Bachelor Thesis

NP-hardness of Invariants in Rational Homotopy Theory

Eric Ritte

Datum der Abgabe
10. 11. 2021

Advisors:

Dr. Andreas Ott
Department of Mathematics

Prof. Dr. Torsten Ueckerdt
Department of Computer Science

Karlsruhe Institute of Technology

Contents

1 Introduction

In algebraic topology, one constructs algebraic invariants of topological spaces in order to classify them. This is done by constructing one's invariants that they are exactly that – invariant under homeomorphism. Then, two spaces with different invariants may not be homeomorphic, the kind of result that is often very tricky to prove otherwise. One of the most powerful invariants is the set of homotopy groups of a space. However, this power comes at a price: the higher homotopy groups of even simple spaces are notoriously hard to compute, and much research has been done into both developing techniques to compute them and develop alternative invariants that trade power for ease of computation. One such invariant can be found in rational homotopy theory, which allows us to describe a certain subclass of topological spaces with the help of an alluringly easy to handle algebraic object. It does this by "forgetting" about unwieldy twists and torsion maps that can occur in regular homotopy theory. The apparent ease of computation it offers raises a new question: does rational homotopy theory actually make computations "easy"? Luckily, complexity theory offers us a powerful theoretical framework to argue about the difficulty of computational problems. We will examine a handful of invariants one generally cares about in algebraic topology and come to a conclusion that may be underwhelming to some (and exciting to others): even though rational homotopy theory makes computation "easier" in an everyday understanding of the word, computing interesting invariants is still "hard" in a formal sense.

2 Complexity Theory

In this chapter, we will give a brief overview over the concepts required from theoretical computer science, specifically computation and complexity theory, to understand the results outlined in this thesis. Roughly speaking, computation theory asks the question “What can we compute?”, whereas complexity theory tries to further classify that which is computable by asking “How efficiently can we compute it?” The object of study of complexity theory thus are computation problems (for example finding the shortest path between two vertices in a weighted directed graph) and routines or procedures[1] that can solve such problems in either the general case or under a specified set of restrictions. The goal then becomes to explicitly construct such procedures and to prove upper and lower bounds of their running time (or other resource use) in dependence on the size of the problem instance to solve (for example the number of edges and vertices in a graph).

Our goal in this chapter is to give a precise, mathematical rigorous meaning to the statements above. Slightly more material than what would be strictly needed to understand the results presented in sections 4 to 5 will be covered, with the goal to better contextualize that which is.

2.1 A Small Vocabulary for Computation

To start out, we introduce some basic terminology from theoretical computer science, which we will need for the definitions in the following sections. A lot of the nomenclature in this section may seem strange to the modern reader, but it is historically motivated and bears witness to a certain influence theoretical linguistics had on early theoretical computer science.

Definition 1 (Alphabet) A finite set of symbols Σ is referred to as an *alphabet*. The term is meant to emphasize that we do not assume a structure on

[1]We are avoiding using the term “algorithm” for now, as we do not yet have developed the necessary theoretical machinery to give it a precise technical meaning.

Σ and view the elements bereft of any (initial) interpretation or context, even if they might already be highly familiar to us. □

Definition 2 (String) Let Σ be an alphabet. For a finite $n \in \mathbb{N}$, an ordered n-tuple ω of symbols in Σ (duplicates allowed) is called a *string* (over Σ). The natural number n is referred to as the *length* of the string, which we denote with $|\omega|$. The set of all strings of length n over Σ is denoted by Σ^n. There is exactly one string of length 0, the *empty string* ϵ. □

Definition 3 (The Set of all Strings) For an alphabet Σ, we define Σ^* to be the set of all finite-length strings over Σ, i.e. $\Sigma^* = \cup_{n\in\mathbb{N}}\Sigma^n$. □

Definition 4 (Language) Given an alphabet Σ, a *language* L (over Σ) is a subset of Σ^*. The elements of L are sometimes called *words*. The complement of a language is the language $L^{\complement} := \{0,1\}^* \setminus L$. □

Example 1 (Binary Representations of Prime Numbers) Let Σ be the set $\{0,1\}$. Then Σ^* is the set of all finite-length bit strings. We can define the language

$$L_{\mathsf{prime}} := \{\omega \in \Sigma^* \mid \omega \text{ is the binary representation of a prime number}\}$$

of all (binary) prime numbers. □

It is generally easier to define a language then to give an explicit description of what it looks like, i.e. a list of all its members, if just for the reason of sheer size. The example L_{prime} was easy to define, but one could not write down a list of all binary representations of prime numbers, as there are an infinite amount of those. More interestingly, given a specific bit-string, e.g. 100011101001101, it is not obvious whether or not this the binary representation of a prime. This motivates the definition of the decision problem.

Definition 5 (Decision Problem) A language L is associated to the *decision problem* of which strings $\omega \in \Sigma^*$ are also elements of L. When trying to decide whether a given $\omega \in \Sigma^*$ is a member of L, we refer to ω as an *instance* of the decision problem.

Define $\chi_L \colon \Sigma^* \to \{0,1\}$ to be the characteristic function of L, that is $\chi_L(\omega) = 1$ if and only if $\omega \in L$. Then the decision problem of L is equivalent to calculating the characteristic function. □

Example 2 (Primes again) Consider the language L_{prime} from example 2.1. For a given bit string $\omega \in \Sigma^*$, evaluating the characteristic function χ_{prime} of L_{prime} at ω means checking whether the number represented by ω is a prime number. More generally, calculating χ_{prime} means constructing an automatable procedure that can check for any bit string whether it represents a prime number. We will return to these notions in the next sections to make them more rigorous. This decision problem is usually referred to as PRIMES. □

2.2 A Model of Computation

Before we start talking about what it means for a computation problem to be "efficiently computable", we first need to clarify what it means for something to be computable in the first place. To do so, we introduce a basic model of computation, the Turing machine. We first give an informal description of how a Turing machine works and what its internals look like, before moving on to a more formal definition. However, Turing machines have a very nice theoretical property that allows us to argue about them on a very high level of abstraction, so the technical details of the definition are often regarded as not as important as a good understanding of how they work.

A Turing machine consists out of a control unit, a read-write head and an infinite-length tape to write on and read from, divided into an infinite amount of uniformly sized cells. Note that the tape is only of infinite length to free us from concerns about space constraints – the machine will never run out of tape space during execution –, but as a result of how a Turing machine works (see further below), it does not allow us to actually surpass the capabilities of an actual computer (which obviously does not have access to an unlimited amount of memory), so at any given point during execution, only a finite amount of cells can actually have been written to/read from.

The control unit consists out of a finite amount of states and possible transitions between them. This can be thought of as a directed graph that permits loops and multi-edges. One state is the designated *starting state*, and two or more designated *halting states*, which are either *accepting* or *rejecting*.

At the start of execution, the tape contains the *input*, a finite string of symbols (for example the binary representation of an integer), one symbol per cell. The read-write head points to the start of the input, the control unit is in the starting state. The start of the input, however, is not the start as the tape, but "somewhere in the middle", that is to say that at the beginning, there is an infinite amount of tape space to both the left and the right of the input. It might be helpful to view the tape cells as indexed by the integers – there is no first cell, same as there is no last cell –, and the input string just starts at some integer.

The execution itself consists of discrete steps. At the beginning of each step, the read-write head reads the content of the cell it is currently pointing at. This then triggers a state transition, even though transitions into the state the machine is already in are allowed (that is to say the machine does not necessarily change state during a transition). During the transition, the machine may write a symbol to the cell the read-write head is currently pointing at, replacing any symbol already inside that cell, and/or move the read-write head one cell to the left or right. If the machine both writes a symbol to the tape and moves the head, the writing is always done first, the moving second. If the transition includes a change of state, this is done last.

If, at any point during execution, the machine transitions into one of the designated halting states, it immediately stops executing. We say that the input was either *accepted* or *rejected*, depending on whether the machine stopped in an accepting or rejecting halting state. Any string of symbols still left on the tape is referred to as the *output* of the execution. However, it is also possible for the machine to never halt, for example when, during execution, the machine enters an ever-repeating circular sequence of transitions into non-halting states. We then say that the machine *loops*.

Definition 6 (Turing Machine) A *(deterministic) Turing Machine* TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{accept}}, q_{\mathsf{reject}})$, consisting out of

- A finite set of states Q
- A finite set of symbols Σ, the *input alphabet*
- A finite set of symbols Γ, the *tape alphabet*, containing Σ, the blank symbol ␣ and any other symbol the machine may write to the tape
- A transition function $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$
- A start-state $q_0 \in Q$
- An accept-state $q_{\mathsf{accept}} \in Q$
- A reject-state $q_{\mathsf{reject}} \in Q$

The input alphabet may not contain the blank symbol. The transition function takes the current state and the symbol written in the cell the read-write head is currently pointing to and maps it to a triple of the new state of the machine (can be the one the machine is already in), the symbol that is to be written to the tape cell (can be the one read from the tape) and the direction to move the read-write head to (left, right or no movement). □

Note that in the literature one can find several slightly different, but ultimately equivalent, definitions. Some may include multiple single-purpose tapes (for example one for input, one for output and one for calculations), each one with its own dedicated read, write or read-write head. Some define the tape cells as indexed by the natural numbers, that is there is a unique starting cell. Some may include a "square" tape, that is one where the cells are indexed by $\mathbb{Z} \times \mathbb{Z}$, and where the read-write head can move up and down as well as left and right. Some may include a fixed input-alphabet. Some may not include rejecting final states, only accepting or even only halting ones.

However, these technical details do not actually matter a great deal: it is a central notion of theoretical computer science that not only is every model

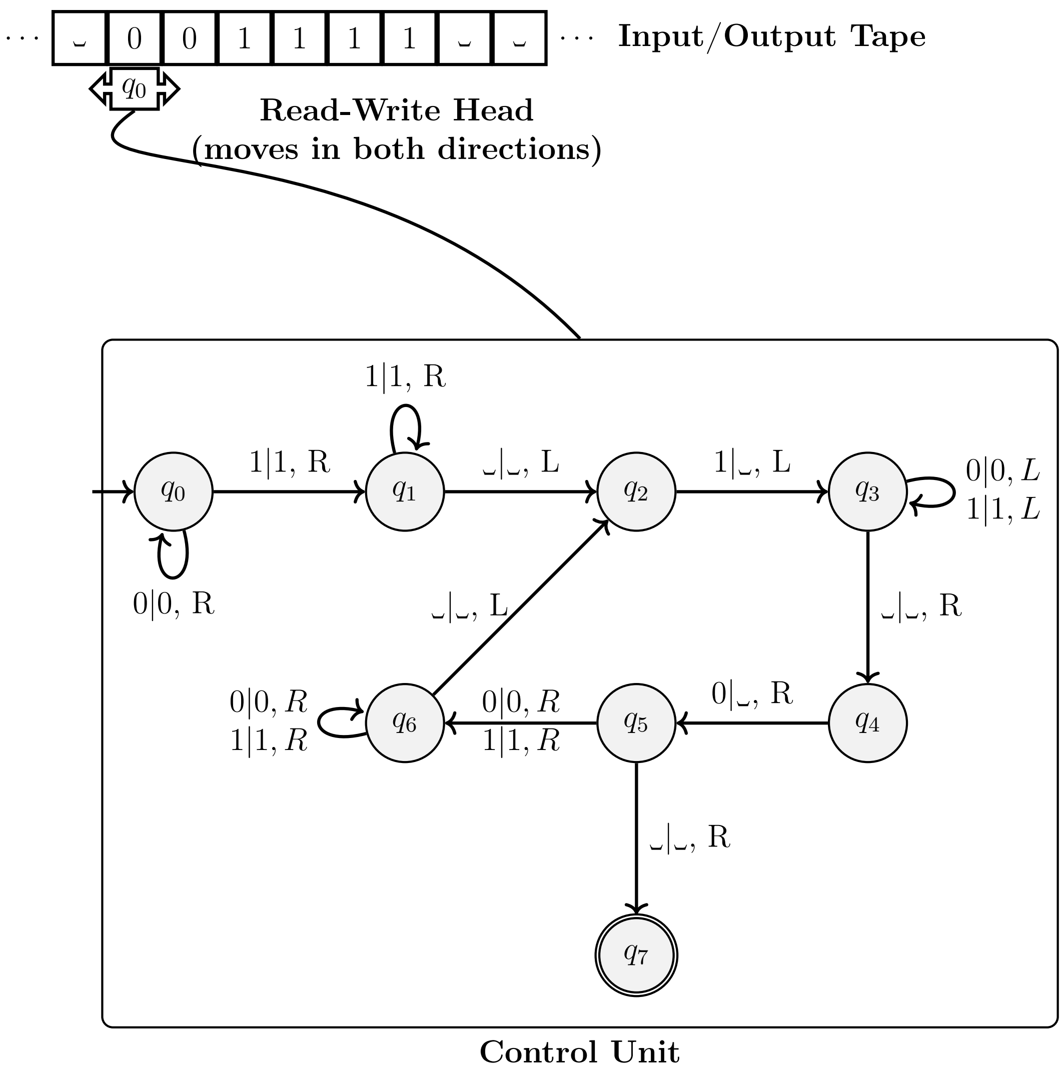

Figure 1: An illustration of a Turing Machine. The transitions should be read as "*input*|*output*, *head movement*". The state diagram of the control unit was lifted from [Wag+ 1] and recognizes the language $L^{=} := \{0^n 1^n \mid n \geq 1\}$. There are a couple of transitions not drawn, all of which lead into the (also not drawn) rejecting state.

of classical computation as powerful as any other one, any model can also simulate any other model efficiently. This is (a slight restriction of) the so-called *extended Church-Turing thesis*. In order to properly state the extended Church-Turing thesis, we first need to specify what we mean when we say "efficiently".

Definition 7 (Polynomial Execution Time) A Turing machine TM halts (or terminates) in *polynomial time* if there is a polynomial p such that TM halts on any input string ω after at most $p(|\omega|)$ steps. □

A bit more reasoning on why it has become custom to interpret computation in polynomial time as efficient will be given once we have defined some basic complexity classes in section 2.3.

Claim 2.1 (Extended Church-Turing Thesis)
Every physically realizable model of computation can be simulated by a Turing machine in polynomial time, that is there is a Turing machine TM a polynomial p such that in order to simulate t execution steps of the model, TM will need at most $p(t)$ steps.

This is the extended (or strong) Church-Turing thesis as stated in [AB09]. One will also often find it stated as "every intuitively realizable model of computation can be efficiently simulated by a Turing machine". While there exists no general proof of the Church-Turing thesis, and its somewhat non-rigorous statement makes it insusceptible to mathematical proof, once one considers a specific model of computation, this equivalence of power can be proven. Note that there are models of computation that are not just differing definitions of Turing machines (for example, every programming language is also a model of computation), which are, of course, also covered by the Church-Turing thesis. See for example section 1.2.1 of [AB09] for an outline of a proof that Turing machines can simulate a general programming language.

Only the advent of quantum computing and the possibility that of a physically realizable universal quantum computer has brought forward a likely candidate for a model of computation that is strictly more powerful than Turing machines. At this point in time, however, this has not been formally proven, and it is still an open research question as to whether quantum computers are more powerful than Turing machines. For more information, see Scott Aaronson's survey [Aar17] [Aar17], specifically section 5.5, or chapter 10 in Arora's book. Still, even with the advent of quantum computing, the extended Church-Turing thesis is still widely held to be true if one only considers models of classical computation, that is models of the kind of deterministic digital computation one finds in modern-day regular computers.

To close this chapter, we will return to the notion of "algorithm" and how Turing machines actually offer a framework to formalize it. In the following, for simplicity's sake, we will only consider Turing machines with a single halting state which is neither accepting nor rejecting. This does, of course, in no way influence the computational power of the definition.[2]

Note that the entire "computational information" of a Turing machine is encoded in its transition function. Like the integrated circuits one would find in an old pocket calculator, they are very much single-purpose – a Turing machine can perform exactly the one kind of calculation described by its transition function. But this is not how a modern computer works. Given that we want Turing machines to offer a theoretical model that allows us to argue about the computational capacities of a computer, the question as to whether we can construct a Turing machine that can calculate anything offers itself up. The answer is yes.[3] The reason for this is that, due to its finite size, the transition function offers itself up to be written as a single string, which can then be given to another Turing machine as input. Then, we only need to

[2]For a Turing machine TM with an accepting and rejecting state, we can construct an equivalent Turing machine TM' with a singular halting state by turning the accepting and rejecting state into non-halting ones, adding a new halting state q_{end} and a set of transitions such that on reaching the old accepting state, the Turing machine TM' empties the entire tape except for a special accepting symbol, e.g. a "1", in a single cell (and, accordingly, new transitions from the old rejecting state that empty the tape except for a single rejecting symbol) and then halts. We can then say that TM' accepts an input ω if it halts with output 1 (or rejects it if it halts with output 0).

[3]While this may not be surprising in the day and age of ubiquitous universal computing, it was considered to be a most remarkable result when it was first proven by Turing in 1936.

construct a Turing machine whose "one kind of computation" it can perform is other Turing machines. More precisely, a Turing machine that, on an input consisting of a string encoding another Turing machine and some input string ω, will simulate the encoded Turing machine on the input ω efficiently. This then uniquely identifies a Turing machine with what can be thought of as an executable bit string consisting of a sequence of instructions describing how to perform a certain kind of calculation. Or in simpler terms: a program.

Definition 8 (Turing Machines as Programs) We now show how any Turing machine can be encoded as a bit string, that is an object $\alpha \in \{0,1\}^*$. While the choice of alphabet is somewhat arbitrary, it has established itself as a standard in complexity theory, and some textbooks, such as [AB09], do not even define languages and Turing machines over any other alphabet.

Let TM be a Turing machine consisting of

- the set of states $Q = \{q_0, \ldots, q_n\}$,
- the input alphabet $\Sigma = \{a_1, \ldots, a_m\}$,
- the tape alphabet $\Gamma = \{a_1, \ldots, a_m, a_{m+1}, \ldots, a_k =\}$,
- the transition function $\delta\colon Q \times \Sigma \to Q \times \Gamma \times \{L, R, N\}$,
- the starting state $q_{\mathsf{start}} \in Q$,
- a single halting state $q_{\mathsf{halt}} \in Q$.

In order to encode δ as a string, we note that we can view it as a subset $Q^2 \times \Sigma \times \Gamma \times D \supset \Delta := \{(q_i, q_j, a_s, a_t, d_\ell) \mid \delta(q_i, a_s) = (q_j, a_t, d_\ell)\}$, where $D = \{d_1, d_2, d_3\}$ with $d_1 = L$, $d_2 = R$ and $d_3 = N$. We introduce an ordering $\prec$ on Δ with

$$(q_i, q_j, a_s, a_t, d_\ell) \prec (q_r, q_k, a_{s'}, a_{t'}, d_{\ell'}) :\Longleftrightarrow i \leq r \text{ and } j \leq k.$$

Note that this is a total order and since Δ is finite, it has a minimal element. We can therefore use it to enumerate the elements in Δ. Should two elements d_1, d_2 be two transitions from the same state into the same state, we order them via the index of the input symbol. We encode an element $(q_i, q_j, a_s, a_t, d_\ell) \in \Delta$ as the bit string $0^i 1 0^s 1 0^j 1 0^t 1 0^\ell$ and write $code_k$ for it, where k is the index of the element in the enumeration given by $\prec$. This allows us to encode Δ by concatenating all the bit strings in the order defined by $\prec$ with sequences of 1's as delimiters, resulting in a string

$$111\mathit{code}_1 11\mathit{code}_2 11\ldots 11\mathit{code}_{|\Delta|}111.$$

Furthermore, by convention, it is often assumed that $q_0 = q_{\mathsf{start}}$ and $q_1 = q_{\mathsf{halt}}$ (since we can always just re-index the states), so that these get encoded at the start. Note that this, like the enumerating we defined above, is not a necessary convention – even with an arbitrary enumeration of the elements in Δ, the construction would still work. It does, however, make the following proofs a bit less messy.

We can now describe any Turing machine as an element $\alpha \in \{0,1\}^*$. We now introduce two further useful conventions. The first is that any string that is not a well-formatted encoding of a Turing machine is an encoding of the Turing machine that immediately halts with output 0. The second one is that if α is an encoding of a Turing machine, $\alpha 1^*$ is an encoding of the same Turing machine. This is to say we can add any amount of trailing ones to an encoding of a Turing machine without changing its computational content. Finally, we write TM_α for the Turing machine encoded by the string α. The two previous conventions guarantee that this is always a meaningful piece of notation, no matter the string α. □

Theorem 1 (The Universal Turing Machine) *There exists a* universal Turing machine TM_U *with input alphabet* $\{0,1\}$ *and tape alphabet* $\{\llcorner\!\lrcorner, 0, 1\}$ *such that for every* $\omega, \alpha \in \{0,1\}^*$*, on the input* (α, ω)*,* TM_U *will simulate* TM_α *on the input* ω*. Moreover, this is an efficient simulation. If* TM_α *halts on the input* ω *within* t *steps, then* TM_U *halts on* (ω, α) *within* $c \cdot t \cdot \log t$ *steps, where* c *depends not on* ω *but only on the size of* TM_α*.* □

Proof For a detailed proof, see section 1.7 in [AB09], we will only give an outline here. For this, we will assume that the Turing machine encoded in the input also uses the binary alphabet, which we will justify afterwards. The rough idea of how the universal Turing machine works is as follows. It reads the description and input and then saves the state-transition function of the Turing machine to be simulated as a look-up table on its tape. Furthermore, it saves the current state and head position of the simulated Turing machine as well. Then it just starts working on input: it reads the first symbol, checks the current state, looks up the transition from the current state on the input symbol, performs it, updates head position, writes new symbol, updates the current state of the simulated Turing machine etc. ■

Lemma 1 *For any Turing machine* TM*, one can construct an equivalent Turing machine* $\tilde{\mathsf{TM}}$ *that uses a binary input alphabet and carries the same computational info as* TM*.* □

PROOF We only give a quick sketch of the proof, by outlining how such a Turing machine $\tilde{\mathsf{TM}}$ can be constructed. The idea is to replace each input symbol with a $2^{\lceil \log k \rceil}$ bit string, where k is the length of the input alphabet of TM. We then extend the amount of states and the state transition function such that $\tilde{\mathsf{TM}}$ can "remember" the bits of the string encoding the input symbol which it has already scanned and perform the state transition function of TM "bit-wise". This construction yields a Turing machine performing a computation equivalent to that of TM and using the binary alphabet. ■

2.3 Two Important Complexity Classes

Having laid the groundwork of developing a model of computation and formalizing what it means for a problem to be solvable by an algorithm, we now properly enter the subject area of complexity theory. Complexity theory is interested in classifying problems by the efficiency properties of the algorithms that solve them, or, spoken more casually, by how hard they are. This desire naturally gives rise to the concept of complexity classes, in which we collect problems of similar difficulty. While we will only consider classifications of runtime efficiency measured in required steps of computation, other kinds of complexity, such as space complexity, also exist.

Definition 9 (Computable Function) A function $f : \Sigma^* \to \Gamma^*$ is called *computable* if there exists a Turing machine TM_f that halts on any input $\omega \in \Sigma^*$ with output $f(\omega)$.

If TM_f is polynomial time, then we say f is *computable in polynomial time*. $\square$

Definition 10 (Decidable Languages) A language L is called *decidable* if its characteristic function is computable. We then say that the Turing machine computing the characteristic function *decides* the language (or the associated decision problem). $\square$

Definition 11 (Semi-decidable Language) A language L is called *semi-decidable* if there is a Turing machine TM_L that accepts any $\omega \in L$, but which may loop indefinitely on an $\omega \notin L$. We then sometimes say that TM_L *verifies* (or *recognizes*) L. $\square$

Definition 12 (The Class P of Problems) The class P is the class of languages (or decision problems) L for which there exists a Turing machine TM_L and a polynomial p such that for each $\omega \in \Sigma^*$, TM_L decides whether $\omega \in L$ in at most $p(|\omega|)$ steps. In other words, the Turing machine TM_L decides L in polynomial time. $\square$

Example 3 (PRIMES is in P) We return to example 2.1 one last time. By our previous note on encoding, we can view this as just the language of all prime numbers or equivalently the decision problem of whether a given natural numbers is prime. As proven in [AKS04], this problem is in P. $\square$

As noted before, computation in polynomial time is often viewed as "efficient computation" in practice. Therefore, the problems in P are often described as the problems which we can solve efficiently in practice. Of course, a polynomial of degree 1000 is still a polynomial, and if one had a problem that was decided by a Turing machine whose runtime was a polynomial of such high degree, its large instances would take an infeasible amount of time to compute in practice. One might therefore justifiably question the identification of “computable in polynomial time” with “efficiently computable”. Thus, it is therefore a most remarkable fact that for almost all problems in P, especially the ones of practical concern, there exist algorithms that run in at most cubic time [Aar17], making this identification a lot more reasonable. We will now introduce another important class of problems, for which it is not known whether or not they can be efficiently solved.

Definition 13 (The Class NP of Problems) The class NP is the class languages (or decision problems) L for which there exists a Turing machine TM_L and a polynomial p such that for all $\omega \in \Sigma^*$, $\omega \in L$ if and only if there exists a *witness* $\mathsf{w} \in \Sigma^{p(|\omega|)}$ (sometimes also referred to as a *certificate*) such that TM_L accepts (w, ω), the concatenation of w and ω separated by a delimiter symbol, in polynomial time depending *only* on $|\omega|$. We then say that TM_L *verifies* L.$\square$

For instances of problems in P, witness strings always exist. Indeed, any string is a witness string of any instance of any problem in P. This is since there exists a Turing machine which decides the problem, so we can construct one that verifies the problem by constructing a Turing machine that first discards the witness and then just decides the input. In other words, for a language $L \in \mathsf{P}$ the empty string ϵ is a witness for any $\omega \in L$ (and any $\alpha \in L^{\complement}$). Thus, P is in NP.

It is often helpful to think of the witness as a suggested solution to the problem instance, and this is how candidates for witness strings are usually chosen in practice. The definition above can therefore be understood as the problems in NP being the problems for which a solution can be verified in polynomial time. It is an open problem in complexity theory whether the inclusion $\mathsf{P} \subset \mathsf{NP}$ is a proper one, i.e. whether there are problems for which there exists no polynomial-time algorithm. However, this is widely assumed to be true (for example in all of modern cryptography). A thorough survey of the current state of research on that question can be found in [Aar17].

In complexity theory, we are interested in classifying problems by their difficulty. One way of doing this is to compare a problem’s difficulty to that of a problem where we already know "how hard" it is. The primary theoretical tool to do this is the proof technique of a Turing reduction. We will, however, not discuss Turing reductions in the entirety of their formal depth and instead focus on a slightly weaker variant usually referred to as a *many-one reduction*, as it is sufficient for the purposes of this thesis.

Definition 14 (Many-One Reduction) Let $\textsc{Pbm}, \textsc{Qst}$ be two languages over the same alphabet Σ. A *many-one reduction* from $\textsc{Pbm}$ to $\textsc{Qst}$ is a computable function $\mathsf{M}\colon \Sigma^* \to \Sigma^*$ such that $\omega \in \textsc{Pbm} \iff \mathsf{M}(\omega) \in \textsc{Qst}$. We then say that $\textsc{Pbm}$ is reducible to $\textsc{Qst}$ or that $\textsc{Qst}$ is "at least as hard as" $\textsc{Pbm}$.

A reduction M is called *polynomial* if there exists a Turing machine TM that computes M in polynomial time. □

Definition 15 (NP-Hardness) A problem $\textsc{Pbm}$ is called NP-hard if it is at least as hard as any other problem in NP, that is if for any other problem $\mathrm{T} \in \mathsf{NP}$, there exists a Turing reduction from T to $\textsc{Pbm}$. □

Note that to show that a problem is NP-hard, it suffices to construct a reduction to another NP-hard problem, as polynomial reductions are closed under composition.

Definition 16 (NP-Completeness) A problem $\textsc{Pbm}$ is called NP-complete if it is NP-hard and $\textsc{Pbm} \in \mathsf{NP}$. □

A lot of NP-hard problems that are considered to be of practical importance are also known to be NP-complete. We will conclude this chapter by giving just three examples, the latter two of which we will use in the proofs of sections 4 to 5.

Problem 2.1 ($k\textsc{Sat}$)
Instance: *A boolean formula in conjunctive normal form with k literals per clause,*

$$(x_{1_1} \vee x_{1_2} \vee \dots \vee x_{1_k}) \wedge (x_{2_1} \vee x_{2_2} \vee \dots \vee x_{2_k}) \wedge \dots \wedge (x_{j_1} \vee x_{j_2} \vee \dots \vee x_{j_k})\,.$$

Question: *Is there a configuration – that is a mapping that assigns each variable a boolean truth value* TRUE *or* FALSE *–, such that the formula is satisfied?*

This problem was first proved to be NP-complete by S.A. Cook in his landmark paper [Coo71]. It was in this paper that the notion of NP-completeness was first defined. One year later, R.M. Karp proved the NP-completeness of a further 21 important problems using reductions to the boolean satisfiability problem in [Kar72], including the following two examples. For these reasons, boolean satisfiability is often considered to be the "grandfather" of NP-complete problems.

Problem 2.2 (kCol)
Instance: *A graph $G := (V, E)$.*
Question: *Does G allow a proper k-coloring, that is exists there a function $\kappa\colon V \to \{1, \dots, k\}$ such that $\kappa(v_i) \neq \kappa(v_j)$ for all $(v_i, v_j) \in E$?*

Problem 2.3 (SubsetSum)
Instance: *A pair consisting out of a multiset M of integers and a target integer t.*
Question: *Does there exist a subset S of M such that the elements of S add up to t?*

3 Rational Homotopy Theory

3.1 Basic Homotopy Theory

Before we start introducing the ideas of rational homotopy theory, we want to give a rundown of some definitions and theorems from homotopy theory and general algebraic topology that we will need later on. These definitions – and, of course, proofs of the mentioned theorems – can be found in most textbooks on algebraic topology, such as [Hat02], [May99] or [Die08]. In the following, any assumed map between topological spaces is continuous, unless explicitly stated otherwise, and we write I for the unit interval $[0,1] \subset \mathbb{R}$ endowed with the subspace topology. Two other kinds of spaces we will often use are the n-spheres S^n, the subspace of points in $\mathbb{R}^{n+1}$ with distance 1 to the origin, and the n-disks D^n, the subspace of points $\mathbb{R}^n$ with distance less than or equal to 1 to the origin. In all of the previous definitions, we assumed $\mathbb{R}^n$ to be equipped with the standard topology induced by the Euclidean metric.

Definition 17 (Pointed Topological Spaces) A pair (X, x_0) consisting of a topological space X and a designated point $x_0 \in X$ is called a *pointed topological space*, and x_0 is referred to as the *base point* of X.

A map of pointed topological spaces $f\colon (X, x_0) \to (Y, y_0)$ is a map $f\colon X \to Y$ such that $f(x_0) = y_0$. □

Definition 18 (Homotopies of Maps) Let X, Y be topological spaces, $f, g\colon X \to Y$ be maps. A *homotopy* between f and g (sometimes from f to g) is a map $h\colon X \times I \to Y$ such that the map $h_0\colon X \to Y, x \mapsto h(x,0) =: h_0(x)$ is equal to f and the map $h_1\colon X \to Y, x \mapsto h(x,1) =: h_1(x)$ is equal to g. Furthermore, we ask that for any $t \in I$, the analogously defined map h_t is continuous and that for any $x \in X$, the map $h_x\colon I \to Y, t \mapsto h_t(x) =: h_x(t)$ is continuous. In less formal language, a homotopy continuously transforms f into g, never passing through a non-continuous state.

A homotopy h is called a *homotopy relative* A for some subspace $A \subseteq X$ if h_a is constant for any point $a \in A$.

A map k from X to Y is called *null-homotopic* if there exists a homotopy from f to the map PT that maps the entirety of X onto a single point in Y. □

Definition 19 (Homotopy Groups) Let (S^n, s_0) be the pointed n-sphere, (X, x_0) be a pointed topological space. We define the n-homotopy group of (X, x_0) as the set of equivalence classes of continuous, base-point preserving maps $(S^n, s_0) \to (X, x_0)$ up to homotopy relative endpoint. □

Note 3.1 (π_1)
The first homotopy group is often referred to as the *fundamental group*.

Note 3.2 (Alternative Definition)
Occasionally, it will be more convenient to instead consider maps from the n-cube I^n into (X, x_0), where the entire boundary of I^n gets mapped to x_0, up to homotopy relative boundary. Since I^n is homeomorphic to D^n and collapsing the boundary of D^n to a single point results in an S^n, this gives us an equivalent definition.

Claim 3.1 (The Homotopy Groups are Groups)
We define an operation $\circ$ on the n-th homotopy group by concatenation. As it is more convenient for writing out the specifics, we will use the alternative definition for this. Specifically, the product $[\alpha] \circ [\beta]$ of the homotopy equivalence classes $[\alpha]$ and $[\beta]$ is given by the homotopy equivalence class of the map

$$(\alpha * \beta)(t_1, t_2, \ldots, t_n) := \begin{cases} \alpha(2t_1, t_2, \ldots, t_n) & \text{if } t_1 \in \left[0, \frac{1}{2}\right] \\ \beta(2t_1 - 1, t_2, \ldots, t_n) & \text{if } t_1 \in \left[\frac{1}{2}, 1\right]. \end{cases}$$

PROOF We quickly outline the proof. Note that since the entire boundary of the I^n gets mapped to the point x_0, we have $\alpha(1, t_2, \ldots, t_n) = \beta(1, t_2, \ldots, t_n) = x_0$ for every $(t_2, \ldots, t_n) \in I^{n-1}$, so the map is indeed continuous. Furthermore, if $\alpha' \in [\alpha]$ and $\beta' \in [\beta]$, then $\alpha' * \beta' \in [\alpha * \beta]$, so the operation does not depend on the choice of representatives. To see this pick a homotopy h_α from α to α' and a homotopy h_β from β to β'. Then the map $h_{\alpha * \beta} := h_\alpha * h_\beta$ is a homotopy from $\alpha * \beta = \alpha' * \beta'$. Finally, one can check that this indeed turns the set of homotopy equivalence classes of maps into a group with the constant map

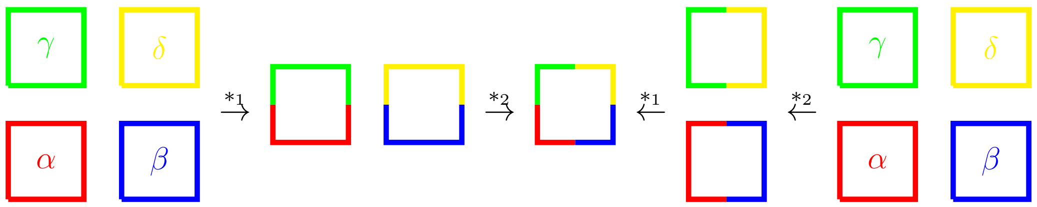

Figure 2: A visual outline of the proof that $*_1$ and $*_2$ commute over each other for $n = 2$.

as the neutral element, the inverse of a map σ being the map σ^{-1} given by $\sigma^{-1}(t_1, \ldots, t_n) = \sigma(1 - t_1, \ldots, t_n)$. ∎

Note 3.3 (The Higher Homotopy Groups are Abelian)
While this is generally not true for the fundamental group, one can see that π_n is abelian for any $n \geq 2$ by defining another operation $*_i$ on the set of homotopy equivalence classes (obviously, $i \leq n$) that concatenates in the i-th instead of the first argument.

One can then apply the Eckmann-Hilton argument, which tells us that for a set S equipped with two operations $\circ, \bullet \colon S \times S \to S$ with a common unit $e \in S$ which commute over each other, meaning that

$$\forall \alpha, \beta, \gamma, \delta \in S \colon\ (\alpha \circ \beta) \bullet (\gamma \circ \delta) = (\alpha \bullet \beta) \circ (\gamma \bullet \delta),$$

the operations $\circ$ and $\bullet$ are the same operation, and that operation is commutative and associative. Both $*$ and $*_i$ have the constant map as a unit and they do indeed commute over each other.

This not only tells us that the higher homotopy groups are abelian, but also that the choice of which argument we concatenate in does not actually matter for the definition.

Definition 20 (Weak Equivalences) A map of topological spaces $f \colon X \to Y$ is called a *weak homotopy equivalence* (or sometimes just *weak equivalence*) if it induces isomorphisms on the homotopy groups. In more detail, for an $n \in \mathbb{N}$ we define the induced map $f_n \colon \pi_n(X) \to \pi_n(Y)$ as postcomposition with f, so $(\sigma \colon S^1 \to X) \mapsto (f \circ \sigma \colon S^1 \to Y)$. Then f is a weak equivalence if $\forall n \in \mathbb{N} \setminus \{0\}$, the maps f_n are group isomorphisms and the map f_0 is a bijection of pointed sets. Noticeably, any homeomorphism is a weak homotopy equivalence, so spaces with non-isomorphic homotopy groups can not be homeomorphic. □

Definition 21 (Path-Connected) A topological space X is called *path-connected* if $\forall x, y \in X$, there is a continuous map $\gamma\colon [0,1] \to X$ (a path) such that $\gamma(0) = x$ and $\gamma(1) = y$. □

Claim 3.2 (The Fundamental Group of a Path-Connected Space) The choice of base-point does not matter when calculating the fundamental group of a path-connected space X, so $\pi_1(X, x_0) \cong \pi_1(X, x_1)$ for any two $x_0, x_1 \in X$.

Proof Let x_0, x_1 be two points in X and let γ be a path from x_0 to x_1. We define a map $\hat{\gamma}\colon \pi_1(X, x_0) \to \pi_1(X, x_1), [\sigma] \mapsto [\gamma^{-1} * \sigma * \gamma]$, where γ^{-1} is the path from x_1 to x_0 defined by $\gamma^{-1}(t) := \gamma(1-t)$. Since the concatenation $\gamma * \gamma^{-1}$ is null-homotopic (and therefore the neutral element of $\circ$), we have

$$\begin{aligned}
\hat{\gamma}([\sigma] \circ [\theta]) &= \hat{\gamma}([\sigma * \theta]) \\
&= \left[\gamma^{-1} * \sigma * \theta * \gamma\right] \\
&= \left[\gamma^{-1} * \sigma * \gamma * \gamma^{-1} * \theta * \gamma\right] \\
&= \left[\gamma^{-1} * \sigma * \gamma\right] \circ \left[\gamma^{-1} * \theta * \gamma\right] \\
&= \hat{\gamma}([\sigma]) \circ \hat{\gamma}([\theta])
\end{aligned}$$

and

$$\begin{aligned}
\left[\gamma * \gamma^{-1} * \sigma * \gamma * \gamma^{-1}\right] &= \left[\gamma * \gamma^{-1}\right] \circ [\sigma] \circ \left[\gamma * \gamma^{-1}\right] \\
&= [\mathrm{PT}] \circ [\sigma] \circ [\mathrm{PT}] \\
&= [\sigma],
\end{aligned}$$

this is a morphism of groups and there is a morphism $\hat{\gamma}^{-1}\colon \pi_1(X, x_1) \to \pi_1(X, x_0)$ such that $\hat{\gamma}^{-1}\hat{\gamma}$ is the identity, and therefore an isomorphism. ■

Definition 22 (Simply and n-Connected) A topological space X is called *simply connected* if it is path-connected and its first homotopy group is trivial.

More generally, a topological space X is called *n-connected* if it is path-connected and its first n homotopy groups are trivial. A simply connected space is therefore also referred to as a *1-connected space.* □

Claim 3.3 (Higher Homotopy Groups of Simply-Connected Spaces) The choice of base-point does not matter when calculating the higher homotopy groups of a path-connected space X, so $\pi_n(X, x_0) \cong \pi_n(X, x_1)$ for any $n \in \mathbb{N}$ and any two $x_0, x_1 \in X$. However, this isomorphism is only canonical if $\pi_1(X)$ is trivial, that is if X is simply connected.

In the following definitions, every topological space is assumed path-connected unless specified otherwise.

Definition 23 (CW complex) A *relative CW complex* (X, A) is a pair of hausdorff topological spaces X and A together with an ascending chain of topological subspaces $X_i \subseteq X$,

$$A = X_{-1} \subseteq X_0 \subseteq X_1 \subseteq X_2 \subseteq X_3 \subseteq \cdots,$$

such that $X = \bigcup_{i \geq -1} X_i$ and for all $k \geq 0$, X_k can be constructed from X_{k-1} by adjoining (or glueing) k-cells, e.g. there exists a pushout-diagramm

$$\begin{tikzcd}
\coprod_{i\in\mathcal{I}} S^{k-1} \arrow[r] \arrow[d, hook] & X_{k-1} \arrow[d, hook] \\
\coprod_{i\in\mathcal{I}} D^{k} \arrow[r] & X_k
\end{tikzcd}$$

where the left arrow is just the disjoint union of the inclusions into the boundary. Furthermore, X then has the final topology with regards to the family X_i, that is the finest topology such that the inclusions $X_i \hookrightarrow X$ are continuous and the triangles

$$\begin{tikzcd}
X_i \arrow[r, hook] \arrow[dr, hook] & X_j \arrow[d, hook] \\
& X
\end{tikzcd}$$

commute for every choice $i < j$. $\square$

Theorem 2 (CW Approximation) *For every space Y there exists a CW-complex X together with a map $f : X \to Y$ such that f is a weak equivalence. This is called the* CW approximation *of Y, and is unique up to homotopy.* $\square$

Proof Section 8.6 in [Die08]. $\blacksquare$

3.2 Rational Spaces and Rationalizations of Spaces

While the homotopy groups are a powerful and valuable topological invariant, they are also notoriously hard to compute. A key difficulty here is that there are maps from higher-dimensional spheres into lower-dimensional ones that are not null-homotopic. The most famous example of such a map is probably the Hopf-fibration $\eta\colon S^3 \to S^2$, a write-up of which can be found in [Lyo]. While it is known that $\pi_n(S^n) \cong \mathbb{Z}$ and $\pi_n(S^m) = 0$ for $n < m$, no general rule for the case $n > m$ is known. Consider the following table of homotopy groups of spheres, lifted from [Bae09]:

$$\begin{array}{lcccc|lcccc}
\pi_3(S^2) & = & \mathbb{Z} & & & \pi_9(S^5) & = & & & \mathbb{Z}/2\mathbb{Z} \\
\pi_5(S^3) & = & & & \mathbb{Z}/2\mathbb{Z} & \pi_{11}(S^6) & = & \mathbb{Z} & & \\
\pi_7(S^4) & = & \mathbb{Z} & \times & \mathbb{Z}/12\mathbb{Z} & \pi_{15}(S^8) & = & \mathbb{Z} & \times & \mathbb{Z}/120\mathbb{Z}
\end{array}$$

While this is obviously just a (very) small sample, the seeming randomness of the non-free part might motivate one to wonder if it is possible to calculate homotopy groups while “ignoring” torsion. From an algebraic perspective, one can “eliminate” torsion in $\mathbb{Z}$-modules by tensoring with $\mathbb{Q}$. But this is an algebraic operation, and its result is not guaranteed to have a sensible interpretation on the level of spaces. For example, consider tensoring $\pi_1(S^1) \cong \mathbb{Z}$ with $\mathbb{Q}$. The result is $\mathbb{Q}$, but if the integer $z \in \mathbb{Z}$ corresponded to the base-point preserving map from S^1 to S^1 given by wrapping it around itself z times, what does the fraction $\frac{1}{z}$ represent? There is no base-point preserving map from S^1 to itself that covers only a fraction of the circle. The question then raises itself if it is possible to perform the algebraic operation of tensoring with $\mathbb{Q}$ on the level of topological spaces, if one can develop a well-behaved theory of "tensoring spaces with the rationals". This is the jump-off point for rational homotopy theory.

Definition 24 (Rational Space) A simply connected space X is called a *rational space* if all its higher homotopy groups are rational vector spaces, that is

$$\pi_i\left(X\right) \otimes \mathbb{Q} = \pi_i\left(X\right) \qquad \forall i > 0.$$

□

Definition 25 (Finite-type Rational Spaces) Let X be some rational space. If all the homotopy groups of X are finite-dimensional vector spaces, then we say X is of *finite-type.* □

Since we will only consider simply connected spaces, as by the previous section, we will drop the base point from the notation when discussing the higher homotopy groups.

Definition 26 (Rational Homotopy Equivalence) A continuous map of topological spaces $f \colon X \to Y$ is called a *rational homotopy equivalence* (or *rational equivalence* for short) if it induces a linear isomorphism on the *rational homotopy groups*, which is to say

$$\pi_\star f \otimes \mathbb{Q} \colon \pi_\star\left(X\right) \otimes \mathbb{Q} \xrightarrow{\cong} \pi_\star\left(Y\right) \otimes \mathbb{Q}$$

. □

Note 3.4 (Weak and Rational Homotopy Equivalences)
Any weak equivalence is a rational equivalence. A map $f : X \to Y$ is a rational equivalence if and only if it is a weak equivalence.

Definition 27 (Rationalizations) A continuous map $f\colon X \to Y$ is called a *rationalization* of X if it is a rational homotopy equivalence and Y is a rational space. □

Theorem 3 (Rational Hurewicz Theorem) *Let X be a simply connected space, r some natural number Then the following two statements hold:*

(i) If $\pi_i(X) \otimes \mathbb{Q} = 0$ for all $i < r$, then the map $H\colon \pi_i(X) \otimes \mathbb{Q} \overset{\cong}{\to} \mathcal{H}_i(X;\mathbb{Q})$ induced by the Hurewicz-map is a natural isomorphism for all $i < 2r-1$ and a surjection for $i = 2r-1$.

(ii) The i-th homotopy group of X is a rational vector space if and only if the i-th homology group is one. □

Proof (i) Can be found in [KK04].

(ii) This is lemma 8.8 in [GM13]. ■

Theorem 4 (Rational Whitehead Theorem) *Let $f : X \to Y$ be a map of simply connected spaces. Then f is a rational equivalence if and only if $\mathcal{H}_*(f;\mathbb{Q})$, the map induced by f in homology with rational coefficients, is an isomorphism.* □

Proof Theorem 8.6 in [FHT01]. ■

Theorem 5 (Rationalization of a Space) *For any simply connected space X, there exists a rationalization $X_\mathbb{Q}$ of X, that is to say a rational space $X_\mathbb{Q}$ such that*

(a) the map

$$\iota_\mathbb{Q}\colon X \hookrightarrow X_\mathbb{Q}$$

is both an inclusion and a rational homotopy equivalence *(b) any other rationalization of X factors over $X_{\mathbb{Q}}$, that is to say if Y is a simply connected, rational space and $f\colon X \to Y$ is a rational homotopy equivalence, then there exists a unique (up to homotopy) map $f_{\mathbb{Q}}\colon X_{\mathbb{Q}} \to Y$ such that the diagram below commutes:*

$$\begin{tikzcd} X \arrow[rr, "f"] \arrow[dr, hook, "\iota_{\mathbb{Q}}"'] & & Y \\ & X_{\mathbb{Q}} \arrow[ur, "f_{\mathbb{Q}}"'] & \end{tikzcd}$$

□

This theorem tells us that it is sensible to speak of *the* rationalization of a space, since the space $X_{\mathbb{Q}}$ from the above theorem is unique up to homeomorphism (to see this, assume another space with the same property and plug the two triangle diagrams together). We will not explicitly prove this theorem due to size constraints, but hopefully, the exemplary constructions below will give a good intuition as to why it holds true.

We will now explicitly construct the rationalization of the 1-sphere and use it to give an intuition as to how one can construct the rationalization of an arbitrary simply connected space. Note that, since the 1-sphere is not a simply connected space itself, it is technically not covered by the above theorem. However, it is an example of a non-simply connected space that still permits a rationalization, since its fundamental group is abelian.[4] For a more in-depth and rigorous account that covers the higher dimensional spheres, CW-complexes and arbitrary simply connected spaces as well, see chapter 9 of [FHT01].

Example 4 (Rationalization of the 1-Sphere) A way of characterizing that an abelian group $\mathcal{A}$ permits the structure of a $\mathbb{Q}$-vector space is that the equation $z\alpha = \beta$ has a unique solution $\alpha \in \mathcal{A}$ for all $z \in \mathbb{Z}\setminus\{0\}$, $\beta \in \mathcal{A}$ [GM13]. More informally, we can "divide by any integer" in $\mathcal{A}$. Since the higher homotopy groups of the 1-sphere are trivial, we can focus on only the fundamental

[4]One can extend rational homotopy theory to 0-connected spaces that satisfy certain properties, see [GHT99]. However, for scope reasons, we will not cover that case here as well.

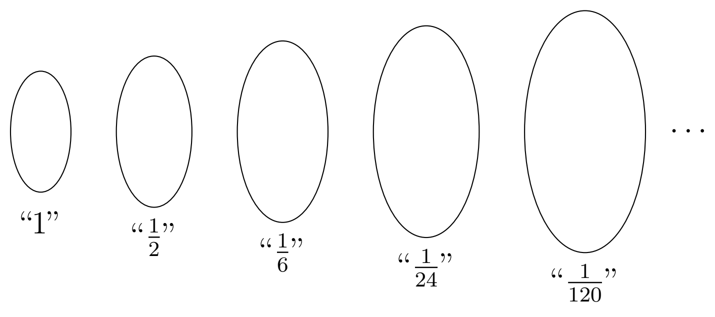

Figure 3: Constructing $S^1_{\mathbb{Q}}$, step 1: Adding copies of the S^1.

group when discussing the rationalization.

Consider that we can identify $\pi_1(S^1)$ with the integers, where the (homotopy equivalence class of the) map $f_z\colon S^1 \to S^1$ wrapping the circle around itself z times is identified with the integer z. From this, we can use our understanding of how division by integers works to derive a geometric intuition of what the rationalization $S^1_{\mathbb{Q}}$ of the S^1 needs to look like as a space.

To develop that intuition, let us return to the equation $z\alpha = \beta$ again, and let β be the identity on S^1. Then the equation is solvable if there is a base-point preserving map $S^1 \to S^1_{\mathbb{Q}}$ that corresponds to “walking” a $1/z$-th of the unit circle. In other words: for each integer z, there needs to be a map from the 1-sphere into the rationalization such that “walking around” the map corresponding to the identity is homotopic to “walking around” that map z times. Furthermore, these need to be compatible with each other, so “walking around” the “z-copy” once has to be homotopic to traversing the “kz-copy” k times.

Now, we face the issue that the maps of the S^1 onto these different copies are not homotopic to each other (and therefore, we can not really speak of one of them corresponding to z times another one). To rectify this issue, we glue a hollow cylinder in between each copy of the S^1 and its successor, the “left end” of the cylinder being glued to the z-th copy of S^1 via the identity and the “right end” being glued to the successor by being wrapped around it $z+1$

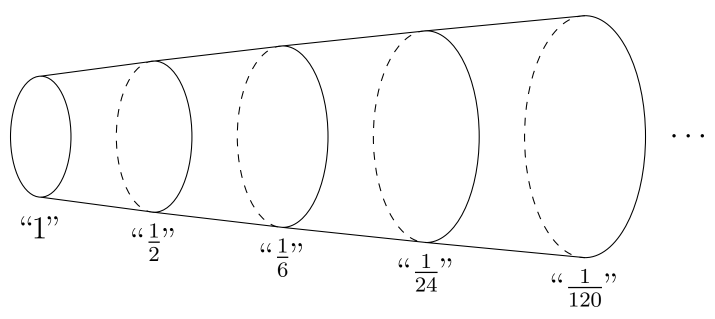

Figure 4: Constructing $S^1_{\mathbb{Q}}$, step 2: Gluing cylinders inbetween the copies of the S^1.

times.

So we end up with a space that looks like some sort of infinite twisted telescope. Let us investigate how this space actually fulfills our algebraic requirement from above. Note that by the way we glued the cylinders in, going around the first circle once is homotopic to going around the second circle twice, and going around the second circle twice is homotopic to going around the third circle thrice, so going around the first circle once is homotopic to going around the third circle six times. In more general terms: going around the first circle once is homotopic to going around the z-th circle $z!$ times, and going around the n-th circle once is going around the z-th circle $\frac{z!}{n!}$ times. So it seems this space allows us to solve all the above equations.

Let us now attempt to turn this intuition into a rigorous description. The first thing we notice is, that since all maps from the S^1 into our space are supposed to be base-point preserving, we can't actually just use disjoint copies of the S^1, but need to identify their base point with each other somehow. Starting out with $S^1(1) = S^1$, we build the rationalization inductively by defining $S^1\,(r+1)$ as the pushout

$$\begin{tikzcd} S^1 \arrow[r, "\phi"] \arrow[d, hook, "i"] & S^1\,(r) \vee S^1 \arrow[d] \\ D^2 \arrow[r] & S^1(r+1), \end{tikzcd}$$

where i is the inclusion into the boundary and ϕ is the composition

$$\begin{tikzcd} S^1 \arrow[r] & S^1 \vee S^1 \arrow[r, "j \vee g_r"] & S^1(r) \vee S^1, \end{tikzcd}$$

with the first map being the collapse of the equator, j being the inclusion into the r-th copy of the S^1 in $S^1(r)$ and g_r being the map wrapping the sphere $r+1$ times around itself. This yields the definition $S^1\,(k) = \bigvee_{i=1}^{k} S^1 \cup_g \coprod_{i=1}^{k-1} D^2$, where $\cup_g$ is supposed to signify that we glue the disjoint union of disks into the smash product of spheres along the maps g_i. Since $S^1\,(r+1)$ is created from $S^1\,(r)$ by adjoining cells, this defines the structure of a CW-complex $S^1_{\mathbb{Q}} \coloneqq \bigvee_{k\geq 1} S^1 \cup_g \coprod_{k\geq 2} D^2$. This space is exactly the rationalization of S^1.

Why is this the case? Note that at any point in the construction, the inclusion $i_r\colon S^1 \hookrightarrow S^1(r)$ into the r-th copy of the S^1 is a weak equivalence, as we can collapse the "telescope" to the last sphere, so $\pi_1\left(S^1(r)\right) = \mathbb{Z}$. Furthermore, consider the maps $i_r\colon S^1 \hookrightarrow S^1(r+1)$ and $i_{r+1}\colon S^1 \hookrightarrow S^1(r)$. By construction, i_r is homotopic to $(r+1)$ times i_{r+1}, so $[i_r] = (r+1)\,[i_{r+1}] \in \pi_1(S^1(r+1))$. That means we can divide the class $[i_r]$ by $(r+1)$ in $\pi_1(S^1(r+1))$ and the map induced by the inclusion $S^1(r) \hookrightarrow S^1(r+1)$ is given by multiplication with $(r+1)$. Since the fundamental group and the first homology group of the S^1 are isomorphic, we get a filtered colimit of homology groups

$$\begin{tikzcd} \mathbb{Z} \arrow[r, "\cdot 2"] & \mathbb{Z} \arrow[r, "\cdot 3"] & \mathbb{Z} \arrow[r, "\cdot 4"] & \mathbb{Z} \arrow[r, "\cdot 5"] & \mathbb{Z} \arrow[r, "\cdot 6"] & \cdots\,. \end{tikzcd}$$

Homology commutes with colimits, and doing this in every degree yields that

$$\mathcal{H}_n\left(S^1_{\mathbb{Q}}\right) = \begin{cases} \mathbb{Q}, & n = 1 \\ 0, & \text{otherwise,} \end{cases}$$

and by the second version of the rational Hurwicz theorem, $S^1_{\mathbb{Q}}$ is a rational space. But then the rational Whitehead theorem tells us that the inclusion $i_1 \colon S^1 \hookrightarrow S^1_{\mathbb{Q}}$ into the initial sphere is a rationalization. The only thing left to show is that every other rationalization factors over it. We will not prove this right now, but prove a more general statement in a second.

Having defined the rational 1-sphere, we can define the rational 2-disk $D^2_{\mathbb{Q}}$ analogous to how we can define the regular D^2: as the cone of $S^1_{\mathbb{Q}}$, that is as the space $S^1_{\mathbb{Q}} \times I / S^1_{\mathbb{Q}} \times \{0\}$, where I is the unit interval. □

Note 3.5 (Rational Spheres and Disks)
Analogous to the constructions above, we can construct $S^n_{\mathbb{Q}} = \bigvee_{k \geq 1} S^n \cup_g \coprod_{k \geq 2} D^{n+1}$ and $D^{n+1}_{\mathbb{Q}} = S^n_{\mathbb{Q}} \times I / S^n_{\mathbb{Q}} \times \{0\}$, the rational n-sphere and rational n-disk.

Lemma 2 *Let X be a rational space and $f \colon S^n \to X$ be a map. Then f factors over $S^n_{\mathbb{Q}}$, that is there exists an extension $f_{\mathbb{Q}} \colon S^n_{\mathbb{Q}} \to X$ such that the diagram*

$$\begin{tikzcd} S^n \arrow[r] \arrow[dr, "f"] & S^n_{\mathbb{Q}} \arrow[d, "f_{\mathbb{Q}}"] \\ & X \end{tikzcd}$$

commutes. This map is determined up to homotopy, that is homotopic maps have homotopic extensions and for any homotopy class of maps from S^n to X, their extension is unique up to homotopy. □

Proof Note that the map f represents an element $[\alpha]$ in the n-th homotopy group of X. Since the n-th homotopy group is a rational vector space by assumption, there exist elements $\frac{1}{2}[\alpha], \frac{1}{3}[\alpha], \ldots$ in it. We pick representatives of these elements and suggestively call them $\frac{1}{2}f, \frac{1}{3}f$ etc. We then extend f

onto $S^1_{\mathbb{Q}}$ by defining $f_{\mathbb{Q}}$ to be equal to $\frac{1}{k!}f$ on the k-th copy of the S^n in $S^n_{\mathbb{Q}}$. One can then verify that this a well-defined map and that it has the desired properties. ■

Example 5 (The Rationalization of a CW-Complex) Having constructed the rationalizations of the n-spheres and -disks, we now use this to construct the rationalization of an arbitrary simply connected CW-complex X. The basic idea is to replace the copies of S^n and D^n in the push-outs with their rationalizations.

We again define the rationalization $X_{\mathbb{Q}}$ inductively, by rationalizing every space X_i in the ascending chain of the CW-structure of X. Since X is simply connected, we can assume it contains no 1-cells, so $X^0_{\mathbb{Q}}$ and $X^1_{\mathbb{Q}}$ are just the single-point space. Now, let

$$\begin{tikzcd}
\coprod_{i\in I} S^k \arrow[r, "\coprod g_i"] \arrow[d] & X^k \arrow[d] \\
\coprod_{i\in I} D^{k+1} \arrow[r] & X^{k+1}
\end{tikzcd}$$

be the pushout defining X^{k+1} and assume the rationalization of X^k has already been constructed. We can post-compose the rationalization map $X^k \to X^k_{\mathbb{Q}}$ to the attaching maps g_i from the S^k to X^k in the pushout diagram defining X^{k+1}, and by the previous lemma, this yields attaching maps g_i' from $S^k_{\mathbb{Q}}$ to $X^k_{\mathbb{Q}}$. Thus, we can construct a pushout

$$\begin{tikzcd}
\coprod_{i\in I} S^k_{\mathbb{Q}} \arrow[r, "\coprod g_i'"] \arrow[d] & X^k_{\mathbb{Q}} \arrow[d] \\
\coprod_{i\in I} D^{k+1}_{\mathbb{Q}} \arrow[r] & X^{k+1}_{\mathbb{Q}}.
\end{tikzcd}$$

We can plug together these two pushouts and, by the universal property of the pushout, this yields a unique map $X^{k+1} \to X^{k+1}_{\mathbb{Q}}$. One can then show that both $X^{k+1}_{\mathbb{Q}}$ is rational and that this map is a rationalization using the Mayer-Vietoris sequence for push-outs and the 5-lemma. □

Corollary 1 *Every simply connected space X admits a rationalization.* □

PROOF Let $Y \xrightarrow{f} X$ be a CW-approximation of X and let $Y \xrightarrow{\phi} Y_{\mathbb{Q}}$ be the rationalization of that approximation. We define $X_{\mathbb{Q}} := X \cup_f (Y \times I) \cup_\phi Y_{\mathbb{Q}}$, with the inclusion $X \overset{\psi}{\hookrightarrow} X_{\mathbb{Q}}$ into the "starting point".

Using excision, we can see that

$$\mathcal{H}_{ast}\left(X_{\mathbb{Q}}, Y_{\mathbb{Q}}\right) \cong \mathcal{H}_{ast}\left(X \cup_f (Y \times I), Y \times 1\right) = 0.$$

By the long exact sequence of homology, we thus get $\mathcal{H}_{ast}(X_{\mathbb{Q}}) \cong \mathcal{H}_{ast}(Y_{\mathbb{Q}})$, and by rational Hurewicz, $X_{\mathbb{Q}}$ is a rational space. Using again the long exact sequence of homology, the five lemma and the already known isomorphisms of two of its parts, we get that $\mathcal{H}_{ast}(X_{\mathbb{Q}}, X; \mathbb{Q}) \cong \mathcal{H}_{ast}(Y_{\mathbb{Q}}, Y; \mathbb{Q}) = 0$, and therefore ψ induces an isomorphism in rational homology. But then the rational Whitehead theorem tells us that $X_{\mathbb{Q}}$ is a rationalization of X. ■

3.3 Sullivan Models of Rational Spaces

In order to understand the machinery we use to take a topological space and "turn it into" an algebraic object one can calculate with, we first need to introduce some algebraic notions. Throughout this chapter, we assume all structures to be over $\mathbb{Q}$. One can define all of the following notions in terms of modules over a ring as well, but as we do not require that generality, we will not bother with it. The main references for this chapter are [FHT01] as well as the STACKS project.

Definition 28 (Graded Vector Space) A vector space V is called a *graded vector space* if it can be written as a direct sum of vector spaces V^i indexed over the natural numbers,

$$V = \bigoplus_{i \in \mathbb{N}} V^i.$$

We will often write V^* instead of just V in order to signify we are considering a graded object.

An element $v \in V$ that is contained within one of the V^i is called *homogeneous of degree i.*

A map of graded vector spaces $\phi : V \to W$ is said to *be of degree n* (or have degree n) if it maps homogeneous elements of degree i to homogeneous elements of degree $i+n$.

If a graded vector space consists only out of homogeneous elements of the same degree, that is $V^* = \bigoplus_{i\in\mathbb{N}} V^i$ with $V^* = V^i$ for some $i \in \mathbb{N}$ and for all $j \neq i$ we have $V^j = 0$, then we say that V^* is concentrated in degree i. □

Definition 29 (Cochain Complex) A *Cochain complex* is a graded vector space C^* together with a map $d\colon C^* \to C^*$ of degree 1 such that $d \circ d = 0$. This is also sometimes written as $d^2 = 0$. The map d is usually referred to as the *coboundary map* or as the *differential*, and the elements in its image are referred to as the *coboundaries*, where as the elements in its kernel are called the *cocycles.* □

Definition 30 (Graded (Commutative) Algebras) A graded vector space A^* is called a *graded algebra* if there exists a bilinear and associative multiplication $\mu\colon A^* \times A^* \to A^*$ such that $\mu\left(A^n, A^m\right) \subseteq A^{n+m}$ and which has a unit $e \in A^0$. For two elements $a, b \in A^*$, we will usually write just ab for $\mu(a,b)$. Furthermore, if for all homogeneous elements $a, b \in A^*$ we have $ab = (-1)^{|a||b|}\, ba$, then we call A^* *graded commutative.* Some authors will also refer to this property as just commutativity.

If there exists a map of graded algebras $\epsilon\colon A^* \to \mathbb{Q}$, with $\mathbb{Q}$ viewed as a graded algebra concentrated in degree 0, we call A^* *augmented* and the map ϵ the *augmentation.* □

Definition 31 ((Commutative) Differential Graded Algebra) A *differential graded algebra* is a graded algebra that also carries the structure of a cochain complex in a compatible way, so it is equipped with a differential $d\colon A^* \to A^*$ that fulfills the *Leibniz rule*

$$d\,(a, b) = (da)\, b + (-1)^{|a|} a\, (db)\,.$$

We then call d a *derivation.* Furthermore, if the multiplication on A^* is graded commutative, we call A^* a *commutative differential graded algebra*, or *cdga* for short. □

Definition 32 (The Free Algebra over a (Graded) Vector Space) Let V be a graded vector space. We define the *free algebra over V* as the graded commutative algebra

$$\Lambda\left(V\right) \coloneqq \bigoplus_{n\in\mathbb{N}} \Lambda^n\left(V\right), \qquad \text{where } \Lambda^n\left(V\right) \coloneqq \left\{v_1 v_2 \cdots v_n \mid v_i \in V\right\}$$

and the multiplication is graded commutative. Another way to think about this is to view the elements of V (obviously, it is sufficient to just consider any basis) as an alphabet. Then $\Lambda^n\left(V\right)$ is set of all words of length n over that alphabet, with a handful of special properties

(i) The words permits a scalar multiplication with elements from $\mathbb{Q}$

(ii) The letters have degrees

(iii) The positions of adjacent letters can be swapped, inducing a multiplication with -1 to the power of the product of the degrees of those letters (this is just graded commutativity)

(iv) As a direct result of the previous property, we have $a^2 = -a^2$ for any odd-degree letter, so odd-degree letters cancel themselves.

It is important to be aware of the difference between word-length and degree. A word of the shape $v_1 v_2 \cdots v_n$ will be of length n, but its degree will be the sum of the degree of the letters, so generally speaking *not* n. We will write $\Lambda^n\left(V\right)$ for the set of all elements of word-length n and $\Lambda\left(V\right)^n$ for the set of all elements of degree n. □

Definition 33 (Reduced and Minimal Algebras) Let A^* be a cdga. If $A^0 = \mathbb{Q}$, then we call A^* *reduced.* Furthermore, if $A^r = 0$ for $0 < r \leq n$, we call A^* *n-reduced.*

A 1-reduced cdga is called *minimal* if it is isomorphic to a free algebra over a graded vector space $(\Lambda(V^*), d)$ that fulfills the following (equivalent) conditions

(i) The graded vector space V^* is trivial in degree 0 and 1, $V^0 = 0$ and $V^1 = 0$, so $(\Lambda(V^*), d)$ is also 1-reduced

(ii) The differential d has the property that $d(V^*) \subset \Lambda^{\geq 2}(V^*)$, so every coboundary is of word-length at least 2. □

Definition 34 (Sullivan Algebras) A cdga S is called a *Sullivan algebra* if it can be written as the free graded commutative algebra over a vector space V, that is $S = (\Lambda V, d)$ such that

(i) V permits a filtration

$$0 = V(-1) \subset V(0) \subset V(1) \subset V(2) \subset \cdots \bigcup_{i \in \mathbb{N}} V(i) = V$$

(ii) $d(V(k)) \subset \Lambda V(k+1)$ □

Theorem 6 (The Main Equivalence) *There exists a bijection between rational homotopy types of finite-type rational spaces and isomorphism classes of minimal Sullivan algebras. We will thus often speak of a space's* (minimal) Sullivan model *to signify the image of its rational homotopy type under this bijection.* □

This is maybe the central result of rational homotopy theory, and it is what enables us to do all of the work in the later sections. There are two proofs of this, one due to Sullivan [Sul77] and one due to Quillen [Qui69], but unfortunately, either is too large in scope and too reliant on abstract machinery to cover here. A more modern treatment can be found in section II of [FHT01].

Proposition 1 *Let X be a finite-type rational space and $(\Lambda V, d)$ its minimal Sullivan model. Then we have*

$$\mathcal{H}^*(\Lambda V, d) \cong \mathcal{H}^*(X; \mathbb{Q}) \qquad \text{and} \qquad V \cong \operatorname{Hom}_{\mathbb{Z}}(\pi_{ast}(X), \mathbb{Q}) .$$ □

Proof Theorem 10.11 and 15.11 in [FHT01]. ■

4 Cohomology of Finite-Type Rational Spaces

In this section, we will carry out Lechuga's and Murillo's proof that the calculation of the cohomology groups of a finite-type rational space is NP-hard. To do this, we will proceed along the following steps: After introducing the necessary mathematical vocabulary and defining the computational problem, we first prove a statement that relates the k-colorability of a graph to the cohomology of a rational space with a carefully constructed Sullivan algebra. Second, we introduce a way to uniquely and unambiguously encode a finite type rational space as a string, using its Sullivan algebra. Finally, we use this encoding to construct a Turing reduction from the problem KCOL to a subclass of the problem $\text{COHOM}_{\mathbb{Q}}$. This, together with the first theorem, will then yield the desired result.

Definition 35 (Elliptic Space) Let X be a finite-type rational space. If all the rational cohomology groups are finitely generated, $\dim \mathcal{H}^* (X; \mathbb{Q}) < \infty$, then we call X *(rationally) elliptic.* □

Note that in this thesis, whenever we consider homology or cohomology of a space, we use these terms to refer to singular cohomology with rational coefficients.

Definition 36 (Finite-Type and Elliptic Sullivan Algebras) A minimal Sullivan algebra $(\Lambda V, d)$ is *finite-type* (and *elliptic*) if and only if the rational space represented by it is. This is a direct consequence of 1. □

Problem 4.1 ($\text{COHOM}_{\mathbb{Q}}$)
Instance: *A rational space S of finite-type.*
Question: *Is S elliptic, that is are the cohomology groups $/H^n (S; \mathbb{Q})$ of S finite dimensional?*

Before we construct the actual reduction, we are first going to prove an important result relating the colorability of a graph to the ellipticness of a carefully constructed Sullivan algebra.

Theorem 7 (Theorem 3, [LM00]) *Let G be an undirected, connected, simple and finite graph with vertex set $V(G) := \{v_1, v_2, \dots, v_n\}$ and edge set $E(G) := \{(v_i, v_j) \mid (i,j) \in J\}$ for an appropriate index set J. Let further be $S_{G,k}$ be the rational space with minimal Sullivan model $(\Lambda Z_{G,k}, \partial)$ where*

$$Z^{\text{even}}_{G,k} = \langle x_1, \dots, x_n \rangle, \text{ where } \forall i \in \{1, \dots, n\}: \deg x_i = 2 \text{ and } \partial x_i = 0$$

$$Z^{\text{odd}}_{G,k} = \left\langle y_{(i,j)} \right\rangle, \text{ where } \forall (i,j) \in J\colon \deg y_{(i,j)} = 2k-3 \text{ and } \partial y_{(i,j)} = \sum_{\ell=1}^{k} x_i^{k-\ell} x_j^{\ell-1}$$

Then G is k-colorable if and only if S is non-elliptic. □

Proof By proposition 32.5 in [FHT01], a minimal Sullivan algebra $(\Lambda Z_{G,k}, d)$ is elliptic if and only if there is a non-trivial morphism of differential graded algebras $\phi\colon (\Lambda Z_{G,k}, d) \to (\mathbb{C}[\alpha], 0)$, where $\deg \alpha = 2$.

Consider the following diagram displaying ϕ:

$$\begin{tikzcd}
& & & & y_{(r,s)} \arrow[rrr, mapsto, dotted] \arrow[ddd, mapsto, dotted] & & & \sum_{\ell=1}^{k} x_r^{k-\ell} x_s^{\ell-1} \arrow[ddd, mapsto, dotted] & \\
\Lambda Z^0_{G,k} \cong \mathbb{Q} \arrow[r] \arrow[d, "\phi_0"] & \Lambda Z^1_{G,k} \arrow[r] \arrow[d, "\phi_1"] & \Lambda Z^2_{G,k} \arrow[r] \arrow[d, "\phi_2"] & \cdots \arrow[rr] & & \Lambda Z^{2k-3}_{G,k} \arrow[r] \arrow[d, "\phi_{2k-3}"] & \Lambda Z^{2k-2}_{G,k} \arrow[rr] \arrow[d, "\phi_{2k-2}"] & & \cdots \\
\mathbb{C}[\alpha]^0 \cong \mathbb{C} \arrow[r, "0"] & \mathbb{C}[\alpha]^1 \arrow[r, "0"] & \mathbb{C}[\alpha]^2 \arrow[r, "0"] & \cdots \arrow[rr, "0"] & & \mathbb{C}[\alpha]^{2k-3} \arrow[r, "0"] & \mathbb{C}[\alpha]^{2k-2} \arrow[rr, "0"] & & \cdots \\
& & & & \phi\left(y_{(r,s)}\right) \arrow[rr, mapsto, dotted] & & 0 & \overset{!}{=} \sum_{\ell=1}^{k} \phi(x_r)^{k-\ell} \phi(x_s)^{\ell-1} &
\end{tikzcd}$$

Since such a morphism needs to fulfill the chain map property (that is each of the squares in the above diagram needs to commute), it follows that such a morphism exists if and only if $\forall (i,j) \in J\colon \phi_{2k-2} \circ \partial y_{(i,j)} = d_{\mathbb{C}[\alpha]} \circ \phi_{2k-3}\left(y_{(i,j)}\right) = 0$, which is equivalent to the system of linear equations $\left\{\sum_{\ell=1}^{k} x_i^{k-\ell} x_j^{\ell-1} \mid (i,j) \in J\right\}$ having a non-trivial solution. Since ϕ, as a morphism of graded algebras, needs to "respect degree", there is a set of $\lambda_i \in \mathbb{C}$ such that $\phi(x_i) = \lambda_i \alpha$, and the λ_i are a non-trivial solution to the system of equations.

We now use this preliminary thought to prove the theorem:

"$\Leftarrow$": Assume that S is non-elliptic. This means that the system of linear equations described by the differentials of the $y_{(r,s)}$ has a non-trivial solution. Let $\{z_1, \ldots, z_n\}$ be such a solution. Again, there is a morphism $\phi\colon (\Lambda V_{G,k}, d) \to (\mathbb{C}[\alpha], 0)$ such that $\phi(x_i) = z_i\alpha$. Furthermore, since G is connected, it follows that, if there is a $z_i = 0$, then $\forall (i,j) \in J\colon z_j = 0$, since $0 = \sum_{\ell=1}^{k} z_i^{k-\ell} z_j^{\ell-1} = z_j^{k-1}$. Therefore, for a path of vertices $P = (v_a, v_b, \ldots, v_z)$ in G, it follows that the z_i corresponding to the vertices of the path must all be zero if at least one of them is zero. But since G is connected, there is a path from each vertex to any other vertex. So if one of the z_i was to be zero, this would "propagate" through the edges of the graph and all z_i would be zero, which contradicts the assumption of the set of z_i being a non-trivial solution. Therefore, all z_i are non-zero.

Note that as a corollary from this, we get that no two "adjacent" z_i are equal. Assume $(i,j) \in J$ such that $z_i = z_j$ (note that, since $(i,j) \in J \iff (v_i, v_j) \in E(G)$ and G has no loops, $(i,j) \in J \Rightarrow i \neq j$). Then $\sum_{\ell=1}^{k} z_i^{k-\ell} z_j^{\ell-1} = k\, z_i^{k-1}$ and this expression can only be zero if z_i is zero.

Furthermore, since

$$\begin{aligned}
0 &= (z_i - z_j) \sum_{\ell=1}^{k} z_i^{k-\ell} z_j^{\ell-1} \\
&= \sum_{\ell=1}^{k} z_i^{k+1-\ell} z_j^{\ell-1} - \sum_{\ell=1}^{k} z_i^{k-\ell} z_j^{\ell} \\
&= z_i^k + \sum_{\ell=2}^{k} z_i^{k+1-\ell} z_j^{\ell-1} - \sum_{\ell=1}^{k-1} z_i^{k-\ell} z_j^{\ell} - z_j^k \\
&= z_i^k - z_j^k\ ,
\end{aligned}$$

it follows that the k-th powers of all z_i are equal (again, the equality "propagates" since G is connected). We can pick to be what we like `! more detail reason !`, for example 1. Then, the z_i are the k-th roots of unity, and $\kappa(v_i) \coloneqq \sigma^{-1}(z_i)$ gives us a k-coloring of G.

"$\Rightarrow$": Assume that G is k-colorable. This means that there is a map $\kappa\colon V(G) \to \{1, \dots, k\}$ such that $\forall (i,j) \in J\colon \kappa(x_i) \neq \kappa(x_j)$. Let further σ be the bijection between the set $\{1, \dots, k\}$ and the set of k-th roots of unity, $\left\{\exp\left(\frac{2pi\pi}{k}\right) \mid p \in \{1, \dots, k\}\right\}$. Since there is a one-to-one correspondence between vertices of G and even-degree generators x_i of Z and a homomorphism of algebras (over a field) is completely defined by how it behaves on the basis of the domain (after all, an algebra is still a vector space), we can define a homomorphism of (graded) algebras $\phi\colon (\Lambda Z_{G,k}, d) \to (\mathbb{C}[\alpha], 0)$, $x_i \mapsto \sigma(\kappa(v_i))\,\alpha$. Let $\zeta_p := \exp\left(\frac{2pi\pi}{k}\right)$ be the p-th k-th root of unity. Using the calculation that we used to show that the k-th powers of the coefficients of the images of the x_i must all be equal from the first part of the proof, we get that

$$\sum_{\ell=1}^{k} \zeta_i^{k-\ell}\zeta_j^{\ell-1} = \frac{\overbrace{\zeta_i^k}^{=1} - \overbrace{\zeta_j^k}^{=1}}{\zeta_i - \zeta_j} = 0\,,$$

so the ζ_j are a non-trivial solution to the system of linear equations described by the differentials of the $y_{(i,j)}$, and by our preliminary considerations, this means that S is non-elliptic. ■

Note 4.1 (Encoding of a Finite-Type Space)
We will now construct a unique encoding for a sub-class of finite-type rational spaces, which we will then use in constructing our Turing reduction.

Consider the class Γ_k of finite-type rational spaces whose minimal models $(\Lambda V, d)$ satisfy the condition that $dV \subset \Lambda^{<k}V$, that is every element in the differential is of word-length less than k. For a space $S \in \Gamma_k$, let $(\Lambda V, d)$ be the minimal Sullivan model of S. Pick a homogeneous basis $\{x_i, \dots, x_m\}$ of V. The differentials of the x_j can be written as

$$dx_j = \sum_{p<k} \lambda^j_{i_1 \cdots i_p} x_i \cdots x_p \text{ for } 1 \leq i_1 \leq i_2 \leq \cdots \leq i_p \leq m.$$

As we allow index repetition, but do not take order into account (due to the commutativity of the algebra), the amount of such coefficients $\lambda^j_{i_1 \cdots i_p}$ is bound from above by the sum of multiset coefficients

$$\sum_{p=1}^{k-1} \left(\!\!\binom{m}{p}\!\!\right) = \sum_{p=1}^{k-1} \binom{m+p-1}{p} = \sum_{p=1}^{k-1} \frac{(m+p-1)!}{(m-1)!p!}.$$

However, since we are considering a minimal Sullivan model, we have $dV \subset \lambda^{\geq 2}V$, so the first summand disappears and the upper bound of the amount of coefficients is actually given by

$$m \sum_{p=2}^{k-1} \frac{(m+p-1)!}{m!p!},$$

which is itself bounded by a polynomial in m.

Since there is a one-to-one correspondence between isomorphism classes of minimal Sullivan models and rational homotopy equivalence classes of finite-type rational spaces, we can uniquely encode S as a string consisting out of the amount of generators m, the degree of each generator and the coefficients of the differentials.

Theorem 8 (Corollary 3, [LM00]) *The problem* coHom$_{\mathbb{Q}}$ *is* NP*-hard.* □

Proof We begin by fixing an integer $k \geq 3$ and then construct a Turing reduction from kCol to the subproblem of coHom$_{\mathbb{Q}}$ that only considers spaces in Γ_k. Note that since this is a proper subclass of the class of finite-type rational spaces, proving that the problem is NP-hard for this subclass automatically extends to a proof that the problem is NP-hard for the entire class. After all, if there was an algorithm that was capable of solving the problem in polynomial time for any arbitrary finite-type rational space, that same algorithm would surely also solve the problem in polynomial time for any space in Γ_k.

Now, let $G = (V, E)$ a graph with n vertices, encoded as its number of vertices and adjacency matrix. Thus, if we view the amount of vertices as the size of the input instance, the length of the encoding is bound from above by the polynomial $\log_2 n + n^2$ (binary encoding). We transform this graph into a Sullivan algebra $(\Lambda Z_{G,k}, \partial)$ as defined in 4 using the encoding outlined above. Note that by definition, $(\Lambda Z_{G,k}, \partial) \in \Gamma_k$. Since the length of the encoding is bound from above by a polynomial in the amount of generators, and the amount of generators itself is equal to the size of G plus the order of G, the size of the encoding of the Sullivan algebra is bound from above by a polynomial in the size of the original input instance. Therefore, this construction can be performed in polynomial time and is a Turing reduction. But then the NP-hardness of kCol, in combination with 4, immediately gives us the desired result. ■

2

b

5 Betti Numbers of Elliptic Spaces

In the previous section, we showed that deciding whether a finite-type rational space was elliptic is NP-hard. In this section, we will prove a result that is maybe even more surprising: even if we consider a space which we know to be elliptic (remember: that means that it has finite dimensional cohomology), computing the exact dimensions of the cohomology (also called the Betti numbers) of that space is NP-hard. To prove this, we will utilize a Turing reduction to the subset sum problem.

Definition 37 (Betti numbers) The n-th Betti number b_n of a topological space X is defined as the rank of the n-th cohomology groups $\mathcal{H}^n(X)$. If, as in our case, the cohomology groups are vector spaces, then b_n is the dimension of $\mathcal{H}^n(X)$. □

Let S be an elliptic space. We again encode S using its minimal Sullivan model $(\Lambda \mathrm{Z}, d)$, using the same encoding described in note 4, except for this time, we do not require the length of the differentials to be bounded from above. For our proof, we will construct a Turing reduction from the problem SubsetSum to the following problem:

Problem 5.1 (BETTI$_{\ell\ell}$)
Instance: *An elliptic space* S *and a positive integer* $n \in \mathbb{N}$.
Question: *Is* $b_n := \dim \mathcal{H}^n(\mathrm{S}; \mathbb{Q})$*, the* n*-th Betti number of* S*, positive?*

Note that this will also give us the NP-hardness of the problem of computing the Betti numbers. After all, if an algorithm existed that could compute the n-th Betti number of a space in polynomial time, one could use it to construct a polynomial-time algorithm for BETTI$_{\ell\ell}$ – just calculate the n-th Betti number and check whether it is positive or not! Thus, it follows as an immediate corollary from the NP-hardness of BETTI$_{\ell\ell}$ that computing the Betti numbers is also NP-hard.

Theorem 9 *Let S be an elliptic space. The computation of the Betti numbers* b_n *of S is NP-hard.* □

PROOF Let $P := \{T, N\}$ be an instance of subset sum, where $T := n_1, \ldots, n_m \subset \mathbb{N}$ and $N \in \mathbb{N}$. We first sort T by parity (or congruence class mod 2), so that the odd numbers are in front and the even numbers are in the back. This can be done in polynomial time since both parity checking and sorting are polynomial, and since the class of polynomial functions in closed under addition, it will not increase the complexity of our reduction.

Let $T_{\text{odd}} = \{n_1, \ldots, n_p\}$ be the set of odd numbers in T and let $T_{\text{even}} = \{n_{p+1}, \ldots, n_m\}$ be the set of even numbers in T. We now construct a Sullivan model $(\Lambda Z_T, \partial)$ corresponding to T. To do this, we set

$$Z_T = \langle x_1, \ldots, x_m, y_{p+1}, \ldots, y_m \rangle, \text{ where } \deg x_i = n_i,\ \partial x_i = 0 \text{ and } \partial y_i = x_i^2.$$

The will illuminate the reasoning behind this construction in more depth in a second. But first, we note that the rational space corresponding to this Sullivan model is indeed both of finite type and elliptic. By definition, the free algebra over a graded vector space is of finite type if the graded vector space itself is finitely generated, and this follows immediately from the construction.

To see this, consider that

$$\forall m \in \mathbb{N} \colon \Lambda Z_T^m = \left\langle \bigwedge_{i_k \in I_p} z_{i_k} \right\rangle_{I_p \in \mathfrak{I}_m},$$

where each z_{i_k} is some generator (duplicates possible) and

$$\mathfrak{I}_m := \left\{ I_p \;\middle|\; I_p = \{i_1, \dots, i_p\} \text{ s.t. } \sum_{k=1}^{p} n_{i_k} = m \right\}.$$

Since we only have finitely many generators, we only have finitely many combinations of indices that give us a product element of a certain degree. Therefore, each degree of the Sullivan algebra can only have finitely many generators, and thus our algebra is of finite type.

Furthermore, the cohomology of the Sullivan model is

$$\mathcal{H}^\star (\Lambda Z_T, d) = \left(\Lambda_{i=1}^{p} \langle x_i \rangle\right) \otimes \left(\Lambda_{i=p+1}^{m} \mathcal{H}^\star \left(\langle x_i, y_i \rangle, d\right)\right).$$

By definition, the cohomology is given by quotienting the cocycles with the coboundaries, that is the elements in the kernel of the differential with the one in the image of the differential. Due to the free nature of the construction of a Sullivan algebra, to calculate cohomology, it is sufficient to consider what happens on the generators. For $i \in \{1, \dots, p\}$, the generator x_i is a cocycle but not a coboundary, therefore also an element in cohomology (the same logic applies to any product of odd-degree generators). This is expressed by the left-hand of the tensor-product. The even-degree generators are also cocycles, but their products with themselves are coboundaries (as they are the differential of the y_j with appropriate degree), so they will disappear in cohomology. This is expresses by the left-hand side of the tensor-product. Therefore, the Sullivan algebra is elliptic, and it follows it is indeed the Sullivan model of an elliptic rational space.

Note that an element $x_i x_j x_p$ in cohomology corresponds to a sum $n_i + n_j + n_p$ (by additivity of degree) in the original subset sum instance. Now the ingenuity of our construction reveals itself. Since, by definition of the exterior algebra, each x_i of odd degree is inverse to itself and the product is graded commutative, we can not have "double copies" of such an x_i in a product of elements. The y_j disappear in cohomology, since they aren't cocycles, and take "double copies" the x_j of even degree with them (since the product $x_j x_j$ is a coboundary for even-degree x_j). Thus, the cohomology of the Sullivan algebra has an element of degree N (so a positive n-th Betti number) if and only if there is a subset of T summing up to N. Since the length of the string encoding the Sullivan algebra is obviously bounded by a polynomial in the size of the SubsetSum-instance, this is a Turing reduction and thus we have proven our statement. ∎

Bibliography

References

[Aar17] Scott Aaronson. “P ?= NP”. Survey. University of Texas - Austin, Texas (USA), 2017.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge ; New York: Cambridge University Press, 2009. ISBN: 978-0-521-42426-4.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. “PRIMES Is in P”. In: *Annals of Mathematics* 160.2 (Sept. 2004), pp. 781–793. ISSN: 0003-486X. DOI: `10.4007/annals.2004.160.781`.

[Bae09] John Carlos Baez. *This Week’s Finds 286: Rational Homotopy Theory*. Blog. Dec. 2009.

[Coo71] Stephen A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing - STOC ’71*. Shaker Heights, Ohio, United States: ACM Press, 1971, pp. 151–158. DOI: `10.1145/800157.805047`.

[Die08] Tammo tom Dieck. *Algebraic Topology*. EMS textbooks in mathematics. Zürich: European Mathematical Society, 2008. ISBN: 978-3-03719-048-7.

[FHT01] Yves Félix, Stephen Halperin, and Jean-Claude Thomas. *Rational Homotopy Theory*. Vol. 205. Graduate Texts in Mathematics. New York, NY: Springer New York, 2001. ISBN: 978-1-4612-6516-0 978-1-4613-0105-9. DOI: `10.1007/978-1-4613-0105-9`.

[GHT99] Antonio Gómez-Tato, Stephen Halperin, and Daniel Tanré. “Rational Homotopy Theory for Non-Simply Connected Spaces”. In: *Transactions of the American Mathematical Society* 352.4 (Nov.

1999), pp. 1493–1525. ISSN: 0002-9947, 1088-6850. DOI: 10.1090/S0002-9947-99-02463-0.

[GM13] Phillip Griffiths and John Morgan. *Rational Homotopy Theory and Differential Forms.* Vol. 16. Progress in Mathematics. New York, NY: Springer New York, 2013. ISBN: 978-1-4614-8467-7 978-1-4614-8468-4. DOI: 10.1007/978-1-4614-8468-4.

[Hat02] Allen Hatcher. *Algebraic Topology.* Cambridge ; New York: Cambridge University Press, 2002. ISBN: 978-0-521-79160-1 978-0-521-79540-1.

[Kar72] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations.* Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2003-6 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

[KK04] Stephan Klaus and Matthias Kreck. "A Quick Proof of the Rational Hurewicz Theorem and a Computation of the Rational Homotopy Groups of Spheres". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 136.3 (May 2004), pp. 617–623. ISSN: 0305-0041, 1469-8064. DOI: 10.1017/S0305004103007114.

[LM00] Luis Lechuga and Aniceto Murillo. "Complexity in Rational Homotopy". In: *Topology* 39.1 (Jan. 2000), pp. 89–94. ISSN: 00409383. DOI: 10.1016/S0040-9383(98)00059-7.

[Lyo] David W Lyons. "An Elementary Introduction to the Hopf Fibration". In: (), p. 16.

[May99] J. Peter May. *A Concise Course in Algebraic Topology.* Chicago Lectures in Mathematics. Chicago: University of Chicago Press, 1999. ISBN: 978-0-226-51182-5 978-0-226-51183-2.

[Qui69] Daniel Quillen. "Rational Homotopy Theory". In: *The Annals of Mathematics* 90.2 (Sept. 1969), p. 205. ISSN: 0003486X. DOI: 10.2307/1970725.

[Sip12] Michael Sipser. *Introduction to the Theory of Computation.* 3rd Ed. Boston, MA: Course Technology Cengage Learning, 2012. ISBN: 978-1-133-18779-0.

[Sul77] Dennis Sullivan. “Infinitesimal Computations in Topology”. In: *Publications mathématiques de l’IHÉS* 47.1 (Dec. 1977), pp. 269–331. ISSN: 0073-8301, 1618-1913. DOI: `10.1007/BF02684341`.

[Wag+ 1] Dorothea Wagner et al. “Vorlesungsskript Theoretische Grundlagen der Informatik”. Lecture Notes. Karlsruhe Institute of Technology - Karlsruhe, Germany, WS 11/12.

Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Ort, Datum