit quaternion, which cannot be represented by just one angle (as is the case for a unit complex number).

We can infer, that with this same method it is not possible to simulate an n quaterbit circuit with only $n+1$ rebits. The following corollary, however, shows that just one extra rebit is sufficient.

Corollary 3.18. *Any temporal chain σ of an n-quaterbit quaternionic circuit can be exactly simulated by an $(n+2)$-rebit real circuit.*

Two proofs are possible. First, we can simply combine the results of Theorems 3.3 and 3.11. More interestingly, however, a direct proof is possible by using the standard representation of quaternions as 4×4 real matrices, which suggests the following tensor $\hat{\mathcal{S}}$

$$\hat{\mathcal{S}} \triangleq \begin{pmatrix} \mathrm{Re} & \mathrm{Im} & -\mathrm{Km} & -\mathrm{Jm} \\ -\mathrm{Im} & \mathrm{Re} & -\mathrm{Jm} & \mathrm{Km} \\ \mathrm{Km} & \mathrm{Jm} & \mathrm{Re} & \mathrm{Im} \\ \mathrm{Jm} & -\mathrm{Km} & -\mathrm{Im} & \mathrm{Re} \end{pmatrix} \tag{3.62}$$

where $\mathrm{Jm}(\hat{\alpha}) \triangleq a_2$ and $\mathrm{Km}(\hat{\alpha}) \triangleq a_3$ are the "other" imaginary parts of quaternion $\hat{\alpha}$. This tensor induces a group isomorphism from $\mathrm{Sp}(N)$ to $\mathrm{SO}(4N)$, which has all the properties required for the simulation to be sound.

It is also interesting to note that the converses of these theorems are not necessarily true. In other words, not all $(n+1)$-rebit/qubit circuits can be simulated by n qubit/quaterbit circuits. This stems from the fact that h and $\hat{h}$ do not span the whole $\mathrm{SO}(2N)$ and $\mathrm{Sp}(2N)$, respectively, as a simple counting argument shows. From a complexity point of view, this gives evidence of how little the actual amplitude structure does to change computational power, and further points to what we believe is the ultimate cause for the "quantum speedup", the possibility for these amplitudes to destructively interfere.

CHAPTER 4

COMPLETE PROBLEMS FOR PROBABILISTIC AND QUANTUM COMPUTING

As we discussed in Chapters 1 and 2, the action of circuit gates can in all cases be thought of as a linear transformation. It is natural then to represent such gates with matrices defining linear transformation with respect to the canonical basis. Even though this is not usual, one can even use this formalism for transformations between spaces of different dimensions

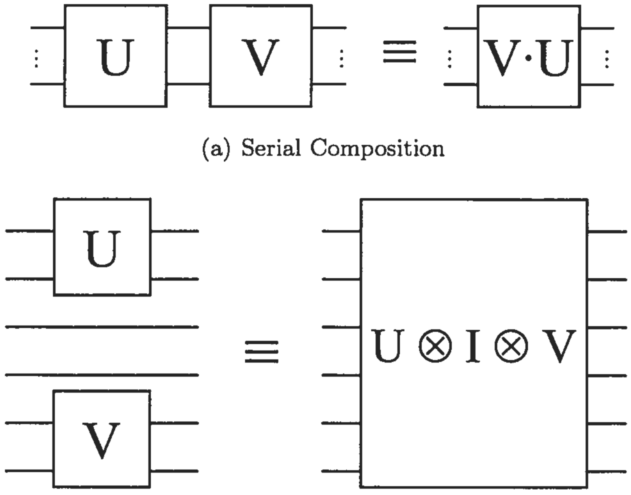
Representing gates in this way brings a significant operational advantage when manipulating and studying circuits. By the semantics given to these matrices as acting on PD- or PA-vectors by right multiplication, we obtain the following intuitive rules:

1. If a gate as k inputs and l outputs, then the corresponding matrix will be of dimension $2^l \times 2^k$.

2. If two gates are put one after the other, i.e. "in series", it is because the number of output of the first gate l_1 coincides with the number of inputs k_2 of the second. This implies that the matrices U and V representing them are multipliable in reverse order. If the underlying algebra is associative, then matrix product will also be associative and thus the matrix VU represents the linear map associated with combining both gates in such a fashion. This is shown in Figure 4.1(a).

3. If two gates are put side-by-side, i.e. "in parallel", then there is no restriction about their sizes. The matrix describing this gate combination is the tensor product $\mathsf{U}\otimes\mathsf{V}$, where U represents the gate on the top. This is shown in Figure 4.1(b).

These rules suggest a direct relationship between circuits and the theory of tensor formulæ, a generalised type of equations involving matrices as variables. In this chapter, we will exploit this relationship to come up with a series of natural complete and promise-complete problems for some of the most significant classical and quantum complexity classes. We start in Section 4.1 by introducing the notion of tensor formula. We then formally describe in Sections 4.2 and 4.3 the relationship between circuits and tensor formulæ. Finally, we introduce in Section 4.4 a type of tensor formula problem whose



(a) Serial Composition

(b) Parallel Composition

variations will be complete or promise-complete for the classes that we are interested in, thus allowing us to complete our unified map of classical and quantum complexity theory.

4.1 Tensor Formulæ and Related Definitions

Let $M_{\mathcal{S}}^{k,\ell}$ denote the set of all *matrices* of order $k \times \ell$ over a semiring $\mathcal{S}$ (Definition 2.54). The (i,j)-th entry of A is denoted by $a_{i,j}$ or $(\mathsf{A})_{i,j}$, the transpose of A by A^{t}, and its inverse, if A is an invertible square matrix, by A^{-1}.

Scalar multiplication, addition and multiplication of matrices form the basis of matrix calculus and are defined in the usual way. Scalar multiplication, addition, and multiplication of matrices over a semiring are compatible with transposition and with conjugate transposition for semirings where a conjugation operation * has been defined, i.e.

$$\begin{aligned}
(a \cdot \mathsf{A})^{\mathsf{t}} &= a \cdot \mathsf{A}^{\mathsf{t}} & \qquad (a \cdot \mathsf{A})^{\dagger} &= a^{*} \cdot \mathsf{A}^{\dagger} \\
(\mathsf{A} \cdot a)^{\mathsf{t}} &= \mathsf{A}^{\mathsf{t}} \cdot a & \qquad (\mathsf{A} \cdot a)^{\dagger} &= \mathsf{A}^{\dagger} \cdot a^{*} \\
(\mathsf{A} + \mathsf{B})^{\mathsf{t}} &= \mathsf{A}^{\mathsf{t}} + \mathsf{B}^{\mathsf{t}} & \qquad (\mathsf{A} + \mathsf{B})^{\dagger} &= \mathsf{A}^{\dagger} + \mathsf{B}^{\dagger} \\
(\mathsf{A} \cdot \mathsf{B})^{\mathsf{t}} &= \mathsf{B}^{\mathsf{t}} \cdot \mathsf{A}^{\mathsf{t}} & \qquad (\mathsf{A} \cdot \mathsf{B})^{\dagger} &= \mathsf{B}^{\dagger} \cdot \mathsf{A}^{\dagger}
\end{aligned}$$

Furthermore, if A and B are invertible square matrices having inverses A^{-1} and B^{-1}, then

$$(\mathsf{A} \cdot \mathsf{B})^{-1} = \mathsf{B}^{-1} \cdot \mathsf{A}^{-1}$$

A matrix U is *unitary* if its conjugate transpose is equal to its inverse, i.e. $\mathsf{U}^\dagger = \mathsf{U}^{-1}$. In semirings where conjugation is equal to the identity operation, i.e. where $\mathsf{U}^t = \mathsf{U}^\dagger$, it is usual to call such matrices *orthogonal.* In the remainder of this chapter, we will use the term unitary for both cases.

Additionally we consider the *tensor product* $\otimes : M_S^{k,\ell} \times M_S^{m,n} \to M_S^{km,\ell n}$ of matrices, also known as Kronecker product[HPS83] or direct product. For $\mathsf{A} \in M_S^{k,\ell}$ and $\mathsf{B} \in M_S^{m,n}$ their tensor product $\mathsf{A} \otimes \mathsf{B} \in M_S^{km,\ell n}$ is defined as follows

$$\mathsf{A} \otimes \mathsf{B} = \begin{pmatrix} a_{1,1} \cdot \mathsf{B} & \dots & a_{1,\ell} \cdot \mathsf{B} \\ \vdots & \ddots & \vdots \\ a_{k,1} \cdot \mathsf{B} & \dots & a_{k,\ell} \cdot \mathsf{B} \end{pmatrix}$$

Hence

$$(\mathsf{A} \otimes \mathsf{B})_{i,j} = (\mathsf{A})_{q,r} \cdot (\mathsf{B})_{s,t},$$

where $i = k \cdot (q-1) + s$ and $j = \ell \cdot (r-1) + t$.

The main properties of the Kronecker product of matrices are gathered in the following identities. These properties are well known[Gra81] and will be restated for reference only. They hold true over arbitrary semirings, unless otherwise stated, whenever the corresponding operations are defined:

$$\mathsf{A} \otimes (\mathsf{B} \otimes \mathsf{C}) = (\mathsf{A} \otimes \mathsf{B}) \otimes \mathsf{C} \tag{4.1}$$

$$(\mathsf{A} + \mathsf{B}) \otimes (\mathsf{C} + \mathsf{D}) = \mathsf{A} \otimes \mathsf{C} + \mathsf{A} \otimes \mathsf{D} + \mathsf{B} \otimes \mathsf{C} + \mathsf{B} \otimes \mathsf{D} \tag{4.2}$$

and, if the underlying semiring is commutative, we have

$$(\mathsf{A} \otimes \mathsf{B}) \cdot (\mathsf{C} \otimes \mathsf{D}) = (\mathsf{A} \cdot \mathsf{C}) \otimes (\mathsf{B} \cdot \mathsf{D}) \tag{4.3}$$

Moreover, for arbitrary semirings the Equation 4.3 also holds if B or C are $\{0,1\}$-valued matrices. Finally, if a is a scalar we have

$$a \otimes \mathsf{A} = a \cdot \mathsf{A} \qquad\qquad \mathsf{A} \otimes a = \mathsf{A} \cdot a$$

And for the transpose and conjugate transpose we have

$$(\mathsf{A} \otimes \mathsf{B})^{\mathsf{t}} = \mathsf{A}^{\mathsf{t}} \otimes \mathsf{B}^{\mathsf{t}} \qquad\qquad (\mathsf{A} \otimes \mathsf{B})^{\dagger} = \mathsf{A}^{\dagger} \otimes \mathsf{B}^{\dagger}$$

and if A and B are invertible square matrices having the inverses A^{-1} and B^{-1}, respectively, we have

$$(\mathsf{A} \otimes \mathsf{B})^{-1} = \mathsf{A}^{-1} \otimes \mathsf{B}^{-1}$$

As a result, note that unitary matrices are closed under multiplication, conjugate transposition and tensor product.

Definition 4.1 (Tensor Formula). The *tensor formulæ* over a semiring $\mathcal{S}$ and their *order* are recursively defined as follows:

1. Every matrix F from $M_{\mathcal{S}}^{k,\ell}$ with entries from $\mathcal{S}$ is an (*atomic*) tensor formula of order $k \times \ell$.
2. If F and G are tensor formulæ of order $k \times \ell$ and $m \times n$, respectively, then
   (a) $(F + G)$ is a tensor formula of order $k \times \ell$ if $k = m$ and $\ell = n$.
   (b) $(F \cdot G)$ is a tensor formula of order $k \times n$ if $\ell = m$.
   (c) $(F \otimes G)$ is a tensor formula of order $km \times \ell n$.
3. Nothing else is a tensor formula.

Let $T_{\mathcal{S}}$ denote the set of all tensor formulæ over $\mathcal{S}$, and define $T_{\mathcal{S}}^{k,\ell} \subseteq T_{\mathcal{S}}$ to be the set of all tensor formulæ of order $k \times \ell$.

In this chapter we will only consider semiring elements whose value can be given with a standard encoding over some finite $\mathcal{G}$. Hence, atomic tensor formulæ, i.e. matrices, can be string-encoded using list notation such as "[[001][101]]." Non-atomic tensor formulæ can be encoded over the alphabet $\Sigma = \{0\} \cup \mathcal{G} \cup \{[,], (,), \cdot, +, \otimes\}$. Strings over Σ which do not encode valid formulæ are deemed to represent the trivial tensor formula $\mathbf{0}$ of order 1×1.

Let F be a tensor formula of order $m \times n$. Its *size*, denoted $|F|$, is $\max\{m, n\}$ and its *length* $L(F)$ is the number of symbols in its string representation. It is easy to show that $|F| \leq 2^{O(L(F))}$. The upper bound is attained when F is an iterated tensor product.

Lemma 4.2. *Testing whether a string encodes a valid tensor formula and if so, computing its order, can be done in deterministic polytime.*

Proof. Let M be the Turing machine which, on an input string x, rejects and halts if the bracketing or operator structure of x are illegal. This can be tested in logspace. If w is legal, then M continues by running the function *order* described by the following pseudo-code:

```
function order (tensor $F$) : (int, int);
var $k$, $\ell$, $m$, $n$ : int;
begin case $F$ in:
          atomic:    determine order of $F$ and store it in $(k,\ell)$;
                     return $(k,\ell)$;
          $(G+H)$:   $(k,\ell) :=$ order$(G)$; $(m,n) :=$ order$(H)$;
                     if $k \neq m$ or $\ell \neq n$ then halt and reject end if;
                     return $(k,\ell)$;
          $(G \cdot H)$:   $(k,\ell) :=$ order$(G)$; $(m,n) :=$ order$(H)$;
                     if $\ell \neq m$ then halt and reject end if;
                     return $(k,n)$;
          $(G \otimes H)$:   $(k,\ell) :=$ order$(G)$; $(m,n) :=$ order$(H)$;
                     return $(k\ell, mn)$;
          end case;
end.
```

The *order* function can be implemented on M, using a tape in a pushdown like fashion to handle the recursive calls. Hence M operates in polytime, since M performs a depth-first search of the formula, and since polynomial space is sufficient to keep track of the orders in binary notation. The initial call *order*(F) thus returns the order of F. □

Definition 4.3. For each semiring $\mathcal{S}$ and each k and ℓ we define $\mathrm{val}_{\mathcal{S}}^{k,\ell} : T_{\mathcal{S}}^{k,\ell} \mapsto M_{\mathcal{S}}^{k,\ell}$, as follows:

$$\mathrm{val}_S^{k,\ell}(F) = \begin{cases} \mathsf{F} & \text{if } F \text{ is atomic} \\ \mathrm{val}_S^{k,\ell}(G) + \mathrm{val}_S^{k,\ell}(H) & \text{if } F = (G + H) \\ \mathrm{val}_S^{k,m}(G) \cdot \mathrm{val}_S^{m,\ell}(H) & \text{if } F = (G \cdot H) \text{ and } G \in T_S^{k,m} \\ \mathrm{val}_S^{k/m,\ell/n}(G) \otimes \mathrm{val}_S^{m,n}(H) & \text{if } F = (G \otimes H) \text{ and } H \in T_S^{m,n}. \end{cases}$$

That is, we associate with each tensor formula F of order $k \times \ell$ its $k \times \ell$ matrix "value" in the natural way.

4.2 From Gate Arrays to Formulæ

In principle, it is possible to associate a tensor formula to any kind of circuit, even a non-reversible one, as long as its gates implement linear operations. However, as we saw in Chapter 2 restricting the dynamics to be reversible does not change the power of computation for any of the complexity classes that we are interested in. In the quantum case, requiring reversibility is in essence equivalent to requiring linearity; in this case reversibility comes "for free." In the deterministic case, we know that any poly-size circuit can be converted into one using only reversible gates, also of poly-size. Finally, we saw that probabilistic poly-time computations can be represented with poly-size classical reversible circuits aided with an adequate number of random input wires.

On the other hand, imposing a reversible dynamics does constrain the circuit topology to be that of a gate array, and therefore simplifies somewhat the type of tensor formulæ that are associated with them. Therefore, we will restrict ourselves in the forthcoming to this special kind of circuit.

In this section, we show how to encode gate arrays into specific tensor formulæ over an appropriate semiring, and in the next, conversely how to obtain a gate array from a particular type of tensor formula F. In particular, we are interested in formulæ with the following properties.

Definition 4.4. A tensor formula F is *sum-free* if and only if none of F and its sub-formulæ has the form $(G + H)$, for tensor formulæ G and H.

A tensor formula is *array-like* if and only if all sub-formulæ of F evaluate to square matrices or column vectors.

Moreover, an array-like tensor formula F is *unitary array-like* if and only if all subformulæ of F evaluate to unitary square matrices, where "unitary" is to be interpreted as *orthogonal* when the formulæ are defined over a semi-ring without conjugation.

We choose the term "unitary array-like" because as we will show, such a formula can be reorganised as a product of a unitary matrix with a column vector, i.e. as the specification of a system of linear equations. Observe, that "sum-free array-like" implies that each sub-formula F of a tensor formula fulfils the following properties: If $F = (G \cdot H)$, then G is a matrix and either H is a matrix or a column vector, and if $F = (G \otimes H)$, either both G and H are matrices or both are column vectors.

In the forthcoming, we use the terminology that a gate array is said to be *reversible* if and only if all gates in the gate array can be described by unitary (orthogonal) matrices. Thus, both quantum and probabilistic gate arrays are reversible gate arrays, as per the discussion above.

The construction of a sum-free tensor formula from a given gate array is rather straightforward and is done as follows:

Lemma 4.5. *Let C be a (reversible) gate array operating on n wires, whose gates can be described by (unitary) square matrices over a semiring $\mathcal{S}$. Then there is a polytime computable function, which given a suitable encoding of C, computes a (unitary) array-like sum-free tensor formula F_C of order $2^n \times 2^n$ such that for each $x = (x_1, \dots, x_n) \in \{0,1\}^n$, if gate array C maps $|\psi\rangle = |x_1 \dots x_n\rangle$ to $|\phi\rangle$, then*

$$|\phi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,2^n}(F_C) \cdot |\psi\rangle,$$

and $|\psi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_x)$ for some polytime computable sum-free tensor formula d_x.

Proof. Let C be an m-levelled gate array, where C_i denotes the i-th level of C, with C_1 being the left-most and C_m the right-most level. Without loss of generality we assume that each level contains only one gate and moreover each gate acts on neighbouring wires. This can be achieved by inserting extra swap gates. In the following we describe how to construct an equivalent tensor formula F_C from C.

If level C_i contains a k-bit gate H with $1 \leq k \leq n$ acting on the wires j up to $j+k-1$, for $j + k - 1 \leq n$, then

$$F_{C_i} = \left(I_1^{\otimes j-1} \otimes H \otimes I_1^{\otimes n-j-k+1}\right)$$

is the tensor formula of order $2^n \times 2^n$ which describes the system evolution in the ith time step.

To complete the description of the sum-free tensor formula F_C over semiring $\mathcal{S}$ let

$$F_C = F_{C_m} \cdot \cdots \cdot F_{C_2} \cdot F_{C_1},$$

since according to the usual convention, the input-to-output direction in a gate array is left-to-right, while in its matrix representation, the array's action on its input is given as a product of matrices with a column vector, and is read right-to-left. It is readily verified that for each $x_i \in \{0, 1\}$ with $1 \leq i \leq n$, if C maps $|\psi\rangle = |x_1 \dots x_n\rangle$ to $|\phi\rangle$, then

$$|\phi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,2^n}(F_C) \cdot |\psi\rangle,$$

and $|\psi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_x)$ for the sum-free tensor formula

$$d_x = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle.$$

where as usual $|0\rangle = \binom{0}{1}$ and $|1\rangle = \binom{1}{0}$. Since F_C and d_x are polytime constructible from a suitable description of the gate array C and its input, the stated claim follows. □

Although Lemma 4.5 only applies to input vectors of the form $|x_1 \dots x_n\rangle$, arbitrary input vectors of the form $|\psi\rangle = \sum_{w\in\{0,1\}^n} \alpha_w |w\rangle$ are appropriately mapped to output vectors due to the linearity of gate array "semantics." Observe, that in general it is not obvious that all possible vectors $|\psi\rangle$ obey sum-free tensor formula representations; in fact, only product states and their images under unitary transformations will. Nevertheless, and without loss of generality, input vectors for probabilistic and quantum computations do obey sum-free tensor formula representations, since for a gate array on n wires with m_1 input bits and $2m_2$ ancilla bits, i.e. $n = m_1 + 2m_2$, we find that for a particular input $x = (x_1, \dots, x_{m_1}) \in \{0, 1\}^{m_1}$ the input vector can be described by

$$|\psi\rangle = \left(\bigotimes_{i=1}^{m_1} |x_i\rangle\right) \otimes \left(\bigotimes_{i=1}^{m_2} \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\right),$$

where $(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ can be explicitly given without summation. Thus, in both cases sum-free tensor formulæ exist.

Moreover, the previous lemma is not restricted to gate arrays operating on n wires carrying (qu)bits only. In fact, one can easily generalise the result of the lemma such that it work on gate arrays with multi-valued logic, in the sense that there is a mapping from $\{1, \dots n\}$ to the natural numbers, defining the "arity" of the wires. This approach is even more general than the multi-valued bit approach presented studied in the literature[MCRS00], where each wire carries (qu)dits of same dimensionality. This more general model allows us to build gates dealing with, e.g. (qu)bits and (qu)trits simultaneously in a single gate.

4.3 From Sum-Free Formulæ to Gate Arrays

In the formula to gate array part, we must deal with the fact that a sum-free tensor formula may contain matrices of various sizes and vectors at atypical locations. In principle, the latter can be regarded as a non-standard manner of specifying the gate array's input. The matrices of various orders, however, cannot be readily interpreted in terms of gate array computations. For instance, consider the sum-free tensor formula

$$(\mathsf{A} \otimes \mathsf{B})(\mathsf{B} \otimes \mathsf{A}),$$

where A is of order 2×2 and B an order 3×3 matrix. These odd-sized orders are discussed in [BFH02], but will not be discussed here; we will restrict ourselves to power of 2 orders, representing qubits and qubit operations.

Before we show how to transform a suitable tensor formula into a gate array acting on (qu)bits, we show that the postulated requirements on a given tensor formula as specified in the discussion above can be verified in deterministic polytime. We omit the proof of the following lemma, because it is quite similar to the proof of Lemma 4.2.

Lemma 4.6. *Testing whether a string encodes a valid tensor formula and if so checking*

(a) *sum-freeness,*

(b) *unitarity of sum-free formulæ,*

(c) *the array-like property, and*

(d) *whether all atomic sub-formula have orders which are powers of two*

can be all be done in deterministic polytime. □

Now we are ready to prove the converse relation, i.e. transforming an array-like sum-free tensor into an equivalent gate array, if the formula obeys some additional easily checkable properties.

Theorem 4.7. *Let F be a (unitary) array-like sum-free tensor formula of order $2^n \times 1$ over semiring $\mathcal{S}$, where the orders of all atomic atomic sub-formulæ are powers of two. Then there is a polytime computable function, which given the tensor formula F, computes a (reversible) gate array C_F over $\mathcal{S}$ operating on n wires and an input $|\psi_F\rangle$ (with $\langle\psi_F|\psi_F\rangle = 1$), such that*

$$|\phi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(F),$$

if gate array C_F maps $|\psi_F\rangle$ to vector $|\phi_F\rangle$, and $|\psi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_F)$ for some sum-free tensor formula d_F.

Proof. We prove the following more general statement, where we call a tensor formula F *closed* if F has order $2^n \times 1$, for some $n \geq 0$, and *open* if the order equals $2^n \times 2^n$, for some $n \geq 1$. Let F be a closed (or open) array-like sum-free tensor formula F over semiring $\mathcal{S}$ having only atomic sub-formulæ whose orders are powers of two. Then there is a polytime computable function, which given the tensor formula F, computes a gate array C_F over $\mathcal{S}$ operating on n wires and an input $|\psi_F\rangle$ (arbitrary, if F is open), such that

$$\begin{aligned} |\phi_F\rangle &= \mathrm{val}_{\mathcal{S}}^{2^n,1}(F) && \text{if } F \text{ is closed} \\ |\phi_F\rangle &= \mathrm{val}_{\mathcal{S}}^{2^n,1}(F) \cdot |\psi_F\rangle && \text{if } F \text{ is open} \end{aligned}$$

if gate array C_F maps $|\psi_F\rangle$ to vector $|\phi_F\rangle$, and $|\psi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_F)$ for some sum-free tensor formula d_F.

The statement is shown by induction on the (unitary) sum-free tensor formula F. If F is an atomic sub-formula, then we distinguish the cases whether F is open or closed:

1. If F is closed, i.e. is of order $2^k \times 1$, then it specifies the amplitudes for all possible combinations of values of k input bits. Thus, the trivial gate array C_F only consisting of k wires with no gates at all and the sum-free tensor formula $d_F = F$ satisfies[1]

$$|\phi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^k,1}(F),$$

[1] If working on a field $\mathcal{S}$ instead of a semiring, one can show the following result: Let $|\phi\rangle$ be a vector over the field $\mathcal{S}$ of length 2^k obeying $\langle\phi|\phi\rangle = 1$. Then there is a matrix A over an extension field of $\mathcal{S}$, whose

since C_F realizes the identity transformation on $|\psi\rangle = \mathrm{val}_{\mathcal{S}}^{2^k,1}(d_F)$; this means $|\phi\rangle$ equals $|\psi\rangle$.

2. If F is a matrix of order $2^k \times 2^k$, i.e. formula F is open, then F is interpreted as the specification of a k-bit gate F. Thus, the gate array C_F consists of the single k-bit gate F acting on k wires and the input to the gate is some 2^k-dimensional vector, for example the unit column vector $|0^k\rangle = |0\rangle^{\otimes k}$.

Now assume that the statement holds for sub-formulæ G and H of the tensor formula F. Thus, by induction hypothesis there are gate arrays C_G and C_H and inputs $|\psi_G\rangle$ and $|\psi_H\rangle$, that can be specified by sum-free tensor formulæ d_G and d_H, respectively. Then we distinguish two cases:

1. If $F = (G \cdot H)$, then we combine the sub-arrays C_H and C_G in sequential manner, where C_H is to the left of C_G, and define the input to be $|\psi_H\rangle$. It is easy to see that C_G and $|\psi_H\rangle$ fulfil the required properties.

2. If $F = (G \otimes H)$, then the sub-arrays are combined in parallel, where C_G is on top of C_H. Thus, the input equals $|\psi_G\rangle \otimes |\psi_H\rangle$, which can be described by the sum-free tensor formula $d_G \otimes d_H$. Again, the induction assertion is fulfilled.

This proves the statement. Observe, that one can easily show, that whenever F is a unitary array-like sum-free tensor formula, then all gates in the gate array C_F can be specified by unitary matrices, and moreover, the input $|\psi_F\rangle$ obeys $\langle\psi_F|\psi_F\rangle = 1$ and has a sum-free tensor description. □

The proof above reveals a significant difference between probabilistic and quantum computation—see the footnote again. In the probabilistic case, the ancilla bits must be given well prepared to the gate array, since the gate array can only perform deterministic computations and thus is not able to prepare them itself. In the quantum case, this preparation is not necessary, since the gate array itself is able to generate them properly. This means, that in the quantum case one can set all ancilla bits to, e.g. $|0\rangle$, without changing the computational power of the underlying device.

first column equals $|\phi\rangle$, which can be decomposed into an orthogonal matrix Q and an upper diagonal matrix R whose upper left element equals 1, and both matrices are over $\mathcal{S}$, satisfying $\mathsf{A} = \mathsf{Q} \cdot \mathsf{R}$. The proof relies on a careful analysis of the Gram-Schmitt[GL96] diagonalisation algorithm, which inductively computes an orthogonal (orthonormal) basis from any set of linearly independent vectors. Therefore, the orthogonal matrix Q may be interpreted as the specification of a k-bit gate Q. Thus, a gate array consisting of a single Q-gate acting on k wires maps input $|\psi\rangle = |0^k\rangle = |0\rangle^{\otimes k}$ to vector $|\phi\rangle$. Observe, that if $\mathcal{S}$ is a (semi)ring, then a similar statement as that for fields is not true in general.

4.4 Completeness Results

In this section, we use the link established in the previous two sections to find complete and promise-complete problems for key classical and quantum complexity classes. We start by defining promise-free formula-based problems and show their completeness for deterministic and non-promise complexity classes. We then introduce the necessary notions of promise problems and promise classes, and reformulate our formula problems. We will discuss our results compared with previous results in this area, and finally provide a "big picture" of the relevant complexity classes in terms of the characterising formula complete problems.

4.4.1 The Formula Sub-Trace Problem

Intuitively, most of our completeness results derive from the following central fact. Let F be a formula representing a circuit C initialised to some state $|\psi\rangle$, then the statistics of the top wire of the circuit are given by summing the elements of the first half of the diagonal of the density matrix of the output, which is represented by the formula $F \cdot F^\dagger$. Since without loss of generality, all relevant complexity classes can be described in terms of circuits examining only the top wire, then the ability to evaluate this sum is tantamount to the computing power of these classes.

Traditionally, the *trace* of an order $n \times n$ square matrix A, denoted by $\mathrm{Tr}(\mathsf{A})$, equals the sum of its diagonal elements, i.e.

$$\mathrm{Tr}(\mathsf{A}) = \sum_{i=1}^{n} (\mathsf{A})_{i,i}$$

We can generalise this notion as follows.

Definition 4.8 (Generalised Trace). For $k \geq 0$, the *k-th sub-trace* of a square $N \times N$ matrix A, for short $\mathrm{Tr}_k(\mathsf{A})$, is the sum of its first k diagonal elements, counting downwards from the upper left corner, i.e.

$$\mathrm{Tr}_k(\mathsf{A}) = \sum_{i=1}^{k} (\mathsf{A})_{i,i}$$

For completeness, if k exceeds the order N of A, then the $k-$th sub trace coincides with the trace of A. Also, we call the *half-trace* of A the sum of the elements of the upper half of its diagonal, i.e. its $n/2$-th sub-trace.

With this generalisation of trace we define the following formula problem:

Definition 4.9 (Sub-trace and Half-trace Evaluation Problems). Let $\mathcal{S}$ be a semiring. Given a tensor formula F over $\mathcal{S}$ of order $N \times 1$ and a natural number k which is a power of two and is written in binary, the *sub-trace evaluation problem* consists in determining the k-th sub-trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\dagger})$. Similarly, the *half-trace evaluation problem* consists in determining the $N/2$-th sub-trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\dagger})$

The decision problem version of these problems are the given as follows:

Definition 4.10 (One and Non-zero Sub-trace Problems). The *one sub-trace* and *non-zero sub-trace* problems over a semiring $\mathcal{S}$ are the set of 2-tuples (F, k) where F is a formula of order $N \times 1$ together with a natural number k, which is a power of two written in binary, for which the k-th subtrace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathrm{t}})$ is equal to 1, and is non-zero, respectively.

4.4.2 Promise Problems and Promise Classes

In order to obtain our completeness results we have to deal with promise versions of the above defined problems. Moreover, we also have to introduce promise complexity classes.

Consider the difference between classical classes PP and BPP. The class PP can be viewed as a "syntactic" class, in the sense that acceptance is defined by simply counting the number of accepting paths, while BPP can be viewed as a "semantic" class, since for a nondeterministic machine to define a language in BPP, it must have the property that for all inputs one of the two outcomes has a "clear majority", i.e. that its probability is significantly bigger than that of the other outcome. As we saw in chapter 2, we can also recast this difference in terms of probabilistic or randomised circuits. For such circuits to accept a language in PP, the (top) output wire must output the correct answer with probability bigger than a fixed value (not necessarily 1/2). On the other hand, for a probabilistic or randomised circuit to decide a language in BPP it must have the property that the right answer always has a clear majority, i.e. a probability significantly bigger than that of the wrong answer or, in other words, it must be bounded away from 1/2.

It is not obvious how to verify this property, for all inputs, given a description of the circuit. Thus, it is necessary to introduce the notion of *promise problems* and *promise* *complexity classes*[ESY84, Sel88].

Definition 4.11 (Promise Problem). A *promise problem* is a formulation of a partial decision problem and can be specified in the form "$R(x)$ given the promise $P(x)$?" where R and P are predicates. That is, on input x, an algorithm solving a promise problem (P, R) has to correctly decide property $R(x)$ provided that the promise $P(x)$ holds; otherwise, it can give an arbitrary answer.

More formally, a language L is said to be a *solution* to (P, R) if $x \in P$ implies that $x \in R \iff x \in L$. In particular, the set R is the unique solution to (Σ^*, R). Thus, the promise problem (Σ^*, R) is identified with the set R.

This definition allows us to extend classes with built-in promises such as BPP or BQP by defining their associated *promise classes*, as follows:

Definition 4.12 (pr$-$BPP)**.** A promise problem (Q, R) belongs to pr$-$BPP if and only if there is a polytime generated family of probabilistic circuits $\mathcal{C} = \{C_n\}$, such that if $x \in P$, we have that

i. One of the answers has clear majority over the other, i.e.

$$\left|\Pr(C_n(x) = 1) - \Pr(C_n(x) = 0)\right| \geq \frac{1}{\mathrm{poly}(n)}$$

ii. If $x \in R$, then the correct answer has clear majority, i.e.

$$\Pr(C_n(x) = 1) - \Pr(C_n(x) = 0) \geq \frac{1}{\mathrm{poly}(n)}$$

or equivalently $\Pr(C_n(x) = 1) \geq 1/2 + 1/\mathrm{poly}(n)$.

Observe, that (Σ^*, L) is in pr$-$BPP if and only if L is in BPP. Also note that we could re-formulate Definition 4.12 in terms of randomised circuits by appropriately substituting probabilities of output with frequency counts over random input choices. For the quantum case, we can similarly define the promise version of BQP denoted pr$-$BQP, by using outcome probabilities of quantum circuits.

4.4.3 Promise Versions of the Sub-trace Problem

Coming back to formula problems, in order to capture the computing power of the promise versions of the "tractable" classes BPP and BQP, and of their promise-free "lucky" equivalent PP, we introduce the following variants of the sub-trace problem.

Definition 4.13 (Majority and Clear Majority Sub-trace Problems). Let $\mathcal{S}$ be a semiring including the rational numbers $\mathbb{N}$ or the Boolean Algebra.

1. The *majority sub-trace problem* over the semiring $\mathcal{S}$ is the set of all unitary tensor formulæ F of order $n \times 1$ together with a natural number k a power of two given in binary, for which the k-th sub-trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathrm{t}})$ is superior to $\frac{1}{2}$.

2. The *clear majority sub-trace problem* is the promise version of the majority partial trace problem with the promise that the sub-trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathrm{t}})$ is in $[0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$.

where in the case of the Boolean Algebra, $\mathbb{S} = \mathbb{B}$, we appropriately extend the order relation in the natural fashion, i.e. $\mathbf{0} < x < \mathbf{1}$, for all $x \in (0,1)$, where $\mathbb{B} = \{\mathbf{0}, \mathbf{1}\}$[2].

Since we will not only be dealing with the Boolean Algebra $\mathbb{B}$, but also with other semi-rings such as the rationals $\mathbb{Q}$ and the positive rationals $\mathbb{Q}_+$, we are not guaranteed that the values of the sub-traces will be $\{0,1\}$-valued. However, some important complexity classes can be formulated in terms of the promise that these sub-traces (or more importantly the probabilities associated with them) are $\{0,1\}$-valued, such as the promise versions of the classes P and EQP, pr−P and pr−EQP, respectively. Therefore, we define the following promise problem.

Definition 4.14 (0-1 promise One Sub-trace Problem). The *0-1 promise* version of the one sub-trace problem is the promise problem (P, R), where R represents the sub-trace evaluation problem (Definition 4.9) and P is the promise predicate indicating that the sub-trace evaluates to 0 or 1, i.e.

$$P(F,k) = \begin{cases} 1 & \text{if } \mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\dagger}) \in \{0,1\} \\ 0 & \text{otherwise} \end{cases}$$

4.4.4 Completeness Results for $\mathbb{Q}_+$ and $\mathbb{Q}$

In order to define a notion of completeness on promise classes we must define the notion of reducibility of promise problems.

Definition 4.15. We say that a promise problem (Q, R) is *uniformly many-one reducible* in polytime to a promise problem (S, T), if there exists a partial polytime computable

[2] By this, and in disagreement with the other co-authors of [BFH02], we "abuse notation" and define the majority problem over the Booleans, even though $1/2 \not\in \mathbb{B}$.

function $f : \{x \in \Sigma^* \mid Q(x)\} \rightarrow \Sigma^*$, such that for every $x \in \Sigma^*$, we have the $Q(x)$ implies:

- $S(f(x))$, i.e. that the original promise is always "carried over" if it true *and*
- $R(x) \iff T(f(x))$, i.e. membership in one language is the reduced to membership in the other.

We are now ready to state the main completeness results for the variants of the sub-trace evaluation problem introduced above.

Theorem 4.16. *Let $\mathcal{F}$ be the set of all unitary, array-like, sum-free formulæ. Then, we have that*

i. *The 0-1 promise version of the one sub-trace problem over the positive rationals $\mathbb{Q}_+$ (and rationals $\mathbb{Q}$, respectively), restricted to formulæ in $\mathcal{F}$, is complete for* pr$-$P *(*pr$-$EQP, *respectively) under polytime many-one reductions.*

ii. *The non-zero sub-trace problem over the positive rationals $\mathbb{Q}_+$ (rationals $\mathbb{Q}$, respectively), restricted to formulæ in $\mathcal{F}$, is complete for* NP *(*NQP, *respectively) under polytime many-one reductions.*

Proof. We only prove the first statement, since the second can be shown by similar arguments. The hardness of the 0-1 promise one partial trace problem on unitary array-like sum-free tensor formulæ is shown by a generic reduction from pr$-$P (pr$-$EQP, respectively). By Theorems 2.63 and 2.71, we start with an m-level reversible gate array C over the positive rationals working on n wires number from 1 to n, whose accepting subspace is defined by setting the first (qu)bit to $|1\rangle$. Now using Lemma 4.5 we build from C an equivalent tensor formula F_C in polytime. Meanwhile we define for the gate array's input (qu)bits x_1 up to x_n a tensor product $d_x = \bigotimes_{i=1}^{n} |x_i\rangle$ of order 1×2^n of n unit column vectors $|x_i\rangle$ each of order 1×2. By Lemma 4.5, the first 2^{n-1} entries along the diagonal of

$$\mathrm{val}_{\mathbb{Q}_+}^{2^n,2^n}\left((F_C \cdot d_x) \cdot (F_C \cdot d_x)^{\mathrm{t}}\right)$$

add up to the probability that the gate array's first (qu)bit is equal to 0, i.e. that C accepts x. Scrutiny of the reduction shows that the constraint on this probability is transported intact from the description of C and x to the partial trace unitary array-like sum-free tensor formula problem over $\mathbb{Q}_+$ instance $F_C \cdot d_x$.

In the other direction, we use Theorem 4.7 to translate an instance $(F, 2^k)$ of the partial trace problem variant under consideration into the description of a reversible gate array C_F over m (qu)bits, where $m \leq n$, if the order of F equals $2^n \times 1$, and of its input $|\psi\rangle$; the 2^kth partial trace of

$$\mathrm{val}_{\mathbb{Q}_+}^{2^m,2^m}(F \cdot F^{\mathrm{t}})$$

represents the probability that the top m (qu)bits are equal to 0. The promise on the partial trace is transported unmodified from the input tensor formula to the reversible gate array. □

In addition, the majority problems of Definition 4.13 characterise the following complexity classes.

Theorem 4.17. *Let $\mathcal{F}$ be the set of all unitary, array-like, sum-free formulæ.*

i. *The majority sub-trace problem over both the positive rationals $\mathbb{Q}_+$ and rationals $\mathbb{Q}$ in general, restricted to formulæ in $\mathcal{F}$, is complete for* PP *under polytime many-one reductions.*

ii. *The clear majority partial trace problem over the positive rationals $\mathbb{Q}_+$ (rationals $\mathbb{Q}$, respectively), restricted to formulæ in $\mathcal{F}$, is complete for* pr−BPP *(*pr−BQP*, respectively) under polytime many-one reductions.*

Proof. We only prove the second statement. For the first statement, observe that PP equals its quantum counterpart as per Theorem 2.53.

The proof of the second assertion parallels that of Theorem 4.16. Hardness follows from Theorems 2.63 and 2.71 and Lemma 4.5, while containment is shown with Theorem 4.7, and the fact that the promise on the partial trace problem is transported unmodified from the input tensor formula to the reversible gate array. □

4.4.5 Completeness Results for the Boolean Algebra $\mathbb{B}$

Theorem 4.18. *The one partial trace, the non-zero partial trace, and the majority problem over the Boolean semiring $\mathbb{B}$, restricted to the domain of unitary array-like sum-free tensor formulæ, is complete for* P *under logspace many-one reductions.*

Proof. Consider the reversible formulation of P. Note that the diagonal of $\mathrm{val}_{\mathbb{Q}_+}^{2^m,2^m}(F \cdot F^{\mathrm{t}})$ will only contain a single 1 and zeros everywhere else. Thus, the half-trace will be 1 iff the top bit of the corresponding reversible is equal to one. It is a simple exercise to verify that in fact since all conversions from and to formulæ to circuits can be done with logarithmic space. □

In fact, what we have here is a quantum-like re-mix of an old result. One of the first problems which was identified as being P-complete was CVAL or Circuit Value problem, which was defined for non-reversible boolean circuits.

4.5 Summary of Results and Open Problems

We summarise our results on variants of the partial trace problem over the positive rationals or rationals in general in Table 4.1.

Sub-trace problem (STP) with appropriate restricted domain

Semiring	one STP with 0-1-promise	non-zero STP	majority STP	strict majority STP
$\mathbb{B}$	P	P	P	P
$\mathbb{Q}_+$	pr−P	NP	PP	pr−BPP
$\mathbb{Q}$	pr−EQP	NQP	PP	pr−BQP
$\mathbb{R}, \mathbb{C}, \mathbb{H}$	-	NQP	PP	pr−BQP

Table 4.1: Completeness results summarised.

Part II

De computatione per res quanticas

Fortunately, Quantum Computation is not just a theory. Several techniques have been proposed and tested in the laboratory, with which limited scale quantum computing experiments have been successfully made reality. Despite the fact that it would have been possible to easily simulate on a classical computer any of the quantum computing experiments implemented to date (the current record is 7 qubits), these experiments and techniques have interest on their own.

First, because of being precisely what they are, an opportunity *to compute with things quantum*. Beyond the speedups promised by quantum theory of computation, there is a need and a natural curiousity for doing and understanding how to build and manipulate these quantum things. Second, they might indeed represent the first step in the long road to universal, scalable, and usable quantum computers. Whatever the ultimate application and motivation behind manipulating and understanding the quantum world, from an engineering point of view these experiments provide a good vehicle for developping and refining methodologies of construction, control, noise reduction and removal, etc. Finally, the outcome and problems encountered in performing real experiments can also provide feedback into the theories that inspired them. In the case of Quantum Computing, one of the most succesful techniques has been that based on Nuclear Magnetic Resonance. We have studied NMR QC and even performed some experiments, which has provided insight in all of the above three facets, as we will see in this second part.

NMR-based Quantum Computing (NMR QC) has allowed teams of researchers worldwide to perform computing experiments involving up to 7-qubits, by using essentially commercial spectrometers with small liquid-state chemical samples at room temperature. While the widespread availability of NMR spectrometers makes these experiments relatively easy to reproduce and work on, these technique has severe limitations, both of a technological and fundamental nature. Most of these difficulties can be dealt with in small-scale experiments, but these techniques cannot be used with larger scale experiments involving more qubits. This has been described as the *NMR QC scalability problem*. In fact, there is not only *one* scalability problem, but several, some more immediate and severe than others.

Among the scalability problems, two of them have particular importance and interest: register initialisation and final measurement. The type of measurement that can be made in NMR QC experiments differ fundamentally from the idealised projective measurement of Quantum Computing Theory. Likewise, the natural initial states of the quantum memory registers in an NMR QC experiment are much different from those that are assumed to be available and are used in standard Quantum Computing models and algorithms.

In this part, we cover the topic of *Algorithmic Cooling* (AC), which is a technique that can be used to solve the initialisation problem in certain cases. The technique is defined and introduced in Chapter 5, where we describe the various theoretical approaches for this technique, including their limitations and strengths. In Chapter 6 we discuss the experimental considerations of AC in NMR, including an analysis of its usefulness in realistic settings and a high level description of the AC experiments that we have performed.

Credits and Acknowledgements

The results described on Algorithmic Cooling described in this part are the fruit of interaction with many different people. In this Alexandre Mikaelian, Raymond Laflamme and Tal Mor deserve special recognition.

Sasha Mikaelian, a co-disciple here at the Université de Montréal, was an early travelling companion in the exploration of the mysteries behind this apparently "magic" invention. He very quickly understood it (and explained to the author) and was able to provide bounds for the heterogeneous cases. Unfortunately, the proof of his results on the generic n-qubit case has been lost but we have reproduced at least one of them here (the 3-qubit case) based on the techniques he developed and showed the author.

The contribution of Raymond Laflamme (now at the University of Waterloo and Perimeter Institute) has been paramount in making us aware of how important this problem was, and how its implications went far beyond Quantum Computing. In particular, he pointed out the work of Sørensen, thus preventing us from reinventing the proverbial wheel. He was key in helping us develop the tools with which to analytically and numerically verify the design and resilience of pulse sequences. Finally, Raymond is responsible for teaching us "the tricks of the trade" of NMR QC, while the author was visiting the Los Alamos National Laboratory.

However, the common denominator throughout has been Tal Mor, now at the Technion in Israel, which has participated in both the theoretical and experimental aspects of this work on Algorithmic Cooling. The non-adiabatic cooling algorithm described at the end of Chapter 5 is joint work with Seth Lloyd from MIT, Vwani Rowchoudhury from UCLA and him.

In addition, Tal Mor and his student Yossi Weinstein have been the driving force [3] in the development and final success of the experiment described in Chapter 6. The findings of this experiment have been described in a joint article with Gilles Brassard, Raymond Laflamme, Tal Mor and Yossi Weinstein [BFLW03]. Since then, Irina Tsyrinsky has been working with us in trying to simulate and understand the possible cause of errors observed in the quest to find a way to overcome the initial marginal results that we obtained.

Since none of us, save Raymond Laflamme, had had experience with NMR spectroscopy before, the help of many other people had to be recruited (willingly or not). Of these, the author wishes to thank particulary Sylvie Bilodeau and Tan Pham Viet, technician at the NMR facilities of the Chemistry Department of the Université de Montreal, Nicolas Boulant (MIT) and Camille Negrevergne (LANL) who both visited us in Montréal and provided helpful advice, and finally Serge Lacelle (Université de Sherbrooke) who also provided advice and guidance, and in whose labs this experiment would have happened first, if it was not for he fact that his lab burned, twice...

[3] Translation: the boot in the behind.

CHAPTER 5

ALGORITHMIC COOLING

"[In NMR] polarisation is like gold..." –*Raymond Laflamme*

5.1 The Quest for Stronger Polarisation

One of the main limitations of NMR spectroscopy, whether it is used for imagery, chemical or biochemical analysis, or Quantum Computing, is that the differences in spin energy levels are very small compared to energy variations due to thermal fluctuation at room temperature. Thus, even when placed within a strong magnetic field the proportion of nuclei from a macroscopic sample whose spin will be aligned with the field, i.e. the *polarisation bias* or simply *polarisation* of the sample, is very small. As a result, the "usable" portion of the sample is proportionately very small and hence the detectable signal that it generates is not very strong either.

Since in order to be able to perform useful tasks, in any of these applications, the signal strength obtained from the sample must be larger than the background noise, this problem is often referred to as the *Signal-to-Noise ratio problem.* The quest for a method to improve polarisation and hence signal strength has been the Holy Grail of NMR spectrometers since its infancy.

In this section, we start by introducing the problem and its physical causes. We continue by reviewing some of the traditional approaches to this problem within the field of NMR. We then quickly describe some of the techniques that have been developed within the context of Quantum Computing to address this problem, which have been described under the generic term *Algorithmic Cooling* (AC). In Section 5.2 we introduce some of the mathematical tools behind (AC) by analysing in detail the simpler cases. We then discuss AC as a scalable technique for improving bias in the context of Quantum Computing and will explain its limitations in Section 5.3. Finally, in Section 5.4 we introduce a new Algorithmic Cooling technique, which makes use of the thermodynamical environment (the "heat bath") in order to break past the limits of Adiabatic Cooling of spin.

5.1.1 Thermal Equilibrium State

The equilibrium bulk spin density matrix of an NMR sample at rooom temperature ξ_0, this is, the steady state solution of the corresponding Schrodinger Equation is called the *thermal state* and is given by

$$\xi_0 = \frac{1}{Z} \exp\left(\frac{-\hbar \hat{H}}{kT}\right) \tag{5.1}$$

where k is the Boltzmann constant, T is the sample temperature, and Z is the Zeeman partition function

$$Z = \mathrm{Tr}(\exp\left(\frac{-\hbar \hat{H}}{kT}\right))$$

which normalises the density matrix ξ_0. If we choose the natural basis of the eigenvectors of the corresponding Hamiltonian, i.e. the good old computational basis, then the diagonal of ξ_0 represents the probability distributions of those eigenvectors at thermal equilibrium.

$$P_b = \langle b|\xi_0|b\rangle = \frac{1}{Z} \exp\left(\frac{-\hbar E_b}{kT}\right) \tag{5.2}$$

Let us consider first the distribution of the population of M molecules with respect to one of the spins, without loss of generality the first one. The probability that in an individual molecule the a given spin is aligned with the magnetic field, i.e. that it is equal to $|z^+\rangle$ or $|0\rangle$, is given by P_0

$$P_0 = \sum_{|b\rangle} P_b \quad \text{s.t. } |b\rangle = |0b_2 \dots b_n\rangle$$

Similarly we define P_1, and we have that $P_0 + P_1 = 1$. Let us define the *polarisation bias* for that spin ε as[1]

$$\varepsilon \triangleq P_0 - P_1 \quad \text{i.e. } P_0 = \frac{1}{2} + \frac{\varepsilon}{2} \tag{5.3}$$

The reduced density matrix for the state of this qubit is diagonal and given by

$$\rho_\varepsilon = \begin{pmatrix} \frac{1}{2} + \frac{\varepsilon}{2} & 0 \\ 0 & \frac{1}{2} - \frac{\varepsilon}{2} \end{pmatrix} \tag{5.4}$$

Since the signal detected by the coils is the sum of each individual molecule's signal,

[1] In some texts, the bias is instead defined as $\varepsilon = (P_0 - P_1)/2$.

and that parallel and anti-parallel molecules will cancel each other's signal, the overall amplitude A of the signal component for that spin is

$$A = M(P_1 - P_0) = M\varepsilon$$

It can be shown that when the spin energies are small compared with kT (e.g. at room temperature), the polarisation bias can be approximated to

$$\varepsilon \approx \frac{\Delta E}{kT} = \frac{\hbar\omega}{kT} \tag{5.5}$$

where ω is the resonance frequency of that spin. With current technology (field strengths in the order of 10 Tesla) and at room temperature, the polarisation bias is pauper, in the order of 10^{-5}.

The situation gets much worse when we consider all n spins simultaneously. Let ε_1 be the polarisation bias for the first spin. Suppose then that we could magically "eliminate" from the sample all $M(1 - \varepsilon_1)$ molecules which do not contribute to the overall signal strength for the first spin (because they cancel each other). Consider the second spin of the $M\varepsilon_1$ molecules left: only roughly $M\varepsilon_1\varepsilon_2$ of them would contribute to the signal for the second spin, where ε_2 is the polarisation bias for the second spin. It is easy to show that for small ε_i (which is the case) the overall amplitude A of a coherent signal including all n modes corresponding to all n spins would be

$$A \approx M \prod_{i=1}^{n} \varepsilon_i$$

Since all ε_i depend on the corresponding γ_i which are all within the same order magnitude, it is easy to see that A will decrease exponentially as we increase n. Unfortunately, this decrease in signal strength cannot be offset indefinitely by signal amplification, as the noise amplitude does remain constant.

This unfortunate fact has been known for long by NMR spectroscopists, and has been their main source of headaches. NMR spectroscopists have been trying for years to find ways to boost it, transfer it, do away with it, etc. giving rise to a plethora of spectroscopy techniques with funny acronyms (see [Jon00] for a small list of them).

In terms of Quantum Computing, this phenomenon also has direct implications on the problem of selecting a particular n-qubit pure state from within such a bulk sample:

the proportion of molecules that would adopt the desired configuration would decrease exponentially as a function of n if the amplitude of the selecting pulses (i.e. that of the rotating field) remains constant. This is the essence of the initialisation scalability problem in NMR-based Quantum Computing.

5.1.2 Engineering and Experimental Techniques

The Electrical Engineering approach to this problem is simple: amplify the signal as much as you can (without distortion), and design all electronics in order to protect it from external noise and minimise internal noise. However, technology has its limits and current NMR spectrometers already use leading-edge microwave electronics. Not much more can be done with current technology to reduce noise and amplify the signal any further[2].

5.1.3 Thermodynamical or "Real" Cooling

A simple look at the formula for bias in Equation 5.5 already suggests some ways in which we can improve it. First, the bias is directly proportional to the strength of the field, so all we need is a bigger magnet. However, this avenue has been pretty much walked all of its length. To give an idea of what we mean, a Commercial-Off-The-Shelf (COTS) spectrometer for which the ^{1}H nucleus spin at 400 or 500 MHz, which is suitable for most types of inorganic and simple organic chemical analysis costs in the order of 1/2 million \$, while a top of the line 900 MHz spectrometer costs several million \$[3].

The other non-constant factor in Equation 5.5 is the ambient temperature T. It is inversely proportional to the bias, and in principle lowering it will increase polarisation. By lowering the ambient temperature, we are in fact reducing the energy values for all of the other non-spin degrees of freedom of the sample molecules, including kinetic energy (linear and angular momentum), intra- and inter-molecular potentials, etc. Because, spin energies only depend on field strength and gyromagnetic strengths, which are in principle unaffected by temperature, the ratio of spin energy to thermal energy will increase and so will the bias.

However, in order to achieve significant polarisation increases, say of an order of

[2] As a rule of thumb, already a bit less than half the cost of a Commercial-Off-The-Shelf spectrometer comes from its electronics, the other half being mostly the cost of its cryogenic superconducting magnet.

[3] It it is fact possible to build much stronger magnets, however these magnets, in addition to their astronomical cost, have the disadvantage they can only be used once...

magnitude, it would be necessary to lower the temperature to close to 30 K, approximately. While this would not be terribly hard to do from a technological point of view (the magnet of a typical spectrometer is already being cooled by liquid Helium which is itself cooled by liquid Nitrogen), at those temperatures the sample would probably not be liquid anymore. Thus, a lot of the simplifications used in liquid-state NMR, in particular the overall isotropy of the spin and the spin-spin interactions would no longer hold. There is, however, no fundamental reason indicating that the more complex mathematical treatment required to describe spin and spin dynamics in this case could not be harnessed to provide a suitable framework for Quantum Computing. This is in fact the subject of current research in solid-state NMR and other NMR-based approaches that have been proposed for experimental QC.

5.1.4 Polarisation Transfer

But suppose that somehow we knew we had an "algorithmic method" for manipulating populations of spins, for example by selectively flipping the spins which were anti-parallel to the field direction. This would result, by definition, in a higher polarisation bias ε' and also result in an increased signal strength. While this operation would not result in a decrease of ambient temperature, if we look again at Equation 5.5 this would result in a new "virtual" temperature T', also called *spin temperature*, as follows

$$\varepsilon = \frac{\Delta E}{kT} \text{ and } \varepsilon' = \frac{\Delta E}{kT'} \implies T' = T\frac{\varepsilon}{\varepsilon'} \tag{5.6}$$

Thus an increase in bias will result in a decrease in spin temperature, which is why the generic term *Algorithmic Cooling* as been coined recently within the NMR QC community for such polarisation increase techniques.

However, while this term is relatively new, the problem of increasing polarisation itself did not appear with the advent of QC. The oldest and simplest method for increasing polarisation, which is commonly used in every day NMR spectroscopy is the Insensitive Nuclei Enhanced by Polarisation Transfer (INEPT). This technique was first proposed and implemented in 1979 [MF79, SE83], and is commonly used for heteronuclear polarisation transfer from the ^{1}H to the ^{13}C, in order to take advantage of the higher natural polarisation of ^{1}H, approximately four times higher than that of the ^{13}C) isotope. From an algorithmic point of view, INEPT has a vague resemblance to a simple `SWAP` gate[4],

[4] In fact, the initial part of an NMR implementation of a perfect SWAP gate is almost identical to the

where the polarisation of the more highly polarised nucleus is transfered to that of the less polarised one. While its simplicity have made it a routine technique used by NMR spectroscopists, its main disadvantages lie in the fact that the polarisation gain is limited by the ratios of the gyromagnetic constants —i.e. the polarisation can only be improved by a factor of 4 in the case of a ^{1}H to ^{13}C transfer— and in the fact that it cannot be used in homonuclear systems.

From a theoretical point of view, the work of Sørensen at the end of late 80's [Sør89, Sør90] provided a sound mathematical framework in which to accomplish two significant achievements in the characterisation of possible polarisation transfer experiments, namely

1. To provide an absolute upper bound on how much transfer could be achieved.

2. Characterise how such optimal transfered could be achieved.

In particular, in the homonuclear scenario where all nuclei of interest have the same initial polarisation ε, Sørensen's work provided upper bounds for how much the bias of a single qubit could be. While Sørensen had already performed in 1989 compression algorithms polarisation transfer from 3 ^{1}H onto a single ^{13}C, as far as we are aware, the first completely homonuclear polarisation transfer was performed at IBM Almaden in 2001 [CVS01], for the case $n = 3$. These experiments will be discussed in more details in Chapter 6.

5.1.5 Entropy, Data Compression and Molecular "Heat Engines"

It is also possible to view polarisation transfer as an essentially information processing operation. For that, let us consider the Shannon entropy of a random variable X over a domain $\mathcal{V} = \{v_1, \ldots, v_N\}$, distributed according to probabilities $p_i = \Pr(\mathsf{X} = v_i)$ which is defined as

$$H(\mathsf{X}) \triangleq \sum_i p_i \log \frac{1}{p_i} \tag{5.7}$$

In particular, consider the entropy of a logical qubit defined by the ensemble of alike nuclei in a sample[5]. For the initial thermal state, if we consider the random variable associated with measuring spins in the direction of the field, the entropy can be expressed

INEPT sequence.

[5] The correct quantum equivalent for generic qubit states is the *von Neumann Entropy*, which coincides with the Shannon entropy for the thermal states.

as a function of bias as follows

$$\begin{aligned}H(\varepsilon) &= \frac{1+\varepsilon}{2}\log\left(\frac{1+\varepsilon}{2}\right) + \frac{1-\varepsilon}{2}\log\left(\frac{1-\varepsilon}{2}\right)\\ &= 1 - \frac{\varepsilon^2}{2\,\mathrm{Ln}\,2} - \frac{\varepsilon^4}{12\,\mathrm{Ln}\,2} - \cdots - \frac{\varepsilon^{2n}}{2n(2n-1)\,\mathrm{Ln}\,2} - \cdots\\ &= 1 - \frac{\varepsilon^2}{2\,\mathrm{Ln}\,2} - O(\varepsilon^4) \qquad (5.8)\end{aligned}$$

Entropy as a function of bias is a monotonically decreasing function. Thus, an increase in polarisation of one of the spins can be seen as reduction of entropy, i.e the amount of uncertainty about the state of the corresponding qubit.

The celebrated work of Shannon in classical Information Theory [Sha48] establishes a link between entropy of an information source (or the random variable associated with it) and the ability to compress its output. Without loss of generality, consider a memoryless information source which outputs bits one at a time, each independently distributed according to some random variable X with domain $\{0,1\}$. A compression scheme can be described as a pair of probabilistic procedures E and D (e.g. described as probabilistic TM's or circuits, such as those introduced in Chapter 2). The first procedure E on input a string of n bits of information produced by the information source X will "encode" it by producing a "compressed" string of $c(n) < n$ bits. The decoding procedure D will then "decode" the encoded string producing and n-bit output which should be (with reasonable probability) equal to the original string. The link between entropy and this compression rate $c(n)$ was established by Shannon in terms similar to the above:

Theorem 5.1 (Noiseless Coding Theorem). [6] *For any $\delta, \epsilon > 0$, and sufficiently large n, there exists a compression scheme with $c(n) \geq n(H(\mathsf{X}) + \delta)$ where D will produce the original output of X with probability greater than $1 - \epsilon$.*

Moreover, no compression scheme exists with $c(n) \leq n(H(\mathsf{X}) - \delta)$ such that the original output is reconstructed with probability greater than ϵ.

The quantum analogue of this theorem was discovered by Schumacher [Sch95], and is very similar except for two significant differences

1. We must consider that in this case what the quantum information source sends are qubits and not of classical bits.

[6] Also referred to as Shannon's "First Theorem" in some Information Theory textbooks.

2. We allow for the compression scheme to be a quantum process, in which the encoding is a "string" of qubits.

3. Instead of the Shannon entropy, one must consider the *von Neumann Entropy* defined on arbitrary density operators as

$$S(\rho) = \mathrm{Tr}(\rho \log \rho) \tag{5.9}$$

where the $\log(\cdot)$ function on linear operator is defined in terms of the Taylor series expansion of the $\log_2 : \mathbb{R}^+ \mapsto \mathbb{R}$ function on the reals, which converges for all positive linear operators (and in particular density operators).

In essence, both the classical and quantum versions of this theorem tell us that it is possible to (almost) attain a compression rate of $nH(\mathsf{X})$ for n (qu)bits, but also that we cannot do any better. Cleve and DiVincenzo went on to show [CD96] that it is possible to construct poly-size (classical) reversible circuits, that correctly perform the required encoding and decoding on strings of n qubits, with exponentially decreasing error probabilities. Their algorithm is essentially classical in nature (it uses only Toffoli gates), but they also show that it is well suited for the quantum case of encoding a string of n qubits.

In both cases (classical and quantum), the top $c(n)$ wires of the encoding circuit E will contain the encoded output, with the other $n - c(n)$ wires being in some "uninteresting" garbage state. The decoding circuit D, when taking as input the encoded $c(n)$ wires, and —this is very important— having the other $n - c(n)$ ancilla wires set to a fixed state $|0^{n-c(n)}\rangle$, will return on its n output wires the original input to E with probability exponentially close to 1, as long as the $c(n)/n$ is appropriately bounded by the Shannon or von Neumann entropy of the input wires. Both circuits require an additional number of ancillary qubits which are set to $|0\rangle$ and returned to that state at the end of the computation; at most $n + O(1)$ such ancillæ are required, or only $\sqrt{n} + O(1)$ if we are willing to sacrifice circuit depth.

Now here comes the link with bias increase and spin cooling. Because the thermal state is essentially classical (i.e. the corresponding density matrix is completely diagonal), let us speak in classical terms without loss of generality. Suppose that the n input wires to the encoding circuit E were distributed with bias ε and hence had each entropy $H(\varepsilon)$, and furthermore that the pair E and D was optimal in the sense of the above theorem, i.e that $c(n) \approx nH(\varepsilon)$. After compression, we expect the entropy of the $c(n)$ compressed bits to be very high, because otherwise some further compression would be possible under the same theorem, and thus the compression scheme would not be optimal. On the other hand, the entropy of the remaining $n - c(n)$ cannot be very high because if it were the error probability of the decoding circuit D, which replaces them with 0's would be high. Thus, as a result of the compression done by E we obtain as a "byproduct" $n(1 - H(\varepsilon))$ bits which with high probability equal to 0, in other words highly biased or "cool" bits.

We can thus see how reversible data compression, classical or quantum, is a type of algorithmic cooling procedure. Its main disadvantage, however, is that in order to cool these $n(1 - H(\varepsilon))$ bits, we need an extra $\Theta(\sqrt{n})$ bits which must be perfectly polarised with bias $\varepsilon = 1$ to start with.

Schulman and Vazirani addressed that issue in [SV99] by providing a poly-size cooling algorithm which works *in-place*, with no requirement for extra qubits. Their procedure allows cooling of approximately $1/20\varepsilon^2 n$ qubits to within $1 - n^{-10}$ probability of being 0, i.e with bias $1 - 2n^{-10}$ [7]. While not arbitrarily close to 1, as in the case of Schumacher's data compression, this polynomial bound is sufficient if the bits cooled to that bias are used as the initial register of a poly-size quantum computation; the final probability of error of the overall computation will still be be polynomially bounded, and thus it can be reduces to any arbitrary fixed value.

From a historical perspective, their result is fundamental because it provided a glimpse of hope to the NMR QC community: here was the first fully scalable polarisation transfer algorithm, suitable for implementation in a NMR spectrometer, i.e. requiring no perfectly biased ancillary qubits. However, it only offered a *glimpse* of hope. At current bias levels of $\varepsilon = 10^{-5}$, approximately 200 billion qubits would be required to purify a single spin to the level of polarisation advertised...

To make things worse, it turns out that one cannot do much better. Reversible compression (classical or quantum) is essentially "adiabatic", or in information-theoretic terms it must preserve the total entropy of the system. Let $d(n)$ be the number of perfectly biased or (almost) entirely biased bits produced by any such compression procedure (including [SV99]), i.e. $d(n) = n - c(n)$. In general we will, have

$$nH(\varepsilon) = H_{\text{initial}} = H_{\text{final}} \geq c(n) \cdot 1 + (n - c(n)) \cdot 0 \qquad (5.10)$$

[7] It is important to note that a polynomially close to 1 bias is sufficient, as long as we will be using quantum fault-tolerant error correcting codes.

having only equality in the (theoretical) optimal compression scenario, in which *all* of the entropy is *pumped out* of the bottom $d(n)$ wires onto the top $c(n)$ wires. Thus, we have an upper bound on the maximum number of perfectly biased bits that such a procedure can produce

Theorem 5.2 (Shannon Bound). *The number of perfectly biased bits $d(n)$ produced by any reversible compression scheme acting on a register of n bits, each in the thermal state with bias ε is bounded by*

$$d(n) \leq n(1 - H(\varepsilon)) = n\frac{\varepsilon^2}{2\,\mathrm{Ln}\,2} + O(\varepsilon^4) \tag{5.11}$$

In particular, we cannot expect to improve the yield of the algorithm of Schulman-Vazirani by more than a factor of $10/\,\mathrm{Ln}\,2 \approx 14.4$.

5.1.6 Non-Adiabatic Cooling

Another significant breakthrough was necessary in order to bring back the rosiness to the cheeks of NMR QC experimentalists. Boykin, Mor, Roychowdhury, Vatan, and Vrijen (BMRVV) [BMR+02] raised to the challenge by importing into algorithmic cooling something a simple invention more than 200 years old: the *radiator*. Indeed, as any mechanical engineer could have pointed out, if you have a "heat" problem, just pump it *out* of your system.

The idea is simple. Once the "top" bits, as it were, have been compressed, their spin temperature is almost infinite, and in particular higher than the surrounding "environment" or lattice. This environment, commonly called the *heat bath* in Thermodynamics, will be in fact *colder* than the compressed bits. The idea then is to pump out the excess entropy towards the heat bath, naturally returning the overheated bits to their initial thermal state. The compression procedure can be then repeated to further increase the bias of previously cooled bits, the overheated bits re-cooled by contact with the environment, and so on.

This new kind of algorithmic cooling in thus intrinsically *non-adiabatic* with respect to the n-bit register that we are concerned with. The operation of obliviously resetting the bits to their initial temperature is inherently non-reversible (non-unitary). In particular, if we consider the entropy of the register alone, it will *decrease*[8], and consequently the

[8] However, cosmologists and quantum physicists need not despair as, in the grander scheme of things,

Shannon Bound of Equation 5.11 no longer applies (yeah!). As long as we don't run out of heat bath (which is really acting as a "cold bath") we can keep cooling our spins...

To be strict from a historic perspective, the term *Algorithmic Cooling* was coined by Boykin et al. in [BMR+02] referring to this technique combining compression and relaxation with the environment. However we have adopted here the now more common usage of using "algorithmic cooling" more generically and dubbing the latter technique "non-adiabatic" AC. We will discuss this technique more in detail in Section 5.4.

However, in order to facilitate that discussion we believe it useful to describe the key techniques and limitations of adiabatic algorithmic cooling (i.e. compression without exchange of heat with environment) in more detail, as these will be the building blocks for the construction and analysis of more complex cooling algorithms.

5.2 Basic Building Blocks

5.2.1 The Basic Compression Subroutine and Its Variants

The first building block of cooling algorithm design is the case where we start with two bits with equal bias. Let p_{b_1,b_2} denote the probability that these bits have value b_1 and b_2. We can consider two cases

1. If both bits had the same value, i.e. $b_1 = b_2$ then the probability for any one of them to be 0 would be

$$\begin{aligned}\frac{p_{00}}{p_{00}+p_{11}} &= \frac{(\frac{1}{2}+\frac{\varepsilon}{2})(\frac{1}{2}+\frac{\varepsilon}{2})}{(\frac{1}{2}+\frac{\varepsilon}{2})(\frac{1}{2}+\frac{\varepsilon}{2})+(\frac{1}{2}-\frac{\varepsilon}{2})(\frac{1}{2}-\frac{\varepsilon}{2})}\\ &= \frac{1+2\varepsilon+\varepsilon^2}{2(1+\varepsilon^2)}\\ &\approx \frac{1}{2}+\frac{2\varepsilon}{2} \qquad \text{for } \varepsilon \ll 1 \qquad (5.12)\end{aligned}$$

thus resulting in an "virtual" approximately two-fold increase of bias.

2. If both bits are different, however, the probability of either of them being 0 is given

the Second Law of Thermodynamics still holds: the operations we perform on the combined holistic system of register and heat bath combined are still unitary.

by

$$\frac{p_{01}}{p_{01}+p_{10}} = \frac{p_{10}}{p_{01}+p_{10}} = \frac{(\frac{1}{2}+\frac{\varepsilon}{2})(\frac{1}{2}-\frac{\varepsilon}{2})}{(\frac{1}{2}+\frac{\varepsilon}{2})(\frac{1}{2}-\frac{\varepsilon}{2})+(\frac{1}{2}-\frac{\varepsilon}{2})(\frac{1}{2}+\frac{\varepsilon}{2})}$$

$$= \frac{1-\varepsilon^2}{2(1-\varepsilon^2)} = \frac{1}{2} \quad (5.13)$$

$$(5.14)$$

and in this second case we got "unlucky" having obtaining a final bias of.

The trick behind this first compression building block is to try to "force our luck" by making sure that somehow we select the cases where both bits have the same value. The simplest possible way of doing this is to perform a CNOT on both bits. If the target bit is equal to 0, then the control bit will have a doubled bias 2ε, as in Equation (5.12). One could measure the second bit: if it is 0 we retain the top bit as a having being "cooled" and discard it if we obtain a 1, because its bias would be 0, as per Equation (5.14). Unfortunately, measuring qubits is not an option in the NMR setting, because of the unavailability of projective measurements.[9]

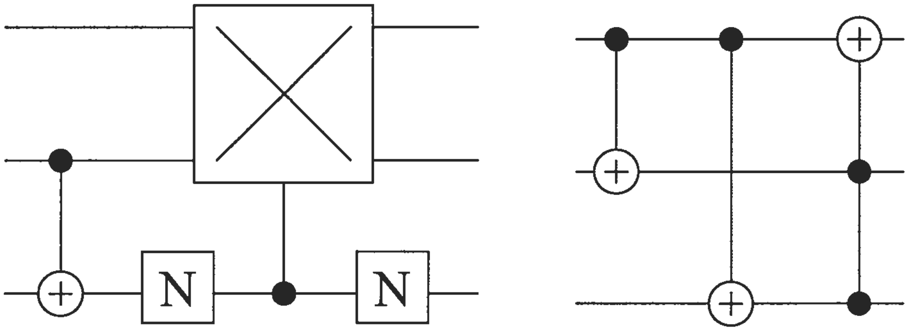
Both the adiabatic Schulman-Vazirani algorithm [SV99] and the non-adiabatic BMRVV algorithm [BMR+02] use this two-qubit technique repeatedly. They both deal with the "unlucky" case by using the Law of Large Numbers to deduce that a significant subset of the control bits must have increased their bias. This makes the analysis a bit complicated and makes the algorithm themselves quite inefficient.

A simple but crucial improvement to the above technique is the following. Suppose that we had a third bit, on top of the first two, also with initial bias ε. If by measuring the target bit we obtained a 1, we could replace the original target bit with this new bit; this can be done with a control swap operation, for example. As a result, we have that the target bit will have a new average bias of

$$\begin{aligned} \bar{\varepsilon} &\approx (p_{00}+p_{11})\,2\varepsilon + (p_{01}+p_{11})\,\varepsilon \\ &= \frac{1}{2}(1+\varepsilon^2)\,2\varepsilon + \frac{1}{2}(1-\varepsilon^2)\,\varepsilon \\ &\approx \frac{3}{2}\varepsilon \end{aligned} \quad (5.15)$$

[9] In fact, if we had projective measurements in NMR the whole issue of initialisation would disappear, as these measurements would provide us with the perfect purification and hence cooling technique.

Thus, we have achieved a small but significant improvement in polarisation of the target bit. This technique is the basis for the construction of the *Basic Compression Subroutine* or BCS[10], which will be the basic building block for the design of cooling algorithms. The circuit for the BCS is shown in Figure 5.1(c). There are other 3-qubit circuits achieving the same result, such as the *Alternate Compression Subroutine* or ACS, a slightly simpler version of BCS suggested by Mikaelian [Mik00], and depicted in Figure 5.1(d).



(c) Basic Compression Subroutine (BCS)

(d) Alternate Compression Subroutine (ACS)

Figure 5.1: Basic building blocks for the design of AC algorithms.

5.2.2 Optimality of the BCS

The BCS, and algorithmic cooling in general, is best analysed in terms of density matrices. In the homonuclear case, where all spins have the same initial bias ε, the initial density matrix ρ_0 is given by a tensor powed of the 1-qubit density matrix ρ_ε given in Equation 5.4. Since ρ_ε is diagonal, so will ρ_0, with its diagonal given by

$$\mathrm{Diag}(\rho_0) = \mathrm{Diag}(\rho_\varepsilon^{\otimes 3}) = \frac{1}{8}\left(\begin{array}{l} (1+\varepsilon)^3 \\ (1+\varepsilon)^2(1-\varepsilon) \\ (1+\varepsilon)^2(1-\varepsilon) \\ (1+\varepsilon)\ (1-\varepsilon)^2 \\ \hline\hline (1+\varepsilon)^2(1-\varepsilon) \\ (1+\varepsilon)\ (1-\varepsilon)^2 \\ (1+\varepsilon)\ (1-\varepsilon)^2 \\ (1-\varepsilon)^3 \end{array}\right) \tag{5.16}$$

[10] Again, to be "historically correct" the term BCS was coined by BMRVV [BMR+02] to denote the 2-qubit CNOT-based technique described above. However, the 3-qubit version was so much simpler that the name stuck to it and is now commonly used.

In general, the probability that the first (or top) bit is 0 is given by the sum of the first half of the diagonal (the top *half-trace*, as we called it in Chapter 4). Because, the initial density matrix ρ_ε is diagonal, the most we can hope to do in order to maximise the sum of the first four elements of the image density matrix is to permute the elements of the diagonal of ρ_ε. By simple examination of Equation 5.16, we can see that the only term in the lower half-diagonal which is bigger than a term in the upper half-diagonal is the 5-th term, corresponding to the base vector $|100\rangle$. This suggests, another "direct" compression subroutine, the DCS, which permutes the base vectors $|011\rangle$ and $|100\rangle$ and leaves all others unchanged. The probability that the top bit is 0 after such an operation would be:

$$\begin{aligned} p_{0xx} &= \frac{1}{8}\left((1+\varepsilon)^3 + 3(1+\varepsilon)^2(1-\varepsilon)\right) \\ &= \frac{1}{2} + \frac{1}{2}\left(\frac{3}{2}\varepsilon - \frac{1}{2}\varepsilon^3\right) \end{aligned} \tag{5.17}$$

Thus, we obtain that the top qubit would now have a new bias

$$\varepsilon' = \frac{3}{2}\varepsilon - \frac{1}{2}\varepsilon^3 \tag{5.18}$$

which is always improved if $0 < \varepsilon < 1$. Even though they are different operations, their effect in terms of compression of the top qubit is the same in this homonuclear case, because the half-trace of the resulting density matrix is the same in all three cases. Furthermore, they constitute in this case the optimal compression subroutine, because no other subset of four elements of the diagonal of the initial thermal state density matrix ρ_0 has a bigger sum.

This is somewhat surprising from an information theoretic point of view. The Shannon bound (Equation 5.11) seems to allow for a higher rate of entropy transfer. Indeed if we solve the entropy conservation equation in the optimal case

$$3H(\varepsilon) = H(\varepsilon_{\max}) + 2H(0) \tag{5.19}$$

we obtain that $\varepsilon_{\max} \approx \sqrt{3}\,\varepsilon$, for $\varepsilon \ll 1$, which is slightly bigger than the $\varepsilon' \approx \frac{3}{2}\varepsilon$ of the BCS. Why this discrepancy? In fact, this discrepancy is also present in the generic n-qubit case, and we will answer that question in Section 5.3, when we study this scenario.

5.2.3 The Heterogeneous Case for 3-qubit Registers

Let us now consider the case where we have 3 spins each with different initial bias ε_1, ε_2 and ε_3, corresponding to the 1st, 2nd and 3rd qubits, respectively. This could correspond to a heteronuclear 3-spin molecule or with a homonuclear molecule where the polarisations have already been manipulated. In this case, we have the following optimality result, jointly due to Sasha Mikaelian and the author.

Theorem 5.3. *Let ε_a, ε_b and ε_c be the ordering of the bias values ε_1, ε_2 and ε_3, such that $\varepsilon_a \geq \varepsilon_b \geq \varepsilon_c$. Then there are two possible scenarios for the optimal compression subroutine onto the 1st qubit, discriminated by the quantity*

$$\Delta \triangleq \varepsilon_a - \varepsilon_b - \varepsilon_c + \varepsilon_a\varepsilon_b\varepsilon_c \tag{5.20}$$

as follows

Case 1. *If $\Delta < 0$ then performing a BCS is optimal.*

Case 2. *If $\Delta \geq 0$ then swapping the most polarised bit to the top position is optimal. In particular, if $\varepsilon_1 = \varepsilon_a$ then the optimal transformation is the identity.*

Proof. Let ρ_0 be the initial thermal density matrix, and let $\rho_1 = \Pi_{\mathsf{BCS}}\,\rho_0\,\Pi_{\mathsf{BCS}}{}^{\mathsf{t}}$ be the density matrix after application of the BCS routine (whose permutation matrix is represented by Π_{BCS}). The partial trace of ρ_1 for the first qubit is:

$$\mathrm{Tr}_{2-3}\left(\rho_1\right) = \begin{pmatrix} \frac{1}{4}(2+\varepsilon_1+\varepsilon_2+\varepsilon_3-\varepsilon_1\varepsilon_2\varepsilon_3) & 0 \\ 0 & \frac{1}{4}(2-\varepsilon_1-\varepsilon_2-\varepsilon_3+\varepsilon_1\varepsilon_2\varepsilon_3) \end{pmatrix} \tag{5.21}$$

By solving the equation $1/2 + \varepsilon'/2 = (\mathrm{Tr}_{2-3}\left(\rho_1\right))_{1,1}$, we obtain a generic expression for the new bias ε' of the top qubit, similar to that of Equation 5.18 for the homogeneous case

$$\varepsilon' = \frac{\varepsilon_1+\varepsilon_2+\varepsilon_3-\varepsilon_1\varepsilon_2\varepsilon_3}{2} \tag{5.22}$$

In particular, note that this equation is completely symmetric on the bias values.

Assume first that the bits are sorted in the circuit in order of decreasing bias, i.e.

that $(\varepsilon_1, \varepsilon_2, \varepsilon_3) = (\varepsilon_a, \varepsilon_b, \varepsilon_c)$. Then, the values in the upper half-diagonal of ρ_0 are

$$\begin{aligned} &\frac{1}{8}(1+\varepsilon_a)(1+\varepsilon_b)(1+\varepsilon_c) \geq \\ &\frac{1}{8}(1+\varepsilon_a)(1+\varepsilon_b)(1-\varepsilon_c) \geq \\ &\frac{1}{8}(1+\varepsilon_a)(1-\varepsilon_b)(1+\varepsilon_c) \end{aligned} \tag{5.23}$$

while the biggest element in the lower half-diagonal is

$$\frac{1}{8}(1-\varepsilon_a)(1+\varepsilon_b)(1+\varepsilon_c) \tag{5.24}$$

The only way to improve the bias of the top bit is if the last term of Equation 5.23 is less than that of Equation 5.24, which is exactly equivalent to the condition of Case 1 of the theorem statement. In that case, the BCS (or any of its variants) will accomplish the required permutation of states. If the condition is not met, then the terms in Equation 5.23 are already the biggest in the diagonal and no permutation can improve the bias of the top bit.

Consider now the general case where the biases are not sorted as per the wires of the circuit. If the condition of Case 1 is met then, the optimal bias yield will be obtained regardless of the ordering of the initial bias values, since new bias for the top bit after the BCS, Equation 5.22 is symmetric with respect to these initial values. If the condition of Case 2 is not met, ε_a is the maximum achievable bias, by our analysis above, and thus the best we can do is swap that bit to the top position, or do nothing if ε_1 is already the maximum. □

This optimality result is somewhat surprising if we consider the following. Similarly to Equation 5.22, we can obtain expressions for the bias of the 2nd and 3rd bits after a BCS:

$$\begin{aligned} \varepsilon_2' &= \frac{\varepsilon_1+\varepsilon_2-\varepsilon_3+\varepsilon_1\varepsilon_2\varepsilon_3}{2} \\ \varepsilon_3' &= -\varepsilon_2\varepsilon_3 \end{aligned} \tag{5.25}$$

While we expect the residual bias of the third bit ε_3' to be negligible, there is still a significant amount of polarisation in the second bit ($\approx \frac{1}{2}\varepsilon$ in the homogeneous case). However, the optimality result tells us that this residual polarisation *cannot* be further pumped onto the first bit, and is thus in some sense wasted. We will discuss the reason why this is so in Section 5.3.2.

5.3 Scalable Adiabatic Cooling Algorithms

Let us now turn to the more generic case where we have an n-bit register with all bits having an identical initial bias ε_0, but where we are only interested in maximising the polarisation of the top qubit. In order to warm up and introduce some of the analysis techniques, let us start with a very simplistic cooling algorithm for that case.

5.3.1 A Very Crude n-qubit Cooling Algorithm

Now that we are properly armed with the BCS, we can use it to construct a simple scalable algorithm for solving the above problem. Let $\texttt{BCS}(i, j, k)$ represent the BCS operation performed on the i-th, j-th and k-th wires of a circuit.

```
function CRUDE-ADIABATIC-AC ($n$:int)
for $i = 1$ to $\lfloor (n-1)/2 \rfloor$
    BCS$(1, 2i, 2i+1)$
    end for;
```

This algorithm simply consists of repeatedly applying BCS $m = \lfloor (n-1)/2 \rfloor$ times to the top bit with "fresh" bits each time, in other words the biases of the 2nd and 3rd bits in the BCS's above will always be $\varepsilon_2 = \varepsilon_3 = \varepsilon$.

The first question to ask is, how good is this algorithm? From a "complexity" point of view, one must view the initial bias and analyse its as efficiency and cost in terms of n. In terms of cost, the circuit performing it has linear size $O(n)$[11]. What is its cooling power, then? Let $\varepsilon^{(i)}$ be the bias of the top bit after the i-th iteration. From Equation 5.22 we obtain a recurrence relation

$$\begin{aligned} \varepsilon^{(i+1)} &= \frac{1}{2}(\varepsilon^{(i)} + 2\varepsilon - \varepsilon^{(i)}\varepsilon^2) \\ &= \varepsilon^{(i)}\frac{1-\varepsilon^2}{2} + \varepsilon \\ &\approx \frac{1}{2}\varepsilon^{(i)} + \varepsilon \qquad\qquad (\text{for } \varepsilon^{(i)} \ll 1) \end{aligned} \tag{5.26}$$

[11] Or at most $O(n^2)$ if we are only allowed operations between neighbouring bits.

Since $\varepsilon^{(0)} = \varepsilon$, we can solve this recurrence thus obtaining the following solution:

$$\begin{aligned} \varepsilon^{(m)} &= \frac{2\varepsilon}{1+\varepsilon^2}\left[1 - \left(\frac{1-\varepsilon^2}{2}\right)^{m+1}\right] \\ &\approx \varepsilon\left(2 - \frac{1}{2^m}\right) \qquad\qquad (\text{for } \varepsilon^{(m)} \ll 1) \end{aligned} \tag{5.27}$$

Also note that the asymptotic limit when we start using more and more bits to boost the top bit will be:

$$\lim_{n\to\infty} \varepsilon^{(\lfloor(n-1)/2\rfloor)} = \frac{2\varepsilon}{1+\varepsilon^2} < 2\varepsilon \tag{5.28}$$

In summary, no matter how many bits we use in our register we will only be able to double (at most) our initial bias. However we will approach that limit exponentially fast as we increase the number n of bits used to cool the top bit.

But is in fact blindingly applying BCS with fresh qubits at each iteration the best strategy? Is it not possible that we might hit a point where doing BCS will do more harm than good, as in Case 2 of Theorem 5.3, and thus that the bias attained $\varepsilon^{(i)}$ might actually start decreasing? The answer is no. We can rewrite Equation 5.26 in terms of the discriminant Δ of Theorem 5.3 as follows

$$\varepsilon^{(i+1)} = \varepsilon^{(i)} - \frac{\Delta}{2}$$

From this, we deduce that the bias will decrease only if Δ is positive. But, in this case

$$\begin{aligned} \Delta &= \varepsilon^{(i)} - 2\varepsilon + \varepsilon^{(i)}\varepsilon^2 \\ &= \varepsilon^{(i)}(1+\varepsilon^2) - 2\varepsilon \\ &= -2\varepsilon\left(\frac{1-\varepsilon^2}{2}\right)^{n+1} < 0\,, \quad \text{since } 0 \le \varepsilon \le 1 \end{aligned} \tag{5.29}$$

In fact this algorithm is far from optimal as one might guess. The fact that one can at most increase the polarisation by a factor two, irrespective of the number of qubits is a big clue.

5.3.2 Upper Bounds on Cooling Efficiency for Adiabatic Algorithms

As mentioned above, the work of Sørensen is a fundamental landmark in understanding the limits of Algorithmic cooling. However, his key result is what he called a "universal bound on spin dynamics", whose implications go far beyond AC, and has in our opinion been unfairly overlooked by the QC community.

Consider the generic situation where we have an arbitrary abstract quantum system (spin-based or otherwise), whose initial state is described by a density matrix ρ_0. Suppose further that, for whichever reason, we would like to transform as much as possible this system into a target state ρ, but in doing so we are limited by the laws of quantum mechanics, i.e. we are constrained to using unitary transformations on the Liouville space of which ρ is a member (i.e. no ancillæ are allowed).

First of all, one must realise that we cannot always fully accomplish our objective, as there does not necessarily exist a U such that $\rho = U\rho_0 U^\dagger$, or in other words, ρ_0 and ρ need not be similar matrices (they might have different eigenvalues). However, we might be content with transforming ρ_0 into "as much of" of ρ as we can, or in other words we seek to transform ρ_0 into a ρ'

$$\rho_0 \xmapsto{U} \rho' = U\rho_0 U^\dagger = a\rho + b\rho^\perp \tag{5.30}$$

where $\rho^\perp$ is "orthogonal" to ρ in the Liouville space , i.e. $\langle \rho, \rho^\perp \rangle = \mathrm{Tr}(\rho^\dagger \rho^\perp) = 0$. The coefficient a represents the projection of ρ' onto the target subspace of the Liouville space spanned by ρ, or "how much of ρ" is in ρ'. Our objective is thus to find the transformation $U_{\max}$ that will maximise the modulus $|a|$ of this coefficient.

Theorem 5.4 (Sørensen). *Let $\lambda_1 \geq \ldots \geq \lambda_N$ and $\mu_1 \geq \ldots \geq \mu_N$ be the eigenvalues of ρ_0 and ρ, respectively, ordered and possibly repeated. Then, the maximum and minimum values of a in Equation 5.30, over all unitary transformations U, are given by*

$$a_{\max} = \frac{\sum_{i=1}^{N} \lambda_i \, \mu_i}{\sum_{i=1}^{N} \mu_i^2}, \qquad \textit{and} \qquad a_{\min} = \frac{\sum_{i=1}^{N} \lambda_i \, \mu_{N-i+1}}{\sum_{i=1}^{N} \mu_i^2} \tag{5.31}$$

We can apply this very general theorem to algorithmic cooling, not by considering the density matrices ρ_0 and ρ but their non-identity part or *deviation matrices*, as they are often called by NMR spectroscopists. The initial deviation matrix is $\delta_0 = \rho_0 - 1/2^n I_n$, where ρ_0 is the thermal density matrix for n bits, each with initial bias ε. The desired "target" density matrix ρ is the tensor product of the 1-bit thermal state density matrix ρ_ε for the first bit, and the identity for all the others bits. This density matrix and the corresponding deviation matrix δ are given by

$$\rho = \rho_\varepsilon \otimes \frac{1}{2^{n-1}} \mathsf{I}_{n-1} \tag{5.32}$$

$$\delta = \frac{1}{2^n} \varepsilon \left(\sigma_z \otimes \mathsf{I}_{n-1}\right) \tag{5.33}$$

where ρ_ε is given in Equation 5.4. The maximum bias transfer ratio is then given by the maximum value of the projection $a_{\max}$ of δ_0 onto the subspace of δ. By applying Theorem 5.4, Sørensen calculated the following absolute bound on polarisation transfer onto a single bit.

Corollary 5.5. *The maximum bias $\varepsilon_{\max}$ achievable on any given qubit by applying a unitary transformation on an n qubit system initialised in the thermal state $\rho_0 = \rho_\varepsilon^{\otimes n}$ is given by*

$$\varepsilon_{\max} = \varepsilon \, \frac{\binom{2m}{m}(2m+1)}{2^{2m}} \tag{5.34}$$

where $m = \lfloor \frac{n-1}{2} \rfloor$ and for $\varepsilon \ll 1$.

It is particularly interesting to compare this value with the Shannon bound. If we take Equation 5.19 and apply it to the n-qubit scenario we get

$$\begin{aligned} nH(\varepsilon) &= H(\varepsilon_{\max}) + (n-1)H(0) \\ &= H(\varepsilon_{\max}) + n - 1 \end{aligned} \tag{5.35}$$

which for $\varepsilon \ll 1$ gives us

$$\varepsilon_{\max} \approx \varepsilon\sqrt{n} \tag{5.36}$$

By using Stirling's formula, we can obtain an approximation of the bound in Equation 5.34 which gives a better intuition on how it grows with n

$$\varepsilon_{\max} \approx \varepsilon\,\sqrt{n}\,\sqrt{\frac{2}{\pi}} \qquad \text{(for } n \text{ odd)} \tag{5.37}$$

From this, we can see that the ratio between the Shannon and Sørensen bounds converges to $\sqrt{2/\pi} \approx 0.80$ as n becomes large.

The fundamental reason why the Sørensen bound is worse, not only in this case but in general, is because ultimately unitary transformations cannot change the eigenvalues of a density matrix: they can only permute them. Since the projection value a depends on these eigenvalues, the Sørensen bound is a natural consequence of this limitation of unitary matrices.

We can draw from the Sørensen bound two further interesting consequences. First of all, the Sørensen bound as computed for this case is not completely tight either. Consider the homogeneous 3-bit case. For $n = 3$, we have $m = 1$ and by Equation 5.34 we have that the maximum achievable ratio is $\frac{3}{2}\,\varepsilon$. We know however from Theorem 5.3 that the BCS is optimal in this case, but only has yield $\frac{3}{2}\,\varepsilon - \frac{1}{2}\,\varepsilon^3$. This difference of $\frac{1}{2}\,\varepsilon^3$ is due to the fact that the initial state ρ_0 used by Sørensen to compute the bound in Equation 5.34 is only the first order approximation of the corresponding thermal state, i.e. in the case $n = 3$ we have that

$$\rho_\varepsilon^{\otimes 3} = \frac{1}{8}(\mathsf{III} + \varepsilon(\mathsf{ZII} + \mathsf{IZI} + \mathsf{ZII}) + \varepsilon^2(\mathsf{ZZI} + \mathsf{ZIZ} + \mathsf{IZZ}) + \varepsilon^3\mathsf{ZZZ}) \tag{5.38}$$

$$\approx \frac{1}{8}(\mathsf{III} + \varepsilon(\mathsf{ZII} + \mathsf{IZI} + \mathsf{ZII})) = \rho_0 \qquad \text{(for } \varepsilon \ll 1) \tag{5.39}$$

Second of all, we can now see that the simple cooling algorithm of Section 5.3.1 is far from optimal. In that case, we could at most double the initial bias, while the Sørensen bound even though more restricted by a constant than the Shannon bound, still allows for the polarisation to grow as $O(\sqrt{n})$ when we increase the number of qubits.

The next question to ask is whether it is possible to find cooling algorithms which attain the optimal yield of the Sørensen bound. The answer is only a semi-satisfactory yes. Sørensen [Sør89] does describe a generic method for obtaining optimal "pulse sequences". In principle, the method is simple

Step 1. Find unitary transformations U and V diagonalising the initial and target deviation matrices δ_0 and δ, i.e. such that $\delta_0^D = U\delta_0 U^\dagger$ and $\delta^D = V\delta V^\dagger$ are both diagonal.

Step 2. Find unitary transformations U' and V' ordering the eigenvalues of δ_0 and δ, i.e. such that the diagonals of $\delta_0^> = U'\delta_0^D U'^\dagger$ and $\delta^> = V'\delta_0^D V'^\dagger$ are in decreasing order from top left to bottom right.

Step 3. Construct an n-wire circuit performing operations $U'U$ and apply it to the initial state ρ_0.

The projection of the final deviation matrix $\delta_0^> = U'U\delta_0 U^\dagger U'^\dagger$ obtained in Step 3 onto δ

will be exactly $a_{\max}$, as we can rewrite Equation 5.31 as

$$a_{\max} = \frac{\langle \delta_0^>, \delta^> \rangle}{\| \delta \|^2} \tag{5.40}$$

In the algorithmic cooling case, because the initial state ρ_0 is already diagonal, we need not worry about Step 1. However, it is not clear that there always exist poly-size circuits for these unitary transformations. The Cleve-DiVincenzo implementation of Schumacher compression does in essence compute this ordering of the elements of ρ_0, and while it can be implemented with poly-size circuits using as little as $n + O(\sqrt{n})$, we believe that it has not been proven or firmly established that such an ordering can be accomplished in-place with poly-size circuits in all cases, or more precisely a polytime algorithm that can generate such circuits.

What about the heterogeneous bias case for n-bits? Sørensen describes and gives the bound for one such simple case, in which the top bit has bias ε_0 and all others have bias ε_1, which corresponds to the typical NMR scenario of transferring polarisation from, say, $n-1$ ^{1}H nuclei to a single ^{13}C nucleus. In this case, the bound is essentially multiplied by the ratio of gyromagnetic constants γ, in this example approximately 4. For more complex cases, such as the analysis of intermediary steps in more complicated cooling algorithms, the universal bound of Theorem 5.4 can be readily applied.

5.4 Non-Adiabatic Algorithmic Cooling

From the NMR QC point of view, the Shannon or Sørensen bound are bad news. As mentioned before, we can at best expect a polarisation ratio of $O(\sqrt{n})$ for single bits, and thus in order to cool m bits close to bias 1, we still need $\Omega(\varepsilon^{-2}m)$ bits at bias ε. Non-adiabatic cooling presents an opportunity to overcome this barrier.

5.4.1 The Basic Non-Adiabatic Case: 3-qubit Register

As before, and to illustrate the principle, let us present a simple case: that in which only three bits are available, but where one of them has the capacity to return to its initial natural bias $\varepsilon^{(0)}$ "on demand." We can view this special bit as a "radiator bit," which has contact with the environment (heat bath), and when it gets "warmed" to higher temperature than that of the environment (i.e. a bias less than $\varepsilon^{(0)}$) has the ability to pump out its entropy onto the heat bath. Abstractly, we model this ability by providing a non-unitary primitive RESET, which immediately returns a bit to its initial thermal state $\rho_0 = \rho_{\varepsilon^{(0)}}$

The algorithm is as follows:

```
function 3-BIT-NON-ADIABATIC-AC (m:int)
for $i = 1$ to $m$
        BCS(1, 2, 3)
        RESET(3)
        SWAP(2, 3)
        RESET(3)
    end for;
```

Note how similar this algorithm is to the simple, crude adiabatic algorithm of Section 5.3.1[12]. At the end of each loop iteration, we will have that the bias of the top bit is $\varepsilon^{(i)}$, while the bias of the second and third bits will have been returned to $\varepsilon^{(0)}$. The next BCS on the next iteration will then have exactly the same effect as $\mathtt{BCS}(1, 2i, 2i+1)$ in the adiabatic algorithm.

We can thus import the analysis of that algorithm and observe that the bias of the top bit after m steps will be given by Equation 5.27. By Theorem 5.3 and Equation 5.29, we know that we are performing at every iteration the optimal internal adiabatic transformation to boost the first qubit. Furthermore, we know that $\varepsilon^{(i)}$ is bounded (Equation 5.28). We thus have the following upper-bound on non-adiabatic cooling on three qubits.

Theorem 5.6. *The maximum achievable polarisation bias $\varepsilon_{\max}$ on a 3-bit register is bounded by $2\varepsilon^{(0)}$, where $\varepsilon^{(0)} = \max\{\varepsilon_1^{(0)}, \varepsilon_2^{(0)}, \varepsilon_3^{(0)}\}$ is the maximum of the natural (thermal) polarisation biases of each bit.*

As discussed in Section 5.2.3, after the BCS step in the above algorithm, there is some non-negligible leftover polarisation on the 2nd bit, which is transfered to the 3rd bit and then "wasted" when we perform the second RESET operation. This seems wasteful but is nonetheless optimal. This kind of "paradox" is actually quite common in Thermodynamics, as any proud owner of a thermal pump or air conditioning unit will know: in the summer, it is more efficient to extract "cold" (by evaporating water) out of the warm air outside than it is to do so from within the colder air inside.

[12] In fact, the only reason we bothered introducing such an inefficient algorithm was as a prelude to the non-adiabatic case

5.4.2 A Truly Scalable Non-Adiabatic Algorithm

The 3-bit non-adiabatic algorithm above is not sufficient for algorithmic cooling purposes. As mentioned in our historical overview, the non-adiabatic cooling algorithm was proposed by BMRVV [BMR+02]. This algorithm uses blocks of m bits and pushes cooled spins to one end of such blocks in the molecule. To obtain significant cooling, the algorithm requires very long registers of hundreds or even thousands of bits, because its analysis relies on the Law of large numbers. As a result, although much better than any adiabatic technique, this algorithm is still far from having any practical implications.

The algorithm presented here [FLMR03] is a significant improvement on this first non-adiabatic algorithm. Besides the fact that it is much simpler to describe and analyse, it requires far fewer bits.

The algorithm is recursive in nature. The base case and initial state is where all the bits are polarised to their natural initial bias $\varepsilon^{(0)}$. Performing a BCS on three bits with such a bias will polarise one of them to a new bias $\varepsilon^{(1)} \approx \frac{3}{2}\varepsilon^{(0)}$. In general, if we have three bits at positions i_1, i_2, and i_3 of a circuit each with bias $\varepsilon^{(j)}$, then the procedure $\mathtt{BCS}(i_1, i_2, i_3)$ will boost the bias of the i_1-th bit to $\varepsilon^{(j+1)} \approx \frac{3}{2}\varepsilon^{(j)}$.

To boost the bias level of the i-th bit to bias level $\varepsilon^{(j)}$, we use the following procedure:

```
function NAC-bit ($i$:posn,$j$:int)
      if $j = 0$
            RESET($i$);
            return;
            end if
      do NAC-bit($i$,$j-1$);
      do NAC-bit($i+1$,$j-1$);
      do NAC-bit($i+2$,$j-1$);
      BCS($i$,$i+1$,$i+2$)
```

The general procedure for cooling m bits to the bias level $\varepsilon^{(j)}$ is simply:

```
function NAC-register ($m$:posn,$j$:int)
for $i = 1$ to $m$
      do NAC-bit ($i$,$j$)
      end for;
```

Let us now consider the performance of this algorithm. In terms of space let $s(j)$ be the number of bits required to cool a bit to level $\varepsilon^{(j)}$. In terms of execution time (or circuit size) let us count the total number of BCS gates used as $c(j)$ and separately the total number of RESET operations as $t(j)$. The recursive nature allows us to setup recurrences for the various resources used by the algorithm as well as for the bias attained:

$$\varepsilon^{(j+1)} = \frac{3}{2}\varepsilon^{(j)} - \frac{1}{2}\varepsilon^{(j)^3} \approx \frac{3}{2}\varepsilon^{(j)}, \qquad \varepsilon^{(0)} = \varepsilon$$

$$s(j+1) = s(j) + 2, \qquad s(0) = 1 \tag{5.41}$$

$$c(j+1) = 3c(j) + 1, \qquad c(0) = 0 \tag{5.42}$$

$$t(j+1) = 3t(j), \qquad t(0) = 1 \tag{5.43}$$

Furthermore, consider the equivalent costs for cooling m bits to the j-th level, by using the *NAC-register* procedure. Then we have that,

$$s(m,j) = m + s(j) \tag{5.44}$$

$$c(m,j) = mc(j) \tag{5.45}$$

$$t(m,j) = mt(j) \tag{5.46}$$

By solving the above recurrences we obtain the following costs:

$$\varepsilon^{(j+1)} \approx \left(\frac{3}{2}\right)^j \varepsilon \tag{5.47}$$

$$s(m,j) = m + 2j \tag{5.48}$$

$$c(m,j) = m\frac{1}{2}3^j - 1 \tag{5.49}$$

$$t(j+1) = m3^{j-1} \tag{5.50}$$

From an asymptotic point of view, this algorithm is "essentially" polynomial and comparable in that respect with the Schulman-Vazirani algorithm. To see this, for most practicable quantum algorithms it will be sufficient to obtain a final bias only polynomially close to one, i.e $\varepsilon^{(j)} = 1 - O(\text{poly}(n))$. Since $\varepsilon^{(0)}$ is constant, this means that j is roughly $O(\log n)$, and that all resources (space, total circuit size and number of bit resets) will be polynomially bounded.

In practice, and if we consider typical polarisation biases in liquid NMR at room temperature, the gains are even more impressive. For an initial bias of 10^{-5}, for example, the Schulman-Vazirani algorithm requires $1/20\varepsilon^2$ bits to cool one single bit, i.e approximatively a 500 million bits, our algorithm can obtain a very respectable final bias of 0.982 with $j = 30$, thus requiring only 60 bits.

Indeed, the interesting feature of this algorithm is that the bias can be boosted exponentially fast in the number of bits used by the machine. This is, however, at the cost of an also exponentially increasing number of operations. Nonetheless, for most practical applications a limited number of steps will be sufficient, with j between 10 and 20 being sufficient before other techniques can take over such as quantum error correction.

Secondly, it is quite clear that this relatively simple algorithm is not "thermodynamically" efficient. We showed in the 3-bit case that even though we were discarding bits with $\varepsilon/2$ leftover polarisation by letting them relax in contact with the environment, this was optimal and that we could not do better. In this case, however, it easy to see that because of the extended workspace and numbers of these half polarised bits, some of the polarisation could be "recycled". Unfortunately, we can no longer use the bounds of Sørensen or Shannon to benchmark these non-adiabatic algorithm, and thus a new theory of efficiency must be developed for them, by importing notions from Thermodynamics such as Q factors (a kind of mechanical/thermal efficiency factor). This we have been working on with our co-authors but is still work in progress.

Even inefficient as it is, it would still be a formidable achievement to implement such an algorithm even for modest values of j. The implications would go far beyond the world of Quantum Computing and might have implications in Chemistry, Molecular Biology, Pharmacology, Medicine and all the other fields which are heavy users of NMR spectroscopy and imagery, which could strongly benefit from an increase in signal strength even if only of an order of magnitude ($j = 6$).

That is why we decided to get our hands wet and set out to implement this technique in the laboratory. The next chapter covers the result of our musings about the viability of implementing such algorithms and the experimental results actually obtained.

CHAPTER 6

EXPERIMENTAL ALGORITHMIC COOLING

6.1 Previous Work and Objectives

At the beginning of Chapter 5 we reviewed the history of NMR experiments in Algorithmic Cooling. Such experiments, of course, precede the coining of this terminology with its beginnings back in the late 70's with the first INEPT experiments. To give an idea of how commonplace INEPT experiments have become, let us just point out that a wide variety of *pret-à-porter* INEPT sequences are part of the standard software distribution that comes with any COTS spectrometer. Because it involves only two nuclei, the transfer limit is simply the ratio of gyromagnetic constants. Nothing more can be done.

Already in 1989 Sørensen reports [Sør89] on an experiment for the case $n = 4$, involving 3 ^{1}H nuclei and 1 ^{13}C nucleus in the methyliodide molecule (CH_3I). The polarisation is transfered from the ^{1}H nuclei to the ^{13}C. If ε is the initial polarisation of the ^{13}C, this scenario allows for a maximum final polarisation of 6ε. Compared to a BCS ($n = 3$), while the presence of a fourth spin does not improve the upper bound, it does in this case significantly simplify the design of the sequence. It uses a combination of non-selective hard pulses on the ^{1}H, which have essentially the same chemical shift and refocusing pulses. This is particularly interesting because it shows an example of successful AC in a molecule which is itself not suitable for NMR QC.

This is in contrast with the experiment performed at IBM Almadén in 2001 [CVS01], which uses the three identical spin 1/2 Fluorine nuclei in the C_2BrF_3 molecule to boost the polarisation of one of them; no other spins are involved in the procedure. Secondly, this experiment used "soft" selective pulses to perform rotations on the individual spins without affecting the others. The experimental results showed a total boost in polarisation of 125%. This is in comparison with a theoretical limit of 150%; a significant achievement nonetheless.

So, as far as adiabatic AC is concerned, proof-of-concept experiments had been performed in both the hetero- and homo-nuclear scenarios. Higher bounds, could only be achieved with a minimum of 5 spins, a non-negligible feat for a beginner such as the author in NMR QC given that the leading edge of that field involved 7-qubit molecules [KLMT99].

The logical choice for the author and his collaborators seemed to be to try our luck with simple non-adiabatic cooling experiments. We therefore set out to find a simple experiment in which we could demonstrate the concept of non-adiabatic AC, by obtaining polarisation biases above those allowed by the Shannon and Sørensen bounds for adiabatic cooling. The next section introduces and discusses the generic problems associated with performing this kind of experiments. In Section 6.3, we introduce the molecule used for our experiments and discuss what can and cannot be accomplished with it. In Section 6.4 we describe the particular objective that we wanted to accomplish and describe the experimental setup that we used. Finally, we discuss in Section 6.5 the results obtained and avenues for further improvements and future research.

6.2 How to Build a Radiator

As we saw in Section 5.4.1, already with a three bit register, it is possible to exhibit bias improvements which beat both the Shannon and Sørensen bounds for adiabatic cooling. However, the main difficulty in performing such experiments is the following: how does one implement this "magical" RESET operation. As we have mentioned in the last chapter, the simplest possible way to do so is to simply let the natural relaxation mechanisms act onto the over-heated spins, which with time will bring them back to their ambient spin temperature and thus to their initial polarisation bias $\varepsilon^{(0)}$.

The rate at which spins return to their natural thermal state is characterised by the relaxation time T1 (usually expressed in seconds). For states whose density matrix is already diagonal (i.e. classical), we can express the rate of return to the initial bias as follows

$$\varepsilon(t) = \varepsilon + (\varepsilon(0) - \varepsilon)e^{-t/\mathrm{T1}} \tag{6.1}$$

where $\varepsilon = \varepsilon^{(0)}$ is the natural polarisation bias ratio, and $\varepsilon(0)$ is the bias at time $t = 0$. Intuitively, as t becomes bigger, the polarisation will quickly increase (or decrease) back to its initial natural value ε, at the rate prescribed by T1, having gone 63% of the way by T1 seconds, and 86% after $2 \cdot \mathrm{T1}$ seconds, and, as a rule thumb, having all but completely returned (99.3%) to its initial bias after $5 \cdot \mathrm{T1}$ seconds. Unfortunately, this relaxation works both ways. While hot bits will cool themselves back to the ambient temperature (a good thing), cooled bits also re-heat themselves to the ambient temperature thus undoing the hard work previously done to cool them. For nuclei typically encountered in liquid-state NMR spectroscopy, T1 values can range from a few tenth of seconds to a few minutes.

In the ideal abstract world, such as of BMRVV [BMR+02], all "computation bits" of our register have no direct contact with the environment and once cooled do not re-heat. In other words, they are perfectly "insulated" from the environment or, more precisely, they have infinite T1. In this idealised model, there are also rapidly-reaching-thermal-relaxation (RRTR) bits, which have small T1 values and thus once heated can rapidly come back to their initial bias. The RESET operations can be applied to such RRTR bits by waiting for 5T1 seconds, for example, and on the non-RRTR, well insulated computation bits by performing a SWAP between a heated computation bit and a RRTR bit at room temperature, which can itself then be quickly reset.

BMRVV further suggest that such an idealised model could be implemented in practice by taking advantage of electron-nuclear interaction, a technique known as ENDOR (for Electron-Nuclear Double Resonance). The gyromagnetic constant of electrons is up to three order of magnitudes larger than that of protons (for example), which by "simple" transfer can boost the nuclear bias by that much. Furthermore, they typically have relaxation rates in the order of milliseconds, which allows for repeated reset-compression cycles before the relaxation of the nuclei (with T1's in a few seconds) can have any appreciable effects. ENDOR experiments have been performed for several decades and are well known. Unfortunately, the electronics of standard COTS NMR spectrometers are not capable of generating nor handling the extremely high microwave frequencies (in the 300-600 GHz range) necessary to manipulate electron spins within a typical NMR-strength magnetic field of 10-15 Tesla.

Again, since constructing a custom-made ENDOR-suitable spectrometer was much beyond the ability and pocketbook of the author and his collaborators, simpler ways had to be found to perform a non-adiabatic experiment.

In a more realistic scenario, we have that T1's of computation bits are finite and we do not necessarily have access to "special" RRTR bits. In other words, we assume that all bits can be used for computation and that they have varying values of T1. The entropy is pumped out of the system by letting the faster bits in the register relax for a certain time t. Of course, the prior compression steps will have cooled the more slowly relaxing bits, which will during that time t loose some of their gained polarisation. After relaxation, the polarisation gained by relaxation can be pumped back into the slowly relaxing qubits, and so on. The hope is that by this process we can pump in polarisation faster than we are losing it by the natural relaxation of cold qubits.

The waiting time t is a free parameter of the procedure, which can be adjusted to maximise the final polarisation of the cooled bits. As it turns out, when we combine the equations for relaxation and for the compression procedures (Equations 6.1, and 5.22 for the BCS), we observe that the final bias only depends on the *ratio* η between the T1 of the bits being cooled and those being used to pump entropy out of the system.

In particular, if we consider the simple non-adiabatic algorithm of Section 5.4.1, the yield of the procedure after m steps of compression/relaxation only depends on the respective initial bias values, the ratio η between the T1 values for the 1st and 3rd bits

$$\eta \triangleq \frac{\mathrm{T1}_1}{\mathrm{T1}_3} \tag{6.2}$$

and on the waiting times $t_1, t_2, \ldots, t_{2m+1}$, where at step i we wait for t_{2i} and t_{2i+1} seconds for the first and second RESET operation, respectively.

6.3 A Heat Engine with Three "Cylinders"

For practical reasons the Trichloroethylene (TCE) molecule C_2HCl_3 was chosen[1] as the basis for performing a 3-qubit non-adiabatic cooling experiment. The sample used consists of doubly labelled TCE at 98 %, i.e. 98 % of the TCE molecules in the sample have two ^{13}C's, in a solution of deuterated chloroform ($^{12}C^2HCl_3$). By convention, we call C1 the carbon nucleus surrounded by Chlorine, C2 the one with the proton H, and they are the first and second bit of our 3-bit register. This a natural choice, since the respective T1's are approximately 30 and 27 seconds, and 5.5 seconds for H, thus having a natural T1-ratio of $\eta \approx 5$. The first question to ask then is how well can we do given these parameters.

Let us consider first the scenario where we seek to increase the polarisation of the first qubit alone. The initial biases are $\varepsilon, \varepsilon, 4\varepsilon$ and by Theorem 5.3, the optimal transformation in this case is simply to perform a swap between C1 and H. After an ideal RESET of H, the polarisation should be $4\varepsilon, \varepsilon, 4\varepsilon$, and then the optimal compression is a BCS, which

[1] It was well known by one of our collaborators who had used it in various successful 3-qubit NMR QC experiment, and a sample was available (they are not cheap...) for our use.

gives a final bias of 4.5ε. This is good news in the sense that it allows us to obtain a theoretical bias value ^{13}C otherwise unattainable in such a 3-bit molecule, hence breaking Sørensen's bound.

Unfortunately, there is not much room for manoeuvre between doing just a swap (or an INEPT) and achieving 4ε, and achieving the theoretical maximum polarisation 4.5ε. For small initial bias ε the final polarisation of C1 after waiting for time t before the BCS is:

$$\varepsilon(\eta, t) = \frac{3}{2}\left(2 - e^{t} + e^{-t/\eta}\right) \tag{6.3}$$

Considering that for a fixed η, we always choose the waiting time t that will maximise polarisation, we can obtain an expression for maximum polarisation as a function of η as follows

$$\varepsilon_{\max}(\eta) = \frac{3}{2}\left(2 + \eta^{1/(1-\eta)} - \eta^{\eta/(1-\eta)}\right) \tag{6.4}$$

whose limit as $\eta \rightarrow \infty$ is 3/2, as we would expect. The more important issue, however, is how high does η need to be in order for us to "break even" and make it worth it for us to use a RESET of H at all. The bad news is that it is not worth it, because $\varepsilon_{\max} > 1$ only if $\eta > 8.5$, which is not the case for this molecule.

How does one get around this problem? The first obvious answer is to identify other molecules for which the ratio η is more favourable. It it not to difficult to do so, and some such as alanine were identified. Another approach is to try to *manipulate* the T1's to our advantage.

The technique used to do so is the insertion of trace amounts of a paramagnetic impurity in the solvent, which has the effect of accelerating the loss of magnetisation of outer electrons and protons with the environment (which now includes the salt). As a result, a marked reduction in the T1 relaxation times is usually observed, depending on the sample molecule. This is a common technique used by NMR spectroscopists to reduce long T1's in order to be able to reduce the waiting time between experiments when the same experiment is run automatically several times in order to improve signal strength by sample averaging.

A commonly used paramagnetic salt is Chromium (III) Acetylacetonate (abbreviated CrAcAc, and also known as Chromium 2,4-pentanedionate). Fortunately, with TCE the addition of CrAcAc results in a dramatic reduction of the T1 for the H, while that of C1 is relatively unaffected (we do not care about the reduction of T1 for C2). Lacing the original sample with a concentration of 233.2 mg/L of CrAcAc salt resulted in the T1 changes shown in Table 6.1.

T1	unsalted	salted
C1	30.85 s	28.3 s
C2	27.45 s	16 s
H	5.46 s	1.88 s
η	5.65	15.05

Table 6.1: T1 relaxation times in trichloroethylene before and after adding the CrAcAc salt to the solvent. The T1 for the H decreased by 65.6% while C1 and C2 changed there relaxation times by 8.3% and 41.7% respectively. This represents a 266.4% increase for the T1 ratio $\eta = \mathrm{T1(C1)/T1(H)}$.

Even so, the new T1 ratio $\eta = 15.05$ only allows for a meager 4.15ε final maximal polarisation, barely 3 % better than doing a simple INEPT... Since this would leave us almost no room for experimental error, instead of trying to beat the 4ε bound of Theorem 5.3 (the heterogeneous version of the Sørensen bound for the 3-bit case), we decided instead to chose another experimental target.

6.4 Bypassing the Shannon Bound

The objective of the proposed experiment was to show that it is possible to use non-adiabatic AC to increase the total polarisation of a molecule beyond what is initially present in its equilibrium thermal state. In other words, we wanted to show that it is possible to bypass the Shannon bound on entropy conservation.

The initial entropy H_0 of the TCE molecule in its thermal state is given by

$$\begin{aligned} H_0 &= H(\varepsilon) + H(\varepsilon) + H(4\varepsilon) \\ &= 3 - \frac{1+1+4^2}{\ln 4}\varepsilon^2 \\ &= 3 - \frac{18}{\ln 4}\varepsilon^2 \end{aligned} \tag{6.5}$$

which represents approximately 2.9999999987 bits, for a typical liquid NMR polarisation of 10^{-5}. In other words, the maximum information storage capacity of our 3-bit register is $13 \cdot 10^{-10}$ bits. More important than these absolute values in bits, are the relative changes in entropy that the register will undergo during our experiment. Assuming that at the end of the experiment we end up with qubit polarisations $c_1\varepsilon$, $c_2\varepsilon$, and $c_3\varepsilon$, then ignoring any correlation (coherence) between the bits (qubits), we can have an upper bound on the final entropy

$$\begin{aligned} H_f &\leq H(c_1\varepsilon) + H(c_2\varepsilon) + H(c_3\varepsilon) \\ &= 3 - \frac{c_1^2 + c_2^2 + c_3^2}{\ln 4}\,\varepsilon^2 \end{aligned} \tag{6.6}$$

What we seek is that there is a net entropy decrease, i.e. $H_f < H_0$, which is achieved if

$$c_1^2 + c_2^2 + c_3^2 > 4^2 + 1 + 1 = 18 \tag{6.7}$$

As we have discussed before, we implement `RESET` operations by waiting for a fixed amount of time. Also, in the case of the TCE molecule, the coupling constants between C1 and C2, and between C2 and H are 103 and 201 Hz, which is sufficient to efficiently perform two-qubit operations between qubits 1 and 2 and 2 and 3. However, the C1-H coupling constant is only 8.9 Hz, and is too small for direct two-qubit interactions between C1 and H. With these constraints in mind, and from an abstract Quantum Computing point of view the simplest possible procedure to allow us to beat the Shannon bound would be the following.

```
function Cool-TCE ($t1$, $t2$: real)
  1 SWAP(2,3);
  2 SWAP(1,2);
  3 WAIT($t1$);
  4 SWAP(2,3);
  5 WAIT($t2$);
```

In practice, performing a perfect `SWAP` operation is too long and unnecessary. Instead, we use the shorter INEPT sequence to do the polarisation transfer from H to C2 and then from C2 to C1. The disadvantage is that at the end of this double INEPT the magnetisation of C1 is on the XY-plane of the Bloch sphere, and hence subject to T2 relaxation. Since T2 is much smaller than T1, in this case in the order of a few hundreds of milliseconds, we must bring back the transfered polarisation parallel to the direction of the field (the positive Z-axis) in order for it to "survive" longer during the first and second waiting times. The same must be done after the INEPT in Step 4 for C2.

A high-level block diagram for the complete experiment is shown in Figure 6.1, and the detailed pulse sequences for each block are given in Figure 6.2.

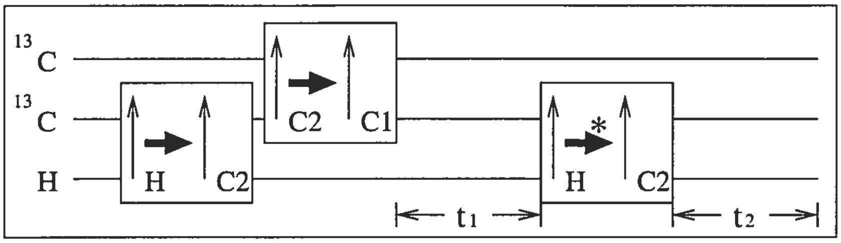


Figure 6.1: Block Diagram of the Complete Non-Adiabatic Cooling Experiment on TCE. The arrow boxes denote an INEPT-based polarisation transfer in the direction of the arrow. Each polarisation transfer sequence is a bit different from the rest, hence the different indices. All the boxes include refocusing for unwanted evolutions due to couplings and the carbon drifting due to chemical shift. The detailed structure of all the boxes are given in Figure 6.2. The time periods t_1 and t_2 are the delay times in which we wait for the H to repolarise.

For our experiment, a Bruker DMX-400 NMR spectrometer was used. The figure "400" indicates that the strength of the magnetic field is such that the resonance frequency of the ^{1}H nucleus is approximately 400 MHz. The probe used was a bi-channel probe. Two transmit channels were available, one used for the ^{13}C frequency range (approx. 100 MHz) and the other for the ^{1}H frequency range. The first channel is calibrated to transmit at the rotation frequency of C2, which causes C1's magnetisation to rotate along the Z axis with the chemical shift frequency of approximately $\Delta\omega$, which was approximately 700 Hz on this spectrometer. Unfortunately, signal acquisition was only possible on the first channel which meant that only the ^{13}C spectra (such as those shown in Figure 6.3) could be obtained. The expected polarisation of H at the end of the experiment had to be calculated as a function of its measure T1.

To keep our experiment as simple as possible, only "hard" non-selective pulses were used on both channels, which meant that every rotation of the ^{13}C had to affect them both. As a general rule, unwanted interactions such as chemical shift, unwanted couplings, etc. were eliminated by using refocusing pulses at appropriate intervals, as is described in Figure 6.2. The sequences were designed and verified by using the NMRCalc package, an NMR-oriented version of the QuCalc Mathematica package developed by Paul Dumais. These resulted in sequence descriptions in terms of NMR-implementable operations such as $\texttt{WAIT}(t)$ and $\texttt{ROTa}(\theta)$, where $a = \texttt{X}, \texttt{Y}$ and $\theta = \pi/2, \pi$. These sequences were then manually converted into Bruker "pulse programs", which were then verified

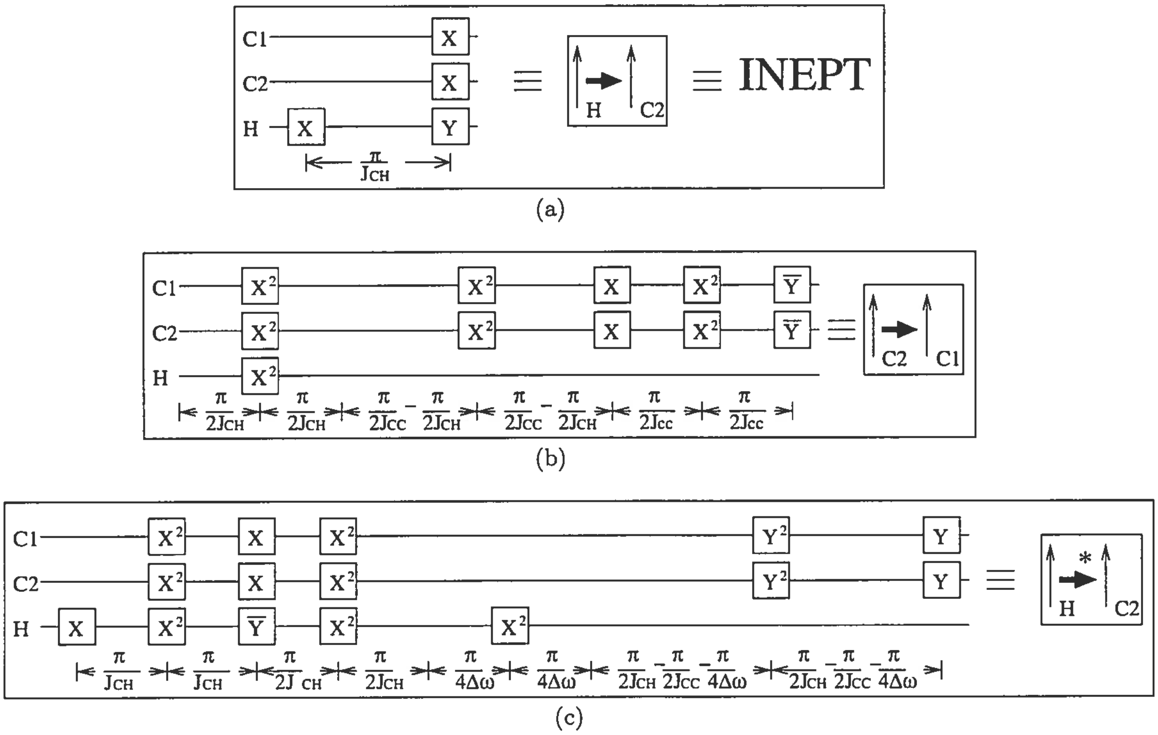


Figure 6.2: The structures of the boxes in the block diagram from Figure 6.1 are portrayed here. The letters X, Y denote $\frac{\pi}{2}$ rotations around the X or Y axis respectively. The superscript 2 denotes a π rotation, and a bar (e.g. $\overline{\mathrm{X}}$) denotes a $-\frac{\pi}{2}$ rotation. The coupling constants are represented by $\mathrm{J_C C}$ and $\mathrm{J_C H}$ for the carbons and between the hydrogen and C2, respectively. The magnitude of the chemical shift between the carbons is represented by $\Delta\omega$. Since the coupling and the chemical shift are given here in units of frequency, the horizontal line is the elapsed time.

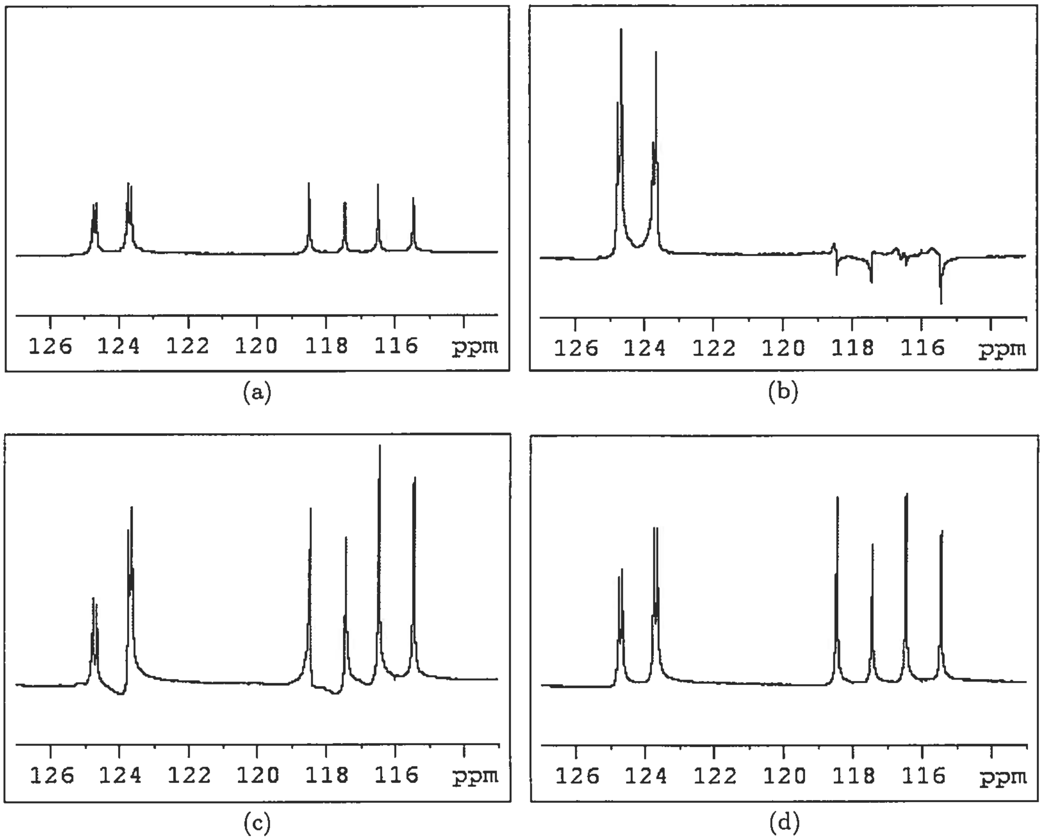
prior to going to the lab under various experimental scenarios (different values of t_1 and t_2, errors in calibration, etc.) with the NMRsim simulator provided by Bruker with its standard software.

6.5 Results and Interpretation

The results shown in Figure 6.3 were obtained in April 2002, at NMR facilities of the Département de chimie, at the Université de Montréal. By integrating the peak shapes for C1 and C2, we obtained values of $c_1 = 1.75$, $c_2 = 2.13$, and $c_3 = 3.72$ for $t_1 = t_2 = 5$s, where the c_3 is calculated as $4(1 - e^{-t/T1})$. The sum of squares is then

$$\sum_{i=1}^{3} c_i^2 = 21.44 > 18. \tag{6.8}$$

From Equation 6.7 we deduce that the final entropy H_f is therefore less than the initial entropy H_0 at the thermal state. We have thus successfully realised the first non-adiabatic spin cooling experiment, successfully beating the Shannon entropy bound.



(a) (b) (c) (d)

Figure 6.3: These are the spectra of the ^{13}C at different stages of the experiment. The intensity is in constant arbitrary units. (a) is the spectrum we get after waiting for the system to fully thermalise and then tipping the spins to the XY plane in order to observe them. We get (b) after the double INEPT sequence from H to C1 through C2. C1 has been polarised while C2 has lost almost all of its polarisation. (c) After waiting for 5 seconds to let H regain most of its polarisation, an INEPT from H to C2 is performed. (d) We wait for another 5 seconds, after which H has regained 93% of its equilibrium polarisation. The polarisation of the ^{13}C's is still higher than at equilibrium and the total entropy has gone down by approximately $\frac{3.5}{\ln 4}\varepsilon^2$.

Nonetheless, the experiment results are somewhat unsatisfactory for several reasons. First of all the decrease in entropy is only marginal. In an ideal scenario with infinite T1's for the ^{13}C and T1 = 0 for H, and a perfect implementation of the sequences, the sum of squares of Equation 6.8 can be as high as $4^2 + 4^2 + 4^2 = 48$. Our calculations had however indicated that with the measured T1times of Table 6.1, and with these t_1 and

t_2 values, we should have obtained relative polarisations values of slightly at least larger than 3 for both c_1 and c_2. This was unfortunately not the case. Having verified that the polarisation of C1 after waiting for t_1 had decreased as predicted by the measured T1 and, similarly, that the polarisation C2 after the second waiting time t_2 decreased as expected, the only explanation possible is that the problem resides with the "efficiency" of the transfer sequences.

In fact, the ultimate objective that we had set out to reach was to polarise the whole molecule to a bias of close to 4ε, allowing us then to obtain a final bias of $\approx 6\varepsilon$ on the C1, thus achieving with only three spins a single spin bias beyond that which could have been achieved according to the Sørensen bound. We went as far as designing a BCS-like sequence for TCE using NMRcalc, and a corresponding pulse program which we verified with NMRsim. We tested the sequence in the laboratory starting at the thermal state. According to Equation 5.22, with initial biases $\varepsilon, \varepsilon, 4\varepsilon$, we should have ended with 3ε polarisation on C1. The result was not even close, with significant leftover magnetisation out of phase and in the XY plane (even though the sequence was designed to bring it all back to the Z axis).

Both the failure of the BCS experiment and the inefficiency of the transfer sequence seem to indicate that the problem is that the effective T2 decoherence time, referred to as T2* is too small. The width of the peaks indicate a T2* of approximately 60–70 ms, which is very bad if compared with the "natural" T2 of 400–500 ms that the TCE nuclei should have. Considering that the BCE sequence is approximately 50 ms long, it is not surprising that it was not successful. The INEPT-based transfer sequences being much shorter, approx. 10–20 ms, were thus more successful, even though not as much we would need them to in order to obtain more decisive results.

The causes for such a low T2* could be varied and have not been fully identified. The author and his collaborators suspect that it is mainly due to bad experimental techniques and setup, such as poor calibration (shimming), impurities (other than CrAcAc) in the sample, inappropriate filling of the NMR sample tubes, etc. At this time, efforts are still ongoing at the Technion by some of the collaborators to repeat and improve on the above experiments. Improving these results with TCE and performing a BCS at the end of the experiment to obtain a final bias as close to 6ε as possible will continue to be our short-term experimental research objectives.

In the mid-term, and within the same line of research, we intend to explore the possibility of performing non-adiabatic cooling with other more suitable molecules with better η ratios, such as alanine, and in doing so identifying generic design and implementation techniques that could make this polarisation improvement tool available to the non-QC part of the NMR community.

In the long-term, we would like to explore the combination of Algorithmic Cooling with more drastic technological approaches such electron-nuclear resonance (ENDOR), and possibly solid-state NMR. Even though it is not clear whether the NMR approach will be the "winner" in the race to build scalable quantum computers, we believe that, no matter the technology, such bias improvement techniques will play a crucial role in the construction and operation of such machines. Furthermore, we suspect that the applications of such techniques will have much farther field of applications than Quantum Computing. This is why we intend to continue exploring both the theoretical and experimental aspects of Algorithmic Cooling.

Conclusions

Nihil sub sole nouum...

> What has been will be again, what has been done will be done again;
> *there is nothing new under the sun.*
>
> Is there anything of which one can say, "Look! This is something new"?
> It was here already, long ago; it was here before our time.
> —*The Koheleth*, Ecclesiastes 1:9.

Unfortunately, there are no earth shattering results on complexity theory in this thesis. Most of the work was already known, intuitively, by the experts in the field, and what was not could even be (and has been) considered trivial. That and more is true. Yet it was worth doing.

First and foremost because it needed to be done: somebody needed to tidy up the house. Gilles Brassard pointed out to me early on my studies that interference was the key to understanding the quantum speed up, and that made sense. How could one make this a stronger statement? The notion of "exorcising" the physics out of Quantum Computing was definitely inspired by Lance Fortnow. Regretably, his ideas were not done justice by the unfortunate complexity tradition of clinging to Turing Machine as the primary model of computation: circuits are far better suited to such generalisations as we have studied here. I do believe that even though nothing "new" was discovered, with this work we now are in a better position to introduce new minds to the field of Quantum Computing, having successfully removed some of the big stones and pitfalls within which one might fall (at least I did!).

> The more I know, the more I know that I know nothing —*Socrates*

Secondly, sometimes the journey matters more than the destination. Among other things it is the lessons learned along the way that are what make the journey worth it. But even more important thant is to discover that we *have not* learned, i.e. what lies ahead, what we researchers customarily calls the *open questions.* What better criterion to judge the quality of a work than by pondering the quality of what it has left undone? Before closing, we will thus come back briefly here to some of the more intriguing and genuinely new mysteries that this work has brought to the light of the Sun.

Sed sub luce solis, noui mysterii...

Extrinsic Quantum Models of Computation

In our quest to exorcise and unify the models of computation, we have come across a new non-standard model of computation, which we also called "extrinsic quantum" computation. It consists of a restriction of quantum computation with arbitrary quantum inputs (with descriptions that can be generated in poly-time), but limited in the dynamics to classical gates (permutations) and possibly phase-shifted permutation. We believe these models to be interesting, because despite their simplicity and apparent closeness to the standard quantum and classical models, no straightforward results of either strict inclusions or equality seem to be forthcoming.

Quaternionic Computing?

We have shown how a somewhat sensible model of computing can be constructed using quaternionic amplitudes. A crucial characteristic of this model is that due to the non-commutativity of quaternions, the output to the circuit will depend on the "evaluation path" of the circuit, as there is no unique circuit operator for all possible ways of re-combining the gates.

However, any such ordering of gates generates a well defined output, which we have shown can be simulated exactly and efficiently by a quantum circuit of similar size and width. This was our main result, which was inspired on a new proof we constructed for the equivalence of complex and real circuits.

We can interpret this theorem as a general result on quaternionic physical models as follows. If somehow Nature chooses and prefers *one* of the possible paths of evolution through the state space, then Nature's behaviour on such quaternionic systems can be efficiently simulated by a quantum system of similar complexity. This, provided that we somehow know which path is prefered.

If this were indeed the case (for example, because the physicists would tell us so), we complexity theorists might rub our hands together in satisfaction and further sing to the robustness of the BQP complexity class.

But what if nature somehow *did not* prefer nor chose one these paths, but somehow *followed them all at the same time.* Would there be any mechanism by which the results of the different computations would be weighed (by probabilities or probability amplitudes)?

Could these paths interfere with each other, in a similar fashion as in quantum models? Destructively? And if it were the case, could we ultimately harness such extra parallelism to achieve a speed-up beyond those achievable with quantum computation models?

The skeptics and realists among the readership might argue that all of these questions are completely sterile and void of interest, because there is not a single kopek (which is now much less than a cent) of evidence to suggest that such quaternionic models correspond to anything in Nature, and even less something in it that we could observe and even less harness for our purposes. If our only objective was to one day build a quaternionic computer, I could no nothing but agree with such skeptics.

Nevertheless, I believe that one of the major contributions of this work has been to find and identify a simple and easily explainable potential reason why there *should not be* quaternion amplitudes involved in Nature: the asymmetry of the possible evolution paths between two space-time events, even without relativistic effects. This, the physicist might argue, is the violation of some fundamental principle, and hence not possible nor likely.

We believe that the continued study of non-standard algebraic models such as those based on quaternions, but also on the reals, the octonions, and even possibly the finite fields will continue to bear fruits in that direction. More concretely, we hope that, at the very least, we might be able to provide more examples of such "weird properties," which can be expressed more simply in the language of information theory than in that of Physics. For example, it would already seem that the notions of entanglement in these different models display significant differences (i.e. over-entanglement in the quaternions, and sub-entanglement in the reals).

One journey ends, but a new day begins, under the Sun...

Bibliography

[Abr61] A. Abragam. *Principles of Nuclear Magnetism.* Oxford University Press, Oxford, England, 1961.

[ADH97] Leonard M. Adleman, Jonathan Demarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, October 1997.

[Aha03] Dorit Aharonov. A simple proof that Toffoli and Hadamard are quantum universal. `quant-ph/0301040`, Jan 2003.

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Electronic pre-print, available at `http://www.cse.iitk.ac.in/news/primality.html`, August 2002.

[AL98] Daniel S. Abrams and Seth Lloyd. Nonlinear quantum mechanics implies polynomial-time solution for NP-complete and #p problems. *Physical Review Letters*, 81:3992–3995, 1998. (also in `quant-ph/9801041`).

[ASV00] Andris Ambainis, Leonard J. Schulman, and Umesh V. Vazirani. Computing with highly mixed states. In *ACM Symposium on the Theory of Computing (STOC): Proceedings*, 2000.

[Bac60] Antonius Bacci. *Lexicon. Eorum uocabularum quae difficilis latine redduntur.* Societas Libraria «Studium», Rome, Italy, third edition, 1960.

[BB92] André Berthiaume and Gilles Brassard. The quantum challenge to structural complexity theory. In *Structure in Complexity Theory : Seventh Annual Conference : Proceedings*, pages 132–137, Boston, June 1992. IEEE Computer Society Press.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, October 1997.

[BBC$^+$95] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995.

[BCJ$^+$99] S.L. Braunstein, C.M. Caves, R. Jozsa, N. Linden, S. Popescu, and R. Schack. Separability of very mixed states and implications for NMR quantum computing. *PRL*, 83:1054–1057, 1999.

[BDG88] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, Germany, 1988.

[BFH02] Martin Beaudry, José M. Fernandez, and Markus Holzer. A common algebraic description for probabilistic and quantum computations. `quant-ph/0212096`, December 2002.

[BFLW03] Gilles Brassard, José M. Fernandez, Raymond Laflamme, and Yossi Weinstein. IMproved POlarization Transfer wtih ENvironment Thermalization. Manuscript in Preparation, 2003.

[BMP$^+$00] Patrick Oscar Boykin, Tal Mor, M. Pulver, Vwani Roychowdhury, and Farokh Vatan. A new universal and fault-tolerant quantum basis. *Information Processing Letters*, 75(3):101–107, August 2000.

[BMR$^+$02] P. Oscar Boykin, Tal Mor, Vwani Roychowdhury, Farrokh Vatan, and Rutger Vrijen. Algorithmic cooling and scalable NMR quantum computers. *Proceedings of the National Academy of Sciences*, 99(6):3388–3393, March 2002. (also in `quant-ph/0106093`).

[Boy03] Michel Boyer. Sur un résultat de José Manuel Fernandez. Unpublished Manuscript, February 2003.

[Bra01] Gilles Brassard. *Quantum Information Process for Computer Scientists.* MIT Press, 2001. In preparation.

[Bra03] Samuel Braunstein. Private communication, July 2003.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, October 1997.

[CD96] Richard Cleve and David P. DiVincenzo. Schumacher's quantum data compression as a quantum computation. *Physical Review A*, 54(4):2636–2650, October 1996. (also in `quant-ph/9603009`.

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A*, 454:339–354, 1998. (also in `quant-ph/9708016`).

[Cle99] Richard Cleve. An introduction to quantum complexity theory. In Chiara Machiavello, Palma G.M., and Anton Zeilinger, editors, *Collected Papers on Quantum Computation and Quantum Information Theory*. World Scientific, 1999. (also in `quant-ph/9906111`).

[CLK$^+$99] D.G. Cory, R. Laflamme, E. Knill, L. Viola, T.F. Havel, N. Boulant, G. Boutis, E. Fortunato, S. Lloyd, R. Martinez, C. Negrevergne, M. Pravia, Y. Sharf, G. Teklemariam, Y.S. Weinstein, and W.H. Zurek. NMR based quantum information processing: Achievements and prospects. *Prepared for Fortschritte der Physik special issue, Experimental Proposals for Quantum Computation*, April 1999.

[CVS01] Derek E. Chang, Lieven M.K. Vandersypen, and Matthias Steffen. NMR implementation of a building block for scalable quantum computation. *Chemical Physics Letters*, 338:337–344, Nov 2001. (also in `quant-ph/0011055`).

[Deu85] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400:96–117, 1985.

[Deu89] David Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A*, 425:73–90, 1989.

[ESY84] S. Even, A. L. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61:159–173, 1984.

[FLMR03] José M. Fernandez, Seth Lloyd, Tal Mor, and Vwani Rowchoudury. Algorithmic cooling of spins: A practicable method for increasing polarisation. Unpublished Manuscript, November 2003.

[For97] Lance Fortnow. Counting complexity. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 81–107. Springer-Verlag, 1997. Survey.

[For00] Lance Fortnow. One complexity theorist's view of quantum computing. In *AQIP'99*, March 2000.

[For02] Lance Fortnow. Private communication, June 2002.

[FR98] Lance Fortnow and John Rogers. Complexity limitations on quantum computation. In *Thirteenth Annual IEEE Conference on Computational Complexity : Proceedings*, pages 202–209, Los Alamitos, CA, June 1998. IEEE Computer Society.

[FS03] José M. Fernandez and William A. Schneeberger. Quaternionic computing. `quant-ph/0307017`, July 2003.

[GL96] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Mathematical Sciences. Johns Hopkins University Press, 1996.

[Gol92] J. S. Golan. *The theory of semirings with applications in mathematics and theoretical computer science.* Pitman Monographs & Surveys in Pure and Applied Mathematics. Longman Scientific & Technical, 1992.

[Gra81] A. Graham. *Kronecker Products and Matrix Calculus With Applications.* Mathematics and its Applications. Ellis Horwood, 1981.

[HPS83] H. V. Henderson, F. Pukelsheim, and S. R. Searle. On the history of the Kronecker product. *Linear and Multilinear Algebra*, 14:113–120, 1983.

[Hug89] R.I.G. Hughes. *The Structure and Interpretation of Quantum Mechanics.* Harvard University Press, Cambridge, MA, 1989.

[Ish95] Chris J. Isham. *Lectures on Quantum Theory.* Imperial College Press, London, 1995.

[Jon00] Jonathan A. Jones. NMR quantum computation: a critical evaluation. `quant-ph/0002085`, February 2000.

[Kit97] Alexei Yu. Kitaev. Quantum computations : Algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

[KL98] Emmanuel Knill and Raymond Laflamme. Power of one bit of quantum information. *PRL*, 81(25):5672–5675, 1998.

[KL01] Emmanuel Knill and Raymond Laflamme. Quantum computing and quadratically signed weight. *Information Processing Letters*, 79(4):173–179, August 2001. (also in quant-ph/9909094).

[KLMT99] E. Knill, R. Laflamme, R. Martinez, and C.-H. Tseng. A cat-state benchmark on a seven bit quantum computer. quant-ph/9908051, August 1999.

[KS86] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1986.

[MCRS00] A. Muthukrishnan and Jr. C. R. Stroud. Multi-valued logic gates for quantum computation. *Physical Reviews, Series A*, 62:052309–1–8, 2000.

[MF79] G.A. Morris and R. Freedman. Enhancement of nuclear magnetic resonance signals by polarization transfer. *J. Am. Chem. Soc.*, 101:760–762, 1979.

[Mik00] Alexandre Mikaelian. Private communication, 2000.

[OED89] *Oxford English Dictionnary*. Oxford University Press, Oxford, UK, 2nd edition, 1989.

[Pou02] David Poulin. Private Communication, November 2002.

[Rab76] M.O. Rabin. Probabilistic algorithms. In J. Traub, editor, *Algorithms and Complexity: New Directions and Results*, pages 21–39. Academic Press, London, UK, 1976.

[RG02] Terry Rudolph and Lov Grover. A 2 rebit gate universal for quantum computing. quant-ph/0210187, October 2002.

[Sch95] Benjamin Schumacher. Quantum coding. *Physical Review A*, 51(4):2738–2747, April 1995.

[SE83] O.W. Sørensen and R.R. Ernst. *Journal of Magnetic Resonance*, 51:477–489, 1983.

[Sel88] A. L. Selman. Promise problems complete for complexity classes. *Information and Computation*, 78:87–98, 1988.

[SH87] Jeremy K.M. Sanders and Brian K. Hunter. *Modern NMR Spectroscopy: A Guide for Chemists*. Oxford University Press, Oxford, England, 1987.

[Sha48] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July and October 1948.

[Shi02] Yaoyun Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computation. *Quantum Information and Computation*, 3(1):84–92, 2002. (also in `quant-ph/0205115`).

[Sip96] Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, Boston, MA, USA, 1996.

[Soc57] Philippus Soccorsi. *De Physica Quantica.* Pontificia Universitas Gregoriana, Vatican, 1957.

[Sør89] Ole W. Sørensen. Polarization transfer experiments in high-resolution NMR spectroscopy. *Progress in NMR Spectroscopy*, 21:503–569, 1989. Submitted as a Habilitation to the Abteilung für Chemie at the ETH Zurich.

[Sør90] Ole W. Sørensen. A universal bound on spin dynamics. *Journal of Magnetic Resonance*, 86:435–440, 1990.

[SS77] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAMJC*, 6:84–85, 1977.

[SV99] Leonard J. Schulman and Umesh V. Vazirani. Scalable NMR quantum computation. In *ACM Symposium on the Theory of Computing (STOC): Proceedings*, pages 322–329, 1999.

[VA03] Felisa Vázquez-Abad. Private communication, February 2003.

[Web03] *Merriam–Webster Online Dictionnary.* Merriam–Webster, Springfield, MA, USA, 2003.

[Yao93] Andrew Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 352–361, Los Alamitos, CA, November 1993. IEEE Computer Society Press.

[YS87] Charles H. Yoder and Charles D. Schaeffer, Jr. *Introduction to Multinuclear NMR Spectroscopy.* Benjamin Cummings Publishing Company, Menlo Park, CA, USA, 1987.