ON A TRANSFER THEOREM FOR THE $\mathcal{P} \neq \mathcal{NP}$ CONJECTURE

GREGORIO MALAJOVICH

Abstract. A model of computation is defined over the algebraic numbers and over number fields. This model is non-uniform, and the cost of operations depends on the height of the operands and on the degree of the extension of the rational defined by those operands.

A transfer theorem for the $\mathcal{P} \neq \mathcal{NP}$ Conjecture is proved, namely: $\mathcal{P} \neq \mathcal{NP}$ in this model over the real algebraic numbers if and only if $\mathcal{P} \neq \mathcal{NP}$ in the classical setting.

Contents

Date: December 14, 1998.

1. Introduction

A model of computation over a ring was introduced by Blum, Shub and Smale in [BSS89]. This model preserves the main features of the classical theory of Computation. Turing computability appears as a particular case (computation over $\mathbb{F}_2 = \mathbb{Z} \mod 2\mathbb{Z}$).

One of the main objectives of this theory was to generalize the $\mathcal{P} \neq \mathcal{NP}$ conjecture by relating it to more traditional mathematical ideas and problems.

A model of computation in the sense of [BSS89] is given essentially by:

1. A ring R with unity and without zero divisors.
2. A subset o of R. Machines will be allowed to decide (to branch on) $x \in o$. Natural choices are $o = \{0\}$ or, when R is ordered, the positive numbers $o = \{x \in R : x > 0\}$.
3. A cost function $c : R^\infty \to \mathbb{R}$. Here, R^∞ stands for the disjoint union of R^k, where $k \in \mathbb{N}$. (i.e., R^∞ is the set of finite lists of elements in R).

We will write for short: $(R, =, c)$ and $(R, >, c)$ for the models of computation over a ring R, with cost function c and branching on $o = \{0\}$ or on $o = \{x \in R : x > 0\}$, respectively.

In this paper, the 'size' of some $x \in R^k \subset R^\infty$ is $kc(x)$, and the 'cost' of a computation is the size of the output, **plus** the number of ring operations times $\max c(s)$ where s ranges over the operands of all the ring operations performed. Concepts of 'polynomial time' and 'nondeterministic polynomial time' arise naturally.

The main reference for computations over a ring is [BCSS98]. In the notation above, the main examples described in [BCSS98] are $(\mathbb{F}_2, =, 1)$, $(\mathbb{Z}, >, x \mapsto \lceil \log |x| \rceil + 1)$, $(\mathbb{R}, >, 1)$, $(\mathbb{C}, =, 1)$.

Given a model of computation, there is a $\mathcal{P} \neq \mathcal{NP}$ conjecture associated to it. A 'transfer' theorem is a theorem relating the $\mathcal{P} \neq \mathcal{NP}$ conjecture in different models.

A few 'transfer' theorems are known for unit-cost models. The conjecture $\mathcal{P} \neq \mathcal{NP}$ is equivalent over the ring of complex numbers $(\mathbb{C}, =, 1)$, over the algebraic closure of the rationals $(\mathbb{Q}^{\mathrm{a}}, =, 1)$, or over any extension in-between [BCSS96]. See also [BCSS98, CG97, CK95, Koi94, Koi97, Mic94]. For a few consequences related to the transfer principle see [MM99].

In most literature on Computability over a ring, the cost function (or the cost of an arithmetic operation over the ring R (complex, real, ...) is usually fixed as $c \equiv 1$. Uniform cost is related to consequences such as the existence of 'wisdom numbers' over the reals, encoding an infinite sequence of 0's and 1's. Or such as the fact that $\mathcal{P} \neq \mathcal{NP}$ is true over $(K, =, 1)$ for all infinite field K not algebraically closed (See [BCSS98] Chapter 7 Theorem 9).

In this paper, a 'weighted' model of computation is introduced. The cost function will be defined as the 'weight', a measure of the complexity of the operands. The definition of weight will encode some natural number-theoretical invariants, such as the height of algebraic numbers or the degree of an extension.

More precisely, if $x_1, \cdots, x_k$ are algebraic over $\mathbb{Q}$, then the weight of x is defined as

$$w(x) = [\mathbb{Q}[x_1, \cdots x_k] : \mathbb{Q}] \, (1 + \lceil \log_2 H(1 : x_1 : \cdots : x_k) \rceil)$$

where H is the projective, multiplicative height function as defined in [Lan97] and [Sil86].

In the particular case $x \in \mathbb{Z}$, the model $(\mathbb{Z}, >, w)$ is precisely the model of computation over the integers 'with height' in [BSS89] or 'with respect to logarithmic cost' in [BCSS98]. This model is known to be equivalent to the Turing model, up to polynomial time.

Under the weighted model of computation, we will obtain a few conditions for the equivalence of the $\mathcal{P} \neq \mathcal{NP}$ conjecture over two rings; we will use those facts to prove that

Theorem A. *Let K be a real number field, and let $\mathbb{Q}^{\mathrm{ra}}$ be the real algebraic closure of rationals. The following are equivalent:*

1. *$\mathcal{P} \neq \mathcal{NP}$ over the Turing model*
2. *$\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Z}, >, w)$.*

3. *$\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Q}, >, w)$.*
4. *$\mathcal{P} \neq \mathcal{NP}$ over $(K, >, w)$.*
5. *$\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Q}^{\mathrm{ra}}, >, w)$.*

In view of the τ-conjecture [BCSS98, Sma98] it is interesting to ask what happens if one removes 'order' from the models above. Is deciding $x > 0$ hard over $(\mathbb{Q}^{\mathrm{ra}}, =, w)$? Actually, $\{x > 0\}$ will turn out to be undecidable, and the following theorem will follow:

Theorem B. *$\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Q}^{\mathrm{a}}, =, w)$ and over $(\mathbb{Q}^{\mathrm{ra}}, =, w)$*

The proof of Theorem A requires some results on the complexity of 'effective' diophantine approximation. Algebraic numbers can be approximated by rationals generated by a machine over $(\mathbb{Z}, >, w)$ (or equivalently, a Turing machine). This machine is 'universal' in the sense that it can be used to approximate any algebraic number x, given a suitable description y of x.

When given $k \in \mathbb{N}$ and $y \in \mathbb{Z}^{\infty}$, where y is a suitable description of x, that machine will produce $a, b \in \mathbb{Z}$ and $c \in \mathbb{N}$ such that

$$|x - (a + ib)2^{-c}| < 2^{-k}$$

The 'suitable information' on x may be its minimal polynomial and an approximation of x.

Theorem C. *There are a machine M over $(\mathbb{Z}, >, w)$ and a function $y : \mathbb{Q}^{\mathrm{a}} \to \mathbb{Z}^{\infty}$ with the following properties:*

If x is an algebraic number, then M with input $k, y(x)$ will output $a, b \in \mathbb{Z}$ and $c \in \mathbb{N}$ such that

$$|x - (a + ib)2^{-c}| < 2^{-k}$$

The running time for $k > 5$ is $k \log k \; w(x) \log w(x) \; \mathrm{Poly}(\deg[\mathbb{Q}[x] : \mathbb{Q}])$. Also, $w(x) \leq \mathrm{Bitsize}(y)$. Moreover, if x is real, then b is always zero in $M(y(x), k)$.

The running time above is better than the $O(k^2)$ in the proof of Theorem (1.4.7)(a) in [Lov86] or in Algorithm 2 Step 3 in [Loo83]. This theorem gives also an implicit bound for the bit-size of $y(x)$. Given $y(x)$, this can be used to decide if an algebraic number is > 0, for instance.

Theorem C makes sense in the Turing model of computation. However, it also makes sense in the weighted model over $\mathbb{Q}^{\mathrm{ra}}$, with or without order. In that model, Turing computations can be simulated by using sequences of 0's and 1's. It is possible to compute 'mixed' expressions, like $f(x)$ where f is an integer polynomial and $x \in \mathbb{Q}^{\mathrm{ra}}$.

Theorem D. *The restriction $y : \mathbb{Q}^{\mathrm{ra}} \to \mathbb{Z}^{\infty}$ of the function y of Theorem D can be computed in polynomial time over $(\mathbb{Q}^{\mathrm{ra}}, >, w)$.*

Unlike Theorem (1.4.7)(b) in [Lov86], the machine in Theorem D does not know *a priori* the weight of the input. Theorem D implies that it is possible to decide in polynomial time whether $x \in \mathbb{Q}$. The proof relies heavily on the Lenstra-Lenstra- Lovasz algorithm [LLL82].

A non-uniform model of computation over number fields was introduced by Adler and Beling [AB94] in the realm of linear programming. This model was used to bound the running time of a simplex algorithm, with input defined over the algebraic numbers. The running time depends (among other parameters) on the degree of the extension and on a measure related to the Height of the input.

I wish to thank Ilan Adler, Lenore Blum, Felipe Cucker, Jim Demmel, Benjamin Diament, Steve Smale, Felipe Voloch, Zhu Hui for their comments and suggestions.

Part of this work was supported by CNPq (Brasil). Another part of this work was done while visiting Mathematical Sciences Research Institute, Berkeley, CA.

2. Heights and Weights

The *height function* is an invariant of $\mathbb{Q}^{a}$, the algebraic closure of the rationals, under automorphisms. It extends naturally to a projective invariant of $\mathbb{P}^n(\mathbb{Q}^{a})$.

The motivation for the introduction of the height function comes from two (related) goals. One of them is counting rational points (or algebraic points) in algebraic varieties. The other goal is bounding above or below algebraic expressions.

The main standard references are Lang [Lan86, Lan97] and Silverman [Sil86]. This section is intended to be an introduction for readers not acquainted with the technicalities of the construction of the height function; those readers coming from a number-theoretical background may skip most of this section and proceed to Subsection 2.6

2.1. Heights in $\mathbb{Z}$ and $\mathbb{Q}$.

Let $x \in \mathbb{Z}$. We may define $H(x) = \max(|x|, 1)$, where $|\cdot|$ is the ordinary absolute value. The following properties are obvious:

(H1) $H(0) = H(1) = H(-1) = 1$
(H2) $H(xy) \leq H(x)H(y)$
(H3) $H(x+y) \leq 2H(x)H(y)$

The reason for writing (H3) instead of $H(x+y) \leq H(x) + H(y)$ will become clear below.

The function H may be extended to $\mathbb{Z}^n$ by putting:

$$H(\vec{x}) = \max\left(H(x_1), \dots, H(x_n)\right)$$

It becomes a projective invariant if one sets:

$$H(x_0 : \cdots : x_n) = \max\left(H(\lambda x_0), \cdots, H(\lambda x_n)\right)$$

where $\lambda \in \mathbb{Q}$ is such that all the λx_i are relatively prime integers.

Note that $H(\vec{x}) = H(1 : x_1 : \cdots : x_n)$, so there is a close relationship between those two formulations. In the sequel, we will prefer the projective one.

We can extend the height function to rational numbers by stating:

$$H\left(\frac{p}{q}\right) = H(p : q)$$

If p and q are relatively prime, this last expression is also $\max(|p|, |q|)$. In case $q_1, q_2 \in \mathbb{Z}$ are relatively prime, we have also:

$$\begin{aligned} H\left(\frac{1}{q_1} + \frac{1}{q_2}\right) &= H\left(\frac{q_1+q_2}{q_1 q_2}\right) \\ &= \max\left(|q_1+q_2|, |q_1 q_2|\right) \\ &= |q_1||q_2| \\ &= H\left(\frac{1}{q_1}\right) H\left(\frac{1}{q_2}\right) \end{aligned}$$

This is why the height of the sum needs to be bounded in terms of the product of the heights. Property (H3) is sharp, in the sense that $H(1+1) = 2 = 2H(1)H(1)$.

It is natural to further generalize the height function to $\mathbb{P}^n(\mathbb{Q})$ by:

$$H(x_0 : \cdots : x_n) = \max|\lambda x_0|, \cdots, |\lambda x_n|$$

and λ is chosen so that the λx_i are integers and relatively prime.

This is **not** the same as $\max H(x_i)$ and should not be. Indeed, the objective of introducing the Height function was to obtain reasonable estimates for expressions.

Example 1. Consider the function: $f(\vec{x}) = \sum x_i$. For instance,

$$f(\frac{1}{2}, -\frac{1}{3}, -\frac{1}{7}, \cdots) = \frac{1}{2 \times 3 \times 7 \cdots}$$

so $H(f(\vec{x})) \leq (\max H(x_i))^n$ is almost sharp ! However, we have also the more elegant bound:

$$H(f(\vec{x})) \leq nH(\vec{x})$$

The above definition of height in $\mathbb{P}^n(\mathbb{Q})$ allows us to bound many integer or algebraic expressions:

Example 2. Let $a(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_0$ be a polynomial with rational coefficients, $a(x) \not\equiv 0$. The following bound is due to Cauchy: if $\alpha \neq 0$ is a root of a, then:

$$|\alpha| > \frac{1}{1 + H(a)}$$

Indeed, we can assume without loss of generality (and without changing the height) that the a_i are relatively prime integers. Also, we can assume that $a_0 \neq 0$. If $|\alpha| < 1$, $a(\alpha) = 0$ implies:

$$\begin{array}{rcl} |a_0| & \leq & |a_1||\alpha| + |a_2||\alpha|^2 + \cdots + |a_d||\alpha|^d \\ & \leq & |\alpha| H(a)(1 + |\alpha| + \cdots + |\alpha|^{d-1}) \\ & < & H(a) \dfrac{|\alpha|}{1 - |\alpha|} \end{array}$$

Hence, $|a_0| < (H(a) + |a_0|)|\alpha|$ and hence,

$$|\alpha| > \frac{1}{\frac{H(a)}{|a_0|} + 1} > \frac{1}{H(a) + 1}$$

The height function H can be extended to the algebraic closure $\mathbb{Q}^{\mathrm{a}}$ of the rationals. The estimate in Example 2 can be generalized to polynomials with algebraic coefficients (the actual statement is a little more subtle, see Theorem 5 below.)

Also, the height of a minimal polynomial of α can be bounded in terms of $H(\alpha)$ (see Corollary 4 below). In that sense, the height function will capture the 'complexity' of algebraic numbers.

2.2. **Absolute values.** In order to generalize the Height into $\mathbb{Q}^{\mathrm{a}}$, a few definitions are needed. First of all, let K be a finite algebraic extension of $\mathbb{Q}$. An absolute value on K is a function

$$| \cdot |_\nu : K \rightarrow \mathbb{R}_+$$

such that:

(A1) $|x|_\nu = 0 \Leftrightarrow x = 0$.
(A2) $|xy|_\nu = |x|_\nu |y|_\nu$
(A3) $|x + y|_\nu \leq |x|_\nu + |y|_\nu$

If $| \cdot |_\nu$ satisfies the condition

(A3') $|x + y|_\nu \leq \max(|x|_\nu, |y|_\nu)$

then $| \cdot |_\nu$ is called a *non-archimedian* absolute value, or *valuation*. Otherwise, $| \cdot |_\nu$ is said to be archimedian.

In the sequel, we will need to consider the set of all possible valuations of K, modulo the following equivalence: $| \cdot |_\mu$ and $| \cdot |_\nu$ are equivalent if and only if there are a, $b > 0$ such that

$$\begin{array}{rcl} |x|_\nu & \leq & a|x|_\mu^b \\ |x|_\mu & \leq & a|x|_\nu^b \end{array}$$

Each of the classes of equivalence above will make K into a metric space, with distance function $d(x, y) = |x - y|_\nu$. Equivalent absolute values give rise to equivalent metric spaces. We will denote by M_K a (carefully chosen) system of representatives of each class of absolute values. There is precisely one class of archimedian absolute value associated to every automorphism σ of K over $\mathbb{Q}$, namely:

$$|x|_\sigma = |\sigma(x)|$$

where $|\cdot|$ is the ordinary absolute value. The case of non-archimedian absolute values is more subtle:

Example 3. Let $K = \mathbb{Q}$. It is a famous theorem by Ostrowski (See [Cas67]) that the absolute values of $\mathbb{Q}$ are precisely, up to equivalence:

- The usual (archimedian) absolute value $|x|$.
- For each prime number $r \in \mathbb{N}$, a valuation $|r^k \frac{p}{q}|_r = r^{-k}$, where it is assumed that r, p, q are relatively prime integers.

Furthermore, we have the *product formula*

$$\prod_{\nu \in M_{\mathbb{Q}}} |x|_\nu = 1$$

where $M_{\mathbb{Q}}$ is the set of absolute values of $\mathbb{Q}$ defined above. A generalization of this product formula will hold in the more general setting of number fields.

Example 4. Let $K = \mathbb{Q}[\sqrt{2}]$. In this case, it is easy to show that the ring of integers of K (i.e., the set of $x \in K$ s.t. $f(x) = 0$ for some monic polynomial f with coefficients in $\mathbb{Z}$) is given by

$$\mathfrak{o}_K = \{a + b\sqrt{2} : a, b \in \mathbb{Z}\}$$

The *norm* $N(a + b\sqrt{2})$ is defined by:

$$N(a + b\sqrt{2}) = \prod_{\sigma \in \mathrm{Gal}(K:\mathbb{Q})} \sigma(a + b\sqrt{2}) = a^2 - 2b^2$$

where σ ranges over the automorphisms of K.

The units u of the ring of integers of K are characterized by the equation: $N(u) = \pm 1$. Those can be proved to be precisely the expressions of the form:

$$\pm(1 + \sqrt{2})^r \ , \ r \in \mathbb{Z}$$

(This follows from a theorem by Dirichlet. See [BS67] Section II.4 Theorem 5, and ibid. Table 1).

The ring of integers of K is an Euclidian domain (hint: Euclidian Algorithm for the division, with 'degree' function $|N(.)|$), hence it admits a unique prime factorization

$$x = u p_1^{a_1} \cdots p_s^{a_s}$$

where $p_1, \cdots, p_s$ are prime numbers in $\mathfrak{o}_K$ and u is a unit.

If we are given $x \in \mathfrak{o}_K$, we can compute the prime number factorization of $N(x)$ over $\mathbb{Z}$:

$$N(x) = \pm q_1^{b_1} q_2^{b_2} \cdots q_s^{b_s}$$

Then for each of the q_i's, two cases may arise:

Case 1: q_i is prime over $\mathfrak{o}_K$. This will be the case if the equation $N(a + b\sqrt{2}) = q_i$ has no solution. For instance, it is easy to check by enumeration that $N(a + b\sqrt{2}) \equiv 0 \mod 3$ has no solution, hence 3 is prime in $\mathfrak{o}_K$. In this case, 3 divides x. Also, $|\cdot|_3$ is a valuation on K and $|x|_3 = \sqrt{|N(x)|_3}$.

Case 2: The prime number q_i factors over $\mathfrak{o}_K$, so that $\pm q_i = p_i p_i'$. Example: $-7 = (1+2\sqrt{2})(1-2\sqrt{2})$ so that p_i or p_i' divide x. There will be valuations associated to p_i and p_i', normalized such that:

$$|p_i^a \frac{q}{r}|_{p_i} = p_i^{-a}$$

for relatively prime p_i, q, r.

We have also two absolute values at infinity,

$$\begin{aligned} |a+b\sqrt{2}|_{\sigma_1} &= |a+b\sqrt{2}| \\ |a+b\sqrt{2}|_{\sigma_2} &= |a-b\sqrt{2}| \end{aligned}$$

where σ_1 is the identity and $\sigma_2 : a+b\sqrt{2} \mapsto a-b\sqrt{2}$.

Example 5. Let $K = \mathbb{Q}[\sqrt{-5}]$. The ring of integers of K is given by $\mathfrak{o}_K = \{a+b\sqrt{-5}, a, b, \in \mathbb{Z}\}$ and its units are ± 1.

This example is interesting because $\mathfrak{o}_K$ is not an Euclidian Domain. The closest we can get from prime number factorization is the primary decomposition into ideals. For instance, the ideal (3) splits in:

$$(3) = \mathfrak{p}_1 \mathfrak{p}_2$$

with $\mathfrak{p}_1 = (3\sqrt{-5}, 1+\sqrt{-5})$ and $\mathfrak{p}_2 = (3\sqrt{-5}, 1-\sqrt{-5})$. Those are prime ideals (Proof: $\mathfrak{o}_K \mod \mathfrak{p}_1$ has three equivalence classes $\mathfrak{p}_1$, $\mathfrak{p}_1 + \sqrt{-5}$ and $\mathfrak{p}_1 + 2\sqrt{-5}$. By writing down all the possibilities for $x \mod \mathfrak{p}_1$ and $y \mod \mathfrak{p}_1$, we conclude that $xy \equiv 0 \mod \mathfrak{p}_1$ implies that x or $y \in \mathfrak{p}_1$. Same proof for $\mathfrak{p}_2$).

To each prime ideal $\mathfrak{p}$, we associate the absolute value

$$|x|_{\mathfrak{p}} = (\mathrm{N}\mathfrak{p})^{-a}$$

where $\mathrm{N}\mathfrak{p}$ is the number of elements of the field $K/\mathfrak{p}$ and $(x) = (\mathfrak{p})^a(\mathfrak{p}_2)^{a_2}\dots$ is the prime ideal decomposition of (x) in $\mathfrak{o}_K$. This definition can be extended to K by setting $|\frac{y}{z}|_{\mathfrak{p}} = |y|_{\mathfrak{p}} - |z|_{\mathfrak{p}}$.

There are also two archimedian valuations, $|x|$ and $|\overline{x}|$.

General case: Let K be a number field (i.e. a finite extension of $\mathbb{Q}$). Then its ring of integers $\mathfrak{o}_K$ is a Dedekind domain (See [AM69], Chapter 9 for definitions and theory on Dedekind domains. This last fact is stated as Theorem 9.5). It is a classical result that :

Theorem 1 (Corollary 9.4 in [AM69])**.** *In a Dedekind domain, every non-zero ideal has a unique factorization as a product of prime ideals.*

This theorem was proved by Dedekind (c. 1871) in its original formulation (number fields), and later generalized to Dedekind domains by Noether (1921, 1927).

Most modern textbooks provide just an existential proof of that factorization. An effective algorithm to compute the prime ideal decomposition of a principal ideal in a number field appears in classical textbooks such as [BS67]. More recent algorithms are discussed in [Coh93].

A non-archimedian valuation $|\cdot|_{\mathfrak{p}}$ is associated to each prime ideal $\mathfrak{p}$. Once we know that $(x) = \mathfrak{p}_1^{a_1} \cdots \mathfrak{p}_s^{a_s}$, it is possible to compute the $|x|_{\mathfrak{p}_i}$.

2.3. **The product formula.** Given a finite extension K of $\mathbb{Q}$, let M_K be the set of absolute values constructed as in the examples above:

1. The archimedian absolute values $|\cdot|_\sigma$ associated to all $\sigma \in \mathrm{Gal}_{\mathbb{Q}} K$
2. The valuations $|\cdot|_{\mathfrak{p}}$ associated to the prime ideals $\mathfrak{p}$ of $\mathfrak{o}_K$.

For each of those valuations $|\cdot|_\nu$, we can define a new metric space K_ν as the completion of K by the metric associated to $|\cdot|_\nu$. This completion is a field, it is indeed a (possibly infinite) field extension of K.

Example 6. The archimedian completion of the rationals is $\mathbb{Q}_\infty = \mathbb{R}$. The non-archimedian completion associated to prime number p is given by all sequences of the form

$$(q_i) = p^n(a_0 + pa_1 + p^2a_2 + \cdots p^i a_i + \cdots)$$

for $n \in \mathbb{Z}$ and $a_i \in \{0; \cdots ; p-1\}$.

Example 7. If $K = \mathbb{Q}[\sqrt{2}]$ then $K_\infty = \mathbb{R}$. If $K = \mathbb{Q}[i]$, we have two archimedian valuations. For any of those, $K_\infty = \mathbb{C}$.

If L is a finite extension of K and $\nu \in M_K$, then ν extends to at least one absolute value ν_0 of M_L.

The completion L_{ν_0} of L is clearly an extension of the completion K_ν of K. This extension turns to be finite. Indeed, let α be a primitive element of L with respect to K. A sequence (l_i) with values in L can always be written as $l_i = \sum k_{ij}\alpha^j$, $k_{ij} \in K$.

Definition 1. The *local degree* n_{ν_0} of the extension $L : K$ in ν_0 is the degree of the extension $[L_{\nu_0} : K_\nu]$.

The local degrees are the multiplicity values that make the product formula true. Indeed,

Theorem 2. *Let K be a number field and let M_K be constructed as above. Then for any $x \in K$,*

$$\prod_{\nu \in M_K} |x|_\nu^{n_\nu} = 1$$

This fact comes from the formula

Theorem 3. *Let L and K be number fields, L a finite extension of K. Let $\nu \in M_K$. Then*

$$\sum_{\nu_0 \text{ extends } \nu} n_{\nu_0} = [L : K]$$

For proofs, refer to [Lan97].

Example 8. Let $K = \mathbb{Q}[\sqrt{2}]$. Let $|\cdot|_{\sigma_1}$ and $|\cdot|_{\sigma_2}$ be the archimedian absolute values. Then

$$n_{\sigma_1} = [K_{\sigma_1} : \mathbb{Q}_{\sigma_1}] = [\mathbb{R} : \mathbb{R}] = 1$$

and same for n_{σ_2}.

As we saw, 3 is a prime in K. $\mathbb{Q}_3$ is the set of expressions $3^a(p_0 + 3p_1 + 3^2p_2 + \cdots)$ with p_i in $\{0; 1; 2\}$. On the other hand, K_3 is the set of expressions $3^a(r_0 + 3r_1 + \cdots)$ with r_i defined modulo 3. Possible values are $0, 1, 2, \sqrt{2}, 1 + \sqrt{2}, 2 + \sqrt{2}$. So $n_3 = 2$.

Now, let us compute $n_{1+2\sqrt{2}}$. As usual, $\mathbb{Q}_7$ is the set of expressions $7^a(p_0 + 7p_1 + 7^2p_2 + \cdots)$ with p_i in $\{0; 1; \cdots ; 7\}$. On the other hand, $K_{1+2\sqrt{2}}$ contains the expressions $(1 + 2\sqrt{2})^a(r_0 + (1 + 2\sqrt{2})r_1 + \cdots)$. Since the r_i are defined modulo $1 + 2\sqrt{2}$, we have only 7 possible values (recall that $7 \equiv 0$ and that $1 \equiv 2\sqrt{2}$). Therefore $K_{1+2\sqrt{2}}$ is indeed $\mathbb{Q}_7$ and $n_{1+2\sqrt{2}} = 1$.

2.4. **Heights.** Let $x = (x_0 : \cdots : x_n) \in \mathbb{P}^n(K)$, where x is a number field. If $x_0, \cdots , x_n \in K$ we define relative height as:

$$H_K(x) = \prod_{\nu \in M_K} \max_i |x_i|_\nu^{n_\nu}$$

Because of the product formula, this definition is invariant with respect to multiplication of x by any $\lambda \in K$:

$$\begin{aligned}
H_K(\lambda x) &= \prod_{\nu \in M_K} \max_i |\lambda x_i|_\nu^{n_\nu} a \\
&= \prod_{\nu \in M_K} \max_i |\lambda|_\nu^{n_\nu} |x_i|_\nu^{n_\nu} \\
&= \left(\prod_{\nu \in M_K} |\lambda|_\nu^{n_\nu} \right) \left(\prod_{\nu \in M_K} |x_i|_\nu^{n_\nu} \right) \\
&= \left(\prod_{\nu \in M_K} |x_i|_\nu^{n_\nu} \right) \\
&= H_K(x)
\end{aligned}$$

However, this definition is still not invariant under multiplication by $\lambda \in \mathbb{Q}^a$, $\lambda \notin K$. In order to obtain an invariant definition in $\mathbb{Q}^a$, it is customary to set:

$$H(x) = H_K(x)^{\frac{1}{[K:\mathbb{Q}]}}$$

In the particular case $\mathfrak{o}_K$ is an Euclidian domain, we can choose representatives $x_0, \cdots, x_n$ of $(x_0 : \cdots : x_n)$ that belong to $\mathfrak{o}_K$ and are relatively prime. Afterwards, only the Archimedian absolute values of x can be different of 1.

2.5. **Properties of Heights.** The basic properties we stated in section 2.1 can be generalized to a convenient bound on $H(f(P))$ in terms of $H(f)$ and $H(P)$, where f is a polynomial or a system of polynomials. That bound is stated below as Theorem 4. This statement generalizes Theorem 5.6 in [Sil86]. Theorem 4 seems well-known to people working in number theory. However, the precise statement does not seem to be written down in the literature, so it will be stated and proved below. A lot of useful properties of heights will then be stated as corollaries.

Before giving the precise statement, it is convenient to clarify some notations: we will reserve the notation $H(x)$ for the projective height of $x \in \mathbb{P}^n(\mathbb{Q}^a)$, as we defined in section 2.1. If we want to write down the affine height of some $y \in (\mathbb{Q}^a)^n$ we will write instead: $H(y, 1)$.

This convention can be extended to polynomials or systems of polynomials. If $f = (f_1, \cdots, f_k)$ is a system of multivariate polynomials, then its height $H(f)$ is the height of the projective point $(\cdots : f_{iI} : \cdots) \in \mathbb{P}^{S(f)-1}(\mathbb{Q}^a)$. Above, $S(f) = S(f_1) + \cdots + S(f_k)$ where $S(f_i)$ is the number of non-zero coefficients of f_i. Of course, when we need the 'affine' height of f, we will write $H(f, 1)$.

With those conventions, we may state the

Theorem 4. *Let*

$$\begin{array}{rclcl} F = (F_0, \cdots, F_m): & \mathbb{C}^{n_1} \times \mathbb{C}^{n_2} \times \cdots \times \mathbb{C}^{n_k} & \to & \mathbb{C}^{m+1} \\ & P^1, \cdots, P^k & \mapsto & F(P^1, \cdots, P^k) \end{array}$$

be a system of multi-homogeneous polynomials with algebraic coefficients, where each F_i has degree d_j in variables P^j. Let the P^j be algebraic. Then

$$H(F(P)) \leq (\max S(f_i)) H(F) H(P^1)^{d_1} \cdots H(P^k)^{d_k}$$

In the particular case $k = 1$, F is a morphism from $\mathbb{P}^n(\mathbb{Q}^a)$ into $\mathbb{P}^m(\mathbb{Q}^a)$ and Theorem 4 is Theorem 5.7 in [Sil86]. In the full statement, Theorem 4 generalizes that fact to morphisms from $\mathbb{P}^{n_1}(\mathbb{Q}^a) \times \cdots \times \mathbb{P}^{n_k}(\mathbb{Q}^a)$ into $\mathbb{P}^m(\mathbb{Q}^a)$.

The proof below is a modification of the proof of Theorem 5.6 in [Sil86].

Proof of Theorem 4. Let $K \subset \mathbb{Q}^a$ be a finite extension of $\mathbb{Q}$, containing all the coefficients of F and all the coordinates of P. We have:

$$\begin{aligned} H_K(F(P)) &= \prod_{\nu \in M_K} \max_i |F_i(P)|_\nu^{n_\nu} \text{ by definition} \\ &= \prod_{\nu \in M_K} \max_i |\sum_J F_{iJ} (P^1)^{J_1} \cdots (P^k)^{J_k}|_\nu^{n_\nu} \end{aligned}$$

where the J_k are multi-indices, $|J_j| = d_j$ and $J = (J_1; \cdots; J_k)$. Let $\epsilon(\nu)$ be equal to 1 when ν is archimedian, and to 0 when ν is a valuation. The triangular inequality implies now: $|\sum_{1 \leq i \leq n} a_i|_\nu \leq n^{\epsilon(\nu)} \max |a_i|_\nu$. Hence.

$$\begin{aligned} H_K(F(P)) &\leq \prod_{\nu \in M_K} S^{\epsilon(\nu) n_\nu} \max_{i,J} |F_{iJ}(P^1)^{J_1} \cdots (P^k)^{J_k}|_\nu^{n_\nu} \\ &\leq S^{\sum \epsilon(\nu) n_\nu} \prod_{\nu \in M_K} \max_{i,J} |F_{iJ}(P^1)^{J_1} \cdots (P^k)^{J_k}|_\nu^{n_\nu} \\ &\leq S^{\deg[K:\mathbb{Q}]} \prod_{\nu \in M_K} \max_{i,J} |F_{iJ}(P^1)^{J_1} \cdots (P^k)^{J_k}|_\nu^{n_\nu} \end{aligned}$$

using the fact $\sum_\nu \epsilon(\nu) n_\nu = \sum_{\nu \text{ archimedian}} n_\nu = \deg[K:\mathbb{Q}]$.

Now, since the degree of $(P^j)^{J_j}$ is precisely d_j, we can write:

$$|(P^j)^J|_\nu \leq \max_l \left(|(P^j_l)|_\nu \right)^{|J|}$$

We can use homogeneity to bound:

$$\begin{array}{rcl} H_K(F(P)) & \leq & S^{\deg[K:\mathbb{Q}]} \prod_{\nu \in M_K} \max_{i,J} |F_{iJ}|_\nu^{n_\nu} \max_l |(P^1_l)|_\nu^{d_1 n_\nu} \cdots \max_l |(P^k_l)|_\nu^{d_k n_\nu} \\ & \leq & S^{\deg[K:\mathbb{Q}]} \left(\prod_{\nu \in M_K} \max_{i,J} |F_{iJ}|_\nu^{n_\nu} \right) \left(\prod_{\nu \in M_K} \max_l \left(|(P^1_l)|_\nu \right)^{d_1 n_\nu} \right) \cdots \\ & & \cdots \left(\prod_{\nu \in M_K} \max_l \left(|(P^k_l)|_\nu \right)^{d_k n_\nu} \right) \\ & = & S^{\deg[K:\mathbb{Q}]} H_K(F) H_K(P^1)^{d_1} \cdots H_K(P^k)^{d_k} \end{array}$$

So, after taking the $\deg[K:\mathbb{Q}]$-th root, we obtain:

$$H(F(P)) \leq S H(F) H(P^1)^{d_1} \cdots H(P^k)^{d_k}$$

□

Also, we may want an affine version of Theorem 4:

Corollary 1. *Let*

$$\begin{array}{rcl} G = (G_1, \cdots, G_m) : \quad \mathbb{C}^{n_1} \times \mathbb{C}^{n_2} \times \cdots \times \mathbb{C}^{n_k} & \to & \mathbb{C}^m \\ Q^1, \cdots, Q^k & \mapsto & G(Q^1, \cdots, Q^k) \end{array}$$

be a system of polynomials with algebraic coefficients, where each G_i has degree at most d_j in variables Q^j. Let the Q^j be algebraic. Then

$$H(G(Q), 1) \leq (\max S(G_i)) H(G, 1) H(Q^1, 1)^{d_1} \cdots H(Q^k, 1)^{d_k}$$

Proof of Corollary 1. Introduce homogenizing variables $q_1, \cdots, q_k$ so that for each G_i, its homogenization $F_i(Q^1, q_1; \cdots; Q^k, q_k)$ has degree precisely d_j in variables (Q^j, q_j). Also, introduce the polynomial $F_0 = q_1^{d_1} \cdots q_k^{d_k}$. Then we have $H(F) = H(G, 1)$. Moreover, $\max S(F_i) = \max S(G_i)$. Specialize $q_1 = \cdots = q_k = 1$, and apply Theorem 4 to obtain

$$\begin{array}{rcl} H(G(Q), 1) & \leq & H(F(P)) \\ & \leq & (\max S(f_i)) H(F) H(P^1)^{d_1} \cdots H(P^k)^{d_k} \\ & \leq & (\max S(G_i)) H(G, 1) H(Q^1, 1)^{d_1} \cdots H(Q^k, 1)^{d_k} \end{array}$$

□

Corollary 2. *Let $x_1, \ldots, x_k \in \mathbb{Q}^a$. Then*

$$H(x_1 x_2 \cdots x_k) \leq H(x_1) H(x_2) \cdots H(x_k)$$

Corollary 3. *Let $x_1, \ldots, x_k \in \mathbb{Q}^a$. Then*

$$H(x_1 + x_2 + \cdots + x_k) \leq n H(x_1) H(x_2) \cdots H(x_k)$$

Corollary 4. *Let $\alpha \in \mathbb{Q}^a$ and let $p(x)$ be the minimal polynomial of α over $\mathbb{Q}$. Then*

$$H(p) = H(p, 1) \leq (2H(\alpha))^{[\mathbb{Q}[\alpha]:\mathbb{Q}]}$$

Proof. By definition p is an irreducible polynomial and its coefficients are relatively prime integers hence $H(P) = H(P, 1)$.

We will use the following fact (Lemma 5.10 in [Sil86]): If $\bar{\alpha}$ is a conjugate of α by the Galois group of $\mathbb{Q}^a$ over $\mathbb{Q}$, then $H(\bar{\alpha}) = H(\alpha)$.

In our case, the roots of the minimal polynomial $p(x)$ are all the conjugates of α by the Galois group of $\mathbb{Q}[\alpha]$ over $\mathbb{Q}$. Hence they have the same height as α.

Consider now the polynomial system:

$$\sigma(x_1, \cdots, x_n) = \begin{bmatrix} \sigma_0(x_1, \cdots, x_n) \\ \sigma_1(x_1, \cdots, x_n) \\ \vdots \\ \sigma_n(x_1, \cdots, x_n) \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 + \cdots + x_n \\ \vdots \\ x_1 \cdots x_n \end{bmatrix}$$

Then σ can be interpreted as having degree at most 1 in each variable x_j. According to Corollary 1,

$$H(\sigma(\alpha_1, \cdots, \alpha_n)) \leq \max S(\sigma_i) H(\sigma, 1) H(\alpha)^n$$

where $\alpha = \alpha_1, \cdots, \alpha_n$ are all the Galois conjugates of α. Furthermore, $H(\sigma, 1) = 1$ and $S(\sigma_i) = \left(\begin{array}{c} n \\ i \end{array} \right) < 2^n$. □

We can generalize the Cauchy bound in example 2 into the following statement using Heights:

Theorem 5. *Let $p(x)$ be a polynomial with algebraic coefficients, and let α be a root of p. Then*

$$H(\alpha) < 2H(p)$$

Proof of Theorem 5. Let $d = \deg p$ and assume without loss of generality that $p_d = 1$. Let $K = \mathbb{Q}[p_0, \cdots, p_d]$. Let M_K be a system of absolute values of K.

We claim that if $\nu \in M_K$,

$$\max(|\alpha|_\nu, 1) \leq s + \max |p_i|_\nu$$

where $s = 1$ is ν is Archimedian and $s = 0$ otherwise. It will follow that $\max(|\alpha|_\nu, 1) \leq (1 + s) \max |p_i|_\nu$.

Indeed, we write

$$\alpha^d = -p_{d-1}\alpha^{d-1} - \cdots - p_0$$

If ν is a valuation, this implies that

$$|\alpha|_\nu^d \leq \max |p_i|_\nu |\alpha|_\nu^i$$

Hence,

$$\max(|\alpha|_\nu, 1)^d \leq (\max |p_i|_\nu) \max(|\alpha|_\nu, 1)^{d-1}$$

and

$$\max(|\alpha|_\nu, 1) \leq (\max |p_i|_\nu)$$

On the other hand, if ν is an Archimedian absolute value,

$$\max(|\alpha|_\nu, 1)^d \leq (\max |p_i|_\nu) \sum_{j=1}^{d-1} \max(|\alpha|_\nu, 1)^j$$

Therefore, if $|x|_\nu > 1$,

$$\max(|\alpha|_\nu, 1)^d < (\max |p_i|_\nu) \frac{\max(|\alpha|_\nu, 1)^d}{\max(|\alpha|_\nu, 1) - 1}$$

It follows at once that

$$\max(|\alpha|_\nu, 1) < 1 + \max |p_i|_\nu$$

If $|x|_\nu \leq 1$, then inequality $\max(|\alpha|_\nu, 1) \leq 1 + \max |p_i|_\nu$ is trivial.

We can now estimate the height of x:

$$\begin{aligned} H_K(x,1) &< \prod_{\nu \in M_K} (1 + s(\nu)) \max |p_i|_\nu^{n_\nu} \\ &\leq 2^{\sum_\nu n_\nu} H_K(p) \end{aligned}$$

where the last sum ranges over all archimedian valuations in M_K.

We can invoke now Theorem 3 to replace $\sum_\nu n_\nu$ by $[K : \mathbb{Q}]$. Taking roots, we obtain:

$$H(x) < 2H(p)$$

□

2.6. Weights.

Definition 2. Let $\alpha \in (\mathbb{Q}^a)^k$. We define the weight function as:

$$w(\alpha) = [\mathbb{Q}[\alpha_1, \cdots, \alpha_k] : \mathbb{Q}](1 + \lceil \log_2 H(\alpha, 1) \rceil)$$

Weights can be used to bound absolute values:

Lemma 1. *Let $\alpha \in \mathbb{Q}^a$, $\alpha \neq 0$. Then $2^{-w(\alpha)} < |\alpha| < 2^{w(\alpha)}$.*

Proof. Let $K = \mathbb{Q}[\alpha]$. Recall that

$$H_K(\alpha) = \prod_{\nu \in M_K} \max(1, |\alpha|_\nu^{n_\nu})$$

There is one archimedian valuation equal to the absolute value, so $H_K(\alpha) \geq |\alpha|$. Hence $H(\alpha) \geq |\alpha|^{\frac{1}{[K:\mathbb{Q}]}}$. It follows that $|\alpha| < 2^{w(\alpha)}$. Since $H(\alpha) = H(\alpha^{-1})$, the first inequality also holds. □

Theorem 6. *Let x be an algebraic number, and let p be its minimal polynomial over $\mathbb{Q}$. Let $d = \deg p = \deg_{\mathbb{Q}}(x)$. Then:*

$$w(p) - 2 \leq w(x) \leq d(1 + w(p)) \leq w(x)^2$$

Proof of Theorem 6. Theorem 5 implies that

$$w(x) \leq d(1 + \lceil \log_2 2H(p) \rceil) \leq d(1 + w(p))$$

On the other hand, Corollary 4 gives:

$$H(p) \leq (2H(x))^d$$

hence

$$\log_2 H(p) \leq d \log_2(1 + H(x)) \leq w(x)$$

and therefore $w(p) \leq 2 + w(x)$. □

The hypothesis of irreducibility of p is essential here.

An interesting question is how to bound the height of an isolated root of a system of polynomial equations. We conclude this section by quoting, for further reference, the following result by Krick and Pardo:

Theorem 7 (Krick and Pardo [KP95], Corollary 6)**.** *Let $f_1, \cdots, f_r$ $r \leq n$ be polynomials in $\mathbb{Z}[x_1, \cdots, x_r]$ of degree and height bounded by $d \geq n$ and η, respectively, and let V denote the algebraic affine variety defined by: $V = \{x : f_1(x) = \cdots f_r(x) = 0\}$.*

Then V has at most d^n isolated points, and their height verifies:

$$\log_2 H(P) \in d^{O(n)}(\log_2 r + \log_2 \eta)$$

3. Weighted model of Computation

3.1. **Machines over a ring.** Let R be a commutative ring with unity and without zero divisors. The following examples will be of interest here:

- The ring $\mathbb{F}_2$ consisting of elements 0 and 1.
- The ring $\mathbb{Z}$ of rational integers.
- The rings $\mathbb{Q}^{\mathrm{a}}$ and $\mathbb{Q}^{\mathrm{ra}}$ of algebraic and real algebraic numbers, resp.
- A number field K, i.e. a finite algebraic extension of $\mathbb{Q}$. However, for simplicity, we consider K as a ring.

Also, let o be a subset of R and let $c : R^{\infty} \to \mathbb{R}^{+}$ be a cost function.

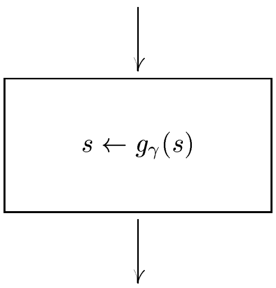
Essentially, a *machine* M maps an input space into an output space. For input and output space, we want to allow all finite, arbitrarily large sequences in our base ring R. Those sequences are denoted by R^{∞}. (In earlier literature, the notation was R^{*}). Eventually, we use the same notation for the machine M and for the function $M : R^{\infty} \to R^{\infty}$, $x \mapsto M(x)$.

Intermediate calculations are stored in a *State Space* $\mathcal{S}$, that can contain arbitrarily large, finite bisequences over R. This is denoted R_{∞}.

A polynomial map $f : R_{\infty} \to R_{\infty}$ is a finite sequence of polynomials in R_{∞}, and therefore it is described by a finite set of coefficients.

One can think of a machine as a *flowchart*, i.e., as a labelled directed graph consisting of *nodes* and *edges*. A first, *initial* node is 'executed'. Then the machine follows and outgoing edge, and finds a new node to 'execute', and so on.

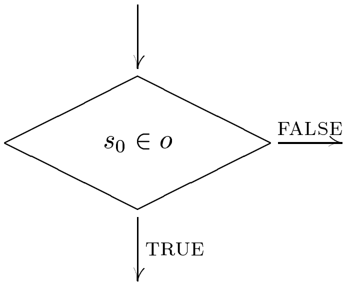
A machine M over (R, o, c) is a defined as a labelled directed graph Γ. The nodes of Γ, indexed 1 to N, are of the following five types:

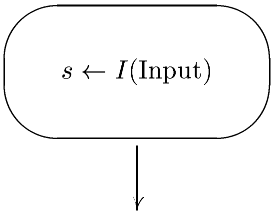


Computation nodes have exactly one outgoing edge, and Computation Node γ is associated to a (finite) polynomial system $g_\gamma : \mathcal{S} \to \mathcal{S}$. The effect of executing γ is to 'replace' $s \in \mathcal{S}$ by $g_\gamma(s) \in (\mathcal{S})$. Replacing means the following. Suppose that $g_\gamma = (g_{-1}, g_0, g_1, g_2)$ where each of the $g_{-1}, \dots, g_2$ is a polynomial in s. Then we replace $s_{-1}, \dots, s_2$ by the values of $g_{-1}(s), \dots, g_2(s)$ and leave all the other coordinates invariant:

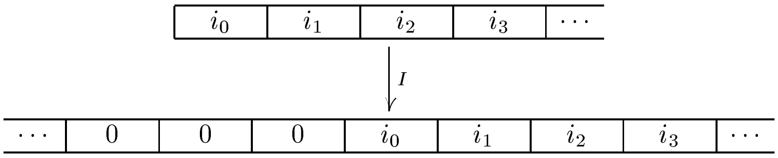
$$
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
\cdots & s_{-3} & s_{-2} & s_{-1} & s_0 & s_1 & s_2 & s_3 & \cdots \\
\hline
\end{array}
$$
$$\Big\downarrow f_\gamma$$
$$
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
\cdots & s_{-3} & s_{-2} & g_{-1}(s) & g_0(s) & g_1(s) & g_2(s) & s_3 & \cdots \\
\hline
\end{array}
$$

Branching nodes have exactly two outgoing edges labelled TRUE and FALSE, and Branching Node γ is associated to the expression $s_0 \in o$. Execution branches accordingly.

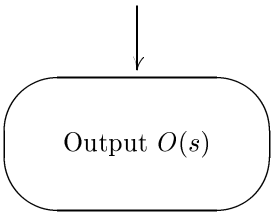


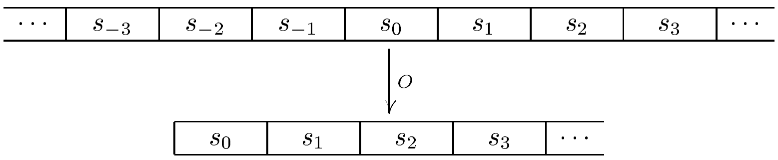


The *Input node* has exactly one outgoing edge, and is associated to the input map $I : \mathcal{I} \to \mathcal{S}$. I just copies the input into non-negative positions in the state space. Input node is indexed 1.

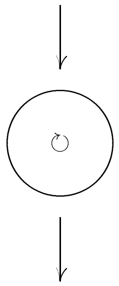


The *Output node* has one outgoing edge to itself, and is associated to an output map $O : \mathcal{S} \to \mathcal{O}$. O copies the non-negative state space into the output space. The output node is indexed N.

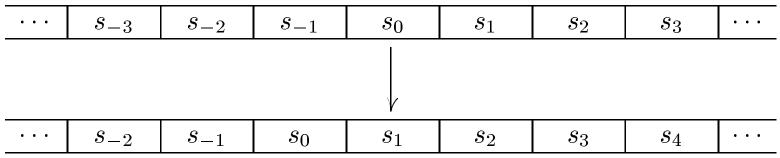




Fifth node



The <u>*Fifth node*</u> shifts $s \in \mathcal{S}$ by 1 to the left or to the right.



The formal definition goes as follows: Let M be a machine with input x, and let γ_0 be the input node. Let $\beta(\gamma)$ denote the node associated to the outgoing edge of γ if γ is not a decision node. In case γ is a decision node, set β_{true} and β_{false} be the corresponding nodes. Let σ_{left} and σ_{right} be the left shift and right shift, respectively. We can define the recurrence:

$$
\text{(R}_1\text{)} \qquad
\begin{aligned}
s^{t+1} &\leftarrow \begin{cases} g_{\gamma^t}(s^t) & \text{if } \gamma^t \text{ is a computation node.} \\ \sigma_{\text{right}}(s^t) & \text{if } \gamma^t \text{ is a right shift node.} \\ \sigma_{\text{left}}(s^t) & \text{if } \gamma^t \text{ is a left shift node.} \\ s^t & \text{otherwise} \end{cases} \\
\gamma^{t+1} &\leftarrow \begin{cases} \beta_{\text{true}}(\gamma^t) & \text{if } \gamma^t \text{ is a decision node and } s_0^t \in o. \\ \beta_{\text{false}}(\gamma^t) & \text{if } \gamma^t \text{ is a decision node and } s_0^t \notin o. \\ \beta(\gamma^t) & \text{Otherwise.} \end{cases}
\end{aligned}
$$

with initial condition

$$
\text{(R}_2\text{)} \qquad
\begin{aligned}
\gamma^1 &= \gamma \\
s^1 &= I(x)
\end{aligned}
$$

Definition 3. The machine M with input x stops after T steps if γ^T is an output node. The output, also denoted as $M(x)$, is $O(s^T)$. The function $x \mapsto M(x)$ is called the input-output mapping, and is defined only for those x such that the machine stops at some time T. The set $\Omega_M = \{x : M(x) \text{ stops}\}$ is said to be the set recognized by machine M, or 'halting set' of M.

Definition 4. A set $X \in K^\infty$ belongs to class re over (K, o, w) if and only if it is the halting set of a machine over (K, o, w). A set X belongs to class r if both X and its complement are in class re

Note that the definitions of classes re and r do not depend on the weighting function. They do depend on the branching set o.

Sets in the class r are also said to be 'decidable'. Indeed, it is easy to construct a machine that 'decides' a set $X \in r$, i.e., $M(x) = 0$ for $x \in X$ and 1 otherwise, in finite time.

Definition 5. The *size* of some $x \in R^k \subset R^\infty$ is defined as:

$$\text{Size } (x) = kc(x)$$

Definition 6. The *cost* or *running time* of computing $M(x)$ is

$$\text{Cost } (M, x) = T \max c(s_j^t) + \text{Size } (M(x))$$

where T is the number of steps (formally, $T = \min t$ s.t. γ^t is an output node) and s is the vector of all s_j^t for all values of $j \in \mathbb{Z}$ and $t \in \mathbb{N}$. Note that it suffices to take $1 \le t \le T$ and $-t-m \le j \le t+m$ where m is the maximum of the length of input x and a constant depending on the machine M.

Definition 7. A machine M is *polynomial time* if and only if there are constants a, b such that for any x, if $M(x)$ is defined, then

$$\text{Cost }(M, x) \leq a(\text{Size } x)^b$$

By defining the size of the input as in Definition 5 for the cost function $c(x) = w(x)$, the sum:

$$k, x_1, \cdots, x_k \mapsto \sum_{i=1}^{k} x_i$$

can be performed in polynomial time. If we had defined the size as $k \max c(x_i)$ instead, with $c = w$, the weight of the output (and of intermediate results) would be exponentially large in the input size.

The motivation for including the size of the output in the definition of the cost is the following property: Under the definitions above, if M and N are polynomial time machines, their composition $M \circ N$ is also polynomial time.

Definition 8. Two machines M and M' are *polynomially equivalent* if and only if they have the same input-output map and the running time of computing $M(x)$ and $M'(x)$ are related by

$$\begin{array}{rcl} \text{Cost }(M, x) & \leq & a\text{Cost }(M', x)^b \\ \text{Cost }(M', x) & \leq & a\text{Cost }(M, x)^b \end{array}$$

where a and b are constants depending on M and M' only.

The definition of 'machine' given above corresponds to a 'machine in normal form' in the terminology of [BCSS98]. It is easy to see that for any machine in the sense of [BCSS98] there is a polynomially equivalent machine in 'normal form'.

Moreover, given M, there is a polynomially equivalent machine M' with all polynomials g_γ of the form $s_0 \leftarrow x + y$ or $s_0 \leftarrow x - y$ or $s_0 \leftarrow xy$ where x and y can be state variables s_j or constants.

It is easy to see that for any machine M, there is a constant C_M independent of the input such that $\max_{i,t} w(s_i^t) \leq C_M \max_{i,t} w((s')_i^t)$ where s' corresponds to an equivalent machine with the restrictions above.

3.2. **The register equations.** Assume a model (R, o, c) of computation. As in [BSS89] or [BCSS98], the recurrence (R_1-R_2) can be rewritten in a more concise form

$$\text{(R')} \qquad \left\{ \begin{array}{rcl} s^{t+1} & = & G_t(\gamma^t, s^t) \\ \gamma^{t+1} & = & B(\gamma^t, \chi(s_0^t)) \end{array} \right.$$

with initial conditions $\gamma^1 = 1$ and $s^1 = I(x)$. Above, G_t and B are polynomials and χ is the characteristic function of the branching-set o:

In the ordered case, $\chi(x) = 1$ if $x > 0$ and 0 otherwise. In the unordered case, $\chi(x) = 1$ if $x = 0$ and 0 otherwise. For a p-adic analogue, see [MW97].

Polynomials G_t and B are polynomials with coefficients in the field of fractions of R. They can be constructed explicitly. Assume that the machine M has N nodes.

Let $p(i, j)$ be the N-th Lagrange interpolating polynomial:

$$p(i, j) = \frac{\prod_{1 \leq j \leq N, k \neq j} (i - k)}{\prod_{1 \leq j \leq N, k \neq j} (j - k)}$$

This is a degree N polynomial in i. If $j \in \{1; \cdots N\}$, its denominator divides $N - 1!$. If $i, j \in \{1; \cdots N\}$ then $p(i, j) = 1$ if $i = j$ and zero otherwise.

Let

$$g_i^t(s) = \left\{ \begin{array}{ll} g_i(s) & \text{if } i \text{ is a computation node} \\ s_{-t+1}, \cdots, s_{t+1} & \text{if } i \text{ is a left shift node} \\ s_{-t-1}, \cdots, s_{t-1} & \text{if } i \text{ is a right shift node} \\ \text{Empty polynomial} & \text{otherwise} \end{array} \right.$$

Then we can set:

$$G_t(\gamma, s) = \sum_{i=1}^{N} p(\gamma, i) g_i^t(s)$$

Also, if we set

$$\beta(i, \chi) = \begin{cases} \beta(i) & \text{if } i \text{ is not a branching node} \\ \chi \beta_{\text{true}}(i) + (1-\chi)\beta_{\text{false}}(i) & \text{if } i \text{ is a branching node} \end{cases}$$

then we can set

$$B(\gamma, \chi) = \sum_{i=1}^{N} p(\gamma, i) \beta(i, \chi)$$

It is also possible to rewrite (R') as a system of algebraic equations with coefficients in R, by introducing new variables u^t and v^t:

$$(\mathrm{R}_1\text{"}) \qquad \begin{cases} \overline{N-1}\,!s^{t+1} &= \overline{N-1}\,!G_t(\gamma^t, s^t) \\ \overline{N-1}\,!\gamma^{t+1} &= \overline{N-1}\,!B(\gamma^t, u^t) \end{cases}$$

In the ordered case $o = \{x \in R : x > 0\}$, variables u^t should satisfy:

$$(\mathrm{R}_{2a}\text{"}) \qquad \begin{cases} s_0^t \,(1 - s_0^t (v^t)^2)\,(1 + s_0^t (v^t)^2) &= 0 \\ u^t - s_0^t (v^t)^2 &= 0 \end{cases}$$

In the unordered case $o = \{0\}$, one needs instead:

$$(\mathrm{R}_{2b}\text{"}) \qquad \begin{cases} s_0^t (1 - s_0^t v^t) &= 0 \\ u^t - s_0^t v^t &= 0 \end{cases}$$

We may summarize all this as:

Theorem 8. *Let M be a machine (in normal form) over $(R, =, w)$ for any R or over $(\mathbb{Q}^{\mathrm{ra}}, >, w)$. Let N be the number of nodes of M. Then there are systems φ_T of polynomial equations such that (γ^t, s^t) satisfy equation (R") for $t = 1, \cdots, T$ if and only if there are $u_j \in \mathbb{Z}$, v_j such that $v_j^2 \in R$, such that $\varphi_T(\gamma, s, u, v) = 0$. Moreover*

1. *The number of equations is $\leq 4T(2T + 1 + C)$ where C is a constant depending on M.*
2. *The coefficients of φ_T are all in R.*
3. $\deg \varphi_T \leq \max(\max_\gamma(\deg f_\gamma) + N - 1, 7)$
4. *The number of non-zero coefficients in each equation of φ_T is bounded by $(N-1)\sum_\gamma S(g_\gamma)$ where $S(g)$ is the number of non-zero coefficients of g and the sum ranges over all the computation nodes.*
5. *The height of the coefficients of φ_T is bounded by $N!2^N H(M)$, where $H(M)$ is the height of the vector of all the coefficients of each g_γ.*
6. *$H(u) = 1$ and $H(v_j^t) \leq \sqrt{H(s)}$, $H(\gamma) \leq N$. Moreover, $u \in \{-1; 0; 1\}$ and $v \in R$ or $v^2 \in R$, respectively.*

This extends Theorem 2 in Chapter 3 of [BCSS98] to the weighted model.

3.3. Non-deterministic machines.

Definition 9. Let (R, o, c) be a model of computation. Let $X \subset R^\infty$. The set X belongs to the class $\mathcal{P}$ over (R, o, c) if and only if there are a machine M with input x and constants a, b such that

$$x \in X \Leftrightarrow M(x) = 1$$

and for any $x \in R^\infty$, $M(x)$ is defined and the cost of computing $M(x)$ is bounded above by $a(w(x)\mathrm{Length}(x))^b$

The class $\mathcal{P}$ stands for Polynomial Time. A lot of sets are known to belong to the possibly larger class $\mathcal{NP}$ (Nondeterministic Polynomial Time). Those are the sets that can be recognized by a 'nondeterministic' machine.

A non-deterministic machine M to recognize a set X is allowed to input some x and a guess g. If x belongs to the set X, then there is a guess that makes $M(x;g)=1$. Moreover, this computation can be performed in polynomial time with respect to the size of x (for at least one guess). On the other hand, given any guess g, if $M(x;g)=1$ then $x \in X$. More formally:

Definition 10. Let (R,o,c) be a model of computation. Let $X \subset R^\infty$. The set X belongs to the class $\mathcal{NP}$ over (R,o,c) if and only if there are a machine M with inputs x and g and constants a, b such that:

1. $\forall x \in R^\infty,\ x \in X \Leftrightarrow \exists\ g$ s.t. $M(x;g)=1$
2. $\forall x \in X, \exists\ g$ s.t. $M(x;g)=1$ and $\mathrm{Cost}(M(x;g)) \leq a(w(x)\mathrm{Length}(x))^b$

A now classical example is Hilbert Nullstellensatz over unit cost models (See [BCSS98] Ch. 5). Let HN(K) be the set of all the $(r,s,f_1,\cdots,f_r)$ where f_i's are polynomials in $x_1,\cdots,x_s$ in sparse representation, and where there is $x \in K^s$ s.t. $f_1(x)=\cdots=f_r(x)=0$

In the unit cost model, the weight of x does not need to be polynomially bounded by the length of the input $(r,s,f_1,\cdots,f_r)$. (Theorem 7 seems quite sharp). However, this is necessary in the weighted-cost model, so we need to give another example of a $\mathcal{NP}$ set. That example will be a subset of the Nullstellensatz, obtained by intersecting the Yes-set of the Nullstellensatz with a set in $\mathcal{P}$.

3.4. The 3-Nullstellensatz problem.

Definition 11. The restricted 3-Nullstellensatz problem over $\mathbb{Q}^{\mathrm{ra}}$, or HN3 for short, is the set of all $(m,n,a,f_1,\cdots,f_m)$ where

1. m, n and a are integers, in unary representation (i.e., they are represented by a sequence of 1's terminated by a 0. This representation has the following feature: the 'bit-size' of a is now a).
2. $f_1,\cdots,f_m$ are polynomials in variables $x_1,\cdots,x_n$, of the form:
$$f_i(x) = x_{j_i} - \left(x_{k_i} \left\{ \begin{array}{c} + \\ - \\ \times \end{array} \right\} x_{l_i} \right)$$
or of the form
$$f_i(x) = x_{j_i} - \text{constant}$$
3. There is x in $(\mathbb{Q}^{\mathrm{ra}})^n$ so that $f(x)=0$ and the size of a multiple of the minimal polynomial of each x_i is $< a$.

It is also very natural to rewrite conditions 1 and 3 in terms of a few extra polynomial equations, with integer (and uniformly bounded) coefficients. This leads to an interpretation of HN3 as a subset of HN. There are no heights in the new definition.

Lemma 2. *HN3 is in $\mathcal{NP}$ over $(\mathbb{Q}^{\mathrm{ra}},>,w)$.*

Proof of Lemma 2. We consider the machine $N = N(m,n,a,f_1,\cdots,f_m;\ x_1,\cdots\ x_n,\ p_1,\cdots,p_m)$. The last part of the input (the x_i's and the p_i's should be interpreted as the guess in a non-deterministic machine). The machine N will check in polynomial time the following facts:

1. $f_i(x)=0$ for all i.
2. $p_i(x_i)=0$
3. The bit-size of p_i is smaller than a.

Clearly, if the answer is yes to all the 3 items, then $(m, n, a, f) \in$ HN3. On the other hand, if $(m, n, a, f) \in$ HN3, then the bit-size of the minimal polynomials of each of the x_i's is bounded by a; furthermore, due to Theorem 6, $w(x_i)$ is polynomially bounded, hence steps 1 to 3 can be performed in polynomial time on (m, n, a, f). □

Theorem 9. *HN3 is $\mathcal{NP}$-Complete over* $(\mathbb{Q}^{\mathrm{ra}}, >, w)$.

It follows from Lemma 9 that both HN3 and HN are $\mathcal{NP}$-hard over $(\mathbb{Q}^{\mathrm{ra}}, >, w)$. However, HN is not likely to be in $\mathcal{NP}$. Indeed, the usual 'guess' for a nondeterministic machine in the unit cost model is the solution of the equations; that solution can have a very large weight. (The bound in Theorem 7 seems to be quite sharp). Therefore, building a nondeterministic polynomial time machine to check HN seems a difficult, maybe impossible task.

Proof of Theorem 9. Let $X \in \mathcal{NP}$ be recognized by the 'nondeterministic' machine $M = M(x; g)$ in time $< a\ \mathrm{size}(x)^b$.

Let $T = a(\mathrm{length}(x) \max W(x_i))^b$ where W is a weight estimator, i.e., a polynomial time function satisfying:

$$w(x) \leq W(x) \leq \mathrm{poly}(w(x))$$

Existence of the weight estimator follows from Theorem D). Therefore, machine M does terminate in time $\leq T$.

Let $\Phi^T(x; g, s, u, v)$ be the register equations (R"). This is a system in g, s, u, v, since x is fixed.

Clearly, there is a polynomial time machine R_1 that given T, x, produces the system $R_1(T, x) = \Phi^T(x; g, s, u, v)$

Also, there is a polynomial time machine R_2 that transforms a system $\varphi(z)$ into a HN3 sort of system $(R_2\varphi)(\hat{z})$; φ has a solution z if and only if $R_2\varphi$ has a solution $\hat{z}$. Moreover, because of Corollary 1, $\max w(z_j)$ and $\max w(\hat{z}_j)$ are related as follows:

$$\max w(z_j) \leq \max w(\hat{z}_j) \leq \max w(z_j)^A$$

We consider the reduction $x \mapsto R(x) = R_2(R_1(T(x), x))$. If $x \in X$, then $R(x)$ has a solution of weight $< T^A$, so the bit-size of the minimal polynomials of the solutions is bounded by T^{2A}.

Reciprocally, if $R(x)$ has a solution $\hat{z}$, then $\Phi^T(x; g, s, u, v) = 1$ and hence there is g s.t. $M(x, g) = 0$. This implies $x \in X$.

□

The situation over $(\mathbb{Z}, =, w)$ is interesting too. The set $\mathbb{N}$ is clearly in $\mathcal{NP}$ over that model, since it is possible to guess the 'bits' of a number x and then decide if $x \geq 0$. (A similar argument works with $(\mathbb{Z}, =, 1)$) [BSS89].

However, assume that the τ-conjecture [Sma98] is true. Then it is hard to decide $x \in \{1; \cdots; n\}$. The input size is defined as $\log n$, both for unit cost or for logarithmic cost. Hence $\mathbb{N} \notin \mathcal{P}$ with any of those cost models. Thus, the τ-conjecture implies $\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Z}, =, w)$.

4. A transfer theorem for the $\mathcal{P} \neq \mathcal{NP}$ Conjecture

.

4.1. **Morphisms and Transfer Theorem.**

Let (R_1, o_1, c_1) and (R_2, o_2, c_2) be models of computation. We do not assume R_2 to be an extension of R_1 or whatsoever. Instead, we will relate those models by a partial mapping of the form

$$\sigma : R_1 \leftarrow R_2^k$$

with $k \in \mathbb{N} \cup \{\infty\}$. This mapping will be used to 'embed' the model of computation (R_1, o_1, c_1) into the model of computation (R_2, o_2, c_2).

This 'covariant' language calls for an explanation: one of the main features of this 'embedding' should be to allow for the simulation of machines in (R_1, o_1, c_1) by machines in (R_2, o_2, c_2). In that case, elements of R_1 should be represented by sequences of elements in R_2.

For instance, if $R_1 = \mathbb{Z}$ and $R_2 = \mathbb{F}_2$, then an integer in R_1 should be represented by a sequence of bits (i.e. a finite sequence in $\mathbb{F}_2$).

If $x \in R_1$, the set $\sigma^{-1}(x)$ will contain all the valid representations of x. Not all the possible lists of elements of R_2 need to make sense, hence σ can be a partial map.

Moreover, representation of a given $x \in R_1$ does not need to be unique. For instance, if $R_1 = \mathbb{Q}^{\mathrm{ra}}$ is the ring of real algebraic numbers, and $R_2 = \mathbb{Z}$, then $x \in R_1$ may be represented by its minimal polynomial, together with a good approximation of x (In the sense of Theorem C).

By the use of coupling functions, σ can be naturally extended to a map $R_1^\infty \leftarrow R_2^\infty$. We will use the same name for σ and for its extension, and specify domain and range when necessary.

An important conceptual point should be stressed: σ is a map, not a computable map. It makes no sense to 'compute' σ in some model of computation. This map is introduced so we can work 'in coordinates', much as in linear algebra or differential geometry. Of course, since σ should transfer a certain 'computable polynomial time' structure from one model of computation into another, a few constraints will be necessary.

Let us fix the notation $\mathrm{Size}_i(x) = \mathrm{Length}(x) c_i(x)$ for x a sequence in (R_i, o_i, c_i) and $i = 1, 2$.

Definition 12. A map $\varphi : (R_1)^\infty \supseteq X \to Y \subseteq (R_2)^\infty$ is polynomially bounded if and only if there are a, b such that for any $x \in X$,

$$\mathrm{Size}(\varphi(x)) < a \mathrm{Size}(x)^b$$

A reciprocal of the condition of "polynomially bounded" will be often necessary. This motivates the definitions below:

Definition 13. A map $\sigma : R_1^\infty \leftarrow R_2^\infty$ is *honest* **over** $Y \subset R_1^\infty$ if and only if there are a, $b > 0$ such that

$$\forall y \in Y, \exists x \in \sigma^{-1}(y) \text{ such that } \mathrm{Size}_1(x) < a\,\mathrm{Size}_2(y)^b$$

A map will said to be honest if and only if it is honest **over** its image.

This definition is closely related to the one of a honest map **on** a set X, in Chapter 5 of [BCSS98]. Here, we do not require honest maps to be computable. It is possible also to define

Definition 14. A map $\sigma : R_1^\infty \leftarrow R_2^\infty$ is *honest* **on** $X \subset R_2^\infty$ if and only if there are a, $b > 0$ such that

$$\forall x \in X, \text{ such that } \mathrm{Size}_1(x) < a\,\mathrm{Size}_2(\sigma(x))^b$$

Clearly, a map σ honest **on** X is always honest **over** $\sigma(X)$. Also, a map σ honest **over** Y is always honest **on** some X contained in $\sigma^{-1}(Y)$.

An example: the projection $\pi : R^2 \to R$, $(x, y) \mapsto x$ is honest **over** R but is not honest **on** R^2. However it is honest on $\{(x, y) \text{ s.t. } \mathrm{Size}(x) < ai\,\mathrm{Size}(y)^b\}$ for a and b fixed.

An alternative definition for the class $\mathcal{NP}$ over (R, o, c) can be given in terms of honest maps. The statement below is somewhat simpler than Proposition 2 in Chapter 5 of [BCSS98]. However, only one direction seems to be valid in general:

Proposition 1. *Let a model of computation (R, o, c) be fixed. Let Y be a set of R^∞. If there exist a set $X \in \mathcal{P}$ and a honest map φ* **over** *Y, computable in polynomial time, and such that $\varphi(X) = Y$, then the set Y is in $\mathcal{NP}$ over (R, o, c).*

The proof is immediate, so it is omitted. The reciprocal is true (and easy) when $c = 1$. A candidate counterexample for the reciprocal over $(\mathbb{Q}^{\mathrm{ra}}, =, w)$ is the following set: $\{(k, x_1, \cdots, x_k) : x_i \geq 0\}$. Obvious guesses are the square roots of the x_i's, but the weight of the guess vector may be exponential in the input size.

In order to transfer 'computable polynomial time' structure from the model of computation (R_1, o_1, c_1) into the model of computation (R_2, o_2, c_2), we can still follow the analogy with differential geometry and define the 'pull-back' by σ of computable functions and machines over (R_1, o_1, c_1).

Definition 15. Let $\sigma : (R_2)^\infty \to (R_1)^\infty$ and let f be a n-ary function,

$$\begin{array}{cccc} f: & (R_1)^n & \to & R_1 \\ & y_1, \cdots y_n & \mapsto & f(y_1, \cdots, y_n) \end{array}$$

Then a function $f^* : (R_2)^\infty \to (R_2 i)^\infty$ is a *pull-back* of f if and only if,

$$\forall x_1, \cdots x_n \in R_2, \sigma(f^*(x_1, \cdots, x_n)) = f(\sigma(x_1), \cdots, \sigma(x_n))$$

The pull-back of a function does not need to be unique. A similar definition, this time unique, can be produced for a total function g from R into $\{0, 1\}$: (i.e. a set): its pull-back g^* is the unique function such that $g^*(x) = g(\sigma(x))$. For example, '$y \in o_1$' has pull-back '$\sigma(x) \in o_1'$.

We can summarize that information in the following commutative diagrams:

$$\begin{tikzcd} (R_2)^\infty \arrow[r, "{(\sigma,\sigma,\cdots\sigma)}"] \arrow[d, dotted, "f^*"'] & (R_1)^n \arrow[d, "f"] \\ (R_2)^\infty \arrow[r, "\sigma"'] & R_1 \end{tikzcd} \qquad \begin{tikzcd} (R_2)^\infty \arrow[r, "\sigma"] \arrow[dr, dotted, "g^*"'] & R_1 \arrow[d, "g"] \\ & \{0; 1\} \end{tikzcd}$$

It is also reasonable to speak of the pull-back of a machine. We will need the following condition in the sequel:

Definition 16. Let $\sigma : R_1^\infty \leftarrow R_2^\infty$. We say that σ has the pull-back property if and only if for any machine M over (R_1, o_1, c_1) there are $a, b > 0$ and a machine M^* over (R_2, o_2, c_2) such that

1. The function computed by M^* is a pull-back of the function computed by M.
2. Cost $(M^*, x) < a$ Cost$(M, \sigma(x))^b$

It is now clear that

Lemma 3. *Let $\sigma : R_1^\infty \leftarrow R_2^\infty$ is polynomially bounded and has the pull-back property. Then $Y \in \mathcal{P}$ over (R_1, o_1, c_1) implies that its preimage $\sigma^{-1}(Y)$ is in $\mathcal{P}$ over (R_2, o_2, c_2)*

If M is a nondeterministic polynomial time machine, the situation is more subtle: Guesses g taken by M should correspond to guesses g' taken by the pull-back, so that $g = \sigma(g')$ and the size of g' is polynomially bounded on the size of g. Moreover, it is necessary to ensure that the guess g' is indeed in the domain of σ.

Definition 17. A *morphism* from (R_1, o_1, c_1) into (R_2, o_2, c_2) is a polynomially bounded, partial mapping $\sigma : R_1 \leftarrow R_2^k$, with $k \in \mathbb{N} \cup \{\infty\}$, so that:

(M1) σ is surjective.
(M2) σ is honest.
(M3) The domain of definition of σ belongs to $\mathcal{NP}$ over (R_2, o_2, w_2).
(M4) σ has the pull-back property.

The main feature of morphisms is to allow for a 'transfer' of the class $\mathcal{N}P$ (and of the class $\mathcal{P}$, also).

Lemma 4. *Let σ be a morphism from (R_1, o_1, c_1) into (R_2, o_2, c_2). Let X be in $\mathcal{NP}$ over (R_1, o_1, c_1). Then $\sigma^{-1}(X)$ is in $\mathcal{NP}$ over (R_2, o_2, c_2)*

Proof of Lemma 4. Let $M = M(x, g)$ be a nondeterministic polynomial time machine to recognize X. Let M^* be a pull-back of M by σ, and let $Y = \sigma^{-1}(X)$.

If $y \in Y$, then $\sigma(y) \in X$ and there is a guess g such that M accepts $\sigma(y)$ in polynomial time (with respect to the size of $\sigma(y)$). Since σ is polynomially bounded, $\sigma(y)$ is accepted in polynomial time with respect to the size of y indeed.

Since σ is surjective, there is g' in $\sigma^{-1}(g)$ such that $M^*(y, g')$ 'accepts' y in polynomial time, again with respect to the size of y.

It is necessary to take a closer look at the preimage g' of the guess g. It can be decomposed into sub-guesses $g'_1, \cdots, g'_s \in (R_2)^\infty$, such that for each coordinate $g_i \in R_1$ of g, we have:

$$\sigma(g'_i) = g_i$$

By Hypothesis (M2) we can assume without loss of generality that $\mathrm{Size}_2(g'_i) < \mathrm{Size}_1(g_i) = c_1(g_i)$. Moreover, since M is polynomial time w.r.t. $\mathrm{Size}_2(y)$, $c_1(g_i)$ is polynomially bounded w.r.t. $\mathrm{Size}_2(y)$.

By Hypothesis (M3), there is a non-deterministic polynomial time machine $D = D(z, g'')$ that recognizes the domain of σ. Setting $z = g'_i$, one can use D to recognize valid guesses g'_i is nondeterministic polynomial time w.r.t. $\mathrm{Size}_2(y)$.

We define a machine N as follows: N inputs y and guesses g', g''_i. It first tests $D(g'_i, g''_i)$. If all tests succeed (i.e. g' is in the domain of σ), the machine tests $M^*(y, g')$ and returns 'Yes' on success.

Clearly, if $y \in Y$, there will be guesses g' and g'' such that $N(y, g', g'')$ will succeed in polynomial time w.r.t. $\mathrm{Size}_2(y)$.

Assume now that y gets accepted by N, with guess (g', g''). Then g' was accepted by D, hence g' is in the domain of definition of σ and $g = \sigma(g')$ is well-defined. It follows from the pull-back property that M accepts $\sigma(y)$, with guess g. Hence, $y \in Y$. □

A few remarks:

Remark 1. Identity is a morphism from (R_1, o_1, c_1) into itself.

Remark 2. The composition of two morphisms is again a morphism

Remark 3. In order to define a Category, we should specify the objects. Those can be chosen to be the class of all (R, o, c) where R is a commutative ring with unity and no zero divisors, o is a proper subset of R and c is a function from R into the positive reals. One can also restrict this to $o = \{0\}$ or to reasonable cost functions in some sense...

Remark 4. The usual category theory functors do not seem to make any sense in this context. However, the word morphism was selected in absence of a better name. $\mathcal{P}$-morphisms are taken, maybe $\mathcal{NP}$-morphisms would be a good name ?

Of the properties in the definition of morphisms, the pull-back property may be the hardest to check. In some conditions, it may be enough to check the pull-back condition for a few basic functions:

Definition 18. A map $\sigma : R_1^\infty \leftarrow R_2^\infty$ is *very honest* **over** $Y \subset R_1^\infty$ if and only if there are $a, b > 0$ and a polynomial time machine S over (R_2, o_2, c_2) such that $\sigma(x) = \sigma(S(x))$ and,

$$\forall x \in \sigma^{-1}(Y), \mathrm{Size}_1(S(x)) < a\mathrm{Size}_2(\sigma(x))^b$$

A map is 'very honest' if and only if it is very honest **over** its image.

Clearly, a very honest map is always honest; the machine S can be seen as a simplification procedure for the representation x of some $y = \sigma(x)$.

Lemma 5. *Let σ be a very honest map, such that the following functions have a polynomial time pull-back: $x, y \mapsto x + y$, $x, y \mapsto xy$, $x \mapsto -x$, $x \mapsto (x \in o_1)$. Then σ has the pull-back property.*

We can now define the following relation of equivalence between two models:

Definition 19. Computation models (R_1, o_1, c_1) and (R_2, o_2, c_2) are *polynomially equivalent* if and only if there are morphisms:

$$\begin{aligned} \sigma_1 &: (R_1, o_1, c_1) \to (R_2, o_2, c_2) \\ \sigma_2 &: (R_2, o_2, c_2) \to (R_1, o_1, c_1) \end{aligned}$$

such that:

(T1) There is a polynomial time machine A_1 over (R_1, o_1, c_1) such that $\sigma_1 \circ \sigma_2 \circ A_1$ is the identity in $(R_1)^\infty$.
(T2) There is a polynomial time machine A_2 over (R_2, o_2, c_2) such that $\sigma_2 \circ \sigma_1 \circ A_2$ is the identity in $(R_2)^\infty$.

$$
\begin{tikzcd}
R_1 \arrow[loop above, "Id"] \arrow[dd, "A_1"'] & \\
 & (R_2)^\infty \arrow[ul, "\sigma_1"'] \\
(R_1)^\infty \arrow[ur, "\sigma_2"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & R_2 \arrow[loop above, "Id"] \arrow[dd, "A_2"] \\
(R_1)^\infty \arrow[ur, "\sigma_2"] & \\
 & (R_2)^\infty \arrow[ul, "\sigma_1"]
\end{tikzcd}
$$

We can now state a transfer theorem:

Theorem 10. *Let computation models (R_1, o_1, c_1) and (R_2, o_2, c_2) be polynomially equivalent. Then $\mathcal{P} \neq \mathcal{NP}$ over (R_1, o_1, c_1) if and only if $\mathcal{P} \neq \mathcal{NP}$ over (R_2, o_2, c_2).*

Note that Theorem 10 does not require the existence of a $\mathcal{NP}$-complete problem over any of (R_1, o_1, c_1) or (R_2, o_2, c_2).

It is clearly enough to prove only one sense, viz. $\mathcal{P} \neq \mathcal{NP}$ over (R_1, o_1, c_1) implies $\mathcal{P} \neq \mathcal{NP}$ over (R_2, o_2, c_2).

Proof of Theorem 10. Assume that $\mathcal{P} \neq \mathcal{NP}$ over (R_1, o_1, c_1).

Step 1: By hypothesis, there is a problem X that is in $\mathcal{NP}$ over (R_1, o_1, c_1), but not in $\mathcal{P}$.
Step 2: Let $Y = \sigma_1^{-1}(X)$. Y is in $\mathcal{NP}$ over (R_2, o_2, c_2) (Lemma 4).
Step 3: Assume by absurd that $\mathcal{P} = \mathcal{NP}$ over (R_2, o_2, c_2). Then Y is in $\mathcal{P}$ over (R_2, o_2, c_2).
Step 4: $Z = \sigma_2^{-1}(Y)$ is in $\mathcal{P}$ over (R_1, o_1, c_1) (Lemma 3).
Step 5: Z contains the image of X by the polynomial time machine A_1 of property (T1).
Step 6: On the other hand, if $A_1(x)$ is in Z, then x is in X.
Step 7: Therefore, machine A_1 reduces the decision of membership in X to membership in Z. This later problem can be solved in polynomial time. Hence $\mathcal{P} = \mathcal{NP}$ over (R_1, o_1, c_1), contradiction. $\square$

As a matter of fact, the same proof shows a more general, weaker statement:

Theorem 11. *Let (R_1, o_1, c_1) and (R_2, o_2, c_2) be computation models, such that there exist mappings*

$$
\begin{aligned}
\sigma_1 &: (R_1, o_1, c_1) \to (R_2, o_2, c_2) \\
\sigma_2 &: (R_2, o_2, c_2) \to (R_1, o_1, c_1)
\end{aligned}
$$

such that σ_1 is a morphism, σ_2 is a polynomially bounded partial mapping that has the pull-back property (M4). Assume that (T1) holds. Then $\mathcal{P} \neq \mathcal{NP}$ over (R_1, o_1, c_1) implies $\mathcal{P} \neq \mathcal{NP}$ over (R_2, o_2, c_2).

A further remark: the technology of Theorem 10 allows to transfer the $\mathcal{P} \neq \mathcal{NP}$ Conjecture between models where R_1^∞ and R_2^∞ have same cardinality.

This is different from the argument in [BCSS98] for transferring between algebraic and complex numbers, that used Tarski's strong transfer principle.

4.2. **Examples.**

Proposition 2. *Models* $(\mathbb{Z}, \geq, w)$ *and* $(\mathbb{F}_2, =, 1)$ *(the Turing model) are polynomially equivalent.*

Proof of Proposition 2. Let

$$\begin{array}{rccl} b: & \mathbb{F}_2^\infty & \to & \mathbb{Z} \\ & (s, 0, x_1, 0, \cdots, x_k, 1) & \mapsto & (-1)^s \sum_{i=0}^k 2^i x_i \end{array}$$

This is clearly a morphism from $(\mathbb{Z}, \geq, w)$ into $(\mathbb{F}_2, =, 1)$. We can also define a morphism from $(\mathbb{F}_2, =, 1)$ into $(\mathbb{Z}, \geq, w)$ by:

$$\begin{array}{rrcl} i: & \mathbb{Z} \supseteq \{0; 1\} & \to & \mathbb{F}_2 \\ & 0 & \mapsto & 0 \\ & 1 & \mapsto & 1 \end{array}$$

Machines A_1 and A_2 are essentially similar. They associate to any input x its binary expansion. In the case of A_2, the input x is supposed to belong to $\mathbb{F}_2$, hence the only possible values are 0 and 1. In the case of A_1, $x \in \mathbb{Z}$.

Since $i \circ b \circ A_2$ and $b \circ i \circ A_1$ are the identity in $\mathbb{F}_2$ and in $\mathbb{Z}$ respectively, those two models are polynomially equivalent. □

Proposition 3. *Models* $(\mathbb{Q}, \geq, w)$ *and* $(\mathbb{Z}, \geq, w)$ *are polynomially equivalent.*

Proof of Proposition 3. In the same spirit, we define:

$$\begin{array}{rrcl} b: & \mathbb{Z}^2 \supseteq \mathbb{Z} \times \mathbb{Z}_* & \to & \mathbb{Q} \\ & p, q & \mapsto & \frac{p}{q} \end{array}$$

$$\begin{array}{rrcl} i: & \mathbb{Q} \supseteq \mathbb{Z} & \to & \mathbb{Z} \\ & x & \mapsto & x \end{array}$$

Then we have to define $A_1 : x \mapsto (x, 1)$ and $A_2 : p/q \mapsto (p, q)$ where p and q are relatively prime. This last machine will just perform Euclidian algorithm (For its running time, see the discussion in [Knu98] Section 4.5.3) □

4.3. **Is** $x \geq 0$ **hard ?** Towards the proof of Theorem B, we will assume Theorem C and show its Corollary:

Corollary 5. *The set* $\mathbb{Q}^{\mathrm{ra}}$ *is in* $\mathcal{NP}$ *over the algebraic numbers* $(\mathbb{Q}^{\mathrm{a}}, =, w)$

Proof of Corollary 5. Given $x \in \mathbb{Q}^{\mathrm{a}}$, guess $y(x)$ where y is the function defined in Theorem C. Then compute a 2^{-k}-approximation of x, of the form $a + bi$. By Theorem C, $b = 0$ if and only if x is real. □

Lemma 6. $\mathcal{P} \neq \mathcal{NP}$ *over* $(\mathbb{Q}^{\mathrm{ra}}, =, w)$ *implies* $\mathcal{P} \neq \mathcal{NP}$ *over* $(\mathbb{Q}^{\mathrm{a}}, =, w)$.

Proof. Define:

$$\begin{array}{rrcl} \sigma_1: & \mathbb{Q}^{\mathrm{ra}} & \to & \mathbb{Q}^{\mathrm{a}} \\ & x & \mapsto & \sigma_1(x) = x \end{array}$$

This is clearly a morphism. (Property (I3) follows from Corollary 5). Define also the morphism:

$$\begin{array}{rrcl} \sigma_2: & \mathbb{Q}^{\mathrm{a}} & \to & (\mathbb{Q}^{\mathrm{ra}})^2 \\ & x + iy & \mapsto & \sigma_2(x + iy) = (x, y) \end{array}$$

Then $\sigma_2 \circ \sigma_1$ is the identity map in R_1, so (T1) holds. Note that $\sigma_1 \circ \sigma_2$ maps $x + iy$ into x, y, so it is not clear if (T2) hold.

Hence, we are under the conditions of Theorem 11 and $\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Q}^{\mathrm{ra}}, =, w)$ implies $\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Q}^{\mathrm{a}}, =, w)$ □

Now we can show that

Theorem 12. *$\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Q}^{\mathrm{ra}}, =, w)$.*

This implies Theorem B

Proof. Let $\mathbb{Q}^{\mathrm{ra}}_+$ be the set of $\{x \in \mathbb{Q}^{\mathrm{ra}} : x \geq 0\}$. Clearly, $\mathbb{Q}^{\mathrm{ra}}_+$ is in $\mathcal{NP}$, since the non-deterministic machine $M(x,y)$ that checks $x - y^2 = 0$ decides $x \in \mathbb{Q}^{\mathrm{ra}}_+$ in polynomial time. (Recall that $H(\sqrt{x}) = \sqrt{H(x)}$ and $[\mathbb{Q}[\sqrt{x}] : \mathbb{Q}] \leq 2[\mathbb{Q}[x] : \mathbb{Q}]$).

Now, suppose that there is a deterministic machine M that decides $x \in \mathbb{Q}^{\mathrm{ra}}_+$ in polynomial time. Let K be the number field generated over $\mathbb{Q}$ by all the coefficients of M.

There is some $\alpha > 0$ algebraic over K with the following property: there is an automorphism σ of $K[\alpha]$ over K such that $\alpha^\sigma < 0$. Choose $z \in K$ such that $\sqrt{z} \notin K$, and set $\alpha = \sqrt{z}$.

The machine M cannot distinguish between α and α^σ. Indeed, let γ^t be the path associated to input α and let $G^t(\alpha) = g_{\gamma^t}(s^t)$ be the polynomial in α associated to each branching node in γ^t. The coefficients of G^t are in K, so σ fixes G^t and

$$G^t(x) = 0 \Leftrightarrow G^t(x^\sigma) = 0$$

This means that replacing α by α^σ does not modify the path γ^t, hence the output of M is the same. ☐

The same argument is true for the unit cost model $(\mathbb{Q}^{\mathrm{ra}}, =, 1)$. However, in the unit cost case, a more general theorem holds, as discussed in [BCSS98] Chapter 7. Namely $\mathcal{P} \neq \mathcal{NP}$ over all $(K, >, 1)$ where K is an infinite field not real closed, and over all $(K, =, 1)$ where K is an infinite field not algebraically closed.

5. Complexity of diophantine approximation

The main goal of this section is to prove Theorem C. A few remarks concerning that Theorem:

It is possible to represent an algebraic number x by its minimal polynomial and an interval containing x. This interval should be taken smaller than the distance to the nearest root. See [Mig83] and [Loo83] for a bound. See also Corollary 1 of Ch. 8 in [BCSS98], and [Mal93] for other 'gap' bounds. However, in order to refine that interval up to 2^{-k} by a method such as bisection, it takes k steps with k-bits arithmetic. This is why Theorem C was stated.

In order to obtain this sharper estimate, we use Newton Iteration instead. Newton iteration is known to converge quadratically once close enough to the root. [KA82, Sma86, SS93]. See also Chapter 8 in [BCSS98].

If we use some sort of approximate computation, we have to allow for some sort of 'approximate' Newton iteration. Theorems on quadratic convergence of 'approximate' Newton iteration appear in [Mal94].

Also, it is necessary to bound the precision necessary to specify the 'starting point' of the Newton iteration. Those are the main ingredients of the proof of Theorem C.

5.1. Approximate Newton Iteration.

Let f be a univariate polynomial. The following invariants [Sma86] are associated to f:

$$\begin{aligned} \beta(x) = \beta(f,x) &= \frac{|f(x)|}{|f'(x)|} \\ \gamma(x) = \gamma(f,x) &= \max_{k \geq 2} \left(\frac{|f^{(k)}(x)|}{k!\ |f'(x)|} \right)^{\frac{1}{k-1}} \\ \alpha(x) = \alpha(f,x) &= \beta(f,x)\ \gamma(f,x) \end{aligned}$$

The following theorem is a particular case of some earlier results in [Mal94]. It generalizes the convergence theorems in [Sma86] to the case of *approximate* Newton iteration. It is stronger than Theorem 4 in Chapter 8 of [BCSS98], which only provides linear convergence for approximate Newton iteration.

Theorem 13. *Let f be a univariate polynomial, and let $s \in \mathbb{N}$, $s \geq 2$. Let x_0 be such that $\alpha(f, x_0) < 2^{-5}$. Let δ be such that $\delta\gamma(f, x_0) < 2^{-4-2^{s+2}}$. Let the sequence x_i satisfy*

$$\left| x_{i+1} - \left(x_i - \frac{f(x_i)}{f'(x_i)} \right) \right| < \delta$$

Then there is a zero x^* of f such that

$$|x_i - x^*| < \frac{2^{1-2^{\min(i,s)}}}{\gamma(f, x_0)}$$

Some of the Lemmas below are standard and may have appeared in [Sma86, SS93, BCSS98]. For later use, we define the function $\psi(t) = 1 - 4t + 2t^2$. The Lemma below is slightly sharper and implies Proposition 3 in Chapter 8 of [BCSS98].

Lemma 7. *Let f be a univariate polynomial, and let x, $h \in \mathbb{C}$. Let $f'(x) \neq 0$ and assume that $|h|\gamma(f, x) < 1 - \sqrt{2}/2$. Then:*

$$\beta(f, x + h) \leq \left| \frac{f(x)}{f'(x)} + h \right| \frac{(\,1 - \gamma(f, x)|h|\,)^2}{\psi(\,\gamma(f, x)|h|\,)} + |h|^2 \gamma(f, x) \frac{1 - \gamma(f, x)|h|}{\psi(\,\gamma(f, x)|h|\,)}$$

In particular,

1. *If $h = -\frac{f(x)}{f'(x)} + \delta$, we get:*

$$\beta(x + h) \leq |\delta| \frac{(\,1 - \gamma(x)|h|\,)^2}{\psi(\,\gamma(x)|h|\,)} + (\,|\beta(x)| + |\delta|\,)^2 \gamma(x) \frac{1 - \gamma(x)|h|}{\psi(\,\gamma(x)|h|\,)}$$

2. *If $\beta(f, x) = 0$, we get:*

$$\beta(x + h) \leq |h| \frac{(\,1 - \gamma(x)|h|\,)^2}{\psi(\,\gamma(x)|h|\,)} + |h|^2 \gamma(x) \frac{1 - \gamma(x)|h|}{\psi(\,\gamma(x)|h|\,)}$$

Proof of Lemma 7. We can break

$$\beta(x + h) = \frac{|f(x + h)|}{|f'(x + h)|} = \frac{|f'(x)|}{|f'(x + h)|} \frac{|f(x + h)|}{|f'(x)|}$$

Using Taylor development, it is easy to estimate, since $f'(x) \neq 0$:

$$\begin{aligned} \frac{|f(x + h)|}{|f'(x)|} &\leq \left| \frac{f(x)}{f'(x)} + h \right| + \sum_{k \geq 2} \left| \frac{f^{(k)}(x)}{k! f'(x)} h^k \right| \\ &\leq \left| \frac{f(x)}{f'(x)} + h \right| + |h| \sum_{k \geq 2} (\,\gamma(f, x)|h|\,)^{k-1} \end{aligned}$$

and hence

$$\frac{|f(x + h)|}{|f'(x)|} \leq \left| \frac{f(x)}{f'(x)} + h \right| + \frac{|h|^2 \gamma(x)}{1 - \gamma(x)|h|} \tag{1}$$

On the other hand, using again the hypothesis $f'(x) \neq 0$,

$$\frac{f'(x + h)}{f'(x)} = 1 + \sum_{k \geq 2} \frac{f^{(k)}(x)}{k - 1! f'(x)} h^{k-1}$$

hence

$$\left| \frac{f'(x + h)}{f'(x)} - 1 \right| \leq \sum_{k \geq 2} (\,k\gamma(x)|h|\,)^{k-1} \leq \frac{1}{(\,1 - \gamma(x)|h|\,)^2} - 1$$

Thus,

$$\frac{|f'(x)|}{|f'(x+h)|} \leq \frac{1}{1-\left|\frac{f'(x+h)}{f'(x)}-1\right|} \leq \frac{1}{2-\frac{1}{(1-\gamma(x)|h|)^2}} \leq \frac{1-\gamma(x)|h|}{\psi(\gamma(x)|h|)} \tag{2}$$

Combining bounds 1 and 2 together, we obtain

$$\beta(x+h) \leq \left|\frac{f(x)}{f'(x)}+|h|\right| \frac{(1-\gamma(x)|h|)^2}{\psi(\gamma(x)|h|)} + |h|^2\gamma(x)\frac{1-\gamma(x)|h|}{\psi(\gamma(x)|h|)}$$

□

Lemma 8. *Let f be a univariate polynomial, and let x, $h \in \mathbb{C}$. Let $f'(x) \neq 0$. Assume also that $\gamma(f,x)|h| < 1-\sqrt{2}/2$. Then:*

$$\gamma(f,x+h) \leq \frac{\gamma(f,x)}{(1-\gamma(f,x)|h|)\psi(\gamma(f,x)|h|)}$$

Since this is Proposition 3 in Chapter 8 of [BCSS98], the proof is omitted.
From Lemma 8 we can deduce that:

Lemma 9. *Let f be a univariate polynomial. Let C be a constant, $C < \frac{1}{10}$. Let $x^* \in \mathbb{C}$ be a zero of f. Let U be a neighborhood of x^*, of radius $\rho < \frac{C}{\gamma(f,x^*)}$. Then for any $x \in U$,*

$$(1-10C)\gamma(f,x^*) < \gamma(f,x) < \frac{1}{1-5C}\gamma(f,x^*)$$

Proof of Lemma 9. From Lemma 8 centered at x^*, we have

$$\begin{aligned}
\gamma(x) &\leq \frac{\gamma(x^*)}{(1-\gamma(x^*)\rho)\psi(\gamma(x^*)\rho)} \\
&\leq \frac{\gamma(x^*)}{(1-C)\psi(C)} \\
&\leq \frac{\gamma(x^*)}{1-5C}
\end{aligned}$$

On the other hand, Lemma 8 also tells us that

$$\begin{aligned}
\gamma(x^*) &\leq \frac{\gamma(x)}{(1-\gamma(x)\rho)\psi(\gamma(x)\rho)} \\
&\leq \frac{\gamma(x)}{(1-\frac{C}{1-5C})\psi\left(\frac{C}{1-5C}\right)} \\
&\leq \frac{1-5C}{1-10C}\gamma(x)
\end{aligned}$$

□

Lemma 10. *Let f be a univariate polynomial. Let x be such that $\alpha(f,x) < 2^{-4}$. Then there is x^* such that $f(x^*) = 0$ and $|x-x^*| < 1.1\beta(f,x)$*

Proof. First of all, let $x_0 = x$ and $x_{i+1} = x_i - f(x_i)/f'(x_i)$. Then $\alpha(x_i) < 2^{-2^{i+1}-2}$. Indeed, by putting Lemma 7 and Lemma 8 together, one gets:

$$\alpha(x_{i+1}) \leq \alpha(x_i)^2 \frac{1}{\psi(\alpha(x_i))^2} < \frac{16}{9}\alpha(x)^2 < 2\alpha(x)^2$$

so

$$\alpha(x_{i+1}) < 2 \times 2^{-4-2^{i+2}} = \times 2^{-3-2^{i+2}}$$

Besides, $\lim \alpha(x_i) = 0$ so x_i converges. Let $x^* = \lim x_i$.

$$|x-x^*| \leq \sum \beta(x_i)$$

From Lemma 7,

$$\beta(x_{i+1}) < \beta(x_i)\alpha(x_i)\frac{1-\alpha(x_i)}{\psi(\alpha(x_i)} < 0.078\cdots\beta(x_i) < 0.08\beta(x_i)$$

Therefore,

$$|x - x^*| < \frac{1}{1-0.08}\beta(x_0) < 1.087\cdots\beta(x_0) < 1.1\beta(x_0)$$

□

Lemma 11. *Let f be a univariate polynomial. Let $x^* \in \mathbb{C}$ be a zero of f. Let U be a neighborhood of x^*, of radius $\rho < \frac{1}{100\gamma(f,x^*)}$. Then for any $x_0 \in U$, set $x_1 = x_0 - f(x_0)/f'(x_0)$*

$$|x_1 - x^*| < 0.1\rho$$

Proof. First of all, we can use Lemma 9 to bound γ in U. We obtain, using $C = \frac{1}{100}$, that for all $x \in U$,

$$0.9\gamma(x^*) < \gamma(x) < \frac{20}{19}\gamma(x^*)$$

Also, from Lemma 7 item 2, we obtain:

$$\beta(x) \leq \rho\frac{(1-\gamma(x^*)\rho)^2}{\psi(\gamma(x^*)\rho)} + \rho^2\gamma(x^*)\frac{1-\gamma(x^*)\rho}{\psi(\gamma(x^*)\rho)} \leq 1.031\cdots\rho < 1.05\rho$$

Hence,

$$\alpha(x) \leq \beta(x)\gamma(x) < (1.05\rho)\frac{20}{19}\gamma(x^*) \leq 0.01105 < 0.012$$

We can also bound

$$\alpha(x_1) < 2\alpha(x)^2$$

Thus,

$$\begin{aligned}\beta(x_1) &\leq \frac{2\alpha(x)^2}{\gamma(x)}\\ &\leq 2\alpha(x)\beta(x)\\ &\leq 2\times 0.012\times 1.05\rho\\ &\leq 0.0252\rho\end{aligned}$$

Hence, using lemma 10,

$$|x_1 - x^*| < 0.028\rho$$

□

Proof of Theorem 13. Lemma 10, together with the hypothesis $\alpha(x_0) < \frac{1}{32}$, implies the existence of a point x^* s.t. $f(x^*) = 0$, with

$$|x_0 - x^*| < 1.1\beta(x) < \frac{1.1}{32\gamma(x)}$$

We apply Lemma 8 to estimate $\gamma(x^*)$:

$$\begin{aligned}\gamma(x^*) &\leq \frac{\gamma(x_0)}{(\,1-\gamma(x_0)|x^*-x_0|\,)\psi(\,\gamma(x_0)|x^*-x_0|\,)}\\ &\leq \frac{\gamma(x_0)}{1-5\frac{1.1}{32}}\\ &\leq 1.21\gamma(x_0)\end{aligned}$$

Let U be a disc centered in x^* and with radius $\frac{1}{20\gamma(x^*)}$. Clearly, $x_0 \in U$. (Proof: $\frac{1.1}{32\gamma(x_0)} < \frac{1.1\times 1.21}{32\gamma(x^*)} < \frac{1}{20\gamma(x^*)}$. Lemma 9 provides the following uniform bound for any $x \in U$:

$$\frac{1}{2}\gamma(x^*) < \gamma(x) < \frac{4}{3}\gamma(x^*) < 2\gamma(x_0)$$

Our induction hypothesis is: $\alpha(x_i) < 2^{-3-2^{i+1}}$ for all $i \leq s$. We also assume by induction that $x_i \in U$ so that $\gamma(x_i)\delta < 2^{-4-2^{s+2}}$. Combining Lemma 7 and Lemma 8 we get:

$$\alpha(x_{i+1}) < \frac{1}{\psi(\alpha(x_i) + \gamma(x_i)\delta)^2}\left(\gamma(x_i)\delta + (\alpha(x_i) + \gamma(x_i)\delta)^2\right)$$

Using the induction hypothesis, we can further simplify:

$$\alpha(x_{i+1}) < 2\alpha(x_i)^2 + 2\gamma(x_i)\delta$$

Replacing $\alpha(x_i)$ by $2^{-3-2^{i+1}}$ and $\gamma(x_i)\delta$ by $2^{-4-2^{s+2}}$ one obtains

$$\alpha(x_{i+1}) < 2^{-5-2^{i+2}} + 2^{-4-2^{s+2}} < 2^{-3-2^{i+2}}$$

since $i \leq s$. Then we need to verify that $x_{i+1} \in U$. We use Lemma 10 item 1:

$$\beta(x_{i+1}) \leq |\delta|\frac{(1-\gamma(x_i)|h|)^2}{\psi(\gamma(x_i)|h|)} + |h|^2\gamma(x_i)\frac{1-\gamma(x_i)|h|}{\psi(\gamma(x_i)|h|)}$$

In the right-hand term, we can bound $\gamma(x_i)|h|$ by $\alpha(x_i) + \gamma(x_i)\delta$. By the induction hypothesis, this is less than $2^{-5} + 2^{-8}$. Hence,

$$\beta(x_{i+1}) \leq \frac{1}{\gamma(x_i)}\left(2^{-8}\frac{\left(1-(2^{-5}+2^{-8})\right)^2}{1-4(2^{-5}+2^{-8})} + (2^{-5}+2^{-8})^2\frac{1-(2^{-5}+2^{-8})}{1-4(2^{-5}+2^{-8})}\right)$$

Besides, $\gamma(x_i) > \frac{1}{2}\gamma(x^*)$, so

$$\beta(x_{i+1}) < \frac{2}{2^7\gamma(x^*)} < \frac{1}{20\gamma(x^*)}$$

This was for $i < s$. When $i = s$, it is easy to bound using Lemmas 9 and 10 that

$$|x_i - x^*| < \frac{2^{-2^{s+1}-1}}{\gamma(x^*)}$$

and then, since $s \geq 2$, we are in the conditions of Lemma 11, for

$$\rho = \frac{2^{-2-2^{s+1}}}{\gamma(x^*)}$$

This is mapped into a $\rho/10$ neighborhood of x^*. Since the error δ is less than $\rho/2$, the sequence of x_i will never leave the ρ-neighborhood. $\square$

We state the following Lemma, that we will need later on:

Lemma 12. *Let f be a polynomial and let x^* be a single root of f. Let $|x - x^*| < 2^{-6}/\gamma(x^*)$. Then $\alpha(f, x) < 2^{-5}$.*

Proof of Lemma 12. By Lemma 7 item 2,

$$\beta(x) < |x - x^*|\frac{(1-\gamma(x^*)|x-x^*|)^2}{\psi(\gamma(x^*)|x-x^*|)} + \gamma(x)|x-x^*|^2\frac{1}{\psi(\gamma(x^*)|x-x^*|)}$$

Lemma 9 implies that $\gamma(x) < \frac{1}{1-4\times 2^{-6}}\gamma(x^*)$. Putting all together,

$$\alpha(x_i) < \frac{2^{-6}+2^{-12}}{(1-4\times 2^{-6})^2} < 2^{-5}$$

$\square$

5.2. Approximation of Algebraic Numbers.

Theorem 14. *Let K be a number field. Let f be a degree d polynomial with coefficients in K. Let x be a single root of f. Let $x_0 \in \mathbb{C}$ be such that $f'(x) \neq 0$ Then the following bounds are true:*

1. $w(\gamma(f, x_0)) \leq [K[x_0] : \mathbb{Q}] \left(d^2 \lceil \log_2 d \rceil + d^3 \lceil \log_2 H(x_0) \rceil + d^2 \lceil \log_2 H(f) \rceil \right)$
2. $w(\gamma(f, x)) \leq 3d^4 w(f)$
3. $\gamma(f, x) \leq 2^{3d^4 w(f)}$
4. *If f is the minimal polynomial of x over $\mathbb{Q}$, then* $\gamma(f, x) \leq 2^{3d^4 w(f)} \leq 2^{3w(x)^5}$

Proof of Theorem 14. In order to bound the weight of

$$\gamma(x_0) = \max_{k \geq 2} \left(\frac{|f^{(k)}(x_0)|}{k! |f'(x_0)|} \right)^{\frac{1}{k-1}}$$

we use Theorem 4 with $F_0(f, y) = f^{(k)}(y)$ and $F_1(y) = k! f'(y)$, $y = (x_0 : 1)$. We get

$$H(F_0(y) : F_1(y)) \leq (d-1) \frac{d!}{d-k\,!} H(f) H(y)^{d-1}$$

A gross estimate of $k!$ and of $d - k\,!$ is $d!$. So we deduce that

$$\begin{aligned} H(\gamma(x_0)) &\leq \max_{k \geq 2} \left(d!(d-1) H(f) H(x_0 : 1)^{d-1} \right)^{\frac{1}{k-1}} \\ &\leq d!(d-1) H(f) H(x_0 : 1)^{d-1} \end{aligned}$$

We can also bound

$$[\mathbb{Q}[\gamma(x_0)] : \mathbb{Q}] \leq (d-1)[K[x_0] : K][K : \mathbb{Q}]$$

Hence,

$$w(\gamma(x_0)) \leq (d-1)[K[x_0] : K][K : \mathbb{Q}] \left(1 + \lceil \log_2 d!(d-1) H(f) H(x)^{d-1} \rceil \right)$$

and the first part of the Theorem follows. The second part is a consequence of Theorem 5. Then we use Lemma 1 to deduce part 3, and Lemma 6 for part 4. $\square$

5.3. Proof of Theorem C.

Proof of Theorem C. Let x be an algebraic number, and let $d = [\mathbb{Q}[x] : \mathbb{Q}]$. Let f be the minimal polynomial of x over $\mathbb{Q}$. By Corollary 4,

$$H(f) \leq 2^{w(x)}$$

and therefore f can be represented by $d + 1$ numbers of $1 + w(x)$ bits.

Theorem 14 item 4 implies:

$$\gamma(x) \leq 2^{3d^4 w(f)}$$

Let x_0 be an approximation of f, such that $|x - x_0| < \frac{2^{-6}}{\gamma(x)}$. It is enough to choose x_0 within a radius of

$$|x - x_0| < 2^{-6-3d^4 w(f)}$$

from x. One possibility is to truncate the real and imaginary part into fixed points of $8 + 3d^4 w(f) + w(x)$ bits of precision (using fixed point representation).

We are under the conditions of Lemma 12, so $\alpha(x_0) < 2^{-5}$.

We can now produce the function $y(x)$. It is a list (d, w, f, x_0) where $w = \lceil w(x) \rceil$, and d, f and x_0 are as above. Theorem 6 yields $w(x) < \text{Bitsize}(y(x))$.

It is time to construct the machine M that will provide the 2^{-k}- approximations of x. First of all, we can bound $\gamma(f, x_0)$ by applying Lemma 8. The following conclusion is not sharp, but is convenient:

$$\frac{1}{2}\gamma(x) \leq \gamma(x_0) \leq 2\gamma(x)$$

The next step is to choose s such that

$$\frac{2^{1-2^s}}{\gamma(x_0)} < 2^{-k}$$

Using $w(\gamma(x_0)) \leq Cd^7 w$, we restrict our choice to:

$$2^{1-2^s} < 2^{-k-Cd^7 w}$$

A good choice is $s = \lceil \log_2 \left(k + 1 + Cd^7 w\right) \rceil$. In that case, $s \leq \log_2 k + 7 \log_2 d + \log_2 w + C'$, for some constant C'. The machine M will perform s approximate Newton iterations, obtaining at each step x_i with accuracy

$$\delta = 2^{-4-Cd^7 w - 2^{s+2}} < \frac{2^{-4-2^{s+2}}}{\gamma(x_0)}$$

This corresponds to $4 + Cd^7 w + 2^{s+2}$ bits of precision in each x_i. This can be achieved by using precision of $kw\mathrm{Poly}(d)$ bits of fixed point, and truncating.

Therefore, the total running time can be bounded in terms of $k \log_2 k \; w \log_2 w \; \mathrm{Poly}(d)$ bit operations, or more precisely:

$$O(k \log_2 k \; w \log_2 w \; d^8 \log_2 d)$$

where the exponent in d is not intended to be sharp. □

6. Bit representation of Algebraic Numbers

Algebraic numbers are essentially discrete objects. They can be represented by a minimal polynomial. We have seen in Theorem 6 and C that the bit-length of such a representation is related to the weight of the algebraic number.

We will start this section by proving Theorem D. The main tool in the proof is the basis reduction algorithm by Lenstra, Lenstra and Lovasz [LLL82] (LLL for short). This algorithm provides an effective procedure to check for algebraic dependence.

The main results about that algorithm are only quoted below. Since this algorithm is defined over the integers (with the bit model), a few extra estimates will be necessary.

Next, we would like to extend the ideas of Theorem C to the case where we have several algebraic numbers $\alpha_1, \ldots, \alpha_n$. It makes sense to represent them as polynomials $q_i(\alpha)$, where α is a primitive element of $\mathbb{Q}[\alpha_1, \cdots, \alpha_n]$.

Definition 20. An integer representation of $\alpha_1, \ldots, \alpha_n \in (\mathbb{Q}^a)^n$ is a list $(y(\alpha), q_1, \cdots, q_n) \in \mathbb{Z}^\infty$ where

1. α is a primitive element of $\mathbb{Q}[\alpha_1, \cdots, \alpha_n]$
2. $y : \mathbb{Q}^a \to \mathbb{Z}^\infty$ is the function defined in Theorem C, where $\mathbb{Q}^a$ is the ring of algebraic numbers.
3. $q_1, \cdots, q_n$ are densely represented polynomials with rational coefficients such that $\alpha_i = q_i(\alpha)$

We will then prove a theorem on the complexity of basic arithmetic operations over integer representations of algebraic numbers. Namely,

Theorem 15. *The following operations can be executed in polynomial time over $(\mathbb{Z}, >, w)$:*

1. *Merging: given an integer representation of $a_1, \cdots, a_n$ and an integer representation of $b_1, \cdots, b_m$, compute one integer representation of $a_1, \cdots, a_n, b_1, \cdots, b_m$.*

2. *Arithmetic: given integer representation for x, y, compute integer representation for $x, y, x+y, xy, x-y$.*
3. *Decision: given an integer representation for x, decide $x = 0$, or $x \in \mathbb{R}$, or if this is the case, whether $x > 0$.*
4. *Restriction: Given an integer representation for $a_1, \cdots, a_n$, produce an integer representation of a_n, of size polynomially bounded in $w(a_n)$.*

Theorem 15 will be proved using a modification of well-known symbolic algebra algorithms (namely Collins' SIMPLE algorithm for primitive elements).

One consequence of Theorem 15 and Theorem D together will be that computation models over the integers $(\mathbb{Z}, >, w)$ and over the real algebraic numbers $(\mathbb{Q}^{\mathrm{ra}}, >, w)$ are polynomially equivalent, finishing the proof of Theorem A.

6.1. **Lattice basis reduction.**

Definition 21. A Lattice is a subset Λ of $\mathbb{R}^n$ of the form:

$$\Lambda = \{Az : z \in \mathbb{Z}^n\}$$

where A is a (real or) integer $n \times n$ matrix.

We can restrict ourselves to the case where A is invertible. In that case, the columns of A are said to be a basis of the Lattice Λ. Of course a lattice admits an infinity of bases, and it is possible to pass from a basis to another basis through elementary column operations (adding a multiple of another column, or column permutation).

Given A, the LLL algorithm will perform those operations and output a 'reduced' basis. Reduced bases keep some of the good properties of orthonormal bases, while maintaining integer arithmetic. The price to pay is to replace orthogonality and norm-1 with certain bounds.

Following the notation of [LLL82], let $(b) = (b_1, \cdots, b_n)$ be a basis for a lattice Λ. The following vectors and numbers are associated to (b):

$$\begin{aligned} b_i^* &= b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j \\ \mu_{ij} &= \frac{< b_i, b_j^* >}{\|b_j^*\|^2} \end{aligned}$$

Vectors b_i^* correspond to the Gram-Schmidt orthogonalization of (b) in $\mathbb{R}^n$.

Definition 22 (LLL)**.** A basis (b) for a lattice Λ is said to be reduced if and only if

$$\begin{aligned} |\mu_{ij}| &\leq \tfrac{1}{2} && , \ 1 \leq j < i \leq n \\ |b_i^* + \mu_{i,i-1} b_{i-1}^*| &\geq \tfrac{3}{4} |b_{i-1}^*|^2 && , \ 1 < i \leq n \end{aligned}$$

The following result is Proposition 1.11 in [LLL82]:

Proposition 4. *Let $L \subset \mathbb{R}^n$ be a lattice with reduced basis $b_1, b_2, \cdots, b_n$. Then $|b_1|^2 \leq 2^{n-1}|x|^2$ for any $x \in L$, $x \neq 0$.*

Also, Proposition 1.9 ibid says that:

Proposition 5. *Let $L \subset \mathbb{R}^n$ be a lattice with reduced basis $b_1, b_2, \cdots, b_n$. Then $\|b_1\| \leq 2^{\frac{n-1}{4}} (\det L)^{\frac{1}{n}}$*

where the determinant of a lattice L is the determinant of any positively oriented basis of L.

Complexity of the basis-reduction algorithm goes as follows (Proposition 1.26 in [LLL82]):

Proposition 6. *Let $L \in \mathbb{Z}^n$ be a lattice with basis $b_1, \cdots, b_n$ and let $B \in \mathbb{R}, B \geq 2$ be such that $|b_i|^2 \leq B$ for $1 \leq i \leq n$. Then the number of arithmetic operations needed by the basis reduction algorithm described in* [LLL82] *equation (1.15) is $O(n^4 \log B)$, and the integers on which those operations are performed each have binary length $O(n \log B)$.*

This was used in [LLL82] to deduce that

Theorem 16. *The cost of factorizing $f \in \mathbb{Z}[x]$ into irreducible factors, in the bit model, is*

$$O((\deg f)^{12} + (\deg f)^9 (\log \|f\|_2)^3)$$

Before giving the proof of Theorem D, we will proceed like it is suggested in [LLL82] 1.39 to obtain the following, preliminary result:

Lemma 13. *There is a polynomial time machine in $(\mathbb{Q}^{\mathrm{ra}}, \geq, w)$ accepting input $w_0, n \in \mathbb{N}$ and $\alpha_0 = 1, \alpha_1, \cdots \alpha_n \in \mathbb{Q}^{\mathrm{ra}}$ with output $p \in \mathbb{Z}^\infty$ with the following properties:*

1. *If $w(\alpha_1, \cdots, \alpha_n) \leq w_0$, and there is a relation $\sum \hat{p}_i \alpha_i = 0$, not all the $\hat{p}_i$ equal to zero, then the machine returns some $p \neq 0$ such that $\sum p_i \alpha_i = 0$.*
2. *If the machine returns $p \neq 0$, then $\sum p_i \alpha_i = 0$.*
3. *The machine always terminates.*

Proof of Lemma 13.

Step 1. Let b be a constant in $\mathbb{N}^*$, to be precised later. Each of the α_i can be approximated by $\frac{a_i}{b}$, $a_i \in \mathbb{Z}$, with error $|\alpha_i - \frac{a_i}{b}| \leq \frac{1}{2b}$. The a_i's can be found by computing $b\alpha_i$ and rounding to the nearest integer.

Step 2. let $m \in \mathbb{Z}^{n+1}$, $m_i < B$ where B is another constant to be computed later. We assume here that $w(\alpha) < w_0$. Then we have two possibilities, namely

$$\sum_{i=0}^{n} m_i \alpha_i = 0$$

or

$$|\sum_{i=0}^{n} m_i \alpha_i| > \delta = (B(n+1))^{-w_0}$$

Indeed,

$$H(\sum m_i \alpha_i) \leq (n+1) \prod H(m_i \alpha_i) = n \prod m_i \prod H(\alpha_i)$$

Let $d = \deg[\mathbb{Q}[\alpha_1, \cdots, \alpha_n]]$ and assume that $\sum m_i \alpha_i \neq 0$. Then

$$\begin{aligned} |\sum m_i \alpha_i| &> H(\sum m_i \alpha_i)^{-d} \\ &> (n+1)^{-d} B^{-nd} \left(\prod H(\alpha_i)\right)^{-d} \\ &> (n+1)^{-d} B^{-nd} 2^{-nw_0} \end{aligned}$$

Also, setting $b > \frac{(n+1)B}{\delta}$, so that $\frac{(n+1)B}{2b} < \frac{\delta}{2}$, one obtains:

$$\left| \sum_{i=0}^{n} m_i \alpha_i - \sum_{i=0}^{n} m_i \frac{a_i}{b} \right| < \frac{\delta}{2}$$

Hence, under the assumptions $w(\alpha) < w_0$, $|m_i| < B$ one obtains:

$$\begin{aligned} |\sum m_i \frac{a_i}{b}| < \frac{\delta}{2} &\Rightarrow \sum m_i \alpha_i = 0 \\ |\sum m_i \frac{a_i}{b}| \geq \frac{\delta}{2} &\Rightarrow \sum m_i \alpha_i \neq 0 \end{aligned}$$

Step 3 We consider now the lattice Λ given by the basis:

$$U = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ \frac{a_0 C}{b} & \frac{a_1 C}{b} & \dots & \frac{a_n C}{b} \end{bmatrix}$$

where C is a multiple of b to be given later. Suppose that $u_1 = U \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_n \end{bmatrix}$ is the first vector of a reduced basis of Λ. According to Proposition 5,

$$\begin{aligned} \|u_1\| &\leq 2^{\frac{n}{4}} (\det U)^{\frac{1}{n+1}} \\ &= 2^{\frac{n}{4}} (\frac{C a_n}{b})^{\frac{1}{n+1}} \end{aligned}$$

Step 4. Now, since $u_1 = \begin{bmatrix} m_0 \\ \vdots \\ m_{n-1} \\ \sum_{i=0}^{n} \frac{C a_i}{b} m_i \end{bmatrix}$ we can estimate:

$$\left| \sum_{i=0}^{n} \frac{C a_i}{b} m_i \right| < 2^{\frac{n}{4}} \left(\frac{C a_n}{b} \right)^{\frac{1}{n+1}}$$

hence

$$\left| \sum_{i=0}^{n} \frac{a_i}{b} m_i \right| < 2^{\frac{n}{4}} C^{-\frac{n}{n+1}} \left(\frac{a_n}{b} \right)^{\frac{1}{n+1}}$$

so if we fix $C > \left(\frac{2}{\delta} 2^{-\frac{n}{4}} |\alpha_n + \frac{1}{2b}|^{-\frac{1}{n+1}} \right)^{1+\frac{1}{n}}$ we obtain

$$\left| \sum_{i=0}^{n} \frac{a_i}{b} m_i \right| < \frac{\delta}{2}$$

Step 5. In order to guarantee the condition in Step 2, we need to give B where $|m_i| < B$ for the reduced basis we will obtain. Let us assume that $w(\alpha) \leq w_0$. Also, we assume that $\sum \hat{p}_i \alpha_i = 0$, with $\hat{p}_i$ minimal, $\hat{p}_i \neq 0$. Then according to Theorem 6, $|\hat{p}_i| < 2^{w_0}$. Proposition 4 lets us choose

$$B = 2^{n+w_0}$$

Then we choose δ and b as above and C as in step 4. All the operations done so far involve integer values with polynomially bounded size. The a_i's can be obtained by binary search (also in polynomial time). The LLL algorithm is also polynomial time. At the end of the program, we check if $\sum p_i \alpha_i = 0$. $\square$

Remark 5. Felipe Cucker pointed out that in the argument above, only the binary search and the final test $\sum p_i \alpha_i = 0$ involve computations over $\mathbb{Q}^{\mathrm{ra}}$. All other computations can be performed in a Turing machine.

Proof of Theorem D. Let $x \in \mathbb{Q}^{\mathrm{ra}}$. If we knew $d = [\mathbb{Q}[x] : \mathbb{Q}]$ and a reasonable bound for w_0, we could just apply Lemma 13 to input $1, x, x^2, \cdots, x^d$ to obtain the minimal polynomial of x. Since we don't, we proceed as follows:

Start with bounds $d = 1, w_0 = 2$, and apply Lemma 13. In case of failure ($p = 0$), multiply d and w_0 by 2 and start again.

The machine in Lemma 13 will eventually succeed, with some $d < 2[\mathbb{Q}[x] : \mathbb{Q}]$ and some $w_0 < 2dw(x)$. Those bounds are polynomial in $w(x)$.

Next, we use Theorem 16 to factorize p over $\mathbb{Q}$, so we obtain the minimal polynomial of x.

At that time, we also have an upper bound w_0 for $w(x)$. Therefore, Theorem 14 implies that:

$$\gamma(x) < 2^{3w_0^5}$$

Recall from Lemma 12 that if $|x - x'| < 2^{-6}/\gamma(x)$, then $\alpha(x') < 2^{-5}$ and x' is a suitable approximation of x. So all that we need is to compute an approximation of x with precision $2^{-6-3w_0^5}$. This can be done by bisection, in polynomial time, in such a way that the denominator is indeed $2^{-6-3w_0^5}$. □

6.2. **Symbolic algebra.**

Proof of Theorem 15. The most difficult part of the proof of Theorem 15 is the first.

We will follow Collins' SIMPLE algorithm, as explained in [Loo83]. However, instead of using isolating intervals, we will use approximate roots. Also, due to the estimates on weights, some of the steps will have to be modified.

Item 1 reduces to the following problem.

Problem 1 (Extension)**.** Let A and B be minimal polynomials of a and b, respectively. Let a' and b' be approximate roots associated to a and b. Produce a minimal polynomial C for a primitive element c of $\mathbb{Q}[a, b]$, together with an approximate root c' of C associated to c. Also, produce rational polynomials q_1 and q_2 such that $a = q_1(c)$ and $b = q_2(c)$.

We need a polynomial time algorithm for solving the Extension Problem. Let us fix some notation. If f and g are polynomials in y, $f = \sum_{i=1}^m f_i y^i$ and $g = \sum_{i=1}^n g_i y^i$, $f_m \neq 0$, $g_n \neq 0$, the Sylvester matrix $\mathrm{Syl}_y(f, g)$ is the matrix associated to the operator:

$$a, b \mapsto af + bg$$

where a and b are polynomials of degree $n - 1$ and $m - 1$, respectively:

$$\mathrm{Syl}_y(f,g) = \begin{bmatrix} f_0 & & & & g_0 & & & \\ f_1 & f_0 & & & g_1 & g_0 & & \\ f_2 & f_1 & \ddots & & \vdots & g_1 & \ddots & \\ \vdots & f_2 & \ddots & f_0 & & \vdots & \ddots & g_0 \\ f_m & & \ddots & f_1 & g_n & & & g_1 \\ & f_m & & f_2 & & g_n & & \vdots \\ & & \ddots & \vdots & & & \ddots & \\ & & & f_m & & & & g_n \end{bmatrix}$$

The resultant $\mathrm{Res}_y(f, g)$ is just

$$\mathrm{Res}_y(f, g) = \det\ \mathrm{Syl}_y(f, g)$$

It is obvious that if f and g have a common root then $\mathrm{Res}_y(f, g)$ vanishes. The converse is easy (assume that $af + bg \equiv 0$ and suppose that f and g have no common root. Then a has to vanish at the roots of g, with multiplicity, etc...)

This is still valid if the coefficients of f and g belong to the transcendental extension $\mathbb{Q}[x]$.

We can now produce a primitive element c for $\mathbb{Q}[a, b]$ as follows. Let

$$r(x, t) = \mathrm{Res}_y(A(x - ty), B(y))$$

where x is a formal variable. We need to compute $r(x, t)$ as a polynomial in x for a fixed value of t. We will choose t such that $r(x, t)$ has no multiple root (as a polynomial in x).

Note that the roots of $r(x,t)$ are the expressions $c_{ij} = a_i + tb_j$ where a_i and b_j are the roots of A and B, respectively.

Therefore, if we set t large enough (namely $t > 2^{2+w(a)+w(b)}$) we get:

$$|a_i - a_j| < 2^{w(a_i - a_j)} \leq 2^{1+w(a)}$$

because of Lemma 1.

$$|b_k - b_l| > 2^{-1-w(b)}$$

by the same reason. Thus

$$|a_i - a_j| < t|b_k - b_l|$$

and hence

$$a_i - tb_k \neq a_j - tb_l$$

It follows that $r(x,t)$ has no multiple root.

We can use Theorem 16 to find out an irreducible factor of $r(x,t)$ vanishing in $a_i + tb_k$. Let $C(x)$ be that factor.

Our primitive element is $c = a_i + tb_k$. The weight of c is polynomially bounded on the input size. An approximation c' can be computed by using estimate on $w(c)$ and Lemma 11.

The polynomial $r(x,t)$ is a polynomial of degree no more than $(\deg A)(\deg B)$ in x. i

For any x in $\{0; 1; (\deg A)(\deg B)\}$, it is possible to compute, in polynomial time, $\mathrm{Res}_y(A(x - ty), B(y))$. Moreover,

$$\mathrm{Res}_y(A(x - ty), B(y))_{|x=i} = \mathrm{Res}_y(A(i - ty), By)$$

so that the coefficients of $r(x,t)$ may be obtained by interpolation.

We can use Theorem 16 to deduce that an irreducible factor of $r(x,t)$ vanishing in c can be computed in polynomial time. Let $C(x)$ be that factor.

Let $D(x) = \gcd(A(c - tx), B(x))$. The polynomial $D(x)$ can be computed symbolically with arithmetic in $\mathbb{Q}[c]$. Its only root is b_k, so it has degree 1, and is of the form: $D(x) = x - q_2(c)$.

We can also set $q_1(x) = x - tq_2(x)$ so that $q_1(c) = a$.

Item 2: Let $x = q_1(a)$ and $y = q_2(a)$. Since A is the minimal polynomial of a,

$$\begin{aligned} x + y &= (q_1 + q_2)(a) \\ x - y &= (q_1 - q_2)(a) \\ xy &= (q_1 q_2 \mod A)(a) \end{aligned}$$

Item 3: If $x = q(a)$ we have, by Corollary 1 that:

$$H(x) \leq SH(q)H(a)^{\deg q}$$

Also,

$$\begin{aligned} w(x) &= [\mathbb{Q}[x] : \mathbb{Q}] \lceil 1 + \log_2 H(x) \rceil \\ &\leq [\mathbb{Q}[a] : \mathbb{Q}] \lceil 1 + (\deg q) \log_2 H(a) + \log_2 H(q) \rceil \\ &\leq (\deg q) w(a) + w(q) \end{aligned}$$

If $x \neq 0$ then $|x| > 2^{-w(x)}$. Thus it is enough to compute x up to accuracy $2^{-w(x)-1}$. If we evaluate a with accuracy 2^{-b}, then $q(a)$ may be evaluated with accuracy $2^{-b}|a|^{\deg q - 1} \max |q_i|$. All this can be done in polynomial time (Theorem C).

Item 4. Let q be such that $a_n = q(a)$. Using arithmetic in $\mathbb{Z}[x] \mod A$, we can compute 1, $q(x)$, $q^2(x) \mod A$, $q^3(x) \mod A$, etc..., and check for linear independence. Once we get linear dependency, we solve the system:

$$Qp = 0$$

where Q is the matrix with columns $1, q(x), \ldots, q(x)^i \mod A(x)$, and where each polynomial is represented by its coefficients. □

6.3. $\mathcal{P} \neq \mathcal{NP}$ **over** $(\mathbb{Z}, >, w) \Leftrightarrow \mathcal{P} \neq \mathcal{NP}$ **over** $(\mathbb{Q}^{\mathrm{ra}}, >, w)$**.** We complete here the proof of Theorem A. Define:

$$\begin{array}{rccl} b: & \mathbb{Z}^{\infty} & \to & \mathbb{Q}^{\mathrm{ra}} \\ & y & \mapsto & \lim_{k \to \infty} M(k, y) \end{array}$$

where M is the machine defined in Theorem C.

Let us recall the construction of M. The list y is interpreted as the coefficients of an irreducible polynomial p and an initial approximate root. $M(k, y)$ computes then a certain number of Newton iterations. If the limit is defined, this means that $b(y)$ is a root of p.

I claim that b is a morphism from $(\mathbb{Q}^{\mathrm{ra}}, >, w)$ into $(\mathbb{Z}, >, w)$. Indeed, surjectivity (M1) and honesty (M2) follow from Theorem C.

The domain of definition of b is the class of y such that Newton iteration will converge to a root of p. Combining Theorem C, Lemma 12 and Theorem 14, we can check the domain of b in polynomial time. This implies property (M3).

Theorem 15 implies that b is 'very honest' and satisfies the hypotheses of Lemma 5. Therefore, b has the pull-back property (M4).

We define also the inclusion morphism from $(\mathbb{Z}, >, w)$ into $(\mathbb{Q}^{\mathrm{ra}}, >, w)$:

$$\begin{array}{rccl} i: & \mathbb{Q}^{\mathrm{ra}} \supseteq \mathbb{Z} & \to & \mathbb{Z} \\ & x & \mapsto & x \end{array}$$

It remains to produce machines A_1 and A_2. The machine A_1 is fairly trivial: it associates, to every $x \in \mathbb{Z}$, its corresponding $y(x)$.

The machine A_2 is the polynomial time machine from Theorem D. It associates to every $x \in \mathbb{Q}^{\mathrm{ra}}$, its $y(x)$

Thus, models $(\mathbb{Z}, >, w)$ and $(\mathbb{Q}^{\mathrm{ra}}, >, w)$ are polynomially equivalent and we can apply Theorem 10 to deduce that

$$\mathcal{P} \neq \mathcal{NP} \text{ over } (\mathbb{Z}, >, w) \ \Leftrightarrow \ \mathcal{P} \neq \mathcal{NP} \text{ over } (\mathbb{Q}^{\mathrm{ra}}, >, w)$$

Almost the same argument shows that $\mathcal{P} \neq \mathcal{NP}$ over $(\mathbb{Z}, >, w)$ if and only if $\mathcal{P} \neq \mathcal{NP}$ over $(K, >, w)$ where K is a number field, considered as a ring. Together with Propositions 2 and 3, this concludes the proof of Theorem A

Define

$$\begin{array}{rccl} \sigma_1: & \mathbb{Z} & \to & \mathbb{Q}^{\mathrm{ra}} \\ & n & \mapsto & n \end{array}$$

and

$$\begin{array}{rccl} \sigma_2: & \mathbb{Q}^{\mathrm{ra}} & \to & \mathbb{Z} \\ & x & \mapsto & y(x) \end{array}$$

where $y(x)$ is the function in Theorem C.

It follows from Theorem 15 that σ_2 is a morphism. Moreover, from Theorem D we can deduce that $\sigma_1 \circ \sigma_2$ is in polynomial time over $(\mathbb{Z}, >, w)$. Hence, we can apply Theorem 10.

References

[AB94] Ilan Adler and Peter A. Beling. Polynomial algorithms for linear programming over the algebraic numbers. *Algorithmica*, 12:436–457, 1994.

[AM69] M. F. Atiyah and I. G. Macdonald. *Introduction to Commutative Algebra*. Addison Wesley, Reading MA, 1969.

[BCSS96] Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. Algebraic settings for the problem $\mathcal{P} \neq \mathcal{NP}$. In Steve Smale James Renegar, Michael Shub, editor, *The Mathematics of Numerical Analysis*, volume 32 of *Lectures in Applied Mathematics*, pages 125–144. American Mathematical Society, 1996.

[BCSS98] Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. *Complexity and Real Computation*. Springer, 1998.

[BS67] Z. I. Borevich and I. R. Shafarevich. *Théorie des Nombres*. Gauthier-Villars, Paris, 1967. Reprinted by Editions Jacques Gabay, 1993.

[BSS89] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: $\mathcal{NP}$-completeness, recursive functions and universal machines. *Bulletin of the AMS*, 21(1), 1989.

[Cas67] J. W. S. Cassels. Global fields. In *Algebraic Number Theory (Proc. Instructional Conf., Brighton, 1965)*, pages 42–84. Thompson, Washington, D.C., 1967.

[CG97] Felipe Cucker and Dima Grigoriev. On the power of real Turing machines over binary inputs. *SIAM Journal on Computing*, 26(1):243–254, 1997.

[CK95] Felipe Cucker and Pascal Koiran. Computing over the reals with addition and order: higher complexity classes. *Journal of Complexity*, 11(3):358–376, 1995.

[Coh93] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[KA82] L. V. Kantorovich and G. P. Akilov. *Functional analysis*. Pergamon Press, Oxford, second edition, 1982. Translated from the Russian by Howard L. Silcock.

[Knu98] Donald E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, Reading MA, third edition edition, 1998.

[Koi94] Pascal Koiran. Computing over the reals with addition and order. *Journal of Complexity*, 133:486–495, 1994.

[Koi97] Pascal Koiran. A weak version of the Blum, Shub, and Smale model. *Journal of Computer and System Sciences*, 54(1, part 2):177–189, 1997. 1st Annual Dagstuhl Seminar on Neural Computing (1994).

[KP95] Teresa Krick and Luis M Pardo. A computational method for diophantine approximation. In *Proceedings of MEGA'94*, Progress in Mathematics. Birkhauser, 1995.

[Lan86] Serge Lang. *Algebraic Number Theory*. Springer-Verlag, New York, 1986.

[Lan97] Serge Lang. *Survey of Diophantine Geometry*. Springer, Berlin, 1997. Corrected Second Printing.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and Laszlo Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[Loo83] R. Loos. Computing in algebraic extensions. In *Computer algebra*, pages 173–187. Springer, Vienna, 1983.

[Lov86] László Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*. CBMS-NSF Reg. Conference Series in Appl. Math. 50. SIAM, Philadelphia, 1986.

[Mal93] Gregorio Malajovich. *On the complexity of path-following Newton algorithms for solving systems of polynomial equations with integer coefficients*. PhD thesis, Berkeley, 1993.

[Mal94] Gregorio Malajovich. On generalized Newton algorithms: quadratic convergence, path-following and error analysis. *Theoret. Comput. Sci.*, 133(1):65–84, 1994. Selected papers of the Workshop on Continuous Algorithms and Complexity (Barcelona, 1993).

[Mic94] Christian Michaux. $\mathcal{P} \neq \mathcal{NP}$ over the nonstandard reals implies $\mathcal{P} \neq \mathcal{NP}$ over $\mathbb{R}$. *Journal of Complexity*, 133:95–104, 1994.

[Mig83] M. Mignotte. Some useful bounds. In *Computer algebra*, pages 259–263. Springer, Vienna, 1983.

[MM99] Gregorio Malajovich and Klaus Meer. On the structure of $\mathcal{NP}_{\mathbb{C}}$. *SIAM Journal on Computing*, 28(1), 1999.

[MW97] Michael Maller and Jennifer Whitehead. Computational complexity over the p-adic numbers. *Journal of Complexity*, 13(2):195–207, 1997.

[Sil86] Joseph A. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics 106. Springer, New York, 1986.

[Sma86] Steve Smale. Newton's method estimates from data at one point. In *The merging of disciplines: new directions in pure, applied, and computational mathematics (Laramie, Wyo., 1985)*, pages 185–196. Springer, New York, 1986.

[Sma98] Steve Smale. Mathematical problems for the next century. *Math. Intelligencer*, 20(2):7–15, 1998.

[SS93] Michael Shub and Steve Smale. Complexity of Bézout's theorem. I. Geometric aspects. *J. Amer. Math. Soc.*, 6(2):459–501, 1993.

Departamento de Matemática Aplicada, Universidade Federal do Rio de Janeiro. Caixa Postal 68530, CEP 21945, Rio de Janeiro, RJ, Brasil

E-mail address: `gregorio@labma.ufrj.br`

Current address: On leave at MSRI, 1000 Centennial Drive, Berkeley CA 94720-5070, e-mail: `gregorio@msri.org`