Computing NP-complete problems in polynomial time by means of Physics

Trabajo de Fin de Grado
Doble Grado en Ingeniería Informática y Matemáticas

UNIVERSIDAD
COMPLUTENSE
MADRID

Autor:
Víctor Carrillo Redondo

Tutores:
Ismael Rodríguez Laguna
Javier Rodríguez Laguna

Departamento de Sistemas Informáticos y Computación

September 2021

"*If computer scientists had been physicists, we'd simply have declared* $\mathbf{P} \neq \mathbf{NP}$ *to be an observed law of Nature. A Nobel Prize would even be gifted for the discovery of that law. And in the unlikely event that someone later proved* $\mathbf{P} = \mathbf{NP}$*, a second Nobel Prize would be awarded for the law's overthrow*"

Scott Aaronson [3]

Abstract

Can NP-complete problems be solved efficiently in the physical universe? Some researchers have claimed to be able to solve NP-complete problems in polynomial time by encoding the problem in the state of a physical system and letting it evolve naturally, according to the laws of physics. However, their proposals have not proven to be very effective in practice. Additionally, there are several reasons to believe that those methods would not work if $\mathsf{P} \neq \mathsf{NP}$.

We present some physical assumptions (both from classical physics and quantum mechanics) that would allow us to provably solve NP-complete problems in polynomial time by means of Physics, even if $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{NP} \not\subset \mathsf{BQP}$. We also study if our proposals are consistent with currently known laws of Physics.

Keywords: Turing machine, cellular automaton, NP-complete, classical physics, quantum mechanics, computational complexity.

Resumen

¿Podemos resolver problemas NP-completos en tiempo polinómico dejando que ciertos entornos físicos evolucionen de manera natural conforme a las leyes de la física? Varios estudios así lo parecen afirmar, sin embargo, los métodos propuestos no han resultado ser muy efectivos en la práctica, y de hecho hay razones para pensar que no sería posible si $\mathsf{P} \neq \mathsf{NP}$.

Planteamos algunas suposiciones físicas (basadas tanto en la física clásica como en la mecánica cuántica) que nos permitirían resolver problemas NP-completos en tiempo polinómico en un Universo con dichas características, aunque $\mathsf{P} \neq \mathsf{NP}$ y $\mathsf{NP} \not\subset \mathsf{BQP}$, y tratamos de estudiar si dichas suposiciones son consistentes con principios conocidos de la Física.

Palabras clave: máquina de Turing, autómata celular, NP-completo, física clásica, mecánica cuántica, complejidad computacional.

Agradecimientos

Gracias a mi familia por haberme apoyado siempre.

Gracias Adrián, Redondo, Casado, Dani, Antonio y Sergio por tantos años aguantándonos (y los que quedan).

Gracias a todos los que habéis aparecido en mi vida a lo largo de estos seis años en Madrid. Gracias Esther, Alex, Carlos, Chema y Jaime, entre muchos otros.

Gracias también a mis compañeros en Devo. A todos los malotes, y en particular a Gonzalo, Juan, Jorge, Fayán, Paula y Paco por vuestro apoyo y por todo lo que estoy aprendiendo de vosotros, no solo sobre desarrollo *software*.

Y por último, gracias a Ismael por introducirme en el mundo de la complejidad computacional y por la pasión que transmite por estos temas. Gracias también a él y a Javier por su apoyo y ayuda durante el desarrollo de este trabajo. Y por todas las interesantes charlas e ideas que se nos iban ocurriendo. Espero que sigamos trabajando juntos en el futuro.

Contents

List of Figures

Al Víctor del pasado.
Lo conseguiste.

Introduction

The concept of NP-completeness was born in 1971, when Cook [16] proved that every problem in NP is reducible in polynomial time to the Boolean Satisfiability Problem, or SAT, as we shall denote it. That means that, in a certain way, SAT is harder that any other problem in NP, because if you could solve it in polynomial time, then you could solve any other NP problem in polynomial time. That was a groundbreaking discovery, because the notion of NP-completeness seemed to characterize the class of problems for which we can efficiently verify whether a claimed solution is valid, but, however, for which there is no better algorithm than exhaustive search over a solution space that grows exponentially on the size of the instance. Moreover, in the following years a lot of important problems were proved to be NP-complete, giving evidence of their hardness. For example, the Protein Folding Problem [9, 17], of great interest to the pharmaceutical industry for the discovery and development of new drugs.

The consequences of proving that $\mathsf{P} = \mathsf{NP}$ (and that there existed a practical algorithm for, *e.g.*, SAT) would be much more profound than, for example, breaking most cryptographic protocols that we currently use on the Internet (for example, to buy safely online). It would have almost metaphysical consequences. Scott Aaronson [1] said:

> If $\mathsf{P} = \mathsf{NP}$, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in "creative leaps", no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss.

Even Gödel noticed it long before the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ question was formulated. In a letter to von Neumann [18] in 1956, Gödel asked him about the computational complexity of deciding if a first-order formula has a proof of length at most n, a problem now known to be NP-complete [11]. He mentions what would be the implications of having a linear or quadratic time algorithm for that problem:

> If there were really a machine with $\varphi(n) \sim K \cdot n$ (or even with $\sim K \cdot n^2$), this would have consequences of the greatest importance. It would apparently mean that in spite of the unsolvability of the Entscheidungsproblem, the reasoning of mathematicians about yes-or-no questions can be completely replaced by machines. [...] Now it seems to me, however, to be completely within the realm of possibility.

Despite all the evidence gathered against $\mathsf{P} = \mathsf{NP}$ since 1971, no one has been able to solve the problem yet. Moreover, complexity theorists have identified several reasons why solving it is so hard [6, 25]. So it is natural to think of it as one of the deepest problems in mathematics. Even a one million dollar prize is awarded to anyone who can solve it [14].

That apparent difficulty to practically solve NP-complete problems motivated some researchers to study if NP-complete problems could be solved by Nature in polynomial time by encoding an instance of the problem in the state of a physical system and letting it evolve naturally, according to the laws of physics [10, 27, 30]. The idea is to exploit Nature's inherent parallelism to tackle the exponential explosion of cases apparently required to be able to solve these problems.

However, those proposals have not proven to be very effective in practice. Moreover, there are several reasons to believe that those methods would not help us to solve NP-complete problems efficiently if $\mathsf{P} \neq \mathsf{NP}$.

Our objective then is to study what physical assumptions (if any) would allow us to provably solve NP-complete problems in polynomial time by means of Physics, even if $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{NP} \not\subset \mathsf{BQP}$, trying to relax them in order to identify the weakest hypotheses that we would need to make and study if they would be consistent with currently known laws of Nature. This way we will try to show that, if those proposals to solve NP-complete problems in polynomial time worked, then we would be living in a Universe very different to the one we think we know.

The text is divided as follows. In Chapter 1 we will introduce the concepts and the computational models that we will be using throughout the work and study their relationship. In Chapter 2 we will suggest three assumptions from classical physics that we think would allow us to provably solve NP-complete problems in polynomial time, the superdense information assumption, the hyperbolic geometry assumption (these two proposals had already been studied) and the instantaneous communication assumption. Next, we will dive into the quantum world in Chapter 3 and, after reviewing the basics of quantum mechanics (and its relation with quantum computation), we will propose some variations on the quantum theory that would let us solve NP-complete problems in quantum polynomial time. Finally, we will share what conclusions we drew from our research and what could be our next steps to continue our study.

Chapter 1

Preliminaries

The mathematical model that we will use to study computation is the Turing machine. As we will see, this apparently simple model seems able to simulate any other "reasonable" model of computation, and moreover, with little loss of efficiency. That last attribute of Turing machines will be crucial for us, because then, the set of "efficiently solvable" computational tasks is at least as large for the Turing machine as for any other "reasonable" model of computation.

Definition 1.1 (Turing machine)**.** A Turing machine consists of $k \geq 1$ tapes, each of which is an infinite line of cells that can contain a symbol from a finite alphabet. Each tape also has a tape head that can read or write symbols to the tape one cell at a time. These tapes are used to store intermediate results during the computation.

Formally, a k-tape Turing machine M is a tuple (Σ, Q, δ), where:

- Σ is the alphabet of the Turing machine, that is, a finite set of symbols that the tapes of M can contain. We assume that Σ always contains the blank symbol, denoted by $\#$.
- Q is a finite set of states in which the machine M can be. We assume that Q contains a special state, the start state q_{start}.
- δ is the transition function of M, which intuitively works as the program of the Turing machine. More specifically, δ is a function

$$\delta \colon Q \times \Sigma^k \to \big(Q \cup \{q_{\text{halt}}, q_{\text{yes}}, q_{\text{no}}\}\big) \times \Sigma^k \times \{\leftarrow, -, \rightarrow\}^k$$

  that describes the rules that M follows in each step.

If currently the machine is in state $q \in Q$, $(\sigma_1, \ldots, \sigma_k) \in \Sigma^k$ are the k symbols being read by the k tape heads and $\delta\big(q, (\sigma_1, \ldots, \sigma_k)\big) = \big(q', (\sigma'_1, \ldots, \sigma'_k), d\big)$, for $d \in \{\leftarrow, -, \rightarrow\}^k$, then, at the next step, the machine will be in state q', the symbols read by the tape will be replaced by the $(\sigma'_1, \ldots, \sigma'_k)$ symbols and each of the k heads will move left, right or will not move depending on the value of $d \in \{\leftarrow, -, \rightarrow\}^k$.

All tapes except the first one are initialized to have the blank symbol, $\#$, in every location, while the first tape (the input tape) initially contains the input string and the blank symbol on the rest of its cells. The first tape's head starts at the beginning of the input string. And the starting state of M is q_{start}.

From this initial configuration, the machine will now continue its execution according to the specification given by the "program" δ until the state reached is either of $\{q_{\text{halt}}, q_{\text{yes}}, q_{\text{no}}\}$, in which case the machine stops its execution, and we will say that it has halted. Moreover, if we reached state q_{yes} we will say that the machine accepts

its input; and if we reached state q_{no} we will say that the machine rejects its input. In either case, we can define the output of the machine on input $x \in \Sigma^*$: if M accepts or rejects x, then $M(x) = 1$ or 0, respectively; and if M halts on input x and the content of the last tape is a string y (ignoring the blank characters of the rest of the tape), then $M(x) = y$. It is also possible that M never halts on input x, and in this case we will write that $M(x) = \nearrow$, but we will only be interested in studying Turing machines that will always eventually halt.

From the previous definition it is not clear that a 1-tape Turing machine would define a computational model as powerful as a k-tape Turing machine for $k > 1$, but it turns out that a 1-tape Turing machine can simulate a k-tape Turing machine with little loss of efficiency (in particular, at most a quadratic slowdown [24, p. 30]) for any $k > 1$. For that reason, we will simply refer to a k-tape Turing machine as a Turing machine when we do not care about k.

Unless explicitly stated, we will suppose that the alphabet of a Turing machine is $\{0, 1, \#\}$. This does not imply any loss of generality, because a Turing machine having any alphabet Σ can be simulated efficiently by a Turing machine with alphabet $\{0, 1, \#\}$ (see [5, p. 16]).

Definition 1.2. Let $f\colon \{0,1\}^* \to \{0,1\}^*$ and $T\colon \mathbb{N} \to \mathbb{N}$ be two functions and let M be a Turing machine. We say that M computes f if for every $x \in \{0,1\}^*$, $M(x) = f(x)$. Moreover, we say that M computes f in $T(n)$-time if its computation on every input $x \in \{0,1\}^*$ requires at most $T\left(|x|\right)$ steps.

In particular, we will say that M decides a language $L \subset \{0,1\}^*$ if it computes the function $f_L\colon \{0,1\}^* \to \{0,1\}$ such that $f_L(x) = 1 \iff x \in L$.

Definition 1.3. Let $T\colon \mathbb{N} \to \mathbb{N}$ be a function. A language L is in $\mathbf{DTIME}\left(T(n)\right)$ if there is a Turing machine that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L.

The next definition aims to capture mathematically the set of languages that can be efficiently decided.

Definition 1.4. We define the complexity class $\mathbf{P}$ to be

$$\mathbf{P} = \bigcup_{c \geq 1} \mathbf{DTIME}\left(n^c\right).$$

The main character of this text, the complexity class $\mathbf{NP}$, is defined as follows.

Definition 1.5. A language $L \subset \{0,1\}^*$ is in $\mathbf{NP}$ if there exists a polynomial $p\colon \mathbb{N} \to \mathbb{N}$ and a Turing machine M running in polynomial time such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \exists u \in \{0,1\}^{p\left(|x|\right)} \text{ such that } M(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p\left(|x|\right)}$ satisfy $M(x,u) = 1$, then we call u a certificate for x (with respect to the language L and machine M).

Clearly, $\mathbf{P} \subset \mathbf{NP}$ because we can choose p to be the zero polynomial (that is, we do not need any certificate).

We will now introduce the concept of $\mathbf{NP}$-completeness, which intuitively represents the hardest problems in $\mathbf{NP}$.

Definition 1.6. A language $L \subset \{0,1\}^*$ is polynomial-time reducible to a language $L' \subset \{0,1\}^*$, denoted by $L \leq_p L'$, if there is a polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in L$ if and only if $f(x) \in L'$.

We say that a language L is **NP**-complete if $L \in \mathbf{NP}$ and $L' \leq_p L$ for every $L' \in \mathbf{NP}$.

It is easy to see that the relation given by $\leq_p$ is transitive, and that if a language L is **NP**-complete, then $L \in \mathbf{P}$ if and only if $\mathbf{P} = \mathbf{NP}$. That means that **NP**-complete problems are, in a sense, the hardest problems in **NP**.

In 1971, Cook [16] introduced the concept of **NP**-completeness and gave the first example of such a problem, the Boolean Satisfiability Problem, or SAT, that is, determining if for a given Boolean formula there exists an assignment of its variables that make it true (*i.e.*, if it is satisfiable).

Soon after that discovery, many other problems were showed to be **NP**-complete by reducing them to other already known **NP**-complete problems. The fact that a language is **NP**-complete, together with the conjecture that $\mathbf{P} \neq \mathbf{NP}$ can be viewed as evidence that the language cannot be decided in polynomial time. Another example of an interesting **NP**-complete problem for us is CNF-SAT, which is the problem of determining if a given Boolean formula in CNF is satisfiable. A Boolean formula is in CNF if it has the form

$$\bigwedge_i \left(\bigvee_j l_{i_j} \right),$$

where each l_{i_j} is either a variable x or its negation $\neg x$. The terms l_{i_j} are called literals of the formula and the terms $\vee_j l_{i_j}$ are called the clauses of the formula.

At first sight, it is not clear from its definition whether Turing machines can encapsulate the notion of arbitrary algorithms. However, we will now see that Turing machines are powerful enough to simulate some simple programming languages.

Definition 1.7 (Random access machine [24])**.** A random access machine, or RAM, is a model of computation that consists of a program acting on an array of registers, each capable of containing an arbitrarily large integer, possibly negative.

Formally, a RAM program is a finite sequence of instructions $(\pi_1, \ldots, \pi_m)$, where each instruction π_i is of one of the types shown in Figure 1.1. At each point during a RAM computation, instruction π_κ is executed, where κ is the program counter, and its value is increased by one after the execution of each instruction (except for instructions of the last five, that already change its value in another way). The program keeps executing instructions until it halts.

Note that we can access the value of the registers directly (access the j-th register for an integer j), or indirectly (access the j-th register, where j is the content of the k-th register for an integer k). Also notice that all logical and arithmetic operations take place in register 0.

The input of a RAM program is a finite sequence of integers, contained in a finite array of input registers, $I = (i_1, \ldots, i_n)$, whose value can be accessed through READ instructions.

Let D be a finite set of finite sequences of integers and let $\phi\colon D \to \mathbb{Z}$. We say that a RAM program Π computes ϕ if, for any $I \in D$, the value stored in register 0 before halting is $\phi(I)$.

Instruction	**Operand**	**Semantics**
READ	j	$r_0 \coloneqq i_j$
READ	$\uparrow j$	$r_0 \coloneqq i_{r_j}$
STORE	j	$r_j \coloneqq r_0$
STORE	$\uparrow j$	$r_{r_j} \coloneqq r_0$
LOAD	$= j$	$r_0 \coloneqq j$
LOAD	j	$r_0 \coloneqq r_j$
LOAD	$\uparrow j$	$r_0 \coloneqq r_{r_j}$
ADD	$= j$	$r_0 \coloneqq r_0 + j$
ADD	j	$r_0 \coloneqq r_0 + r_j$
ADD	$\uparrow j$	$r_0 \coloneqq r_0 + r_{r_j}$
SUB	$= j$	$r_0 \coloneqq r_0 - j$
SUB	j	$r_0 \coloneqq r_0 - r_j$
SUB	$\uparrow j$	$r_0 \coloneqq r_0 - r_{r_j}$
HALF		$r_0 \coloneqq \lfloor r_0/2 \rfloor$
JUMP	j	$\kappa \coloneqq j$
JPOS	j	**if** $r_0 > 0$ **then** $\kappa \coloneqq j$
JZERO	j	**if** $r_0 = 0$ **then** $\kappa \coloneqq j$
JNEG	j	**if** $r_0 < 0$ **then** $\kappa \coloneqq j$
HALT		$\kappa \coloneqq 0$

FIGURE 1.1: RAM instructions and their semantics.

The running time of a RAM program is the number of instructions executed, and we will measure it as a function in the length of its input, $l(I)$, defined as

$$l(I) = l(i_1, \ldots, i_n) = \sum_{j=1}^{n} l\left(i_j\right),$$

where $l(i)$ for an integer i is the length of its binary representation with no redundant leading 0s and with a minus sign in front, if negative. That is, let $f\colon \mathbb{N} \to \mathbb{N}$ and suppose that a RAM program Π computes a function $\phi\colon D \to \mathbb{Z}$ such that for any $I \in D$, the RAM executes k instructions before halting, where $k \leq f\left(l(I)\right)$. Then we say that Π computes ϕ in time $f(n)$.

And it turns out that a Turing machine can simulate any RAM program with only a polynomial loss of efficiency [24, Theorem 2.5].

Theorem 1.8. *A RAM program computing a certain function ϕ in time $f(n)$ can be simulated by a 7-tape Turing machine in time $\mathcal{O}(f(n)^3)$.*

The converse is also true: a Turing machine can be simulated with a RAM program in linear time [24, Theorem 2.4].

This shows that the class of functions computable in polynomial time on a RAM is the same as class of functions computable in polynomial time on a Turing machine, so the class of problems efficiently solvable on each model is the same.

Definition 1.9 (Cellular automaton)**.** A cellular automaton is a tuple $A = (d, S, N, f)$, where:

- d is the dimension of A.

- S is a finite set of the possible states of each cell of A. We assume that there is an empty state, represented by #.
- $N = \{n_1, \ldots, n_v\}$ is a set of neighbourhood vectors in $\mathbb{Z}^d$ that indicates the relative position of the neighbours of each cell.
- $f\colon S^{v+1} \to S$ is the local rule of the cellular automaton that specifies how the state of a specific cell evolves depending on its state and the state of its neighbours. We will suppose that f will not change the state of an empty cell whose neighbourhood is also empty.

At each time step, the state of each cell will be transformed depending of the current state of the cell and the state of its neighbours, according to the local rule f.

We will be interested in two-dimensional cellular automata with the Moore neighbourhood (see Figure 1.2), that is, $d = 2$ and

$$N = N_M \coloneqq \{(-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1,0), (1,1)\}.$$

The local rule f of a cellular automaton will be specified as a finite set of rules of the form shown in Figure 1.2, where the value on each cell represents a state from S, and the rule specified is applied to those cells whose neighbourhood match the states shown in the rule. If none of the rules apply, then the cell does not change its state. We shall write the symbol '$*$' to represent that any state from S is valid.

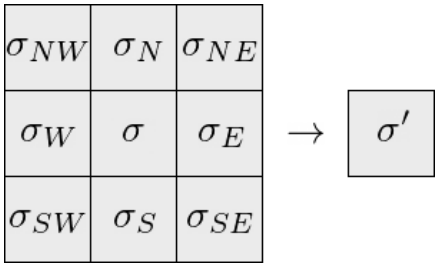


FIGURE 1.2: Form of the rules of a cellular automaton. In this case the cell with state σ would change to have state σ'.

This apparently simple computational model is powerful enough to simulate a Universal Turing machine efficiently [13, 22]. Moreover, we can simulate the execution of a cellular automaton on a Turing machine with at most a polynomial slowdown, having in particular the following result.

Theorem 1.10. *Let $A = (2, S, N_M, f)$ be a cellular automaton. The execution of t time steps of A can be simulated by a Turing machine in a number of steps polynomial in t.*

The idea of the proof is to write a RAM program that simulates the execution of t time steps of A and then simulate that RAM program on a Turing machine in polynomial time as shown in Theorem 1.8. The RAM program that simulates the execution of A would just iterate over all non-empty cells (that we know that are initially finite), updating their state according to the local rule f (this would require to have a copy of the states of all non-empty cells to not overwrite them during the computation). It is also important to note the fact that we can access registers indirectly in RAM programs and that the size of the non-empty area may grow after

every time step, however, after t steps it can only grow a number of cells that is quadratic on t. Thus the algorithm would only require $\mathcal{O}(t^3)$ steps.

This result will allow us to work with the cellular automaton model of computation without worrying about computational complexity issues. If the number of steps needed to reach a specific configuration in a cellular automaton is polynomial on the size of the input (the initial state of the non-empty cells), then we know that we can simulate its execution in polynomial time on a Turing machine. This means that if we modelled Nature as a cellular automaton, then we would not be able to solve **NP**-complete problems in polynomial time by means of Physics if $\mathbf{P} \neq \mathbf{NP}$. In Section 2.3 we will study an extension of the cellular automaton model of computation based on the physical assumption that instantaneous communication is possible in our Universe.

Chapter 2

Classical physics

In this chapter we will propose three assumptions about the physical Universe that would allow us to provably solve NP-complete problems in polynomial time using the resources of Nature, supposed that $\mathsf{P} \neq \mathsf{NP}$. These are the superdense information assumption, the hyperbolic geometry assumption and the instantaneous communication assumption. The first two proposals have already been studied, so we will just mention a few results. The last proposal will be discussed in more detail.

2.1 Superdense information

We describe the superdense information assumption as the ability of storing and manipulating arbitrarily big amounts of information in a finite region of space and constant time.

Such assumption would allow us to solve NP-complete problems in polynomial time by means of Physics by encoding an instance of CNF-SAT in a huge integer (of exponential size) and using appropriate arithmetic operations (accessing the bit representation of the number) to evaluate a Boolean formula in parallel on all its possible assignments. Schönhage [28] showed how to do it given the ability to compute $x + y$, $x - y$, $x \cdot y$, x/y and $\lfloor x \rfloor$ in constant time. He also proved that PSPACE-complete problems can be solved in polynomial time using a similar technique.

However, this assumption is not valid with our current understanding of the Universe, since it violates some well-established principles of Physics, in particular the Bekenstein bound [7]. This principle states that the Shannon entropy, *i.e.*, the amount of information, that can be contained within a given finite region of space that has a finite amount of energy is bounded.

2.2 Hyperbolic geometry

According to general relativity, the density of matter and energy in the Universe determine the curvature (and shape) of space. Depending on that, the curvature of the Universe could either be zero (the Universe is flat), positive (the Universe is locally a region of a sphere) or negative (the Universe is locally a region of a hyperboloid); and each possibility would have consequences on the geometry of space: we would have Euclidean, spherical or hyperbolic geometry, respectively.

Experiments show that our Universe has zero curvature in average (up to some experimental error), so it would seem contradictory to expect it to have a non-Euclidean

geometry. However, would it have any consequence to computation if we dropped that assumption?

The answer is yes, as shown by Margenstern and Morita [21]. They proved that NP-complete problems can be solved in polynomial time in a cellular automaton defined on a grid of the hyperbolic plane. The idea of their proof, and the reason why it is important to do it in a hyperbolic space, is that then the number of neighbours of each cell grows fast enough as we move away from the origin that we can reach an exponential number of cells on the distance covered.

2.3 Instantaneous communication

Would the ability of sending information arbitrarily fast (that is, the instantaneous communication assumption) allow us to provably solve NP-complete problems in polynomial time using the resources of the physical Universe?

Based on an extension of the 2-dimensional cellular automaton model of computation where we can take advantage of the instantaneous communication assumption, we will show a polynomial time algorithm for CNF-SAT.

The extension consists on a new way of changing the state of a cell in addition to the evolution rules that apply locally. Now, a cell can send a message (identifying a state of the cellular automaton) a specific number of cells away in the left or right direction. More specifically, these new rules are described as follows:

FIGURE 2.1: If the current cell is in state σ, then the state of the cell being c positions to the left (or to the right, respectively) will change its state to $\overline{\sigma}$. More specifically, c will be of the form $c = 2^k$, where k is a counter that will be increased by one after each time step and that we can initialize back to 0 when needed.

Note that when a cell receives a message, its state during the next step of the computation will be the one received, regardless of any other rules that may apply to the destination cell.

Next we will specify our algorithm for CNF-SAT. But first we want to informally describe how it works in Section 2.3.1 and then we will show the algorithm in detail in Section 2.3.2. Moreover, to make the explanation easier, we will assume that we can change the local rule of the cellular automaton during its computation (after a specific number of time steps). Later in Section 2.3.3 we will show why we can make that assumption.

2.3.1 Informal explanation of the algorithm

Let φ be a CNF Boolean formula of n variables and m clauses. We want to determine if φ is satisfiable or not. The idea of our algorithm is to exploit the parallelism of cellular automata together with the ability of sending messages arbitrarily far away (in the left or right direction) to evaluate φ on every possible assignment at the same time and then check if any of them actually make the formula true.

The input of our cellular automaton will be the CNF formula φ encoded such that clauses are separated with a blank symbol, and each clause is encoded as a string of length n, where the i-th symbol indicates whether the i-th variable appears in the clause as a positive literal, as a negative literal or does not appear, having a value of 1, -1 or 0, respectively.

Note that the proposed encoding of a CNF Boolean formula of n variables and m clauses is a string of $(n+1)\,m-1$ symbols. This is a reasonable encoding, as we will prove later (see page 12), because the size of the encoded string grows at most quadratically compared to other standard encodings like the DIMACS CNF encoding [19].

The idea is to generate a string for each possible assignment of the n variables of φ, encoding it so that we can easily evaluate the Boolean formula in parallel. Given an assignment of the n variables, we will encode it as another string of length $(n+1)\,m-1$ divided into m substrings of length n separated by a blank symbol. Each of those m substrings will be identical to the others, because its i-th symbol will indicate the value assigned to the i-th variable (0 or 1). Note that, then, if we see both the encoding of φ and the encoding of a certain assignment in vertical and one string next to the other (see Figure 2.2a), we would have that, for each clause of the formula, next to the symbol encoding if the i-th variable appears as a literal in that clause there is a symbol encoding the value assigned to that variable, so we can easily substitute the value in the formula with an appropriate rule of the cellular automaton (see Figure 2.2b). Then we would just need to check if any literal in each clause is true and then if every clause is true to decide if the given assignment satisfies φ.

1	0
0	0
−1	0
0	1
0	0
−1	0
1	0
1	1

(a) Encoding of the Boolean formula (left) next to the encoding of the assignment (right).

0	0
0	0
1	0
0	1
0	0
1	0
0	0
1	1

(b) After substituting the value of each variable in the formula we are left with a binary string that can be evaluated as we will show later.

FIGURE 2.2: Encoding of the Boolean formula $(x_1 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \wedge x_4)$ and of the assignment $x_1 = x_2 = x_3 = 0, x_4 = 1$ before and after substituting each variable with the corresponding value.

Now the problem is how to generate such a configuration where we have the encoding of the input formula next to the encoding of a unique assignment in 2^n different locations of the cellular automaton grid. First we will generate the encoding of the assignments $x_1 = \cdots = x_n = 0$ and $x_1 = 1$, $x_2 = \cdots = x_n = 0$, and then, sending the appropriate messages, we will be keeping the invariant that at each step we have generated all the possible assignments for the first k variables (and the rest of them

having value 0), so in n steps we would have generated all 2^n possible assignments to φ. We will specify it later.

Finally, we want to evaluate the binary string that resulted from substituting the value of each variable on the encoding of the input formula (as in Figure 2.2b). Note that it represents a CNF formula whose variables have already been substituted with their actual value. To compute the value of each clause we just have to check if at least one literal is true, and this can be done following a rule that makes any cell with state 1 infect any other cell in that column for $n-1$ time steps. If there is at least one cell with state 1 in a certain clause, after $n-1$ steps all of them would be in state 1. If all of them are in state 0, they will not change their state. We then repeat the same logic to check whether every clause was true or not, but in this case we need to swap the roles of the states 0 and 1, because the formula will only be true if every clause is true. Following a rule that makes any cell with state 0 infect any other cell in that column for $(n+1)\,m-n-1$ time steps.

Before showing the details of the algorithm, we are going to prove the claim that we made previously (in page 11) regarding the length of the proposed encoding. That is, that its length will grow at most quadratically compared to the size of other standard encodings such as the DIMACS CNF encoding. The DIMACS CNF encoding of a CNF formula of n variables and m clauses is the concatenation of the encoding of each clause separated by a symbol 0 (and adding blank symbols when needed). And each clause is encoded as the concatenation of the number of the variables that appear on it, where negated variables appear as negative integers (also adding blank symbols when needed).

Proof. Let φ be a CNF Boolean formula of n variables and m clauses, where each clause has l_c literals, for $c \in \{1,\dots,m\}$. Let f be the function that maps the length of the DIMACS CNF encoding of φ to the length of our encoding of φ, and let k be the length of the DIMACS CNF encoding of φ. We have that k can be, at most, equal to

$$\sum_{c=1}^{m} (2l_c+1)\log_2(2n+2) = \sum_{c=1}^{m} (2l_c+1)\left(1+\log_2(n+1)\right),$$

because we need to encode $2n+2$ symbols as bit strings and for each clause we have $2l_c+1$ symbols (each literal is followed by a space and at the end we have to write the symbol 0), for $c \in \{1,\dots,m\}$. And in that case,

$$k = \sum_{c=1}^{m} (2l_c+1)\left(1+\log_2(n+1)\right) \geq \sum_{c=1}^{m} l_c =: l \geq m,$$

where the first inequality is due to the fact that $\log_2(n+1) \geq 0$ and the last holds because in every clause there must be at least one literal (we could remove that clause if not). Moreover, note that $n \leq l$, because if $n > l$, then some variable would never appear as a literal in the formula, so we could remove it.

Now we can prove our claim, because we have that

$$\begin{aligned} f(k) &= f\left(\sum_{c=1}^{m} (2l_c + 1)\log_2(2n+2)\right) \\ &= C\left((n+1)\,m - 1\right) \\ &\leq C\left((k+1)\,k - 1\right) \in \mathcal{O}(k^2), \end{aligned}$$

where $C = \log_2 |S|$, that is, C is the number of *bits* needed to encode every possible state of the cellular automaton (note that C is a constant number). That is, f grows at most quadratically in k. ■

2.3.2 Detailed explanation of the algorithm

We will assume that we can change the local rule of the cellular automaton after a certain number of steps during its execution to explain the algorithm easier. However, we will show in Section 2.3.3 how that assumption can be eliminated.

To see how our algorithm for CNF-SAT works we will be following the computation steps for a specific instance of the problem. We want to check whether the Boolean formula $\varphi(x_1, x_2, x_3) = (x_1 \vee \neg x_3) \wedge (x_2 \vee x_3)$ in CNF is satisfiable or not. Since this is a very small instance (just three variables and two clauses), we can easily verify that φ is indeed satisfiable, because, for example, the assignment $x_1 = x_2 = x_3 = 1$ makes it true. We recommend the reader to watch this video showing the full execution of the algorithm [12].

To begin, we have to represent the CNF formula φ as a string that will be the input to our cellular automaton. Using the encoding that we proposed on the previous section we have that the input of our cellular automaton is as shown in Figure 2.3.

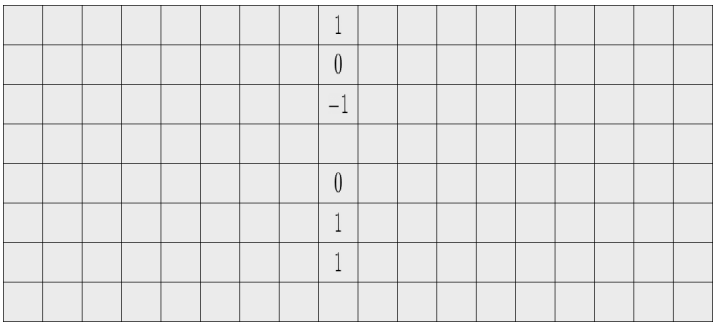
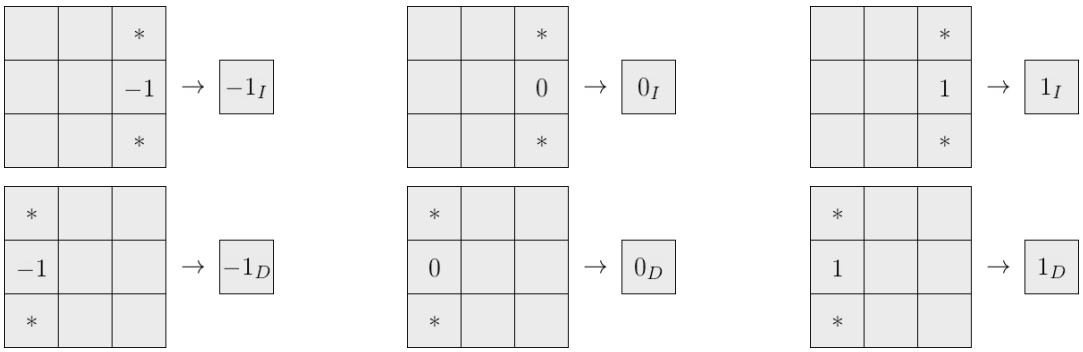


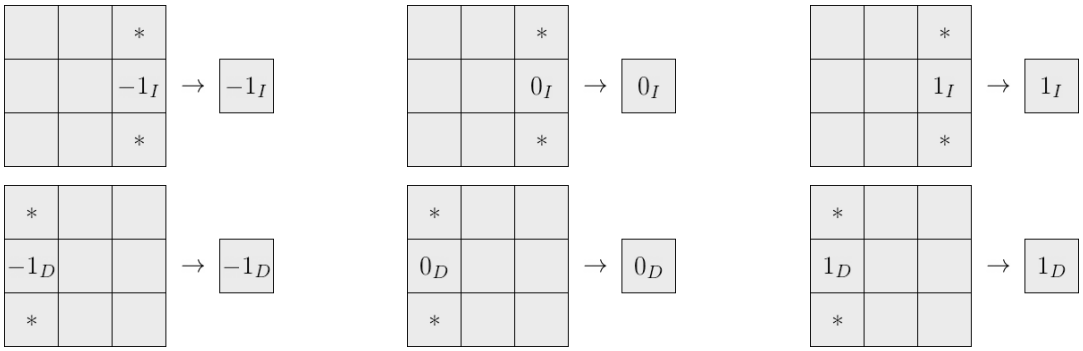
FIGURE 2.3: Encoding the CNF formula $(x_0 \vee \neg x_2) \wedge (x_1 \vee x_2)$ as the input of our cellular automaton.

The idea now is to replicate the information about φ in 2^n different locations (sufficiently far away from each other) together with a unique assignment of the n variables to evaluate all 2^n of them in parallel. Note that the ability of sending information at arbitrarily long distances (in particular, distances that may grow exponentially) is necessary in order to populate an exponential number of cells in polynomial time, because the number of reachable cells grows only quadratically with the distance (unlike the case of the cellular automaton on the hyperbolic plane, as we mentioned before in page 10).

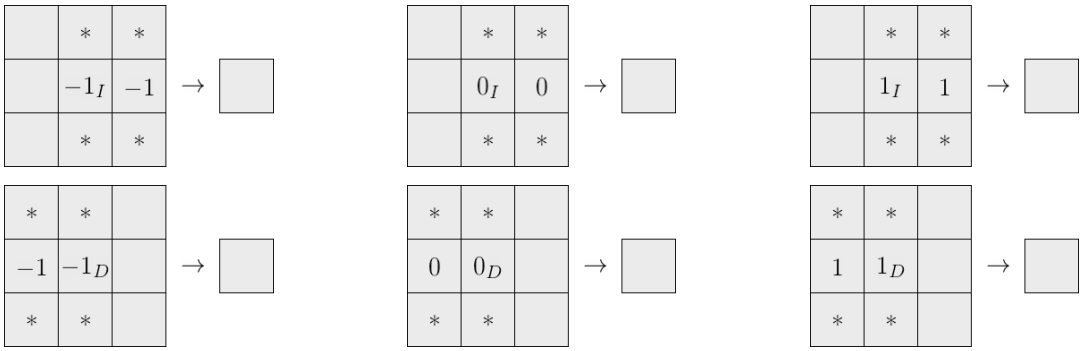
First, we will generate the encoding of the assignments $x_1 = \cdots = x_n = 0$ and $x_1 = 1$, $x_2 = \cdots = x_n = 0$. This can be easily achieved following the rules in Figure 2.4 for two steps and then the rules in Figure 2.5 for one step. Reaching the state shown in Figure 2.6.



(a) Moving the input formula one step to the left and to the right, annotating it properly.

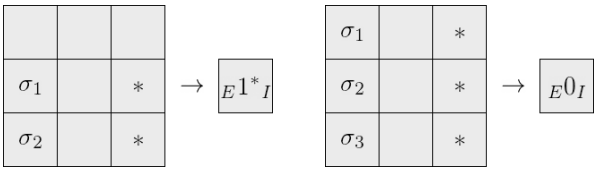


(b) Moving the input formula a second step to the left and to the right.



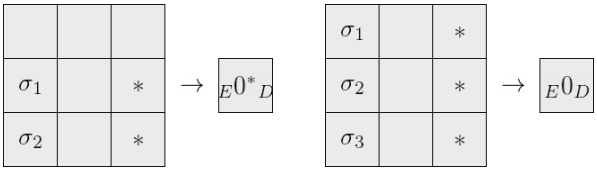
(c) Removing the intermediate result produced when the formula was moved a first step to the left and to the right.

FIGURE 2.4: Moving the input formula two steps to the left and to the right.



for $\sigma_1, \sigma_2 \in \{-1_I, 0_I, 1_I\}$ and $\sigma_3 \in \{-1_I, 0_I, 1_I, \#\}$.

(a) Creating the first assignment to the left of the input formula, where the first variable is assigned the value 1 and the rest are assigned the value 0.



for $\sigma_1, \sigma_2 \in \{-1_D, 0_D, 1_D\}$ and $\sigma_3 \in \{-1_D, 0_D, 1_D, \#\}$.

(b) Creating the first assignment to the right of the input formula, where all variables are assigned the value 0.

FIGURE 2.5: Creating the first pair of possible assignments to the input formula. The asterisk at the first symbol of each assignment will be used to track which of the 2^n possible assignments have already been produced.

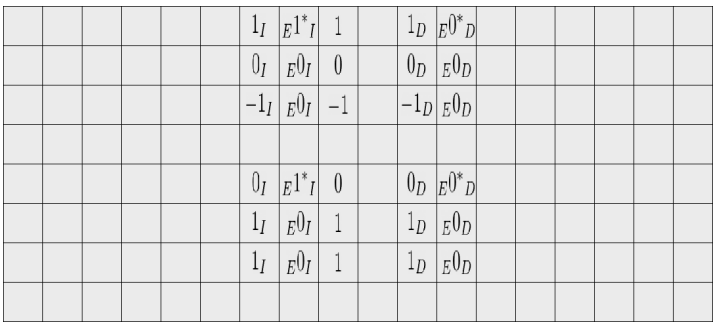
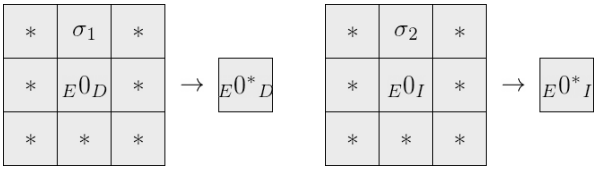


FIGURE 2.6: State of our cellular automaton after generating the first pair of assignments.

Note in Figure 2.6 that the first cell of each block of assignment cells (those with the subscript E) is annotated with an asterisk. That asterisk will be used to track which (and how many) assignments have already been generated. In particular, if the asterisk is in the i-th cell, it means that we have generated all 2^i possible assignments for the first i variables (the rest of the variables would have the value 0), so when it reaches the n-th cell it would mean that we have generated all 2^n possible assignments. We will keep this invariant throughout the following steps.

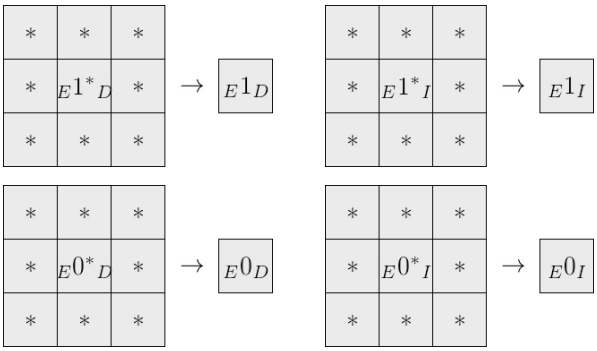
For the next $2\,(n-1)$ steps we will be repeating the following sequence of rules $n-1$ times. First, we will move the asterisk one cell down following the rules from Figure 2.7 for one step. Note that, then, we have to generate new assignments to keep the invariant true, which means doubling the number of already generated assignments and making sure that we do not repeat any of them. It is easy to check that we achieve it by following the rules in Figure 2.8 for another step, where the value k is initially 1 and will be increased in each iteration up to $n-1$. The reason is that the already

generated assignments are not changed, but we also copy them in another place where we change the value assigned to the new variable, so the number of assignments is doubled and they are all new.



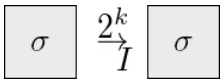
for $\sigma_1 \in \{{}_E1^*{}_D, {}_E0^*{}_D\}$ and $\sigma_2 \in \{{}_E1^*{}_I, {}_E0^*{}_I\}$.

(a) Moving the asterisk one cell down.

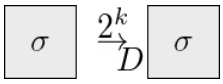


(b) Removing the asterisk from the cells already having it.

FIGURE 2.7: The execution of these rules at the same time has the effect of moving the asterisk one cell down and removing it from the cell where it was previously.

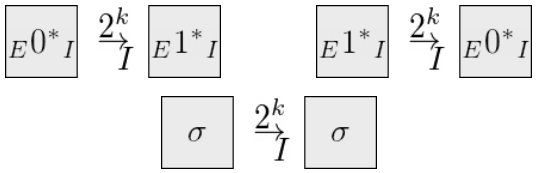


for $\sigma \in \{-1_I, 0_I, 1_I\}$.

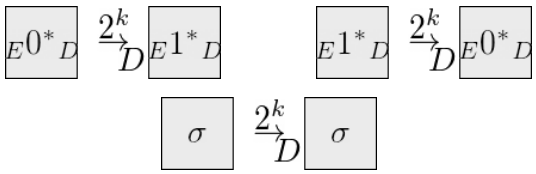


for $\sigma \in \{-1_D, 0_D, 1_D\}$.

(a) Sending the encoding of φ 2^k cells to the left and to the right.



for $\sigma \in \{{}_E0_I, {}_E1_I\}$.

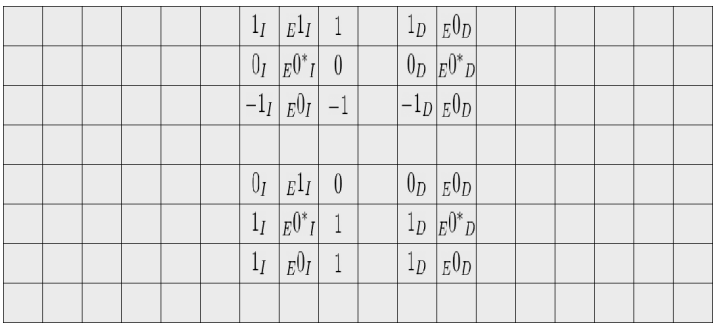


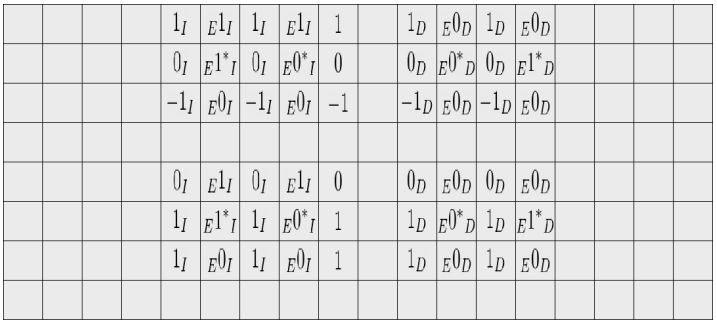
for $\sigma \in \{{}_E0_D, {}_E1_D\}$.

(b) Generating a new assignment from each one already calculated.

FIGURE 2.8: The appropriate messages are sent to the left and to the right to send the encoding of φ and to generate new assignments. The value of k is a variable of the cellular automaton that initially has value 1 and then keeps increasing by one every time this rule is executed (up to $n-1$, because we are repeating this rule $n-1$ times).

Note that there are three variables (*i.e.*, $n = 3$), so we would be repeating the sequence of rules explained before just two times, going through the states shown in Figure 2.9.





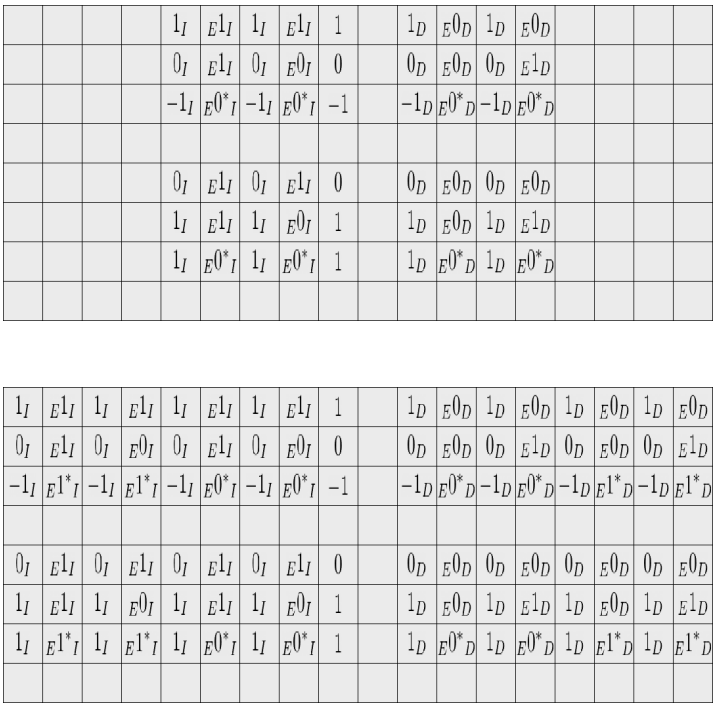


FIGURE 2.9: States of our cellular automaton during the generation all possible assignments.

To remove the asterisks we just have to follow the rules from Figure 2.7 for one more step, reaching the state shown in Figure 2.10, where we have already generated every possible assignment.

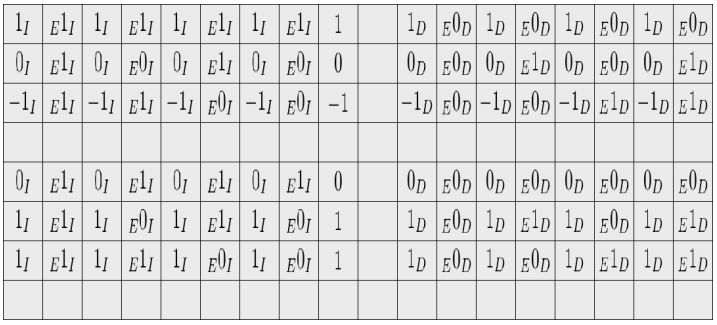
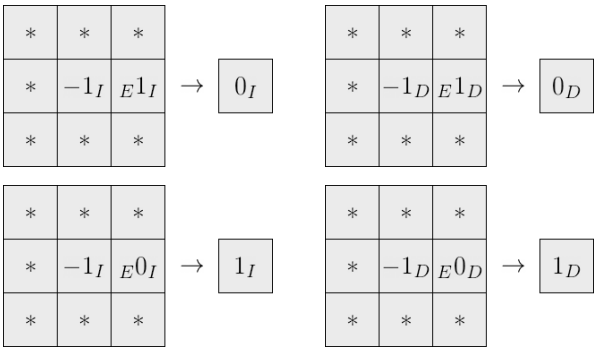
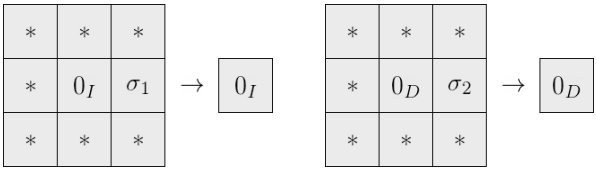


FIGURE 2.10: State of our cellular automaton when every possible assignment has been generated.

At this moment we can evaluate φ on each assignment in parallel. To do that, first we are going to substitute the value of each literal in the formula with the corresponding value depending on the assignment to its right. This can be achieved by following the rules in Figure 2.11 for one step, reaching the state shown in Figure 2.12.

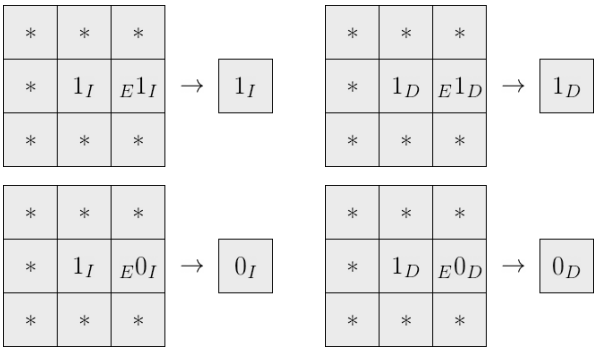


(a) Since the variable appears negated in the clause, its value will be substituted with the opposite of the assigned.



for $\sigma_1 \in \{{}_E1_I, {}_E0_I\}$ and $\sigma_2 \in \{{}_E1_D, {}_E0_D\}$.

(b) The variable does not appear in the clause, so its value will be substituted with a 0, because clauses are disjunctions of literals.



(c) The variable is substituted with the assigned value because it appears as a positive literal.

FIGURE 2.11: Substituting each literal with the appropriate value depending on the assignment of the variables.

1_I	${}_E1_I$	1_I	${}_E1_I$	1_I	${}_E1_I$	1_I	${}_E1_I$	1		0_D	${}_E0_D$	0_D	${}_E0_D$	0_D	${}_E0_D$	0_D	${}_E0_D$
0_I	${}_E1_I$	0_I	${}_E0_I$	0_I	${}_E1_I$	0_I	${}_E0_I$	0		0_D	${}_E0_D$	0_D	${}_E1_D$	0_D	${}_E0_D$	0_D	${}_E1_D$
0_I	${}_E1_I$	0_I	${}_E1_I$	1_I	${}_E0_I$	1_I	${}_E0_I$	-1		1_D	${}_E0_D$	1_D	${}_E0_D$	0_D	${}_E1_D$	0_D	${}_E1_D$
0_I	${}_E1_I$	0_I	${}_E1_I$	0_I	${}_E1_I$	0_I	${}_E1_I$	0		0_D	${}_E0_D$	0_D	${}_E0_D$	0_D	${}_E0_D$	0_D	${}_E0_D$
1_I	${}_E1_I$	0_I	${}_E0_I$	1_I	${}_E1_I$	0_I	${}_E0_I$	1		0_D	${}_E0_D$	1_D	${}_E1_D$	0_D	${}_E0_D$	1_D	${}_E1_D$
1_I	${}_E1_I$	1_I	${}_E1_I$	0_I	${}_E0_I$	0_I	${}_E0_I$	1		0_D	${}_E0_D$	0_D	${}_E0_D$	1_D	${}_E1_D$	1_D	${}_E1_D$

FIGURE 2.12: State of our cellular automaton after substituting the value of each literal.

Then, we have to check the value of each clause. Since a clause is a disjunction of literals, we have to check if any of the literals on it is true, that is, if it has a value of 1. Following the rules from Figure 2.13 for $n-1$ steps we would make all the cells of the clause to be equal to 1 if any literal was true, and nothing would change if all literals were false. Thus, we would reach the state shown in Figure 2.14. And following the rules from Figure 2.15 for one step we would reach the state shown in Figure 2.16.

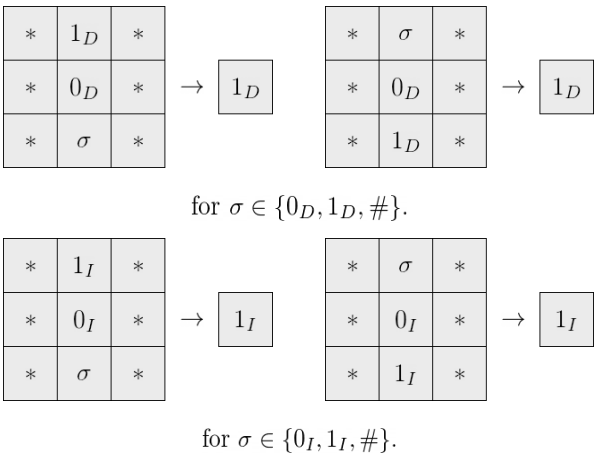


FIGURE 2.13: If any literal in a clause has value 1, then the whole clause evaluates to 1, so each cell will be replaced with a 1. If all literals have value 0, none of these rules apply.

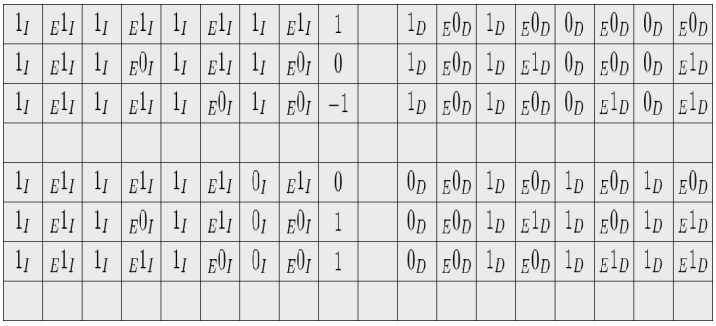


FIGURE 2.14: State of our cellular automaton after calculating the value of each clause depending on the assignment.

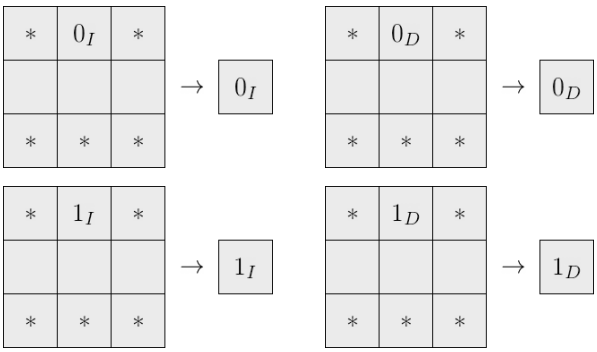


FIGURE 2.15: The cell under each clause (initially empty) is changed with the value of the whole clause.

1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E1_I$	1		1_D	$_E0_D$	1_D	$_E0_D$	0_D	$_E0_D$	0_D	$_E0_D$
1_I	$_E1_I$	1_I	$_E0_I$	1_I	$_E1_I$	1_I	$_E0_I$	0		1_D	$_E0_D$	1_D	$_E1_D$	0_D	$_E0_D$	0_D	$_E1_D$
1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E0_I$	1_I	$_E0_I$	−1		1_D	$_E0_D$	1_D	$_E0_D$	0_D	$_E1_D$	0_D	$_E1_D$
1_I		1_I		1_I		1_I				1_D		1_D		0_D		0_D	
1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E1_I$	0_I	$_E1_I$	0		0_D	$_E0_D$	1_D	$_E0_D$	1_D	$_E0_D$	1_D	$_E0_D$
1_I	$_E1_I$	1_I	$_E0_I$	1_I	$_E1_I$	0_I	$_E0_I$	1		0_D	$_E0_D$	1_D	$_E1_D$	1_D	$_E0_D$	1_D	$_E1_D$
1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E0_I$	0_I	$_E0_I$	1		0_D	$_E0_D$	1_D	$_E0_D$	1_D	$_E1_D$	1_D	$_E1_D$
1_I		1_I		1_I		0_I				0_D		1_D		1_D		1_D	

FIGURE 2.16: State of our cellular automaton after changing the value of the cell under each clause.

Finally, we want to check the value of the formula for each assignment. Following the same idea as before, but swapping the roles of 0 and 1 (because now the formula is a conjunction of clauses), we would check whether any clause is false repeating the rules from Figure 2.17 for $(n+1)\,m - n - 1$ steps. In our case, we would reach the state shown in Figure 2.18.

$*$	0_D	$*$
$*$	1_D	$*$
$*$	σ	$*$

$\rightarrow$ $\boxed{0_D}$

$*$	σ	$*$
$*$	1_D	$*$
$*$	0_D	$*$

$\rightarrow$ $\boxed{0_D}$

for $\sigma \in \{0_D, 1_D, \#\}$.

$*$	0_I	$*$
$*$	1_I	$*$
$*$	σ	$*$

$\rightarrow$ $\boxed{0_I}$

$*$	σ	$*$
$*$	1_I	$*$
$*$	0_I	$*$

$\rightarrow$ $\boxed{0_I}$

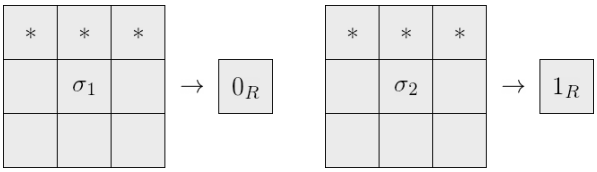
for $\sigma \in \{0_I, 1_I, \#\}$.

FIGURE 2.17: If any clause has value 0, then the whole formula evaluates to 0, because it is the conjunction of clauses. So each cell is replaced with a 0 if there is any other 0. If all clauses have value 1, then none of these rules apply.

1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E1_I$	0_I	$_E1_I$	1		0_D	$_E0_D$	1_D	$_E0_D$	0_D	$_E0_D$	0_D	$_E0_D$
1_I	$_E1_I$	1_I	$_E0_I$	1_I	$_E1_I$	0_I	$_E0_I$	0		0_D	$_E0_D$	1_D	$_E1_D$	0_D	$_E0_D$	0_D	$_E1_D$
1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E0_I$	0_I	$_E0_I$	−1		0_D	$_E0_D$	1_D	$_E0_D$	0_D	$_E1_D$	0_D	$_E1_D$
1_I		1_I		1_I		0_I				0_D		1_D		0_D		0_D	
1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E1_I$	0_I	$_E1_I$	0		0_D	$_E0_D$	1_D	$_E0_D$	0_D	$_E0_D$	0_D	$_E0_D$
1_I	$_E1_I$	1_I	$_E0_I$	1_I	$_E1_I$	0_I	$_E0_I$	1		0_D	$_E0_D$	1_D	$_E1_D$	0_D	$_E0_D$	0_D	$_E1_D$
1_I	$_E1_I$	1_I	$_E1_I$	1_I	$_E0_I$	0_I	$_E0_I$	1		0_D	$_E0_D$	1_D	$_E0_D$	0_D	$_E1_D$	0_D	$_E1_D$
1_I		1_I		1_I		0_I				0_D		1_D		0_D		0_D	

FIGURE 2.18: State of our cellular automaton after calculating the value the formula depending on the assignment. Observe that there are four assignments that satisfy φ.

Next, before collecting the results, we would follow the rules from Figure 2.19 for one step, and thus reaching the state in Figure 2.20.



for $\sigma_1 \in \{0_D, 0_I\}$ and $\sigma_2 \in \{1_D, 1_I\}$.

FIGURE 2.19: The value of the formula for each assignment is confirmed.

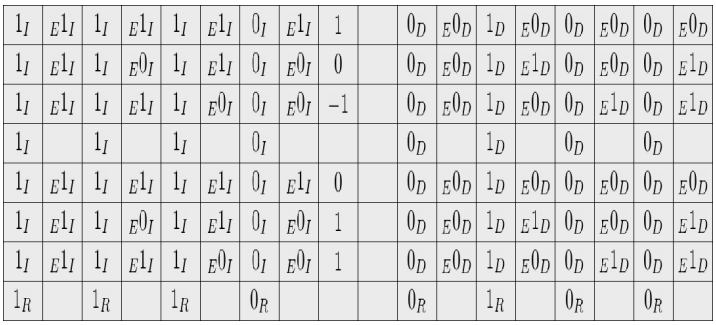
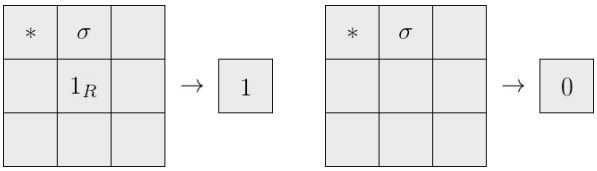


FIGURE 2.20: State of our cellular automaton after confirming the value the formula depending on the assignment.

Finally, from every cell with symbol 1_R we would be sending that message 2^k cells to the right and to the left, for k ranging from 1 to $n+1$ to make sure that if any assignment satisfied the formula, we would get that information in the cell under the input formula. That is, we would be following the rules from Figure 2.21 for $n+1$ steps. Note that we may be sending messages to empty cells out of the grid shown in the figures.

FIGURE 2.21: To check if any assignment satisfies the Boolean formula φ, for any satisfying assignment we send the symbol 1_R 2^k cells to the left and to the right to confirm it, and that message will eventually reach the cell under the input formula.

We get the result of the algorithm checking if the cell under the encoding of the input formula has value 1_R. On the other hand, if there was no satisfying assignment, that cell would be blank. So following the rules from Figure 2.22 for one step would yield the result of the algorithm.



for $\sigma \in \{-1, 0, 1\}$.

FIGURE 2.22: The result of the algorithm, that is, if φ is satisfiable or not, is stored one cell down the input formula.

The state of our cellular automaton in this case would be the one shown in Figure 2.23. Note that the algorithm correctly determined that φ was satisfiable.

1_I	$E1_I$	1_I	$E1_I$	1_I	$E1_I$	0_I	$E1_I$	1		0_D	$E0_D$	1_D	$E0_D$	0_D	$E0_D$	0_D	$E0_D$
1_I	$E1_I$	1_I	$E0_I$	1_I	$E1_I$	0_I	$E0_I$	0		0_D	$E0_D$	1_D	$E1_D$	0_D	$E0_D$	0_D	$E1_D$
1_I	$E1_I$	1_I	$E1_I$	1_I	$E0_I$	0_I	$E0_I$	-1		0_D	$E0_D$	1_D	$E0_D$	0_D	$E1_D$	0_D	$E1_D$
1_I		1_I		1_I		0_I				0_D		1_D		0_D		0_D	
1_I	$E1_I$	1_I	$E1_I$	1_I	$E1_I$	0_I	$E1_I$	0		0_D	$E0_D$	1_D	$E0_D$	0_D	$E0_D$	0_D	$E0_D$
1_I	$E1_I$	1_I	$E0_I$	1_I	$E1_I$	0_I	$E0_I$	1		0_D	$E0_D$	1_D	$E1_D$	0_D	$E0_D$	0_D	$E1_D$
1_I	$E1_I$	1_I	$E1_I$	1_I	$E0_I$	0_I	$E0_I$	1		0_D	$E0_D$	1_D	$E0_D$	0_D	$E1_D$	0_D	$E1_D$
1_R		1_R		1_R		1_R		1		1_R		1_R		1_R		1_R	

FIGURE 2.23: State of our cellular automaton at the end of the algorithm.

2.3.3 Simplification of the algorithm

In the previous section we assumed that we could change the local rule of the cellular automaton that specifies how the states of cells evolve after a certain number of steps. We will argue how we can eliminate that assumption.

If the number of times that a local rule is followed before changing to another one is constant, then we can easily simulate it in the same number of steps.

Proposition 2.1. A cellular automaton A can simulate another cellular automaton that follows a local rule f_1 for a constant number c of time steps before it changes its local rule to another function f_2. Moreover, the simulation can be performed in the same number of steps.

Proof. We assume that $d = 2$, that N is the Moore neighbourhood (so $v = 9$) and that S is the finite set of states used. The cellular automaton A is given by $(d, S \times \{0, \ldots, c\}, N, f)$, where f is built from f_1 and f_2 as follows:

$$f\left((s_1, p_1), \ldots, (s_{10}, p_{10})\right) = \begin{cases} \left(f_1\left(s_1, \ldots, s_{10}\right), p + 1\right) & \text{if } 0 \leq p \coloneqq p_1 = \cdots = p_{10} < c \\ \left(f_2\left(s_1, \ldots, s_{10}\right), c\right) & \text{if } c = p_1 = \cdots = p_{10} \end{cases}$$

It is clear that then the cellular automaton will follow the local rule f_1 for c steps and then follow the local rule f_2. ■

This trick does not work if c is not a constant, because then we cannot assume that the set of states is finite. Also, we can only apply this trick for a constant number of changes of the local rule.

Then, we just need to study four more cases: how to change the local rule after generating all the possible assignments (expansion phase), after repeating the rules from Figure 2.13 for $n-1$ steps (flooding the clause phase), the rules from Figure 2.17 for $(n+1)\,m-n-1$ steps (flooding the formula phase) and the rules from Figure 2.21 for $n+1$ steps (collecting the results phase). The general idea is to repeat those rules until a specific condition (that can be easily checked by the cellular automaton) takes place. Then we could use a similar trick as in Proposition 2.1 to annotate the cells that detect that condition and start communicating it to their neighbours.

Flooding the clause phase

We need to make sure that we have repeated that rule for, at least, $n-1$ steps to make sure that if any literal was true, then every cell in that clause has already changed its state to 1. So we could annotate the first cell of each clause and applying the same rule, but also annotating the cell below an already annotated one until we notice that we cannot annotate any other cell because we have reached an empty cell. Since the length of each clause is n, we would repeat the new rule for the required time steps.

Flooding the formula phase

We could apply a similar idea to the previous one, but focusing in this case in the whole formula. We need to repeat the flooding rule for at least $(n+1)\,m-n-1$ steps, but using the previous trick we would need to annotate every cell in the formula, of length $(n+1)\,m-1$.

Collecting the results phase

In this case we could end before the $n+1$ steps just checking if the cell under the input formula changes its state. Note that the input formula is the only column of cells with states from $\{-1,0,1\}$.

Expansion phase

In this case we are moving the asterisk one cell down each step, so we could finish this phase when we cannot move the asterisk more cells down because we already reached an empty cell.

Chapter 3

Quantum mechanics

What if we tried to take advantage of the quantum rules of Nature to solve NP-complete problems in polynomial time? When quantum computing came along, it was believed that the inherent parallelism of quantum mechanics would allow us to finally achieve it. Indeed, you can naively argue that you just need to prepare a superposition of all possible assignments to a Boolean formula and evaluate it in parallel to check whether it is satisfiable or not, and thus solving SAT or TAUT in polynomial time. However, as we shall see, it is not that easy: performing such computation would leave us with a state that gives us little information about the satisfiability of the formula. Moreover, it is known that NP $\not\subset$ BQP[1] relative to an oracle [8]. And while this result does not rule out the possibility of NP $\subset$ BQP, it tells us that we would need to exploit the structure of NP-complete problems in a nontrivial way in order to solve them in quantum polynomial time.

This leaves us with the question of how should the theory of quantum mechanics change to be able to provably solve NP-complete problems in polynomial time with a quantum computer. We will address that problem later in this chapter, but before that, we shall review the basics of quantum mechanics and how it applies to quantum computation.

3.1 Introduction to quantum mechanics

In order to study the behaviour of quantum systems and how to apply them to computation, we will introduce the postulates of quantum mechanics, the mathematical formalization of the quantum theory. We refer the reader to Appendix A for a review of some of the mathematical concepts and notation that we will be using.

First, it is important to see how to mathematically represent a physical system and its state.

Postulate 1. Each isolated physical system is associated with a complex vector space with inner product (*i.e.*, a Hilbert space), known as the *state space* of the system. The state of the system is described by a unit vector in its state space, known as the *state vector*.

Note that quantum mechanics does not tell us what is the state space of a given physical system and that modelling a physical system is a nontrivial task for physicists. However, we will not have to deal with such difficulties here. Indeed, we are concerned with one of the simplest quantum systems, the *qubit*.

[1] BQP is the class of decision problems solvable in quantum polynomial time with an error probability of at most 1/3 for all instances.

Example. The state space of a *qubit* is $\mathbb{C}^2$. Let $\{|0\rangle, |1\rangle\}$ be an orthonormal basis for that state space, we shall refer to it as the *computational basis*. Then, the state vector $|\psi\rangle$ of a *qubit* can be written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$. The condition that $|\psi\rangle$ is a unit vector is equivalent to $1 = \langle\psi|\psi\rangle = \overline{\alpha}\alpha + \overline{\beta}\beta = |\alpha|^2 + |\beta|^2$.

The state vectors $|0\rangle$ and $|1\rangle$ are named in that way in analogy with 0 and 1, the possible states of a *bit*, the basic unit of classical information. However, *qubits* can also be in a *superposition* of those states, *i.e.*, $\alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. α and β are called the *amplitudes* of the states $|0\rangle$ and $|1\rangle$, respectively. ■

Since we will be working with systems made up of multiple *qubits*, we need a way to represent the state space of such systems. The next postulate describes how the state space of a composite system is built up from the state spaces of the component systems.

Postulate 2. The state space of a composite physical system is the tensor product of the state spaces of the component systems. In addition, the state vector of the composite system is the tensor product of the state vectors of the component systems. That is, if there are n component systems and the i-th system is in the state $|\psi_i\rangle$, for $i \in \{1, \ldots, n\}$, then the composite system is in the state $|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$.

Note that it does not mean that any state vector of a composite physical system can be written as the tensor product of state vectors of the component systems. In fact, there are state vectors like

$$\frac{|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle}{\sqrt{2}},$$

known as *entangled states*, that cannot.

Example (Multiple *qubits*)**.** From postulate 2 we know that the state space of a system of n *qubits* is $\bigotimes_{i=1}^{n} \mathbb{C}^2 = \mathbb{C}^{2^n}$. Note that the dimension of the state space grows exponentially on the number of *qubits*, so we can have a superposition of an exponential number of states.

The computational basis for that space is given by

$$\left\{|0\rangle, |1\rangle\right\}^{\otimes n} = \left\{|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle : |\psi_i\rangle \in \left\{|0\rangle, |1\rangle\right\}\right\}.$$

For example, for $n = 2$, the computational basis of $\mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^4$ is

$$\left\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\right\},$$

that we will write as $\left\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\right\}$ for the sake of readability.

In general, for n *qubits* we might write the vectors of the computational basis as

$$\left\{|x\rangle : x \in \{0,1\}^n\right\}$$

in lexicographic order (as in the $n = 2$ example). ■

In order to perform computations with quantum systems, we are interested in how they may change with time. The following postulate describes such changes.

Postulate 3. The time evolution of a closed quantum system is described by *unitary* transformations. That is, the state of the system at time t is related to the state of the system at time t_0 by a unitary operator U as follows

$$\left|\psi\left(t\right)\right\rangle = U\left|\psi\left(t_0\right)\right\rangle.$$

That means that the transformations that we are allowed to perform on any quantum system, and in particular on *qubits*, must be given by a unitary operator. This is important because the unitarity of quantum states is preserved by unitary transformations (see definition A.12).

Example. To see some examples, consider the unitary transformation X given by

$$\mathrm{X}: \begin{array}{ccc} |0\rangle & \mapsto & |1\rangle \\ |1\rangle & \mapsto & |0\rangle, \end{array}$$

which is the quantum analogous of the classical NOT gate.

A more interesting example is the Hadamard transformation H, given by

$$\mathrm{H}: \begin{array}{ccc} |0\rangle & \mapsto & \frac{|0\rangle+|1\rangle}{\sqrt{2}} = |+\rangle \\ |1\rangle & \mapsto & \frac{|0\rangle-|1\rangle}{\sqrt{2}} = |-\rangle. \end{array}$$

This unitary transformation has interesting properties, like being self-adjoint (so it is its own inverse), *i.e.*, $\mathrm{H} = \mathrm{H}^\dagger$, and that if we apply it to n *qubits* all with state $|0\rangle$, we would get a superposition of the 2^n states in the computational basis of n *qubits*. That is,

$$\mathrm{H}^{\otimes n}\left|0^n\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle.$$

More generally, it verifies that for any $y \in \{0,1\}^n$ we have that

$$\mathrm{H}^{\otimes n}\left|y\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot y}\left|x\right\rangle,$$

where $x \cdot y$ is the inner product of x and y seen as vectors in $\{0,1\}^n$. We will use this identity later. ■

Finally, we are interested in how we may access the information stored on a *qubit*. While getting the state of a *bit* has no consequences, we cannot measure the state of a *qubit* without changing its state, and moreover, the result of a measurement may be random! The following postulate shows how the measurement of the state of a quantum system works and what effects it has.

Postulate 4. Quantum measurements are described by a collection $\{M_m\}_m$ of *measurement operators* on the state space of the system being measured. The index m refers to the possible measurement outcomes. The measurement operators must verify the *completeness equation*

$$\sum_m M_m^\dagger M_m = I.$$

If the state of the system is $|\psi\rangle$, the probability of getting the outcome m when measuring is

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle,$$

and in that case, the state would *collapse* to

$$\frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}}.$$

The completeness equation shows that the probabilities of getting any outcome m sum up to one, because

$$1 = \langle\psi|\psi\rangle = \langle\psi| I |\psi\rangle = \langle\psi| \left(\sum_m M_m^\dagger M_m\right) |\psi\rangle = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle = \sum_m p(m).$$

Example (Measurement of a *qubit* in the computational basis)**.** For example, we may want to perform the measurement of a *qubit* with state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in the computational basis, which is described by the measurement operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. We can easily check that they verify the completeness equation, for example, seeing the matrix representation of the measurement operators in the computational basis. Since

$$M_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } M_1 = |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},$$

we have that $M_0^\dagger M_0 + M_1^\dagger M_1 = M_0^2 + M_1^2 = M_0 + M_1 = I$. We could have also noticed that M_0 and M_1 are projections that verify that $M_m^\dagger = M_m$ and $M_m^2 = M_m$ for $m \in \{0, 1\}$ and because of proposition A.4.

Measuring the state $|\psi\rangle$, we would obtain the outcome 0 with probability

$$p(0) = \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle = \langle\psi|0\rangle\langle 0|\psi\rangle = \overline{\alpha}\alpha = |\alpha|^2,$$

and in that case the state would collapse to

$$\frac{M_0 |\psi\rangle}{|\alpha|} = \frac{|0\rangle\langle 0|\psi\rangle}{|\alpha|} = \frac{\alpha}{|\alpha|}|0\rangle.$$

On the other hand, we would obtain the outcome 1 with probability

$$p(1) = \langle\psi| M_1^\dagger M_1 |\psi\rangle = \langle\psi| M_1 |\psi\rangle = \langle\psi|1\rangle\langle 1|\psi\rangle = \overline{\beta}\beta = |\beta|^2,$$

and the state would collapse to

$$\frac{M_1 |\psi\rangle}{|\beta|} = \frac{|1\rangle\langle 1|\psi\rangle}{|\beta|} = \frac{\beta}{|\beta|}|1\rangle.$$

Note that $|\alpha|^2 + |\beta|^2 = 1$. ■

Another important observation from the previous example is that after the measurement the state collapses to one of the states of the computational basis, but multiplied by a certain unitary complex number. In general we can ignore that factor. To see

it, let $|\psi\rangle$ and $|\phi\rangle$ be two states such that $|\psi\rangle = \mathsf{e}^{\mathsf{i}\theta}|\phi\rangle$ for some $\theta \in [0, 2\pi)$ and let $\{M_m\}_m$ be a collection of measurement operators. Then, when measuring the state $|\psi\rangle$, the probability of obtaining the outcome m is

$$\begin{aligned}\langle\psi| M_m^\dagger M_m |\psi\rangle &= \left(\overline{\mathsf{e}^{\mathsf{i}\theta}}\langle\phi|\right) M_m^\dagger M_m \left(\mathsf{e}^{\mathsf{i}\theta}|\phi\rangle\right) \\ &= \mathsf{e}^{-\mathsf{i}\theta}\mathsf{e}^{\mathsf{i}\theta}\langle\phi| M_m^\dagger M_m |\phi\rangle \\ &= \langle\phi| M_m^\dagger M_m |\phi\rangle ,\end{aligned}$$

which equals the probability of obtaining the outcome m when measuring the state $|\phi\rangle$.

In most cases we will be concerned with a special class of measurements, known as *projective* measurements. Indeed, it can be shown that they are equivalent to general measurement operators.

A projective measurement is described by a collection $\{P_m\}_m$ of measurement operators that, in addition to satisfying the completeness equation, also satisfy the conditions that P_m are orthogonal projectors. That is, the P_m are self-adjoint and $P_m P_{m'} = \delta_{m,m'} P_m$. Let us note that in that case the completeness equation can be rewritten as

$$I = \sum_{i=1}^{n} P_m^\dagger P_m = \sum_{i=1}^{n} P_m^2 = \sum_{i=1}^{n} P_m.$$

So, if $|\psi\rangle$ is the state of the system, the probability of getting outcome m when measuring is

$$p(m) = \langle\psi| P_m |\psi\rangle ,$$

and in that case, the state would collapse to

$$\frac{P_m|\psi\rangle}{\sqrt{\langle\psi| P_m |\psi\rangle}}.$$

This allows us to define what measuring in a basis is. Let $\{|\psi_i\rangle : 1 \le i \le n\}$ be an orthonormal basis of our state space. Then, a measurement with respect to that basis is the projective measurement with projectors $\{P_i = |\psi_i\rangle\langle\psi_i| : 1 \le i \le n\}$.

Example. For example, let us suppose that we have the following two-*qubit* state

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

that we want to measure in the computational basis. Then, we would get the outcome, *e.g.*, 01, with probability

$$p(01) = \langle\psi| P_{01} |\psi\rangle = \langle\psi|01\rangle\langle 01|\psi\rangle = \overline{\beta}\beta = |\beta|^2.$$

And in that case, the state would collapse to $|01\rangle$, modulo a unitary complex number. ■

Example. Another example of a measurement that we could perform is measuring just one *qubit* of a multiple *qubit* system. Following the previous example, let

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

be the state of our two *qubit* system. And suppose that we only want to measure the second *qubit* in the computational basis. That measurement is described by projectors $\left\{I \otimes |0\rangle\langle 0|, I \otimes |1\rangle\langle 1|\right\}$. Thus, we would obtain the outcome 0 with probability

$$\begin{aligned}
p(0) &= \langle\psi| \left(I \otimes |0\rangle\langle 0|\right) |\psi\rangle \\
&= \langle\psi| \left(I \otimes |0\rangle\langle 0|\right) \Big(\big(\alpha|0\rangle + \gamma|1\rangle\big) \otimes |0\rangle + \big(\beta|0\rangle + \delta|1\rangle\big) \otimes |1\rangle\Big) \\
&= \langle\psi| \Big(\big(\alpha|0\rangle + \gamma|1\rangle\big) \otimes \big(\langle 0|0\rangle\,|0\rangle\big) + \big(\beta|0\rangle + \delta|1\rangle\big) \otimes \big(\langle 0|1\rangle\,|0\rangle\big)\Big) \\
&= \langle\psi| \big(\alpha|00\rangle + \gamma|10\rangle\big) \\
&= \overline{\alpha}\alpha + \overline{\gamma}\gamma \\
&= |\alpha|^2 + |\gamma|^2,
\end{aligned}$$

leaving the state as

$$\frac{\left(I \otimes |0\rangle\langle 0|\right)|\psi\rangle}{\sqrt{|\alpha|^2 + |\gamma|^2}} = \frac{\alpha|00\rangle + \gamma|10\rangle}{\sqrt{|\alpha|^2 + |\gamma|^2}}.$$

Or we could obtain the outcome 1 with probability

$$\begin{aligned}
p(1) &= \langle\psi| \left(I \otimes |1\rangle\langle 1|\right) |\psi\rangle \\
&= \langle\psi| \left(I \otimes |1\rangle\langle 1|\right) \Big(\big(\alpha|0\rangle + \gamma|1\rangle\big) \otimes |0\rangle + \big(\beta|0\rangle + \delta|1\rangle\big) \otimes |1\rangle\Big) \\
&= \langle\psi| \Big(\big(\alpha|0\rangle + \gamma|1\rangle\big) \otimes \big(\langle 1|0\rangle\,|1\rangle\big) + \big(\beta|0\rangle + \delta|1\rangle\big) \otimes \big(\langle 1|1\rangle\,|1\rangle\big)\Big) \\
&= \langle\psi| \big(\beta|01\rangle + \delta|11\rangle\big) \\
&= \overline{\beta}\beta + \overline{\delta}\delta \\
&= |\beta|^2 + |\delta|^2,
\end{aligned}$$

and collapsing the state to

$$\frac{\left(I \otimes |1\rangle\langle 1|\right)|\psi\rangle}{\sqrt{|\beta|^2 + |\delta|^2}} = \frac{\beta|01\rangle + \delta|11\rangle}{\sqrt{|\beta|^2 + |\delta|^2}}.$$

This example can be intuitively seen if we rewrite the state $|\psi\rangle$ as

$$|\psi\rangle = \big(\alpha|0\rangle + \gamma|1\rangle\big) \otimes |0\rangle + \big(\beta|0\rangle + \delta|1\rangle\big) \otimes |1\rangle.$$

■

Up to this moment we have reviewed the mathematical formulation of quantum mechanics, but we are now interested in its applications. In particular, we will discuss two important consequences of the quantum theory: the no-cloning theorem and the impossibility of distinguishing arbitrary quantum states (and how they relate to each other).

The no-cloning theorem states that, unlike with classical information, we cannot create a perfect copy of a quantum system. This is shown by the following result.

Theorem (No-cloning theorem)**.** *Let $\mathcal{H}$ be the state space of a given quantum system and let $|\gamma\rangle \in \mathcal{H}$ be a unit vector. Then, there does not exist any linear operator U on $\mathcal{H} \otimes \mathcal{H}$ such that $U\left(|\psi\rangle \otimes |\gamma\rangle\right) = |\psi\rangle \otimes |\psi\rangle$ for any $|\psi\rangle \in \mathcal{H}$. That is, we cannot create a copy of an arbitrary quantum state.*

Proof. By contradiction, let us assume that such unitary operator U exists, and let $|\psi\rangle, |\phi\rangle \in \mathcal{H}$ be two states we want to clone. So we have that

$$U\left(|\psi\rangle \otimes |\gamma\rangle\right) = |\psi\rangle \otimes |\psi\rangle \text{ and } U\left(|\phi\rangle \otimes |\gamma\rangle\right) = |\phi\rangle \otimes |\phi\rangle .$$

Taking the inner product we have that

$$\left(\langle\psi| \otimes \langle\gamma|\right) U^\dagger U \left(|\phi\rangle \otimes |\gamma\rangle\right) = \left(\langle\psi| \otimes \langle\gamma|\right)\left(|\phi\rangle \otimes |\gamma\rangle\right) = \langle\psi|\phi\rangle \langle\gamma|\gamma\rangle = \langle\psi|\phi\rangle$$

and, on the other hand,

$$\left(\langle\psi| \otimes \langle\psi|\right)\left(|\phi\rangle \otimes |\phi\rangle\right) = \langle\psi|\phi\rangle \langle\psi|\phi\rangle = \langle\psi|\phi\rangle^2 .$$

Since both values must be equal, we have that $\langle\psi|\phi\rangle = \langle\psi|\phi\rangle^2$, which leaves us with two possibilities: either $\langle\psi|\phi\rangle = 0$ or $\langle\psi|\phi\rangle = 1$. That is, either $|\psi\rangle$ and $|\phi\rangle$ are the same state or they are orthogonal. Anyway, we cannot clone arbitrary quantum states, which contradicts the hypothesis. ■

We can start feeling that the restriction that quantum transformations must be given by unitary operators is limiting us. We will see this again later.

Although the previous result prevents us from cloning arbitrary quantum states, it is still possible to do it in some cases. The following example shows one of them.

Example. Let $|\psi\rangle, |\phi\rangle \in \mathbb{C}^2$ be two orthogonal *qubits*. Without loss of generality we may assume them to be the states $|0\rangle$ and $|1\rangle$ (if not, we just change the states to the computational basis). And let $|\gamma\rangle$ be an unknown state that is equal to one of them.

Let CNOT be the two *qubit* unitary transformation given by

$$\text{CNOT}: \begin{array}{ccc} |00\rangle & \mapsto & |00\rangle \\ |01\rangle & \mapsto & |01\rangle \\ |10\rangle & \mapsto & |11\rangle \\ |11\rangle & \mapsto & |10\rangle . \end{array}$$

Where the first *qubit* is the control *qubit* whose value determines if the second *qubit* is negated or not.

We can see that CNOT is unitary because it transforms an orthonormal basis into another orthonormal basis (see proposition A.13). Moreover, notice that it allows us to clone the state $|\gamma\rangle$ given that it is either $|0\rangle$ or $|1\rangle$. ■

Another problem that classically is trivial but cannot be solved with certainty for quantum systems is the problem of distinguishing quantum states. Let $\{|\psi_i\rangle : 1 \leq i \leq n\}$ be a set of quantum states and let $|\gamma\rangle$ be an unknown state that belongs to that set. We have to identify which state it is.

This problem is related to the problem of cloning quantum states. Actually, if we could distinguish a pair of arbitrary quantum states, then we would be able to clone them.

Example. Let $|\psi\rangle$ and $|\phi\rangle$ two *qubits*, not necessarily orthogonal, let $B_1 = \{|\psi\rangle, |\psi'\rangle\}$ and $B_2 = \{|\phi\rangle, |\phi'\rangle\}$ two orthonormal basis in $\mathbb{C}^2$ and let $|\gamma\rangle$ be an unknown state equal to one of $|\psi\rangle$ and $|\phi\rangle$. We are going to show that if we can identify which state it is, then we could clone it.

Let $U_1 = |\psi\rangle\langle 0| + |\psi'\rangle\langle 1|$ and $U_2 = |\phi\rangle\langle 0| + |\phi'\rangle\langle 1|$ be the unitary operators that change from the computational basis to B_1 and B_2, respectively. If we can distinguish if $|\gamma\rangle$ is equal to $|\psi\rangle$ or $|\phi\rangle$, we just have to apply to a *qubit* with state $|0\rangle$ the transformation U_1 or U_2, respectively. ■

But it turns out that we can distinguish a quantum state from an orthonormal set.

Example. Let $\{|\psi_i\rangle : 1 \leq i \leq n\}$ be an orthonormal set of states and let $|\gamma\rangle$ be an unknown state from that set. Consider the set of projections $\{P_i = |\psi_i\rangle\langle\psi_i| : 1 \leq i \leq n\}$, that together with $P_0 = I - \sum_{i=1}^{n} P_i$ define a projective measurement.

Then, if $|\gamma\rangle$ is $|\psi_{i_0}\rangle$ for a certain $i_0 \in \{1, \ldots, n\}$ and we measure that state with the measurement defined by projections $\{P_i : 0 \leq i \leq n\}$, we would get the outcome i_0 with probability

$$p(i_0) = \langle\gamma| P_{i_0} |\gamma\rangle = \langle\psi_{i_0}|\psi_{i_0}\rangle\langle\psi_{i_0}|\psi_{i_0}\rangle = 1.$$

And thus, reliably distinguishing the state. ■

Finally, we will prove that it is not possible to distinguish a state from a set of non-orthogonal states.

Proof. Let $|\psi_1\rangle$ and $|\psi_2\rangle$ be two non-orthogonal quantum states that we want to distinguish. The idea is to perform a measurement of our unknown state $|\gamma\rangle$ described by measurement operators $\{M_m\}_m$. Depending on the outcome m, we will try to guess what the value of the index $i \in \{1, 2\}$ was according to a predefined rule f such that $f(m) = i$.

Let

$$E_1 = \sum_{m:\, f(m)=1} M_m^\dagger M_m \text{ and } E_2 = \sum_{m:\, f(m)=2} M_m^\dagger M_m,$$

that verify that $E_1 + E_2 = I$ because of the completeness equation. Also note that E_1 and E_2 are positive-semidefinite operators (see proposition A.17).

If we could distinguish the states $|\psi_1\rangle$ and $|\psi_2\rangle$ with the measurement described above, we would have that the probability of measuring and obtaining an outcome m such that $f(m)$ matches the desired index is 1. And that condition can be expressed as follows

$$\langle\psi_1| E_1 |\psi_1\rangle = 1 \text{ and } \langle\psi_2| E_2 |\psi_2\rangle = 1.$$

Since

$$\langle\psi_1| E_1 |\psi_1\rangle = 1 = \langle\psi_1|\psi_1\rangle = \langle\psi_1| (E_1 + E_2) |\psi\rangle,$$

we must have that $\langle\psi_1| E_2 |\psi_1\rangle = 0$. Moreover, since E_2 is a positive-semidefinite operator, there exists an operator S such that $S^\dagger S = E_2$, so $0 = \langle\psi_1| E_2 |\psi_1\rangle = \langle\psi_1| S^\dagger S |\psi_1\rangle$, which is equivalent to $S|\psi_1\rangle = 0$.

This will lead to a contradiction, because if we pick an orthonormal basis $\{|\psi_1\rangle, |\phi\rangle\}$ of the vector space spanned by vectors $|\psi_1\rangle$ and $|\psi_2\rangle$, we can write $|\psi_2\rangle = \alpha|\psi_1\rangle + \beta|\phi\rangle$ for some $\alpha, \beta \in \mathbb{C}$ such that $0 < |\alpha|, |\beta| < 1$. Applying S to $|\psi_2\rangle$ we get that $S|\psi_2\rangle = \alpha S|\psi_1\rangle + \beta S|\phi\rangle = \beta S|\phi\rangle$, so

$$1 = \langle\psi_2| E_2 |\psi_2\rangle = \langle\psi_2| S^\dagger S |\psi_2\rangle = |\beta|^2 \langle\phi| S^\dagger S |\phi\rangle < \langle\phi| E_2 |\phi\rangle \leq 1.$$

Where the last inequality comes from the fact that

$$\langle\phi| E_2 |\phi\rangle \leq \langle\phi| E_1 |\phi\rangle + \langle\phi| E_2 |\phi\rangle = \langle\phi| (E_1 + E_2) |\phi\rangle = \langle\phi|\phi\rangle = 1$$

because both E_1 and E_2 are positive-semidefinite operators. ■

That means that it would be against all known laws of Physics if we could reliably clone arbitrary quantum states or distinguish non-orthogonal quantum states.

3.2 Variations on quantum mechanics

Now we will explore some variations on quantum mechanics that would allow us to provably solve NP-complete problems in quantum polynomial time. In particular, we want to study what would happen if we could perform transformations other than unitary to *qubits* and if the measurement of quantum systems was a bit different.

3.2.1 Linear transformations

Many arguments against quantum computers claim that we do not really know whether the theory of quantum mechanics will remain valid in the regime tested by quantum computers. However, most of the proposals for how quantum mechanics could be wrong suggest a world with more computational power than BQP (see [2, p. 7]).

For example, what would happen if we added a tiny nonlinear term to the Schrödinger equation? Abrams and Lloyd [4] proved that they could solve NP-complete (and even #P-complete) problems in quantum polynomial time if they could use some nonlinear transformations that were naturally defined based on a nonlinear model of quantum mechanics described by Weinberg [29]. Moreover, if we could use arbitrary nonlinear operators then we could solve PSPACE-complete problems in polynomial time [2, p. 8].

That begs the question whether we could relax that hypothesis and define a model of quantum mechanics where we could perform certain types of linear transformations to quantum states that would allow us to provably solve NP-complete problems in quantum polynomial time, and what physical consequences it would have. In particular, we focused on the following types of linear transformations:

1. Unitary operators. This leads to the model of quantum mechanics that we have already introduced. As we mentioned before, at the time of writing this it is unclear whether NP $\subset$ BQP, but there is no evidence that it is the case and that we would be able to solve NP-complete problems with standard quantum computers in polynomial time.
2. Linear operators with unitary complex eigenvalues.
3. Linear operators whose determinant is a unitary complex number.
4. Invertible linear operators.

Note that each type is contained in the following ones in the list (see, for example, proposition A.14), so we are interested in identifying what is the slightest modification of quantum mechanics that would let us solve NP-complete problems in polynomial time (if any among those above applies). It is also important to notice that we could have focused on another classification of linear transformations that may (or not) have led to a better understanding of the problem, but we consider that this approach will shed light on that question.

In all those cases (except the first one), we would need to modify the postulate of quantum mechanics that describes how quantum states evolve, because the unitarity of states may no longer be preserved.

Postulate 3. The time evolution of a closed quantum system is described by linear transformations of type 2, 3 or 4. That is, the state of the system at time t is related to the state of the system at time t_0 by a linear operator L of the corresponding type as follows

$$\left|\psi\left(t\right)\right\rangle = \frac{L\left|\psi\left(t_0\right)\right\rangle}{\sqrt{\left\langle\psi\left(t_0\right)\right| L^\dagger L\left|\psi\left(t_0\right)\right\rangle}}.$$

Note that we could apply multiple linear transformations and normalize the vector state at the end of the computation.

Let φ be a Boolean formula on n variables and let $s \in \{0, \ldots, 2^n\}$ be the number of satisfying assignments of φ. We want to find out whether φ is satisfiable or not, that is, if $s > 0$ or not.

Starting with $n+1$ *qubits* with state $\left|0^n\right\rangle\left|0\right\rangle$, we apply the Hadamard transformation to the first n of them, obtaining the state

$$\left(\mathrm{H}^{\otimes n} \otimes I\right)\left(\left|0^n\right\rangle\left|0\right\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} \left|x\right\rangle\left|0\right\rangle.$$

Then, we evaluate the Boolean function φ applying the oracle U_φ (see example A.15) to get the state

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} \left|x\right\rangle\left|\varphi\left(x\right)\right\rangle.$$

Finally, we apply the Hadamard transformation (as shown in page 27) to the first n *qubits* once more to be left with the state

$$\left|\psi\right\rangle = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y} \left|y\right\rangle\left|\varphi\left(x\right)\right\rangle.$$

Note that if $s = 0$, measuring the last *qubit* in the computational basis would always yield the outcome 0; and that if $s = 2^n$, it would always yield the outcome 1.

We can check that if we measured the first n *qubits* in the computational basis we would obtain the outcome 0^n with probability

$$\begin{aligned}
p\left(0^n\right) &= \left\langle\psi\right|\left(\left|0^n\right\rangle\left\langle 0^n\right| \otimes I\right)\left|\psi\right\rangle \\
&= \frac{1}{2^n}\left\langle\psi\right|\left(\sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}\left\langle 0^n|y\right\rangle\left|0^n\right\rangle\left|\varphi\left(x\right)\right\rangle\right) \\
&= \frac{1}{2^n}\left\langle\psi\right|\left(\sum_{x\in\{0,1\}^n} \left|0^n\right\rangle\left|\varphi\left(x\right)\right\rangle\right) \\
&= \frac{1}{2^{2n}} \sum_{x,z,y\in\{0,1\}^n} (-1)^{z\cdot y}\left\langle y|0^n\right\rangle\left\langle\varphi\left(z\right)|\varphi\left(x\right)\right\rangle
\end{aligned}$$

$$= \frac{1}{2^{2n}} \sum_{x,z \in \{0,1\}^n} \langle \varphi(z) | \varphi(x) \rangle .$$

Let $\varphi_0 = \{x \in \{0,1\}^n \colon \varphi(x) = 0\}$ and $\varphi_1 = \{x \in \{0,1\}^n \colon \varphi(x) = 1\}$, and note that $s = |\varphi_1|$ and $|\varphi_0| = 2^n - s$. So we can simplify the previous expression as follows

$$\sum_{x,z \in \{0,1\}^n} \langle \varphi(z) | \varphi(x) \rangle = \sum_{x,z \in \varphi_0} 1 + \sum_{x,z \in \varphi_1} 1 = (2^n - s)^2 + s^2.$$

That is,

$$p(0^n) = \frac{(2^n - s)^2 + s^2}{2^{2n}} \geq \frac{\left(2^n - 2^{n-1}\right)^2 + \left(2^{n-1}\right)^2}{2^{2n}} = \frac{2\left(2^{n-1}\right)^2}{2^{2n}} = \frac{1}{2},$$

where the inequality comes from the fact that the function $h(x) = x^2 + (1-x)^2$, for $x \in [0,1]$, reaches its minimum value at $x = 1/2$.

And in that case, the state would collapse to

$$\begin{aligned} \frac{1}{\sqrt{(2^n - s)^2 + s^2}} \sum_{x \in \{0,1\}^n} |0^n\rangle |\varphi(x)\rangle &= |0^n\rangle \otimes \frac{1}{\sqrt{(2^n - s)^2 + s^2}} \left(\sum_{x \in \varphi_0} |0\rangle + \sum_{x \in \varphi_1} |1\rangle \right) \\ &= |0^n\rangle \otimes \frac{(2^n - s)|0\rangle + s|1\rangle}{\sqrt{(2^n - s)^2 + s^2}}. \end{aligned}$$

Thus, discarding the first n *qubits*, and with probability at least $1/2$, we are left with the state

$$|\text{AL}\rangle = \frac{(2^n - s)|0\rangle + s|1\rangle}{\sqrt{(2^n - s)^2 + s^2}}. \tag{AL}$$

This state, that gets its name from Abrams and Lloyd [4], will be the starting point for the algorithms that we will propose in this section, because we just need to be able to distinguish the cases $s > 0$ and $s = 0$ to decide whether φ is satisfiable or not. Next we will show how we can do it.

Note that we have not used any non-unitary transformation yet.

Consider the invertible linear operator L_0 given by $L_0|0\rangle = |0\rangle$ and $L_0|1\rangle = 2|1\rangle$. Then, given a Boolean formula φ we can decide whether it is satisfiable or not with a probability of success of $2/3$ in polynomial time using L_0.

Proof. Using the matrix representation of L_0 in the computational basis, we observe that applying it k times to the state $|\text{AL}\rangle$ from AL would yield the state

$$\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}^k \begin{pmatrix} 2^n - s \\ s \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 2^k \end{pmatrix} \begin{pmatrix} 2^n - s \\ s \end{pmatrix} = \begin{pmatrix} 2^n - s \\ 2^k s \end{pmatrix}$$

before normalizing. That is, the state

$$|\phi\rangle = \frac{(2^n - s)|0\rangle + 2^k s|1\rangle}{\sqrt{(2^n - s)^2 + 2^{2k} s^2}}.$$

We want to pick k such that if $s > 0$ and we measure the state in the computational basis we would get the outcome 1 with high probability. Note that if $s = 0$ we would always get the outcome 0 and that if $s = 2^n$ we would always get the outcome 1. So, picking $k = n$ and comparing the size of the amplitudes of states $|1\rangle$ and $|0\rangle$ (dividing the former by the later) for $0 < s < 2^n$, we get that

$$\frac{\left|\langle 1|\phi\rangle\right|}{\left|\langle 0|\phi\rangle\right|} = \frac{|2^n s|}{|2^n - s|} = \frac{2^n s}{2^n - s} \geq \frac{2^n}{2^n - 1} \geq 1$$

for any $0 < s < 2^n$. That is, for $0 < s < 2^n$, $\langle\phi|1\rangle\langle 1|\phi\rangle = \langle\phi|0\rangle\langle 0|\phi\rangle = 1/2$ in the worst case.

That means that for $s > 0$ we would get the outcome 1 with a probability of $1/4$ in the worst case (note that we are only obtaining the initial state with probability at least $1/2$). But that is enough to build an algorithm that would return 1 if φ is satisfiable and 0 if not with a probability of success of $2/3$. To do it, we would simply repeat the experiment above a number c of times (to be chosen later). If any of the c outcomes was 1, then we know for sure that φ is satisfiable, because if $s = 0$ we would always get outcome 0. On the other hand, if all outcomes were 0, we would say that φ is unsatisfiable. However, it might be the case that φ was satisfiable and all outcomes were 0, but it would only happen with a probability of, at most, $\left(3/4\right)^c$. Since we want this value to be lower that $1/3$, we would need

$$\begin{aligned}\left(\frac{3}{4}\right)^c < \frac{1}{3} &\iff c\log_2\left(\frac{3}{4}\right) < \log_2\left(\frac{1}{3}\right)\\ &\iff c > \frac{\log_2\left(1/3\right)}{\log_2\left(3/4\right)} \approx 3.81.\end{aligned}$$

So we can pick $c = 4$. ■

We proved the following result.

Theorem. *If the time evolution of closed quantum systems is described by invertible linear operators, we can decide whether a given Boolean formula φ is satisfiable or not with a probability of success of $2/3$ in polynomial time.*

We also proved it using another linear operator L_1 given by $L_1|0\rangle = |0\rangle$ and $L_1|+\rangle = |1\rangle$.

A slight improvement over our last result lets us prove that we can solve SAT in quantum polynomial time with a probability of success of $2/3$ using linear operators whose determinant is a complex number of unit norm. In particular, applying the linear operator L_2 given by $L_2|0\rangle = (1/2)|0\rangle$ and $L_2|1\rangle = 2|1\rangle$ to the state $|\text{AL}\rangle$ a polynomial number of times.

Theorem. *If the time evolution of closed quantum systems is described by linear operators whose determinant is a complex number of unit norm, we can decide whether a given Boolean formula φ is satisfiable or not with a probability of success of $2/3$ in polynomial time.*

Proof. The proof is almost the same as before. Applying the linear operator L_2 to the state $|\text{AL}\rangle$ k times would yield the state

$$\frac{\left(\frac{2^n-s}{2^k}\right)|0\rangle + 2^k s\,|1\rangle}{\sqrt{\left(\frac{2^n-s}{2^k}\right)^2 + 2^{2k}s^2}}.$$

And as before, we can compare the size of both amplitudes dividing one by the other, so we have that

$$\frac{\left|\langle 1|\phi\rangle\right|}{\left|\langle 0|\phi\rangle\right|} = \frac{\left|2^k s\right|}{\left|\frac{2^n-s}{2^k}\right|} = \frac{2^{2k}s}{2^n - s},$$

and picking $k = n/2$ we have the same relationship as before, so we can repeat the argument. ■

Note that even though we restricted ourselves to use a smaller class of linear operators, we could still solve the same problem and in less time (the difference is just a constant factor, but it is still an improvement). This seems to suggest that it does not make any difference to distinguish between linear operators with unitary determinant and invertible linear operators.

We have not been able to extend this result for linear operators with unitary eigenvalues, although we think that it might not be possible. The issue deserves further investigation.

However, as expected, such a modification of the quantum theory would lead to other unexpected physical consequences that contradict some known well-established physical principles. In particular, we can easily show how we could distinguish two arbitrary quantum states if the time evolution of quantum states was described by arbitrary invertible linear operators.

Let $|\psi\rangle$ and $|\phi\rangle$ two distinct states, not necessarily orthonormal, that we would like to distinguish. And let $|\Psi\rangle$ and $|\Phi\rangle$ be two orthonormal states (that we *can* distinguish). Then, we could just apply the linear transformation C given by $C\,|\psi\rangle = |\Psi\rangle$ and $C\,|\phi\rangle = |\Phi\rangle$ to transform the states into two distinguishable ones. Note that C is invertible because it transforms a basis into another basis (or at least it could be extended to a linear operator that transforms a basis into another one acting on a bigger Hilbert space that is the state space of some quantum system).

We showed that if quantum states could evolve according to invertible linear operators or linear operators with complex eigenvalues of module 1, then we could solve NP-complete problems efficiently, however, that would be against our understanding of Nature.

3.2.2 Deterministic measurement

In which possible ways could we modify the way quantum measurement works to be able to exploit the parallelism of quantum mechanics to solve NP-complete problems more efficiently? This question is what motivated us to research what consequences would a "deterministic" measurement of quantum systems have.

We propose the following modification of the measurement postulate of quantum mechanics.

Postulate 4. Quantum measurements are described by a collection $\{M_m\}_m$ of *measurement operators* on the state space of the system being measured. The index m refers to the possible measurement outcomes. The measurement operators must verify the *completeness equation*

$$\sum_m M_m^\dagger M_m = I.$$

If the state of the system is $|\psi\rangle$ at the moment of measuring, we would get the only outcome m that maximizes the value

$$\langle\psi| M_m^\dagger M_m |\psi\rangle.$$

If there are multiple possible outcomes that maximize that value, then the outcome would be any of them uniformly at random.

And in the case of obtaining the outcome m, the state would *collapse* to

$$\frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}}.$$

Note that there still may be some randomness in measurements.

For example, if we measured the state $|+\rangle = \left(|0\rangle + |1\rangle\right)/\sqrt{2}$ in the computational basis (that is, with measurement operators $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$), we would have the outcome 0 or 1 with equal probability $1/2$. But if we performed the same measurement on the state $\left(\sqrt{3}|0\rangle + |1\rangle\right)/2$ we would have the outcome 0, because

$$\frac{1}{4}\left(\sqrt{3}\langle 0| + \langle 1|\right)|0\rangle\langle 0|\left(\sqrt{3}|0\rangle + |1\rangle\right) = \frac{1}{4}\left(\sqrt{3}\langle 0|0\rangle\right)^2 = \frac{3}{4} > \frac{1}{2}.$$

Let φ be a Boolean formula on n variables and let $s \in \{0, \ldots, 2^n\}$ be the number of satisfying assignments of φ. If we can determine if $s > 0$, we would determine if φ is satisfiable. To do that, we will proceed as in the previous section. In fact, after repeating the first three steps we would be left with the state

$$|\psi\rangle = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle\left|\varphi(x)\right\rangle.$$

We have also already shown that

$$\langle\psi|\left(|0^n\rangle\langle 0^n| \otimes I\right)|\psi\rangle \geq \frac{1}{2},$$

and it is easy to see that for any other $y \in \{0,1\}^n$ we would have that

$$\langle\psi|\left(|y\rangle\langle y| \otimes I\right)|\psi\rangle < \frac{1}{2},$$

which means that if we measured the first n *qubits* of $|\psi\rangle$ in the computational basis we would get the outcome 0^n and the state would collapse to

$$|0^n\rangle \otimes \frac{(2^n - s)|0\rangle + s|1\rangle}{\sqrt{(2^n - s)^2 + s^2}}.$$

That is, ignoring the first n *qubits*, the state $|\text{AL}\rangle$ from AL.

Note that we want to distinguish the case where $s = 0$ from the case where $s > 0$. To do that, we can try to view each of the possible $2^n + 1$ possible states for each value of s on the Bloch sphere and try to separate the state where $s = 0$ from the rest. For example, for $n = 3$ we would have the 9 states shown as blue dots on figure 3.1.

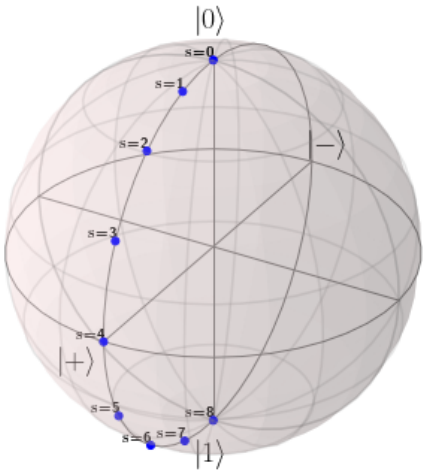


FIGURE 3.1: State $|\text{AL}\rangle$ for each possible value of $s \in \{0, \ldots, 2^n\}$ and $n = 3$.

The idea is, then, to apply a rotation to the state $|\text{AL}\rangle$ in a way that would leave the state with $s = 0$ above the equator and the rest under it, so we could distinguish both cases with just one measurement. However, that transformation depends on n, so we will try a different approach: perform a rotation of $\pi/2$ radians on the y-axis. This transformation will leave the states from figure 3.1 as shown in figure 3.2.

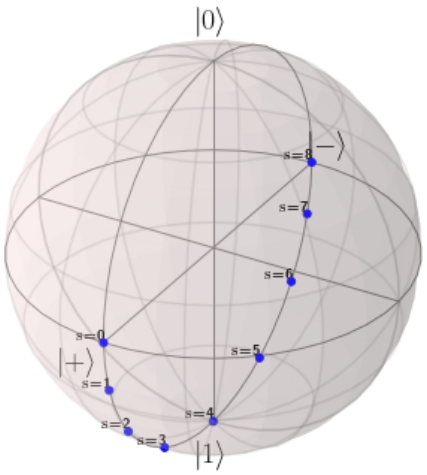


FIGURE 3.2: State $R_y(\pi/2)\,|\text{AL}\rangle$ for each possible value of $s \in \{0, \ldots, 2^n\}$ and $n = 3$.

Rotations around the y-axis are given by the unitary transformation $R_y(\theta)$ such that

$$R_y(\theta): \begin{array}{rcl} |0\rangle & \mapsto & \cos\left(\theta/2\right)|0\rangle + \sin\left(\theta/2\right)|1\rangle \\ |1\rangle & \mapsto & -\sin\left(\theta/2\right)|0\rangle + \cos\left(\theta/2\right)|1\rangle . \end{array}$$

In our particular case, where $\theta = \pi/2$, we would have the transformation given by

$$R_y(\pi/2): \begin{array}{rcl} |0\rangle & \mapsto & \cos\left(\pi/4\right)|0\rangle + \sin\left(\pi/4\right)|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \\ |1\rangle & \mapsto & -\sin\left(\pi/4\right)|0\rangle + \cos\left(\pi/4\right)|1\rangle = \frac{-1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) . \end{array}$$

Note that $R_y\left(\pi/2\right) = \mathrm{XH}$.

So applying that rotation to our state we would have that

$$\begin{aligned} R_y\left(\pi/2\right)\left(\frac{(2^n-s)|0\rangle + s|1\rangle}{\sqrt{(2^n-s)^2+s^2}}\right) &= \frac{(2^n-s)\,R_y\left(\pi/2\right)|0\rangle + s\,R_y\left(\pi/2\right)|1\rangle}{\sqrt{(2^n-s)^2+s^2}} \\ &= \frac{(2^n-s)\left(|0\rangle+|1\rangle\right) + s\left(-|0\rangle+|1\rangle\right)}{\sqrt{2}\sqrt{(2^n-s)^2+s^2}} \\ &= \frac{(2^n-2s)|0\rangle}{\sqrt{2}\sqrt{(2^n-s)^2+s^2}} + \frac{2^n\,|1\rangle}{\sqrt{2}\sqrt{(2^n-s)^2+s^2}} \\ &= |\phi\rangle . \end{aligned}$$

If we measured that state in the computational basis, since

$$\langle\phi|0\rangle\langle 0|\phi\rangle = \frac{(2^n-2s)^2}{2\left((2^n-s)^2+s^2\right)}$$

and

$$\langle\phi|1\rangle\langle 1|\phi\rangle = \frac{2^{2n}}{2\left((2^n-s)^2+s^2\right)},$$

we would have that

$$\langle\phi|0\rangle\langle 0|\phi\rangle < \langle\phi|1\rangle\langle 1|\phi\rangle$$

if and only if $|2^n - 2s| < |2^n|$, which will be true for every $s \in \{1, \ldots, 2^n - 1\}$, and we would get the outcome 1. On the other hand, for $s \in \{0, 2^n\}$ we would have that

$$\langle\phi|0\rangle\langle 0|\phi\rangle = \langle\phi|1\rangle\langle 1|\phi\rangle ,$$

getting any outcome with probability 1/2.

But note that we can discard the case $s = 2^n$ simply by evaluating our Boolean function φ on any assignment. If that assignment makes φ true, then we have already decided if it is satisfiable or not. On the other hand, if it does not, then we know that $s < 2^n$.

Now we can specify an algorithm to decide if φ is satisfiable in polynomial time. First, we evaluate $\varphi\left(0^n\right)$ to decide whether it is satisfiable or $s < 2^n$. If 0^n does not make φ true, we continue with the algorithm by preparing the state $|\mathrm{AL}\rangle$ from AL

and measure it. If the outcome of the measurement is 0, then we know that φ is not satisfiable. On the other hand, if it is 1, it could be the case that φ is unsatisfiable but we obtained the wrong outcome with probability 1/2. We would just need to repeat the algorithm once more. If we obtained outcome 1 once again, it could be possible that φ is unsatisfiable, but only with a probability of $1/4 < 1/3$. That is, the overall probability of getting the right result is $3/4 > 2/3$.

That proves the following result.

Theorem. *If quantum measurement was deterministic in the way we proposed, then we could solve* SAT *with a probability of success of* $2/3$ *in quantum polynomial time.*

This variation of quantum mechanics would also lead to some consequences that contradict well-established physical principles. In particular, this kind of measurement would also allow us to distinguish two arbitrary *qubits*.

To see it, imagine the two *qubits* that we want to distinguish as points on the Bloch sphere. If we manage to rotate them across it so that one is above the equator and the other is under it, a measurement in the computational basis would allow us to distinguish them: the outcome of measuring the state above the equator would be 0, while the outcome of measuring the other state would be 1. So the idea of the proof involves determining what is their middle point on the sphere and rotate it so that it now lays on the equator. Then, since the angles between states are preserved by unitary transformations, one of the states would be in the northern hemisphere and the other would be in the southern hemisphere (unless they were initially in the same parallel, but that case can be solved by changing the basis appropriately).

Formally, we will first show how to reliably distinguish any *qubit* with state $|\psi\rangle$ from the one with state $|0\rangle$ and then extend it to any pair of *qubits*.

Let $|\gamma\rangle$ be an unknown state that is equal to $|0\rangle$ or to $|\psi\rangle$. We are going to show how to transform that state so that measuring it would yield the result 0 if it is the state $|0\rangle$ and it would yield the outcome 1 if it is the state $|\psi\rangle$. Using the Bloch sphere representation, we could write $|\psi\rangle = \cos\left(\theta/2\right)|0\rangle + \mathsf{e}^{\mathsf{i}\phi}\sin\left(\theta/2\right)|1\rangle$ for certain $\theta \in [0,\pi]$ and $\phi \in [0,2\pi)$. Since we know the state $|\psi\rangle$ that we are trying to distinguish from $|0\rangle$, we are going to suppose that we also know the values of θ and ϕ.

If $\theta > \pi/2$, we could just measure the state in the computational basis to distinguish if it is $|0\rangle$ or not. If $|\gamma\rangle$ is equal to $|\psi\rangle$, we would have that

$$\langle\psi|1\rangle\langle 1|\psi\rangle = \sin^2\left(\theta/2\right) > \frac{1}{2} > \cos^2\left(\theta/2\right) = \langle\psi|0\rangle\langle 0|\psi\rangle\,,$$

where the inequalities hold for $\theta \in \left(\pi/2,\pi\right]$, so we would measure the outcome 1. Intuitively, this works because for $\theta > \pi/2$ the state $|\psi\rangle$ would already be in the southern hemisphere.

On the other hand, if $\theta \in \left(0,\pi/2\right]$, we would perform a rotation around the z-axis of $-\phi$ radians followed by another rotation of $(\pi-\theta)/2$ radians around the y-axis before measuring the state in the computational basis.

Rotations around the z-axis are given by the unitary transformation $R_z(\theta)$ such that

$$R_z(\theta):\ \begin{array}{rcl} |0\rangle & \mapsto & \mathsf{e}^{-\mathsf{i}\theta/2}|0\rangle \\ |1\rangle & \mapsto & \mathsf{e}^{\mathsf{i}\theta/2}|1\rangle\,. \end{array}$$

So in our case we would have the transformations given by

$$R_z\left(-\phi\right) : \begin{array}{ccc} |0\rangle & \mapsto & \mathsf{e}^{\mathsf{i}\phi/2}\,|0\rangle \\ |1\rangle & \mapsto & \mathsf{e}^{-\mathsf{i}\phi/2}\,|1\rangle \end{array}$$

and

$$R_y\left(\frac{\pi-\theta}{2}\right) : \begin{array}{ccc} |0\rangle & \mapsto & \cos\left(\frac{\pi-\theta}{4}\right)|0\rangle + \sin\left(\frac{\pi-\theta}{4}\right)|1\rangle \\ |1\rangle & \mapsto & -\sin\left(\frac{\pi-\theta}{4}\right)|0\rangle + \cos\left(\frac{\pi-\theta}{4}\right)|1\rangle\,. \end{array}$$

Those rotations would act on the state $|0\rangle$ in the following way

$$\begin{aligned} R_y\left(\frac{\pi-\theta}{2}\right) R_z\left(-\phi\right)|0\rangle &= \mathsf{e}^{\mathsf{i}\phi/2}\, R_y\left(\frac{\pi-\theta}{2}\right)|0\rangle \\ &\equiv R_y\left(\frac{\pi-\theta}{2}\right)|0\rangle \\ &= \cos\left(\frac{\pi-\theta}{4}\right)|0\rangle + \sin\left(\frac{\pi-\theta}{4}\right)|1\rangle \\ &= |\psi_0\rangle\,. \end{aligned}$$

Then, the outcome of measuring that state in the computational basis would be 0, because

$$\langle\psi_0|0\rangle\,\langle 0|\psi_0\rangle = \cos^2\left(\frac{\pi-\theta}{4}\right) > \frac{1}{2} > \sin^2\left(\frac{\pi-\theta}{4}\right) = \langle\psi_0|1\rangle\,\langle 1|\psi_0\rangle\,,$$

where the inequalities $\cos^2(x) > 1/2 > \sin^2(x)$ hold for $x \in \left[0, \pi/4\right)$, and since $\theta \in \left(0, \pi/2\right]$, we then have that $(\pi-\theta)/4 \in \left[\pi/8, \pi/4\right) \subset \left[0, \pi/4\right)$.

The effect of those rotations on the state $|\psi\rangle$ would be

$$\begin{aligned} R_y\left(\frac{\pi-\theta}{2}\right) R_z\left(-\phi\right)|\psi\rangle &= \mathsf{e}^{\mathsf{i}\phi/2}\, R_y\left(\frac{\pi-\theta}{2}\right)\Big(\cos\left(\theta/2\right)|0\rangle + \sin\left(\theta/2\right)|1\rangle\Big) \\ &\equiv R_y\left(\frac{\pi-\theta}{2}\right)\Big(\cos\left(\theta/2\right)|0\rangle + \sin\left(\theta/2\right)|1\rangle\Big) \\ &= \cos\left(\theta/2\right)\left(\cos\left(\frac{\pi-\theta}{4}\right)|0\rangle + \sin\left(\frac{\pi-\theta}{4}\right)|1\rangle\right) \\ &\qquad + \sin\left(\theta/2\right)\left(-\sin\left(\frac{\pi-\theta}{4}\right)|0\rangle + \cos\left(\frac{\pi-\theta}{4}\right)|1\rangle\right) \\ &= \cos\left(\frac{\pi+\theta}{4}\right)|0\rangle + \sin\left(\frac{\pi+\theta}{4}\right)|1\rangle \\ &= |\psi_1\rangle\,, \end{aligned}$$

where we used the identities $\sin(t+s) = \sin(t)\cos(s) + \sin(s)\cos(t)$ and $\cos(t+s) = \cos(t)\cos(s) - \sin(s)\sin(t)$. And in this case, the outcome of measuring $|\psi_1\rangle$ in the computational basis would be 1, because

$$\langle\psi_1|1\rangle\,\langle 1|\psi_1\rangle = \sin^2\left(\frac{\pi+\theta}{4}\right) > \frac{1}{2},$$

because $\sin^2(x) > 1/2$ holds for $x \in (\pi/4, 3\pi/4)$ and $(\pi+\theta)/4 \in (\pi/4, 3\pi/8] \subset (\pi/4, 3\pi/4)$.

And thus, we can reliably distinguish both states.

To extend the result for arbitrary *qubits* $|\psi\rangle$ and $|\phi\rangle$, we would just need to consider a unitary transformation U such that $U|\psi\rangle = |0\rangle$, and then distinguish the states $|0\rangle$ and $U|\phi\rangle$.

In brief, although deterministically measuring quantum states would let us decide if a Boolean formula is satisfiable in polynomial time, it would also let us reliably distinguish any two *qubits*, and that would be contradictory the laws of Physics.

3.2.3 Probability stabilizers and enhancers

In an attempt to generalize all the previous results, we noticed that we were always juggling around with how the probabilities of the different outcomes should change in order to achieve the desired behaviour for our algorithms. In particular, how we needed big amplitudes to diminish so that states with smaller amplitudes could increase their probability of being measured. Applying this to the state $|\text{AL}\rangle$, we wanted to increase the amplitude of the state $|1\rangle$ compared to the one of $|0\rangle$ so that we could measure the outcome 1 with a probability big enough so that we could distinguish between the $s = 0$ and $s > 0$ cases.

This led to the concept of probability-stabilizers that we will be covering in this section.

Definition 3.1. Let $|\psi\rangle$ be a quantum state and consider the quantum measurement described by measurement operators $\{M_m\}_m$. A probability-stabilizer of degree $\alpha \geq 1$ is any transformation on the state $|\psi\rangle$ such that if $\{p_m\}_m$ is the set of probabilities of obtaining any outcome m when measuring $|\psi\rangle$, then those will be changed in the following way

$$p_m \mapsto p'_m \coloneqq \frac{p_m^{1/\alpha}}{\sum_l p_l^{1/\alpha}},$$

so that

$$\sum_m p'_m = \sum_m \frac{p_m^{1/\alpha}}{\sum_l p_l^{1/\alpha}} = \frac{1}{\sum_l p_l^{1/\alpha}} \sum_m p_m^{1/\alpha} = 1.$$

Note that the case where $\alpha = 1$ corresponds to the identity transformation. And that making $\alpha \to \infty$ causes the probabilities that are greater than 0 to be all equal, because for any m_1 and m_2 such that $p_{m_1}, p_{m_2} > 0$, we have that

$$\frac{p'_{m_1}}{p'_{m_2}} = \frac{p_{m_1}^{1/\alpha}}{p_{m_2}^{1/\alpha}} = \left(\frac{p_{m_1}}{p_{m_2}}\right)^{1/\alpha} \xrightarrow[\alpha\to\infty]{} 1.$$

The following proposition shows why this definition may be useful.

Proposition 3.2. Applying a probability-stabilizer of sufficiently high degree α to the state $|\text{AL}\rangle$ (see AL) and measuring in the computational basis would let us distinguish the cases where $s = 0$ and $s > 0$, and thus deciding if a Boolean formula φ is satisfiable. Moreover, $\alpha \in \Omega(n)$ is enough.

Proof. Starting with state

$$|\text{AL}\rangle = \frac{(2^n - s)\,|0\rangle + s\,|1\rangle}{\sqrt{(2^n - s)^2 + s^2}}$$

we have that the probabilities of getting each outcome are given by

$$p_0 = \langle \text{AL}|0\rangle \langle 0|\text{AL}\rangle = \frac{(2^n - s)^2}{(2^n - s)^2 + s^2}$$

for the probability of getting the outcome 0, and

$$p_1 = \langle \text{AL}|1\rangle \langle 1|\text{AL}\rangle = \frac{s^2}{(2^n - s)^2 + s^2}$$

for the probability of getting the outcome 1.

If $s = 0$ we have that $p_0 = 1$ and $p_1 = 0$, so those probabilities will not be affected by any probability-stabilizer. It also happens if $s = 2^n$, because in that case $p_0 = 0$ and $p_1 = 1$.

However, a probability-stabilizer would affect the probabilities if $0 < s < 2^n$, because then $0 < p_0, p_1 < 1$. In that case, we would have that

$$\frac{p_1'}{p_0'} \xrightarrow[\alpha\to\infty]{} 1$$

understanding that quotient as a function of α. Which means that there exists a value of $\alpha \geq 1$ such that a probability-stabilizer of degree α would have the property that, for example,

$$\frac{p_1'}{p_0'} \geq \frac{2}{3},$$

which would imply that $p_1' \geq 2/5$, for $0 < s < 2^n$, since $p_0' + p_1' = 1$.

Now we could distinguish the cases $s = 0$ and $s > 0$ as follows. We would apply the probability-stabilizer to state $|\text{AL}\rangle$ and measure it a number c of times (to be chosen later). If any of the c outcomes is 1, we certainly know that $s > 0$. On the other hand, if all outcomes were 0 we would say that $s = 0$. However, it might be the case where $s > 0$ and all outcomes were 0 with probability at most $\big(3/5\big)^c$. We want that value to be lower than $1/3$, so we would need that

$$\begin{aligned}\left(\frac{3}{5}\right)^c < \frac{1}{3} &\iff c\log_2\left(\frac{3}{5}\right) < \log_2\left(\frac{1}{3}\right) \\ &\iff c > \frac{\log_2\big(1/3\big)}{\log_2\big(3/5\big)} \approx 2.15.\end{aligned}$$

So taking $c = 3$ we would be able to distinguish between the two cases with a probability of success of at least $2/3$.

To see the last part of the statement, we simply notice that

$$\frac{p_1'}{p_0'} = \left(\frac{s^2}{(2^n - s)^2}\right)^{1/\alpha} \geq \left(\frac{1}{(2^n - 1)^2}\right)^{1/\alpha} = \frac{1}{(2^n - 1)^{2/\alpha}},$$

where the inequality is tight (the equality holds for $s = 1$). So if we want that value to be greater that $2/3$, we have that

$$\frac{1}{(2^n-1)^{2/\alpha}} \geq \frac{2}{3} \iff \frac{2}{\alpha}\log_2(2^n-1) \geq \log_2(3/2)$$
$$\iff \alpha \geq \frac{2\log_2(2^n-1)}{\log_2(3/2)}.$$

With

$$\frac{2\log_2(2^n-1)}{\log_2(3/2)} \in \Theta(n)$$

because

$$\frac{2}{\log_2(3/2)}n \geq \frac{2\log_2(2^n-1)}{\log_2(3/2)} \geq \frac{2}{\log_2(3/2)}(n-1).$$

So $\alpha \in \Omega(n)$ is enough to achieve $p'_1 = 2/5$ with a probability-stabilizer of degree α. ■

We suspect that if $\alpha \in o(n)$, that is, if $\alpha/n \xrightarrow[\alpha\to\infty]{} 0$, then a probability-enhancer of degree α would not let us distinguish the cases $s = 0$ and $s > 0$ repeating the experiment at most a polynomial number of times. We could not prove it, though.

This result leaves us with the question of how could we build a probability-stabilizer in polynomial time to be able to solve NP-complete problems efficiently. This does not seem to be easy because the definition of a probability-stabilizer sounds very restrictive. Could we relax the definition and still being able to prove a result like proposition 3.2?

The answer is that we can.

Definition 3.3. Let $|\psi\rangle$ be a quantum state and consider the quantum measurement described by measurement operators $\{M_m\}_m$. A k-probability-enhancer of degree $\alpha \geq 1$ is any transformation on the state $|\psi\rangle$ such that if $\{p_m\}_m$ is the set of probabilities of getting any outcome m when measuring $|\psi\rangle$, then those will be changed to $\{p'_m\}_m$ such that

$$p'_k \geq \frac{p_k^{1/\alpha}}{\sum_l p_l^{1/\alpha}}.$$

Note that in this definition we only require one of the possible outcomes to increase its probability at the expense of the probabilities of the other possible outcomes.

The relationship between probability-stabilizers and k-probability-enhancers is that a probability-stabilizer of degree α is a k-probability-enhancer of degree α for any possible outcome k. The reciprocal is also true: any transformation that is a k-probability-enhancer of degree α for any possible outcome k is also a probability-stabilizer of degree α.

Proposition 3.4. Applying a 1-probability-enhancer of sufficiently high degree α to the state $|\text{AL}\rangle$ and measuring it in the computational basis would let us distinguish the cases where $s = 0$ and $s > 0$, and thus deciding if a Boolean formula φ is satisfiable.

Proof. The proof is analogous to the one of proposition 3.2, because for $0 < s < 2^n$ we have that

$$\frac{p_1'}{p_0'} \geq \frac{p_1^{1/\alpha}}{p_0^{1/\alpha}} = \left(\frac{p_1}{p_0}\right)^{1/\alpha} \xrightarrow[\alpha\to\infty]{} 1$$

as before, so we can guarantee that there exists a value of $\alpha \geq 1$ such that a 1-probability-enhancer of degree α would have the property that

$$\frac{p_1'}{p_0'} \geq \frac{2}{3}.$$

The rest of the proof would be the same. ■

Finally, we would like to study if the proposed variations of quantum mechanics would allow us to build a 1-probability-enhancer for the state $|\text{AL}\rangle$ measuring it in the computational basis.

The probabilities of getting outcomes 0 and 1 when measuring the state $|\text{AL}\rangle$ are, respectively,

$$p_0 = \frac{(2^n - s)^2}{(2^n - s)^2 + s^2} \text{ and } p_1 = \frac{s^2}{(2^n - s)^2 + s^2}.$$

Example (Building a 1-probability enhancer with deterministic measurements)**.** Applying the algorithm we described for the deterministic measurements, we would have that the probabilities would change in the following way

$$p_0' = \begin{cases} 1 & \text{if } s = 0 \\ 0 & \text{if } 0 < s < 2^n \\ 1/2 & \text{if } s = 2^n \end{cases} \text{ and } p_1' = \begin{cases} 0 & \text{if } s = 0 \\ 1 & \text{if } 0 < s < 2^n \\ 1/2 & \text{if } s = 2^n. \end{cases}$$

And we can see that the given transformation is indeed a 1-probability-enhancer, because for $s = 0$ we have that

$$p_1' = 0 = \frac{p_1^{1/\alpha}}{p_0^{1/\alpha} + p_1^{1/\alpha}}$$

for all $\alpha \geq 1$. And for $0 < s < 2^n$ we have that

$$p_1' = 1 \geq \frac{p_1^{1/\alpha}}{p_0^{1/\alpha} + p_1^{1/\alpha}}.$$

for all $\alpha \geq 1$. However, for $s = 2^n$,

$$p_1' = \frac{1}{2} \ngeq 1 = \frac{p_1^{1/\alpha}}{p_0^{1/\alpha} + p_1^{1/\alpha}}$$

for all $\alpha \geq 1$. But we already saw that we could discard that case: we could evaluate our Boolean formula in any assignment, and we would have that $s < 2^n$ or that the formula was satisfiable (so we would not need any 1-probability-enhancer!). ■

Example (Building a 1-probability enhancer with invertible linear operators)**.** If the time evolution of quantum systems was described by arbitrary invertible linear operators, we would also be able to build a 1-probability-enhancer. Consider the invertible linear operator L given by $L|0\rangle = |0\rangle$ and $L|1\rangle = 2|1\rangle$. Then, as we already saw, applying that transformation n times to the state $|\text{AL}\rangle$ would yield the state

$$|\phi\rangle = \frac{(2^n - s)|0\rangle + 2^n s|1\rangle}{\sqrt{(2^n - s)^2 + 2^{2n}s^2}}.$$

And then, we would have that

$$p_1' = \langle\phi|1\rangle\langle 1|\phi\rangle = \frac{2^{2n}s^2}{(2^n - s)^2 + 2^{2n}s^2}.$$

So we have to check if

$$\frac{2^{2n}s^2}{(2^n - s)^2 + 2^{2n}s^2} = p_1' \geq \frac{p_1^{1/\alpha}}{p_0^{1/\alpha} + p_1^{1/\alpha}} = \frac{s^{2/\alpha}}{(2^n - s)^{2/\alpha} + s^{2/\alpha}}.$$

for some value of $\alpha \geq 1$.

The inequality holds for $s = 0$ and for $s = 2^n$ for any $\alpha \geq 1$. For $0 < s < 2^n$, we have that

$$\begin{aligned}
&\frac{2^{2n}s^2}{(2^n - s)^2 + 2^{2n}s^2} \geq \frac{s^{2/\alpha}}{(2^n - s)^{2/\alpha} + s^{2\alpha}} \\
\iff\ & 2^{2n}s^2\left((2^n - s)^{2/\alpha} + s^{2/\alpha}\right) \geq s^{2/\alpha}\left((2^n - s)^2 + 2^{2n}s^2\right) \\
\iff\ & 2^{2n}s^2(2^n - s)^{2/\alpha} \geq s^{2/\alpha}(2^n - s)^2 \\
\iff\ & 2^{2n} \geq \left(\frac{2^n - s}{s}\right)^{2-2/\alpha}.
\end{aligned}$$

Since

$$\left(\frac{2^n - s}{s}\right) \leq 2^n - 1,$$

and the inequality is tight (the equality is reached for $s = 1$), we are looking for which values of $\alpha \geq 1$ we have that

$$2^{2n} \geq (2^n - 1)^{2-2/\alpha} \geq \left(\frac{2^n - s}{s}\right)^{2-2/\alpha}.$$

But it turns out that the inequality holds for any $\alpha \geq 1$, because

$$2^{2n} = (2^n)^2 > (2^n - 1)^2 > (2^n - 1)^{2-2/\alpha},$$

where we used that $0 \leq 2 - 2/\alpha < 2$ for $\alpha \geq 1$ in the last inequality. ■

Conclusions

Can **NP**-complete problems be solved efficiently in the physical Universe? Our objective was to identify if any physical assumption would let us provably solve **NP**-complete problems in polynomial time by the means of Nature, even if $\mathbf{P} \neq \mathbf{NP}$ and $\mathbf{NP} \not\subset \mathbf{BQP}$, and trying to relax them to study what would be the physical consequences if those assumptions were true.

We identified five assumptions that allowed us to solve **NP**-complete problems in polynomial time using the resources of the physical Universe: three of them from classical physics and two from quantum mechanics. However, two of those proposals (the superdense information assumption [28] and the hyperbolic geometry assumption [21], both from classical physics) had already been studied. But, as far as we know, the other approaches have not been investigated before: the instantaneous communication assumption, the deterministic measurement (of quantum mechanics) assumption, and the non-unitary transformations assumption (this last approach had already been followed using non-linear operators [4], but based on that work we focused on studying the consequences of using a certain classification of invertible linear operators).

We can draw two main conclusions from our research:

1. In order to be able to solve **NP**-complete problems in polynomial time, even if $\mathbf{P} \neq \mathbf{NP}$, by the means of Physics we apparently need to be able to handle an exponential amount of information in polynomial time. This can be clearly seen from the superdense information assumption. But also from the hyperbolic geometry assumption and the instantaneous communication assumption: note that what those two assumptions allowed us to do is to be able to populate an exponential number of cells of a cellular automaton in polynomial time.

2. We could also solve **NP**-complete problems in quantum polynomial time if we were able to separate two exponentially close quantum states in polynomial time. Recall from AL's state

$$|\mathrm{AL}\rangle = \frac{(2^n - s)\,|0\rangle + s\,|1\rangle}{\sqrt{(2^n - s)^2 + s^2}}$$

that if we were able to distinguish the cases where $s = 0$ and $s > 0$, then we could decide if a Boolean formula is satisfiable or not. However, unitary transformations preserve the angles between states, so it seems that we would not be able to distinguish those two cases using only linear operators.

This leads us to talk about our future work on this and related questions.

Future work

We do not think that we can relax any of the previous assumptions any further while being able to solve NP-complete problems in polynomial time. However, we left an open question during our research that would be interesting to study. That is if we can solve any NP-complete problem in quantum polynomial time using linear operators with unitary eigenvalues. Note that the only difference with unitary operators is that they can be diagonalized orthogonally (for being normal operators), so maybe we can exploit the non-orthogonality of the eigenvectors of linear operators with unitary eigenvalues to separate two exponentially close states in polynomial time, and thus solving SAT in quantum polynomial time.

Appendix A

Linear algebra

We know from Postulate 1 that each physical system is associated with a complex vector space (more specifically, a Hilbert space). Moreover, since we will mostly work with systems composed of a finite number n of *qubits* that have $\mathbb{C}^{2^n}$ as their state space (see Postulate 2), we'll be dealing with finite dimensional complex vector spaces.

The idea of this appendix is to introduce some linear algebraic concepts and prove some results that will be useful throughout the work.

A vector is represented as $|\psi\rangle$ where ψ is a label to identify it, whereas $\langle\psi|$ represents the dual vector to $|\psi\rangle$, that is, $\langle\psi|$ is a linear form that maps any vector $|\phi\rangle$ to $\big(|\psi\rangle, |\phi\rangle\big)$, the inner product of $|\psi\rangle$ and $|\phi\rangle$. This is called the *bra-ket* notation.

Definition A.1. Let V and W be two vector spaces. A linear operator $A\colon V \to W$ is any function that is linear on its arguments, that is,

$$A\left(\sum_{i=1}^{n} \alpha_i |v_i\rangle\right) = \sum_{i=1}^{n} \alpha_i A\left(|v_i\rangle\right).$$

A common way to study linear operators is through their matrix representation. Let $A\colon V \to W$ be a linear operator and let $\big\{|v_i\rangle : 1 \le i \le n\big\}$ and $\big\{\big|w_j\big\rangle : 1 \le j \le m\big\}$ be basis of V and W, respectively. Then, for any $i \in \{1, \dots, n\}$, the vector $A\left(|v_i\rangle\right) \in W$ can be written as

$$A\left(|v_i\rangle\right) = \sum_{j=1}^{m} \alpha_{i,j} |w_j\rangle$$

for some $\alpha_{i,j} \in \mathbb{C}$ for $j \in \{1, \dots, m\}$. The matrix $M = (\alpha_{i,j})_{i,j}$ of size $m \times n$ will be the matrix representation of A for the chosen basis, because we have that $A\left(|v\rangle\right) = M\,|v\rangle$ for any $|v\rangle \in V$.

Reciprocally, let M be an $m \times n$ matrix and let $\big\{|v_i\rangle : 1 \le i \le n\big\}$ and $\big\{\big|w_j\big\rangle : 1 \le j \le m\big\}$ be basis of V and W. Then, we can define a linear operator $A\colon V \to W$ given by $A\left(|v\rangle\right) = M\,|v\rangle$ for any $|v\rangle \in V$.

That is, fixed basis of V and W, there is a correspondence between linear operators and matrices of the right size. So we will be using both concepts interchangeably.

Definition A.2. An inner product in a complex vector space V is any transformation $(\cdot, \cdot) : V \times V \to \mathbb{C}$ with the following properties:

1. $\big(|v\rangle, |v\rangle\big) \ge 0$ for any $|v\rangle \in V$, and $\big(|v\rangle, |v\rangle\big) = 0$ if and only if $|v\rangle = 0_V$.

2. $\big(|u\rangle, |v\rangle\big) = \overline{\big(|v\rangle, |u\rangle\big)}$ for any $|u\rangle, |v\rangle \in V$, where $\overline{z}$ denotes the complex conjugate of $z \in \mathbb{C}$.
3. It's linear in its second argument, that is, for any $|u\rangle, |v_1\rangle, \ldots, |v_m\rangle \in V$

$$\left(|u\rangle, \sum_{i=1}^{m} |v_i\rangle\right) = \sum_{i=1}^{m} \big(|u\rangle, |v_i\rangle\big).$$

This allows us to have a notion of distance in complex vector spaces where we have defined an inner product. The norm of a vector $|v\rangle$ is given by

$$\big\||v\rangle\big\| = \sqrt{\big(|v\rangle, |v\rangle\big)},$$

and the distance between two vectors is the norm of their difference.

Definition A.3. We say that two vectors $|u\rangle, |v\rangle$ are orthogonal if $\big(|u\rangle, |v\rangle\big) = 0$. A set $\big\{|v_1\rangle, \ldots, |v_n\rangle\big\}$ is orthogonal if each pair of vectors in it are orthogonal. Additionally, the set is said to be orthonormal if all vectors are unitary, that is, their norm is equal to one.

Let $\{|v_i\rangle : 1 \leq i \leq n\}$ be an orthonormal basis in $\mathbb{C}^n$, then we can define the inner product of two vectors $|u\rangle = \alpha_1 |v_1\rangle + \cdots + \alpha_n |v_n\rangle$ and $|v\rangle = \beta_1 |v_1\rangle + \cdots + \beta_n |v_n\rangle$ as follows

$$\big(|u\rangle, |v\rangle\big) = \sum_{i=1}^{n} \overline{\alpha_i}\beta_i.$$

This is the inner product that we will be using from now on, unless explicitly specified.

Also, we will denote the inner product of two vectors $|u\rangle$ and $|v\rangle$ as $\langle u|v\rangle$. Note that for the inner product defined previously in $\mathbb{C}^n$ we have

$$\langle u|v\rangle = \sum_{i=1}^{n} \overline{\alpha_i}\beta_i = \begin{pmatrix} \overline{\alpha_1} & \cdots & \overline{\alpha_n} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix},$$

so we can represent the dual vector $\langle v|$ as the conjugate transpose of $|v\rangle$.

The *bra-ket* notation can also be useful to represent linear operators. Let $|v\rangle \in V$ and $|w\rangle \in W$, then $|w\rangle\langle v|$ is a linear operator from V to W given by $\big(|w\rangle\langle v|\big)|u\rangle = \langle v|u\rangle |w\rangle$ for any $|u\rangle \in V$. That is, it multiplies the vector $|w\rangle$ by the scalar $\langle v|u\rangle$. In fact, we have the following result.

Proposition A.4. Let $\big\{|v_i\rangle : 1 \leq i \leq n\big\}$ be an orthonormal basis of a vector space V, and let A be a linear operator such that $|w_i\rangle = A|v_i\rangle$ for $i \in \{1, \ldots, n\}$. Then, we can write A in the following way

$$A = \sum_{i=1}^{n} |w_i\rangle\langle v_i|.$$

Proof. Since a linear operator is uniquely determined by the image of the vectors of a base, we just need to check that both operators have the same value when applied to the vectors of $\big\{|v_i\rangle : 1 \leq i \leq n\big\}$.

Let $i \in \{1, \ldots, n\}$, we have that

$$\left(\sum_{k=1}^{n} |w_k\rangle \langle v_k| \right) |v_i\rangle = \sum_{k=1}^{n} |w_k\rangle \langle v_j|v_i\rangle = \sum_{i=1}^{n} \delta_{i,k} |w_k\rangle = |w_i\rangle ,$$

which completes the proof. ■

In particular, for any orthonormal basis $\{|v_i\rangle : 1 \leq i \leq n\}$ on a vector space V we have that

$$\sum_{i=1}^{n} |v_i\rangle \langle v_i| = I_V,$$

where I_V is the identity operator on V. The subscript may be removed when it's not necessary.

Definition A.5. Let A be a linear operator on a vector space V. We say that $|v\rangle \in V$ is an eigenvector of A if there exists $\lambda \in \mathbb{C}$ such that $A|v\rangle = \lambda |v\rangle$. In that case, we say that λ is an eigenvalue of A associated to the eigenvector $|v\rangle$.

Note that if the dimension of V is n, then A has n eigenvalues, because of the Fundamental Theorem of Algebra. Another important property is that the determinant of A is equal to the product of its eigenvalues.

Definition A.6. Given any linear operator A on a Hilbert space V, the adjoint of A is the only linear operator $A^\dagger$ on V such that for all $|u\rangle, |v\rangle \in V$

$$\left(|u\rangle, A|v\rangle\right) = \left(A^\dagger |u\rangle, |v\rangle\right).$$

It turns out that if A is the matrix representation of a linear operator, its adjoint $A^\dagger$ is given by the conjugate-transpose of A. In particular, for any vectors $|u\rangle$ and $|v\rangle$ we have that $|u\rangle^\dagger = \langle u|$ and $\left(|u\rangle \langle v|\right)^\dagger = |v\rangle \langle u|$.

Definition A.7. A matrix A is said to be self-adjoint if $A = A^\dagger$.

An important example of self-adjoint operators are projections.

Definition A.8. Let V be a Hilbert space and $\{|v_i\rangle : 1 \leq i \leq n\}$ an orthonormal basis on it. Let W be a subspace of V such that $\{|v_i\rangle : 1 \leq i \leq m\}$ is an orthonormal basis on W. Then, we say that the linear operator given by

$$P = \sum_{i=1}^{m} |u_i\rangle \langle u_i|$$

is a projection onto the subspace W. Note that $Q = I - P$ is another projection, but in this case it's a projection onto the orthogonal complement of W, because

$$Q = I - P = \sum_{i=m+1}^{n} |u_i\rangle \langle u_i| .$$

It can be shown (and it's not hard) that the definition doesn't depend on the choice of the basis, it just depends on the subspace W onto which we project. We'll skip that proof, however.

Proposition A.9. If P is a projection, then $P^2 = P$.

Proof. By definition, we can write

$$P = \sum_{i=1}^{m} |u_i\rangle \langle u_i|$$

for a given orthonormal basis $\{|u_i\rangle : 1 \leq i \leq n\}$. Thus,

$$\begin{aligned} P^2 &= \left(\sum_{i=1}^{m} |u_i\rangle \langle u_i| \right) \left(\sum_{k=1}^{m} |u_k\rangle \langle u_k| \right) \\ &= \sum_{i,j=1}^{m} \left(|u_i\rangle \langle u_i| \right) \left(|u_k\rangle \langle u_k| \right) \\ &= \sum_{i,j=1}^{m} |u_i\rangle \left(\langle u_i|u_k\rangle \right) \langle u_k| \\ &= \sum_{i,j=1}^{m} \delta_{i,k} |u_i\rangle \langle u_k| \\ &= \sum_{i=1}^{m} |u_i\rangle \langle u_i| = P. \end{aligned}$$

■

Definition A.10. A linear operator A is normal if $AA^\dagger = A^\dagger A$.

Note that every self-adjoint operator is normal. The converse is not always true, but we have the following characterization.

Proposition A.11. Let A be a normal operator. A is self-adjoint if and only if all its eigenvalues are real numbers.

Definition A.12. A linear operator U is unitary if $U^\dagger$ is its inverse.

An important observation is that unitary operators preserve inner products between vectors. That is, let $|u\rangle, |v\rangle \in V$ and U a unitary operator on V. Then,

$$\left(U|u\rangle, U|v\rangle \right) = \langle u| U^\dagger U |v\rangle = \langle u| I |v\rangle = \langle u|v\rangle = \left(|u\rangle, |v\rangle \right).$$

Proposition A.13. Unitary operators transform orthonormal basis into orthonormal basis. Moreover, if a linear operator transforms an orthonormal basis into another orthonormal basis, then it's unitary

Proof. Let U be a unitary operator and let $\{|v_i\rangle : 1 \leq i \leq n\}$ be an orthonormal basis. Since

$$\langle v_i| U^\dagger U |v_j\rangle = \langle v_i|v_j\rangle = \delta_{i,j},$$

we have that the set given by $\{L|v_i\rangle : 1 \leq i \leq n\}$ is an orthonormal basis.

Reciprocally, let $\{|v_i\rangle : 1 \leq i \leq n\}$ and $\{|w_i\rangle : 1 \leq i \leq n\}$ be two orthonormal basis, and let U be the linear operator given by

$$U = \sum_{i=1}^{n} |v_i\rangle \langle w_i| .$$

We have that

$$U^\dagger = \sum_{i=1}^{n} |w_i\rangle \langle v_i| ,$$

so

$$\begin{aligned} U^\dagger U &= \left(\sum_{i}^{n} |w_i\rangle \langle v_i| \right) \left(\sum_{j=1}^{n} |v_j\rangle\langle w_j| \right) \\ &= \sum_{i,j=1}^{n} |w_i\rangle \langle v_i|v_j\rangle\langle w_j| \\ &= \sum_{i,j=1}^{n} \delta_{i,j} |w_i\rangle \langle w_j| \\ &= \sum_{i=1}^{n} |w_i\rangle \langle w_i| = I. \end{aligned}$$

The proof that $UU^\dagger = I$ is analogous. ■

Proposition A.14. The eigenvalues of a unitary matrix U are unitary complex numbers.

Proof. Since U is unitary, it's also normal, so we can apply the spectral theorem and decompose U in the following way

$$U = \sum_{i=1}^{n} \lambda_i |v_i\rangle \langle v_i| ,$$

where λ_i is the eigenvalue associated to the eigenvector $|v_i\rangle$ of U, for $i \in \{1, \ldots, n\}$, such that the set $\{|v_i\rangle : 1 \leq i \leq n\}$ is an orthonormal basis. Then, for $i \in \{1, \ldots, n\}$ we have that

$$1 = \langle v_i|v_i\rangle = \langle v_i| U^\dagger U |v_i\rangle = \overline{\lambda_i}\lambda_i \langle v_i|v_i\rangle = |\lambda|^2.$$

■

Then, the determinant of U is also a unitary complex number.

Example A.15. Let φ be a Boolean function on n variables. Then, the linear transformation U_φ given by

$$U_\varphi \left(|x\rangle |y\rangle \right) = |x\rangle \left| y \oplus \varphi(x) \right\rangle$$

for $x \in \{0,1\}^n$ and $y \in \{0,1\}$ is unitary, where $\oplus$ denotes addition modulo 2. We can see that this unitary function gives us a way to quantumly evaluate a Boolean function, because

$$U_\varphi \left(|x\rangle |0\rangle \right) = |x\rangle \left| \varphi(x) \right\rangle .$$

Definition A.16. We will say that a linear operator A on V is positive-semidefinite if $\langle v| A |v\rangle \geq 0$ for any $|v\rangle \in V$. Additionally, we will say that it's positive if $\langle v| A |v\rangle > 0$ for any $|v\rangle \in V \setminus \{0_V\}$.

Positive-semidefinite operators are self-adjoint, and in particular, normal. So we can apply the spectral theorem to a positive-semidefinite operator A and decompose it as

$$A = \sum_{i=1}^{n} \lambda_i |v_i\rangle \langle v_i| ,$$

where λ_i is the eigenvalue associated to the eigenvector $|v_i\rangle$ of U, such that the set $\{|v_i\rangle : 1 \leq i \leq n\}$ is an orthonormal basis. Then, we have that

$$0 \leq \langle v_i| A |v_i\rangle = \lambda_i \langle v_i|v_i\rangle = \lambda_i$$

for $i \in \{1, \ldots, n\}$, and we can consider the linear operator B given by

$$B = \sum_{i=1}^{n} \sqrt{\lambda_i} |v_i\rangle \langle v_i| .$$

Note that B is self-adjoint and that $B^2 = A$, which means that we can define the square root of a positive-semidefinite operator.

Proposition A.17. Let A be a lineal operator. Then, $A^\dagger A$ is positive-semidefinite.

Proof. Simply see that $\langle v| A^\dagger A |v\rangle \geq 0$ because $\big(A |v\rangle\big)^\dagger = \langle v| A^\dagger$. ■

Finally, we will be talking about the tensor product and some of its properties. As we saw in Postulate 2, it gives us a way to combine state spaces to represent composite systems.

Let V and W be two complex vector spaces of dimension n and m, respectively. Then, $V \otimes W$ is a complex vector space of dimension $n \cdot m$ whose elements are linear combinations of tensor products of the form $|v\rangle \otimes |w\rangle$ for $|v\rangle \in V$ and $|w\rangle \in W$. Moreover, if $\{|v_i\rangle : 1 \leq i \leq n\}$ and $\{|w_i\rangle : 1 \leq i \leq m\}$ are basis of V and W respectively, then the set $\{|v_i\rangle \otimes |w_k\rangle : 1 \leq i \leq n,\ 1 \leq k \leq m\}$ is a basis on $V \otimes W$.

We may denote the tensor product of two vectors $|v\rangle \otimes |w\rangle$ as $|v\rangle |w\rangle$, $|v, w\rangle$ or even $|vw\rangle$ depending on the context.

Let A and B be two matrices of size $m \times n$ and $p \times q$, respectively. Then, the tensor product $A \otimes B$ is a matrix of size $mp \times nq$ given by

$$A \otimes B = \begin{pmatrix} A_{1,1}B & \cdots & A_{1,n}B \\ \vdots & \ddots & \vdots \\ A_{m,1}B & \cdots & A_{m,n}B \end{pmatrix},$$

where $A = \big(A_{i,j}\big)_{i,j}$ and $A_{i,j}B$ is the $p \times q$ result of multiplying B by the scalar $A_{i,j}$.

The tensor product has the following properties.

1. For any scalar $z \in \mathbb{C}$ and any vectors $|v\rangle \in V$, $|w\rangle \in W$ we have that

$$z\big(|v\rangle \otimes |w\rangle\big) = \big(z |v\rangle\big) \otimes |w\rangle = |v\rangle \otimes \big(z |w\rangle\big) .$$

2. For any vectors $|v\rangle \in V$, $|w_1\rangle, |w_2\rangle \in W$ we have

$$|v\rangle \otimes \big(|w_1\rangle + |w_2\rangle\big) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle .$$

3. For any vectors $|v_1\rangle, |v_2\rangle \in V$, $|w\rangle \in W$ we have

$$\big(|v_1\rangle + |v_2\rangle\big) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle .$$

Let A and B be two linear operators on V and W, respectively. Then, we can define a linear operator $A \otimes B$ in $V \otimes W$ given by

$$(A \otimes B)\big(|v\rangle \otimes |w\rangle\big) = A|v\rangle \otimes B|w\rangle ,$$

such that it's extended to all $V \otimes W$ by linearity. That is,

$$(A \otimes B)\left(\sum_{i=1}^{k} \alpha_i |v_i\rangle \otimes |w_i\rangle\right) = \sum_{i=1}^{k} \alpha_i A|v_i\rangle \otimes B|w_i\rangle .$$

We can also define an inner product in $V \otimes W$ based on the inner products defined in V and W as follows. The inner product of

$$\sum_{i=1}^{n_1} \alpha_i |v_i\rangle \otimes |w_i\rangle \quad \text{and} \quad \sum_{j=1}^{n_2} \beta_i |\tilde{v}_j\rangle \otimes |\tilde{w}_j\rangle$$

is

$$\left(\sum_{i=1}^{n_1} \alpha_i |v_i\rangle \otimes |w_i\rangle , \sum_{j=1}^{n_2} \beta_i |\tilde{v}_j\rangle \otimes |\tilde{w}_j\rangle\right) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \overline{\alpha_i}\beta_j \langle v_i|\tilde{v_j}\rangle\langle w_i|\tilde{w}_j\rangle.$$

Proposition A.18. If A and B are unitary, self-adjoint, positive-semidefinite or projections, then so is $A \otimes B$.

Bibliography

[1] Aaronson, S. Reasons to believe. https://www.scottaaronson.com/blog/?p=122.

[2] Aaronson, S. Guest Column: NP-complete problems and physical reality. *ACM SIGACT News 36*, 1 (Mar. 2005), 30–52.

[3] Aaronson, S. P =? NP. In *Open Problems in Mathematics*, J. Nash, John Forbes and M. T. Rassias, Eds. Springer International Publishing, 2016, pp. 1–122.

[4] Abrams, D. S., and Lloyd, S. Nonlinear Quantum Mechanics Implies Polynomial-Time Solution for NP-Complete and #P Problems. *Physical Review Letters 81*, 18 (Nov. 1998), 3992–3995.

[5] Arora, S., and Barak, B. *Computational Complexity: A Modern Approach*, 1st ed. Cambridge University Press, USA, 2009.

[6] Baker, T., Gill, J., and Solovay, R. Relativizations of the P =? NP Question. *SIAM Journal on Computing 4*, 4 (Dec. 1975), 431–442.

[7] Bekenstein, J. D. Universal upper bound on the entropy-to-energy ratio for bounded systems. *Physical Review D 23*, 2 (Jan. 1981), 287–298.

[8] Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing 26*, 5 (Oct. 1997), 1510–1523.

[9] Berger, B., and Leighton, T. Protein folding in the hydrophobic-hydrophilic (*HP*) is NP-complete. In *Proceedings of the second annual international conference on Computational molecular biology* (New York, NY, USA, Mar. 1998), RECOMB '98, Association for Computing Machinery, pp. 30–39.

[10] Bringsjord, S., and Taylor, J. P=NP. *arXiv:cs/0406056* (June 2004). arXiv: cs/0406056.

[11] Buss, S. R. On Gödel's Theorems on Lengths of Proofs II: Lower Bounds for Recognizing k Symbol Provability. In *Feasible Mathematics II* (Boston, MA, 1995), P. Clote and J. B. Remmel, Eds., Progress in Computer Science and Applied Logic, Birkhäuser, pp. 57–90.

[12] Carrillo Redondo, V. 2-dimensional cellular automaton (instantaneous communication assumption), Sept. 2021. https://www.youtube.com/watch?v=8IIsXnJl7hM.

[13] Cook, M. Universality in elementary cellular automata. *Complex Systems 15* (2004).

[14] Cook, S. The P versus NP problem. In *Clay Mathematical Institute; The Millennium Prize Problem* (2000).

[15] Cook, S. The importance of the P versus NP question. *Journal of the ACM 50*, 1 (Jan. 2003), 27–29.

[16] Cook, S. A. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing* (New York, NY, USA, May 1971), STOC '71, Association for Computing Machinery, pp. 151–158.

[17] Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., and Yannakakis, M. On the Complexity of Protein Folding. *Journal of Computational Biology 5*, 3 (Jan. 1998), 423–465.

[18] Hartmanis, J. Godel, von Neumann and the P=?NP Problem.

[19] Johnson, D. J., and Trick, M. A. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, USA, 1996.

[20] Johnson, D. S. A brief history of NP-completeness, 1954–2012. *Documenta Mathematica* (2012).

[21] Margenstern, M., and Morita, K. NP problems are tractable in the space of cellular automata in the hyperbolic plane. *Theoretical Computer Science 259*, 1 (May 2001), 99–128.

[22] Neary, T., and Woods, D. P-completeness of Cellular Automaton Rule 110. In *Automata, Languages and Programming* (Berlin, Heidelberg, 2006), Lecture Notes in Computer Science, Springer, pp. 132–143.

[23] Nielsen, M. A., and Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Cambridge, 2010.

[24] Papadimitriou, C. H. *Computational Complexity*. Addison-Wesley, 1994.

[25] Razborov, A. A., and Rudich, S. Natural Proofs. *Journal of Computer and System Sciences 55*, 1 (Aug. 1997), 24–35.

[26] Rieffel, E., and Polak, W. An introduction to quantum computing for non-physicists. *ACM Computing Surveys 32*, 3 (Sept. 2000), 300–335.

[27] Saito, K., Aono, M., and Kasai, S. Amoeba-inspired analog electronic computing system integrating resistance crossbar for solving the travelling salesman problem. *Scientific Reports 10*, 1 (Nov. 2020), 20772.

[28] Schönhage, A. On the power of random access machines. In *Automata, Languages and Programming* (Berlin, Heidelberg, 1979), H. A. Maurer, Ed., Lecture Notes in Computer Science, Springer, pp. 520–529.

[29] Weinberg, S. Testing quantum mechanics. *Annals of Physics 194*, 2 (Sept. 1989), 336–386.

[30] Zhu, L., Kim, S.-J., Hara, M., and Aono, M. Remarkable problem-solving ability of unicellular amoeboid organism and its mechanism. *Royal Society Open Science 5*, 12 (Dec. 2018), 180396.