Capturing the Computational Complexity of Geometric Problems by the First-Order Theory of the Reals

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Paul Jungeblut

Tag der mündlichen Prüfung: 26.04.2024

1. Referent:	PD Dr. Torsten Ueckerdt
2. Referent:	Prof. Marcus Schaefer

Acknowledgements

I would like to express my sincere gratitude to everyone who supported me during my time as a PhD student in the *Algorithmics* group at the Karlsruhe Institute of Technology under the supervision of PD Dr. Torsten Ueckerdt. This thesis would not have been possible without the continuous support of all of you. In particular, I would like to thank

- PD Dr. Torsten Ueckerdt for his valuable guidance during the last years, starting with the supervision of my master's thesis in 2019, before offering me a position as his first PhD student. Thank you for all the time you invested in me, for suggesting interesting problems to work on, for the many joint projects, and for introducing me to the right people to learn from and to work with.
- Prof. Dr. Dr. h.c. Dorothea Wagner for welcoming me to her research group, for generously supporting me traveling to numerous workshops and conferences, and her valuable advice regarding the whole graduation process.
- Prof. Marcus Schaefer for serving as a reviewer for this thesis, as well as dedicating a lot of time to discuss $\exists\mathbb{R}$ and related concepts with me.
- Prof. Dr. Reussner for agreeing to serve as an examiner in my thesis defense.
- Torsten Ueckerdt, Laura Merker and Miriam Goetze for providing valuable and honest feedback to preliminary versions of this thesis and earlier publications.
- Linda Kleist and Tillmann Miltzow for introducing me to the $\exists\mathbb{R}$-world and for the exciting workshop experiences.
- my co-authors Lukas Barth, Daniel Bertschinger, Nicholas Bieker, Thomas Bläsius, Guido Brückner, Emil Dohse, Miriam Goetze, Minh Tuan Ha, Christoph Hertrich, Robert Hickingbotham, Linda Kleist, Laura Merker, Tillmann Miltzow, Marcel Radermacher, Samuel Schneider, Torsten Ueckerdt, Simon Weber, David R. Wood and Paweł Żyliński. I learned a lot from you and it was always a pleasure to work with you.
- everyone in the Algo and Scale groups, i.e., Lukas Barth, Thomas Bläsius, Guido Brückner, Valentin Buchhold, Adrian Feilhauer, Miriam Goetze, Lars Gottesbüren, Max Göttlicher, Sascha Gritzbach, Michael Hamann, Jean-Pierre von der Heydt, Maximilian Katzmann, Tamara Mchedlidze, Laura Merker, Marcel Radermacher, Jonas Sauer, Torsten Ueckerdt, Dorothea Wagner, Franziska Wegner, Christopher Weyand, Marcus Wilhelm, Matthias Wolf, Wendy

Yi, Tim Zeitz, Michael Zündorf and Tobias Zündorf. I really enjoyed the open, creative and stimulating atmosphere. The regular coffee breaks have always been a great opportunity to informally discuss “bold claims” and “stupid ideas”, sometimes leading to serious insights. Also, not to forget the many high quality (!) table soccer matches and the amusing but top-secret “Gerüchteküche”.

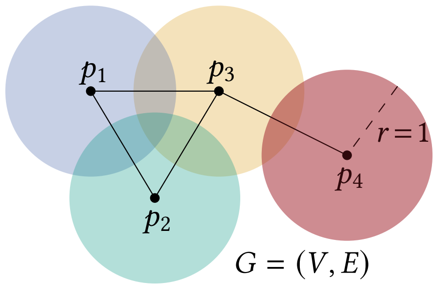
- Lilian Beckert, Isabelle Junge, Ralf Kölmel and Tanja Wehrmann, for their administrative and technical support that allowed me to focus on my research.
- my family and friends, for keeping me motivated, for providing feedback, for deviating my thoughts away from computer science, and for keeping up the “Mensa-Mittwoch” tradition.

Contents

1 Introduction

In this thesis, we explore a young branch of computational complexity that deals with problems whose solutions consist of *real* numbers subject to a given set of polynomial constraints. Naturally, many of these problems come from computational geometry (the solutions may be coordinates of points), but examples are found in nearly all branches of computer science and mathematics. In particular, we study a complexity class called $\exists\mathbb{R}$ that was proposed by Schaefer in 2009 [133]. It contains NP [144] and is itself contained in PSPACE [37]. The class $\exists\mathbb{R}$ allows us to precisely capture the computational complexity of many algorithmic problems whose complexity remained open in terms of the more established classes NP and PSPACE. Before we give a formal definition in Section 1.1, let us consider two examples for such problems. As we are going to see, their commonality is the underlying geometry, even though this might not be evident on first sight.

A famous family of problems from computational geometry is to decide whether a given graph admits various kinds of *intersection representations*: Here, the vertices are mapped to sets such that two vertices are adjacent if and only if the two corresponding sets intersect. Our first example problem is the recognition of *unit disk graphs*, i.e., graphs that admit an intersection representation by disk with radius 1 in $\mathbb{R}^2$, see Figure 1.1.



$$V = \{p_1, p_2, p_3, p_4\}$$

$$E = \left\{\{p_i, p_j\} \mid i \neq j \wedge \|p_i - p_j\| \leq 2\right\}$$

Figure 1.1: A unit disk graph with a corresponding intersection representation.

Our second example is the problem to train a neural network, more specifically to tweak its parameters such that it performs well on a given set of training data. While not as evident as above, this can also be understood as a geometric problem: A neural network realizes a function and the graph of this function can be studied from a geometric perspective. The functions realized by simple neural networks are very tame, and the geometry of their graphs is widely studied and well understood [91]. Figure 1.2 presents a neural network with two inputs, a single hidden layer, and one output. It realizes a continuous piecewise linear function $f\colon \mathbb{R}^2 \to \mathbb{R}$, whose graph consists of polygonal cells with constant gradient.

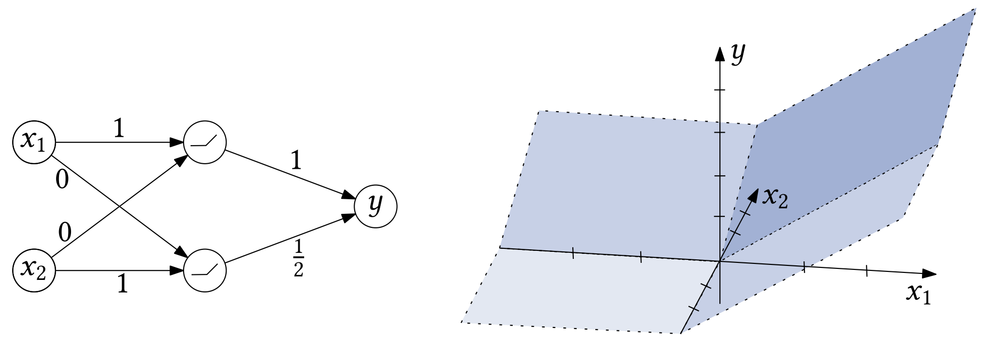


Figure 1.2: A neural network realizing a continuous piecewise linear function.

We consider both of these and more geometric problems from a complexity perspective, in particular focusing on their computational hardness. It is not surprising that both problems are difficult to solve, and indeed both are known to be NP-hard for more than twenty years [25, 31]. However, NP-membership and therefore also NP-completeness remains an open problem in both cases. Only recently it has been proven that recognizing unit disk graphs and training neural networks are both $\exists\mathbb{R}$-complete and therefore "equally difficult" [4, 18, 112] (the result about training neural networks is contained in this thesis). In particular, under the assumption that NP $\subsetneq \exists\mathbb{R}$, this implies that neither problem is in NP.

In this thesis, we study several $\exists\mathbb{R}$-complete problems with the goal to learn more about the complexity class $\exists\mathbb{R}$ itself. We prove $\exists\mathbb{R}$-completeness of interesting and important problems from the literature whose complexity remained open until now. In doing so, we sharpen the tools used for previous $\exists\mathbb{R}$-hardness results and also develop new techniques. In addition, we explore natural extensions of the class $\exists\mathbb{R}$ to be able to determine the exact computational complexity of even more difficult geometric problems.

1.1 The Existential Theory of the Reals and $\exists\mathbb{R}$

We have yet to define $\exists\mathbb{R}$ formally. Intuitively, $\exists\mathbb{R}$ contains all problems that can be expressed as a polynomial system of equations and inequalities. To this end, we define the following decision problem that serves as a canonical complete problem for $\exists\mathbb{R}$:

Definition 1.1 (Existential Theory of the Reals (ETR)).
Input: *An integer $n \in \mathbb{N}$, symbols $X_1, \dots, X_n$ for variables and a quantifier-free first-order formula $\varphi(X_1, \dots, X_n)$ whose atoms are polynomial equations and inequalities in these variables.*
Question: *Is the sentence*

$$\exists X_1, \dots, X_n \in \mathbb{R} : \varphi(X_1, \dots, X_n)$$

true, i.e., are there real numbers $x_1, \dots, x_n \in \mathbb{R}$ such that $\varphi(x_1, \dots, x_n)$ is true?

We define the syntax and semantics of first-order formulas rigorously in Section 2.1 below. For now, let us stick to some examples. The sentence $\exists X \in \mathbb{R} \colon X^2 < 0$ is false because squares of real numbers are always non-negative. On the other hand, the sentence $\exists X, Y, Z \in \mathbb{R} \colon Y = X^2 \wedge Z = X^3$ is true: Its set of solutions is $(X, Y, Z) = (t, t^2, t^3) \in \mathbb{R}^3$ for all $t \in \mathbb{R}$ and known as the *twisted cubic.*

Definition 1.2 ($\exists\mathbb{R}$). *The complexity class* $\exists\mathbb{R}$ *contains exactly* ETR *and all decision problems that are polynomial-time many-one reducible to it.*

We mentioned two $\exists\mathbb{R}$-complete problems already: the recognition of unit disk graphs and the training problem for neural networks. A short survey of further $\exists\mathbb{R}$-completeness results across different disciplines of computer science and mathematics follows in Section 1.4, illustrating that $\exists\mathbb{R}$ indeed appears in a wide variety of topics.

The canonical way to prove $\exists\mathbb{R}$-membership is by a reduction to ETR. For example, this is straightforward for the recognition problem of unit disk graphs: Given a graph $G = (V, E)$ with vertices $v_1, \dots, v_n$, we seek points $p_1, \dots, p_n \in \mathbb{R}^2$ such that for all $i, j \in \{1, \dots, n\}$ the distance between p_i and p_j is at most 2 if and only if v_i and v_j are adjacent. This question is equivalent to the following ETR instance[1]:

$$\exists p_1, \dots, p_n \in \mathbb{R}^2 \colon \bigwedge_{\{u,v\} \in E} \|p_u - p_v\|^2 \leq 2^2 \wedge \bigwedge_{\{u,v\} \notin E} \|p_u - p_v\|^2 > 2^2 \tag{1.1}$$

The length[2] of sentence (1.1) is polynomial in $|V| + |E|$. Thus, we obtain $\exists\mathbb{R}$-membership of recognizing unit disk graphs. In contrast, proving $\exists\mathbb{R}$-hardness is much more elaborate. It is obtained by a reduction from the so-called SimpleStretchability problem [112]. As a matter of fact, the vast majority of $\exists\mathbb{R}$-hardness reductions in the literature start with either ETR or SimpleStretchability. The reductions contained in this thesis are no exceptions.

1.2 Our Contribution

We present new reductions to determine the computational complexity of several important algorithmic problems. Most of these problems are (much) older than the notion of $\exists\mathbb{R}$-completeness itself, and in this case there were so far at most partial complexity results like NP-hardness or PSPACE-membership. As it turns out, studying these problems in the context of $\exists\mathbb{R}$ allows us to settle the complexity question of these problems.

Apart from the problems themselves, we advance the theory of $\exists\mathbb{R}$-completeness in several directions: We provide the first $\exists\mathbb{R}$-hardness results in hyperbolic geometry, demonstrate how to utilize geometry to study hardness of machine

1 We have to work with squared distances in (1.1) because $\|\cdot\|$ is the square root of a polynomial and square roots are not allowed in first-order formulas.

2 See Section 2.1.3 for a formal definition of the *length* of a sentence.

learning, and identify the first natural problem complete for the complexity class $\forall\exists\mathbb{R}$. Our contribution is spread across four main chapters. Each of these focuses on one specific problem, thereby exploring a different aspect of $\exists\mathbb{R}$:

Chapter 3: Stretchability and Hyperbolic Geometry Many results concerning $\exists\mathbb{R}$-hardness in the literature are obtained by a reduction from SimpleStretchability, placing it among the most important $\exists\mathbb{R}$-complete problems. We study SimpleStretchability in the hyperbolic plane $\mathbb{H}^2$, thereby exploring how geometric problems that are $\exists\mathbb{R}$-hard in $\mathbb{R}^2$ behave in $\mathbb{H}^2$.

Our main result is that SimpleStretchability is equivalent in $\mathbb{R}^2$ and $\mathbb{H}^2$. This allows us to adapt the known $\exists\mathbb{R}$-hardness reduction for recognizing unit disk graphs to prove $\exists\mathbb{R}$-hardness of recognizing *hyperbolic* unit disk graphs as well. Exploiting the mathematical similarities between $\mathbb{R}^2$ and $\mathbb{H}^2$, we generalize this reduction into a framework that is applicable to other geometric problems, in particular to those concerning geometric intersection representations in $\mathbb{H}^2$.

Chapter 4: Lombardi Graph Drawing A *Lombardi drawing* of a graph maps its vertices to points in the plane $\mathbb{R}^2$ and its edges to circular arcs such that each vertex has *perfect angular resolution*, i.e., equal angles between all incident edges. This drawing style has received considerable attention in the graph drawing community since its introduction in 2010, proving the existence or non-existence of Lombardi drawings for many graph classes [55, 56, 57, 59, 60, 61, 63, 99, 124]. Notably, there are no complexity results yet.

We close this gap by proving that it is $\exists\mathbb{R}$-complete to decide whether a given graph admits a Lombardi drawing. Our reduction is from SimpleStretchability and makes an elegant "detour" through hyperbolic geometry, building on our main result from the previous chapter.

Chapter 5: Training Neural Networks Artificial neural networks are currently the most successful tool in machine learning. Consequently, their training problem, i.e., finding their parameters to fit a given set of training data well, is highly important.

By a reduction from ETR we prove that the general training problem is $\exists\mathbb{R}$-complete. The proof heavily exploits the underlying geometry of structurally simple neural networks. Our reduction also yields a so-called *algebraic universality* theorem: This means that there exists instances, that are particularly difficult to train, because their solutions require irrational numbers of arbitrarily large degree.

Our result improves previous work by showing that $\exists\mathbb{R}$-hardness does not stem from adversarial network architectures but is indeed inherent to the problem. Furthermore, it explains why a celebrated NP-membership result for extremely simple network architectures [12] withstood any generalization to even slightly more complicated architectures.

Chapter 6: The Hausdorff Distance and ∀∃ℝ The *Hausdorff distance* is a classical similarity measure for sets. It has applications in many areas of computer science, mathematics and beyond.

We prove that computing the Hausdorff distance between two semi-algebraic sets is complete for the complexity class ∀∃ℝ. This class contains ∃ℝ, so computing the Hausdorff distance is supposedly even more difficult than all problems in ∃ℝ. Actually, computing the Hausdorff distance is among the first natural problems that were proven to be ∀∃ℝ-complete. More formally, ∀∃ℝ contains all problems that polynomial-time many-one reduce to deciding first-order formulas with two blocks of quantifiers of the form

$$\forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m \colon \varphi(X, Y).$$

To obtain our result, we utilize sophisticated results and algorithms from real algebraic geometry, in particular the so-called Ball Theorem and singly exponential quantifier elimination.

In analogy to NP and Σ_2^P being on the first and second level of the classical polynomial hierarchy, the classes ∃ℝ and ∀∃ℝ are on the first and second level of the *real polynomial hierarchy*. The study of the real polynomial hierarchy has only recently gained momentum [47, 54, 96, 139], partly because of our result.

1.3 Implications of ∃ℝ-Completeness

Our definition of ∃ℝ relies on ETR as a canonical complete problem. ETR is known to be NP-hard and in PSPACE [37, 144], however its precise computational complexity is still unknown (and it is considered likely that it is neither NP- nor PSPACE-complete). Therefore, proving a problem to be ∃ℝ-complete only tells us its difficulty relative to ETR. Nevertheless, proving ∃ℝ-completeness is a valuable contribution, even for problems that are known to be NP-hard, because ∃ℝ-completeness has several implications:

- As pointed out by Schaefer, ∃ℝ-completeness of a problem shifts the focus away from the problem itself to its underlying algebraic nature: “Knowing that a problem is ∃ℝ-complete does not tell us more than that it is NP-hard and in PSPACE in terms of classical complexity, but it does tell us where to start the attack: [...] A solution will likely not come out of graph drawing or graph theory but out of a better understanding of real algebraic geometry and logic.” [133]
- Often (but not always), ∃ℝ-hardness leads to algebraic universality results, stating that optimal solutions require algebraic numbers of arbitrarily large degree. Recall that this was the case for the neural network training problem.

  There is no algebraic universality result for the recognition problem of unit disk graphs. In fact, there is always an intersection representation in which all coordinates are rational [112]. Still, ∃ℝ-hardness has an impact on the required precision: There are n-vertex unit disk graphs such that every intersection representation with integer coordinates requires disk centers whose binary encoding has length exponential in n [112]. Note that this makes NP-membership unlikely, because the disk centers are not suited as an NP-witness.

- Knowing that a problem is ∃ℝ-complete gives some hints towards the algorithmic challenges that need to be overcome. While many NP-complete problems can be solved well in practice by extremely optimized off-the-shelf SAT- or ILP-solvers, no such general purpose tools are available for ∃ℝ-complete problems. In fact, to the best of our knowledge, finding the optimal solution for any ∃ℝ-complete problem requires algorithms from real algebraic geometry, for example a decision procedure for existentially quantified first-order sentences like (1.1) (after reducing the problem to ETR). However, these algorithms are very slow (all of them at least exponential in the number of variables) and therefore infeasible for large instances [123]. Different heuristics have been employed successfully [8, 51], however, they are all tailored towards the specific problem at hand.
- Lastly, problems from many areas of computer science and mathematics turn out to be ∃ℝ-complete. In Section 1.4 below, we compile a comprehensive but non-exhaustive list of many important examples. Knowing that they are equally difficult might lead to synergy effects in algorithm design.

1.4 Importance of ∃ℝ in the Literature

Since ∃ℝ has first appeared in a publication in 2009 [133], numerous problems across various disciplines have been shown to be ∃ℝ-complete[3]. Among others, these include:

Computational Geometry Just like for unit disk graphs, the recognition problems for many other types of geometric intersections graphs are ∃ℝ-complete [39, 38, 97, 110, 112, 133]. Other well-known ∃ℝ-complete problems include the art gallery problem [3], geometric packing [7], covering polygons with convex pieces [2], and geometric embeddings of simplicial complexes [5].

Moreover, the ∃ℝ-complete (Simple)Stretchability problem asks whether a (simple) arrangement of curves can be "stretched" to an equivalent arrangement of lines with the same intersection pattern [115, 130, 144] (see

3 Actually, some of the following results were obtained long before 2009. Consequently, these are stated as NP-hardness results in the literature, even though the reductions imply ∃ℝ-hardness in today's language. Often, NP-membership is explicitly posed as an open problem. Knowing that these problems are ∃ℝ-complete makes NP-membership unlikely (as this would imply NP = ∃ℝ).

Section 3.1 for a formal definition). Many ∃ℝ-hardness results in the literature are obtained by a reduction from SimpleStretchability [20, 89, 92, 103, 133, 134].

Graph Drawing The first explicit definition of ∃ℝ as a complexity class in a publication was at the Graph Drawing conference in 2009 [133], so it is no surprise that this field offers many ∃ℝ-complete problems. Often, they involve finding a drawing of a graph that satisfies geometric constraints, for example Lombardi graph drawing [92], the realization of linkages [1, 134], or recognizing many beyond-planar graphs like RAC-graphs [137] and geometric k-planar graphs [135] (for sufficiently large k).

Furthermore, decision problems about graph parameters like the rectilinear crossing number [20], the planar slope number [89] and the segment number [121] are ∃ℝ-complete. Deciding whether k graphs with a shared vertex set have a simultaneous straight line drawing is ∃ℝ-complete, even if k is a (large enough) constant [40, 105, 136].

Machine Learning The neural network training problem mentioned above is known as EmpiricalRiskMinimization. Its input is a network architecture and a set of training data. The goal is to compute the parameters of the network to fit the given training data well. Preliminary work proved ∃ℝ-hardness for adversarial network architectures [4]. In this thesis, we present an ∃ℝ-hardness proof for fully connected two-layer neural networks [18].

∃ℝ-hardness is also studied in other contexts of neural networks, e.g., for different activation functions [83] or reachability problems [153].

Mathematics In linear algebra, finding a non-negative matrix factorization is ∃ℝ-complete [143], as are many continuous constraint satisfaction problems from combinatorial optimization [113]. In probability theory, the conditional independence problem is co∃ℝ-complete (the class co∃ℝ is usually denoted by ∀ℝ) [28].

Game Theory Problems concerning the existence of (special) Nash equilibria in multi-player games are often NP-complete for two players, and ∃ℝ-complete for three or more players [22, 68].

2 Preliminaries

In this chapter, we revisit the theoretical background that is required to read this thesis. We provide formal definitions of all occurring concepts and establish our own (notational) conventions. Readers who are familiar with mathematical logic and/or the complexity class $\exists\mathbb{R}$ may skip this chapter or just look-up individual definitions if needed.

Chapter Outline The chapter is split into three parts: First, we lay the logical foundations in Section 2.1, defining syntax and semantics of the first-order theory of the reals. The second part, Section 2.2, covers computational questions. We consider decidability of the first-order theory of the reals and review existing algorithms for it. The last part concerns $\exists\mathbb{R}$-completeness, discussing the main approaches to prove $\exists\mathbb{R}$-hardness and $\exists\mathbb{R}$-membership.

2.1 The First-Order Theory of the Reals

We are going to formally define the *first-order theory of the reals*, a highly expressive mathematical theory. Assuming some prior knowledge of the reader, we start by providing some intuition and some examples before diving into the precise definitions below.

For our purposes, a *(first-order) formula* is a logical combination of polynomial equations and inequalities in real *variables*, possibly containing *quantifiers*. For example, the formula $\varphi(X,Y) :\equiv X^2 + Y^2 = 1$ defines the unit circle $S_\varphi := \{(x,y) \in \mathbb{R}^2 \mid \varphi(x,y)\}$. Similarly, the set $S_\psi := \{(x,y) \in \mathbb{R}^2 \mid \psi(x,y)\}$, where $\psi(X,Y) :\equiv Y \geq X^2$, contains all points above a parabola.

A *sentence* is a formula without free variables, and as such either true or false when interpreted over the real numbers. For example, the sentence

$$\exists X, Y \in \mathbb{R} \colon \varphi(X,Y) \land \psi(X,Y)$$

is true because S_φ and S_ψ have non-empty intersection, i.e., $S_\varphi \cap S_\psi \neq \emptyset$. The first-order theory of the reals can also express more complex statements:

$$\forall X \in \mathbb{R} \colon X \neq 0 \implies (\exists Y \in \mathbb{R} \colon XY = 1)$$

is true, because every real number except 0 has a multiplicative inverse. For our last example, let ϕ be a first-order formula with n free variables and let us consider $S_\phi := \{x \in \mathbb{R}^n \mid \phi(x)\}$. The set S_ϕ is open if and only if

$$\forall X \in S_\phi \,.\, \exists R \in \mathbb{R}_+ \,.\, \forall Y \in \mathbb{R}^n \colon \|X - Y\| < R \implies Y \in S_\phi.$$

In the remainder of this section, we make above intuition precise: We start by specifying the *syntax* of first-order formulas in Section 2.1.1. The *semantics* in Section 2.1.2 provides meaning to sentences, by determining their truth value. Together, syntax and semantics define the first-order theory of the reals. Sections 2.1.3 and 2.1.4 lay the foundation for the following complexity section.

2.1.1 Syntax

The *syntax* is a set of rules that define what well-formed first-order formulas are. We obtain a formal language. Let us stress that syntax is only concerned with which sequences of which symbols are considered to be well-formed. It does not provide any meaning to them.

The core concepts of the syntax are variables, atoms, formulas and sentences:

Variables A *variable* is a distinguished symbol without a value. Initially, all variables in A are said to be *free*.

As a convention, we use upper case letters (like X, Y, Z) for variables. Often, we group several individual variables into a vector of variables. In that case, we write $X = (X_1, \ldots, X_n)$ for a vector of n variables.

Atoms Let $X = (X_1, \ldots, X_n)$ be the symbols for n variables and let $P, Q \in \mathbb{Z}[X] = \mathbb{Z}[X_1, \ldots, X_n]$ be polynomials in these variables with integer coefficients. For simplicity, the only allowed operations in the polynomials are addition, subtraction, and multiplication. An *atom* A is a polynomial equation or inequality, i.e., an expression of the form $P \circ Q$, where $\circ$ is one of the binary relations in $\{<, \leq, =, \neq, \geq, >\}$.

Formulas & Sentences A *formula* is a combination of several atoms according to the following inductive definition:

- *Atoms:* If A is an atom, then A is a formula.
- *Negation:* If φ is a formula, then $\neg\varphi$ is a formula.
- If φ_1 and φ_2 are formulas, then the following are formulas:
  - *Conjunction:* $\varphi_1 \wedge \varphi_2$
  - *Disjunction:* $\varphi_1 \vee \varphi_2$

  Additionally, we define the following shorthand notation:
  - *Implication:* $\varphi_1 \implies \varphi_2$ denotes $\neg\varphi_1 \vee \varphi_2$
  - *Equivalence:* $\varphi_1 \iff \varphi_2$ denotes $(\varphi_1 \implies \varphi_2) \wedge (\varphi_2 \implies \varphi_1)$
- *Quantifiers:* If φ is a formula with a free variable X, then

  $$\exists X \in \mathbb{R} \colon \varphi \quad \text{and} \quad \forall X \in \mathbb{R} \colon \varphi$$

  are formulas in which X is *bound* by the respective quantifier. Each variable can be bound at most once, so bound variables are not considered free anymore.

A *sentence* is a formula without free variables. On the other hand, if a formula φ contains a free variable X, then we sometimes write $\varphi(X)$ to make this explicit.

Remark 2.1 (Operator Precedence). The negation operator "$\neg$" binds stronger than any of the binary operators "$\wedge$", "$\vee$", "$\Longrightarrow$" or "$\Longleftrightarrow$". On the other hand, we do not specify any precedence among the binary operators. Instead, we use parenthesis to avoid ambiguities. ⌟

2.1.2 Semantics

The *semantics* provide a "meaning" to sentences by assigning them a truth value, either true or false. Together, syntax and semantics define a theory: A logical *theory* is exactly the set of all true sentences in some formal language. Here, truth is defined by the semantics, while the syntax defines the formal language.

Let X be a free variable in a formula $\varphi(X)$. An *instantiation* of X in $\varphi(X)$ by a real number $x \in \mathbb{R}$ is obtained by replacing each occurrence of X by the specific x. Throughout this thesis, we use lower case letters for real-valued instantiations.

By definition, a sentence φ contains no free variables and is therefore either true or false. Its truth value is determined according to the following rules:

- An atom in which all variables are instantiated is true if and only if its interpretation over the real numbers is true.
- A sentence $\neg\varphi$ is true if and only if φ is false.
- A sentence $\varphi_1 \wedge \varphi_2$ is true if and only if φ_1 and φ_2 are both true.
- A sentence $\varphi_1 \vee \varphi_2$ is true if and only if at least one of φ_1 and φ_2 is true.
- A sentence $\exists X \in \mathbb{R}\colon \varphi(X)$ is true if there is at least one instantiation of X by some real number $x \in \mathbb{R}$ such that $\varphi(x)$ is true. Similarly, $\forall X \in \mathbb{R}\colon \varphi(X)$ is true if $\varphi(x)$ is true for every possible instantiation of X by $x \in \mathbb{R}$.

Two formulas $\varphi_1(X)$ and $\varphi_2(X)$ with the same free variables $X = (X_1, \ldots, X_n)$ are *equivalent* if they evaluate to the same truth value for all instantiations of X. Formally, we write $\varphi_1(X) \equiv \varphi_2(X)$ if and only if

$$\forall X \in \mathbb{R}^n\colon \varphi_1(X) \iff \varphi_2(X)$$

is true.

Definition 2.2 (First-Order Theory of the Reals). *The* first-order theory of the reals *contains all true sentences (according to the semantics in Section 2.1.2) in the formal language of first-order formulas (according to the syntax in Section 2.1.1).*

2.1.3 Encoding and Length

In essence, the length of a first-order formula φ, denoted by $|\varphi|$, is the number of bits required to write it down. There are some subtleties, so let us give a more precise definition here. This is crucial because it might depend on the exact method we measure length in, whether a reduction takes polynomial time or not. Or definition is inductive, mimicking the inductive definition of the syntax a formula in Section 2.1.1.

We start with the *length of an atom* A, denoted by $|A|$. To this end we restrict the *signature* of the polynomials, i.e., the available set of symbols, to $\{0, 1, +, -, \cdot, (,), X_1, \ldots, X_n\}$. In particular, we only allow 0 and 1 as coefficients and constants (see Remark 2.3 why this does not need to bother us). Then, each of these $O(n)$ symbols can be represented by a unique string of $O(\log n)$ bits. The length of A is the number of bits needed in this encoding.

For the *length of a formula* φ, we consider φ's structure: If φ is an atom A, then $|\varphi| = |A|$. If φ is constructed from smaller subformulas, then its length is the sum of the lengths of its parts. For example, the formula $\varphi :\equiv \varphi_1 \wedge \varphi_2$ has its length defined as the length of its subformulas φ_1 and φ_2 plus a constant number of bits for each additional symbol ($\wedge$ in this case): $|\varphi| = |\varphi_1| + |\varphi_2| + O(1)$

Remark 2.3 (Integer Coefficients)**.** Using only the constants 0 and 1, an arbitrary integer coefficient $k \in \mathbb{N}$ can be constructed by an expression of length $O(\log k)$. The idea is to use Horner's rule on the binary expansion of k. For example, we can express 13 as

$$13_{10} = 1101_2 = 1+2\cdot(0+2\cdot(1+2\cdot 1)) = 1+(1+1)\cdot(0+(1+1)\cdot(1+(1+1)\cdot 1)).$$

Therefore, the length of a formula increases by at most a factor of $O(\log k)$ when transforming a formula with arbitrary integer coefficients into one with only 0 and 1, assuming that k is the largest coefficient. This logarithmic difference in length has no effect on whether a reduction takes polynomial time or not. ⌟

Remark 2.4 (Exponentiation)**.** There is no syntax for exponentiation. In fact, the notation X^k for some constant $k \in \mathbb{N}$ is just an abbreviation for the k-fold product $X \cdot \ldots \cdot X$. For example, the length of the polynomial $(\cdots((X^2)^2)\cdots)^2$ with n second powers is in $O(2^n)$ (and not in $O(n)$) [110]. This exponential difference is indeed important for *polynomial-time* reductions. ⌟

2.1.4 Normal Forms

Formulas adhering to a restricted syntax are often easier to work with and allow for more streamlined reductions. For this reason, several *normal forms* exist. Considering just polynomials, we need just one normal form:

Monomial Normal Form A polynomial is in *monomial normal form* if it is written as a sum of monomials. In particular, no parentheses are used. Monomials containing the same variables to the same powers are grouped together.

Transforming a polynomial into monomial normal form can lead to an exponential increase in the length of the polynomial: For example, the polynomial $(X_1 + 1)(X_2 + 1)\ldots(X_n + 1)$ has length in $O(n)$ while its monomial normal form has length $O(2^n)$ [110].

On the other hand, a polynomial in n variables and maximum degree d has at most $(d+1)^n$ different monomials.

There are several normal forms for formulas. Every formula can be transformed into each of these, so all normal forms have the same descriptive power.

Atom Normal Form A formula is in *atom normal form* if all of its atoms are of the form $P \circ 0$, where P is a polynomial and $\circ \in \{<, \leq\}$.

Atom normal form can easily be achieved by elementary operations. This can be done in linear time and increases the length by at most a constant factor.

Negation Normal Form A formula is in *negation normal form* if the negation operation is only applied to atoms.

Negation normal form is achieved by repeatedly applying De Morgan's law

$$\neg(\varphi_1 \land \varphi_2) \;\equiv\; \neg\varphi_1 \lor \neg\varphi_2 \quad \text{and} \quad \neg(\varphi_1 \lor \varphi_1) \;\equiv\; \neg\varphi_1 \land \neg\varphi_2$$

as well as double negation $\neg\neg\varphi \equiv \varphi$ and the negation of quantifiers

$$\neg\exists X \in \mathbb{R} \colon \varphi \;\equiv\; \forall X \in \mathbb{R} \colon \neg\varphi \quad \text{and} \quad \neg\forall X \in \mathbb{R} \colon \varphi \;\equiv\; \exists X \in \mathbb{R} \colon \neg\varphi.$$

This increases the length of the formula only by a constant factor and takes linear time.

Prenex Normal Form A formula φ is in *prenex normal form* if all quantifiers appear at the beginning, i.e., φ starts with several blocks of quantified variables followed by a quantifier-free subformula (also called the *matrix* of the formula).

We assume that each variable is denoted by its own, unique symbol. Then a formula in negation normal form can be transformed into prenex normal form by simply moving the quantification of the bound variables to the front of the formula (keeping their relative ordering intact). This transformation does not change the length of the formula and takes linear time.

If the formula is not in negation normal form, care has to be taken when moving a quantifier out of the scope of a negation. Still, the length of the formula only changes by a constant factor and the transformation takes linear time.

Conjunctive/Disjunctive Normal Form A quantifier-free formula in negation normal form is in *conjunctive normal form (CNF)* if it is a conjunction of disjunctions. Similarly, it is in *disjunctive normal form (DNF)* if it is a disjunction of conjunctions.

Every formula can be transformed into an equivalent formula in CNF or DNF by repeatedly applying the distributive law

$$\begin{aligned} \varphi_1 \land (\varphi_2 \lor \varphi_3) \;&\equiv\; (\varphi_1 \land \varphi_2) \lor (\varphi_1 \land \varphi_3) \quad \text{and} \\ \varphi_1 \lor (\varphi_2 \land \varphi_3) \;&\equiv\; (\varphi_1 \lor \varphi_2) \land (\varphi_1 \lor \varphi_3). \end{aligned}$$

Converting into CNF or DNF takes time linear in the length of the resulting formula. However, both can be exponentially longer than the original formula.

2.2 Decidability and Algorithms

Given a mathematical theory, the obvious question is whether it is *decidable*. In other words: Is there an algorithm to decide the truth value of a given sentence? Variables in the first-order theory of the reals can be instantiated with any of the uncountably many real numbers. Therefore, iterating over all possible instantiations is not possible. Still, the first-order theory of the reals is decidable, as proven by Tarski in the 1940s. He provides a decision algorithm, however there is no elementary bound on its time complexity [146]. Current algorithms are based on Collins' *cylindrical algebraic decomposition (CAD)* method and require doubly exponential time in the number of variables [44].

Theorem 2.5 ([16, Theorem 14.17]). *Consider a first-order formula*

$$\Phi \;:\equiv\; \mathrm{Q}_1 X_1 \in \mathbb{R}^{n_1} \,.\, \mathrm{Q}_2 X_2 \in \mathbb{R}^{n_2} \ldots \mathrm{Q}_\omega X_\omega \in \mathbb{R}^{n_\omega} : \varphi(X_1, X_2, \ldots, X_\omega)$$

in prenex normal form, where ω is the number of blocks of quantifiers, and $\mathrm{Q}_i \in \{\forall, \exists\}$. Further, φ is quantifier-free, contains s atoms, and each polynomial has degree at most d. Then there is an algorithm to decide truth of Φ in time

$$s^{(n_1+1)\cdots(n_\omega+1)} d^{O(n_1)\cdots O(n_\omega)}.$$

The so-called *∃-fragment* ("existential fragment") of the first-order theory of the reals consists of all true sentences in prenex normal form with a single block of existentially quantified variables, i.e., sentences of the form

$$\exists X_1, \ldots, X_n \in \mathbb{R} : \varphi(X_1, \ldots, X_n),$$

where φ is quantifier-free. This theory is also known as the *existential theory of the reals*. The decision problem whether such a sentence is true corresponds exactly to the problem ETR from Definition 1.1 (the definition was incomplete back then because we did not yet define "syntax", "semantics" and "prenex normal form"). By Theorem 2.5, there is an algorithm taking time $s^{n+1} d^{O(n)}$ (take $\omega = 1$ as there is just a single existential block of quantifiers). The first singly exponential time algorithms to decide ETR are by Renegar [126, 127, 128], a PSPACE-algorithm is given by Canny [37]. NP-hardness follows easily by a reduction from SAT to ETR [144].

Similarly, the *∀∃-fragment* ("universal existential fragment") of the first-order theory of the reals consists all true sentences in prenex normal form with two blocks of quantifiers of the form

$$\forall X_1, \ldots, X_n \in \mathbb{R} \,.\, \exists Y_1, \ldots, Y_m \in \mathbb{R} : \varphi(X_1, \ldots, X_n, Y_1, \ldots, Y_m),$$

where φ is quantifier-free. This theory can also be decided in PSPACE. Again, by applying Theorem 2.5, now with $\omega = 2$. In fact, it requires only singly exponential time for every constant number of quantifier alternations.

Remark 2.6 (Decidability of other Existential Theories)**.** There are many other interesting and related existential theories besides the existential theory of the reals:

- The *existential theory of* $\mathbb{F}_2$ asks for Boolean variables satisfying a given formula. This corresponds to the well-known SAT problem, therefore deciding this theory is NP-complete.
- The *existential theory of the integers* asks for integer solutions of so-called *Diophantine equations*. The corresponding decidability problem is known as Hilbert's tenth problem and known to be undecidable [109].
- The *existential theory of the rationals* asks for rational solutions of a first-order formula. It is an open problem whether this theory is decidable.
- The *existential theory of the complex numbers* asks for the existence of complex solutions. Koiran showed that, under the Generalized Riemann Hypothesis, this theory can be decided in $\mathrm{RP}^{\mathrm{NP}}$ [101], which is in the second level of the polynomial hierarchy (Π_2^P).
- Tarski's *exponential function problem* asks whether the existential theory of the reals together with the exponential function $\exp(\cdot)$ is decidable. This question is still open.

  Adding trigonometric functions like $\sin(\cdot)$ and $\cos(\cdot)$ to the existential theory of the reals yields an undecidable theory [129]. ⌟

2.3 ∃ℝ-Completeness

Recall the definition of the complexity class ∃ℝ from Definition 1.2. A problem is *in* ∃ℝ if there is a polynomial-time many-one reduction from it to ETR. Conversely, a problem is *∃ℝ-hard* if ETR can be polynomial-time many-one reduced to it. In both cases, the reduction must be executed in the bit model of computation, or equivalently, on a deterministic Turing machine. We say that a problem is *∃ℝ-complete* if it is both ∃ℝ-hard and in ∃ℝ.

∃ℝ-Hardness Most ∃ℝ-hardness reductions in the literature are either from ETR or from SimpleStretchability. As a matter of fact, the reductions appearing in Chapters 3 and 4 are based on SimpleStretchability, and the reductions in Chapters 5 and 6 are based on (extensions of) ETR.

A general ETR instance may have a very complicated structure: Its matrix may be nested several levels deep and/or contain arbitrary combinations of the Boolean operators. To simplify reductions from ETR, a number of special cases with restricted syntax have been proven to be ∃ℝ-complete as well (just like 3SAT is an NP-complete syntactical restriction of SAT). Among other, the following restrictions of ETR remain ∃ℝ-complete:

- INEQ: The matrix is a conjunction of atoms, each in monomial normal form.

- STRICT-INEQ: Variant of INEQ where each atom is a strict inequality.
- FEASIBILITY: Variant of INEQ with only a single atom $p(X_1, \ldots, X_n) = 0$. Here, $p \in \mathbb{Z}[X_1, \ldots, X_n]$ is a multivariate polynomial of degree at most 4.
- ETR-INV: Variant of INEQ in which each atom has the form

  $$X + Y = Z \quad \text{or} \quad XY = 1,$$

  where X, Y and Z are symbols for variables. A promise version of ETR-INV allows us to assume that there is either no solution or a solution with all variables in $\left[\frac{1}{2}, 2\right]$. For example, our reduction in Chapter 5 is from ETR-INV.

∃ℝ-Membership By definition, ∃ℝ-membership is proven by a polynomial-time many-one reduction to ETR. Often this is straightforward, as we have seen in Section 1.1 for the recognition of unit disk graphs. Another helpful property is that ∃ℝ is closed under polynomial-time many-one NP-reductions [148], i.e., reductions executed on a non-deterministic Turing machine. Thus, reductions to prove NP-membership may use non-determinism to guess parts of the resulting ETR instance.

Another approach to prove ∃ℝ-membership is to apply a real-valued analog of the Cook-Levin theorem proven by Erickson, van der Hoog and Miltzow [64]. It states that ∃ℝ-membership is equivalent to the existence of a so-called *real verification algorithm*. This is an algorithm that verifies a (possibly real-valued) witness in polynomial time on a *real RAM* machine. The real RAM is a model of computation that is similar to the classical word RAM but with additional registers that can hold real values with arbitrary precision. Addition, subtraction, multiplication, division, and even taking square roots of real numbers is possible in constant time. The real RAM is omnipresent in computational geometry, in fact, the correctness of many famous algorithms relies on the ability to compute with unbounded precision [108].

3 Stretchability and Hyperbolic Geometry

This chapter is based on our paper "Recognizing Unit Disk Graphs in Hyperbolic Geometry is ∃ℝ-Complete" presented at the *39th European Workshop on Computational Geometry (EuroCG 2023)*. It is joint work with Nicholas Bieker, Thomas Bläsius and Emil Dohse.

In this chapter, we consider the Stretchability problem. Certainly, this is one of the most important ∃ℝ-complete problems because many ∃ℝ-hardness reductions in the literature start from Stretchability or – more commonly – its restricted version SimpleStretchability. Examples include Lombardi drawability [92], the rectilinear crossing number [20], the planar slope number [89], the recognition of various types of geometric intersection graphs [103, 133] and the realization of graphs and linkages with prescribed edge lengths [1, 134].

Classically, Stretchability and SimpleStretchability are considered in the Euclidean plane $\mathbb{R}^2$. Our main contribution of this chapter is to establish that both remain ∃ℝ-complete in the hyperbolic plane $\mathbb{H}^2$. We use this insight to prove that the recognition of hyperbolic unit disk graphs is ∃ℝ-complete. Actually, our reduction for this result is very simple: It uses the known reduction by McDiarmid and Müller [111] for Euclidean unit disk graph recognition as a black box, thereby hiding all the complicated details.

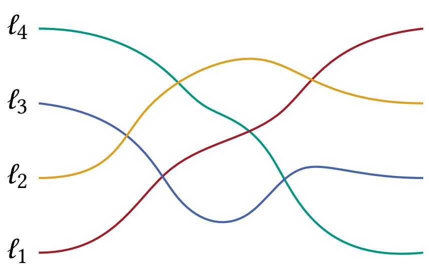
Besides their simplicity, most arguments of the reduction are easily adaptable to other geometric problems than recognizing hyperbolic unit disk graphs. We utilize this to present a general framework to translate ∃ℝ-hardness reductions in the literature from Euclidean to hyperbolic geometry.

Moreover, as a second application, ∃ℝ-hardness of SimpleStretchability in $\mathbb{H}^2$ also plays an important role in the next chapter, where we prove ∃ℝ-hardness of a purely Euclidean graph drawing problem.

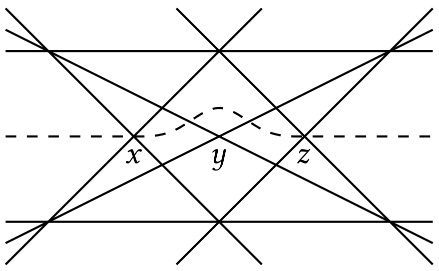
Chapter Outline We start by defining Stretchability and SimpleStretchability formally in Section 3.1. Afterward, we give a short introduction to hyperbolic geometry in Section 3.2. The equivalence of SimpleStretchability in the Euclidean plane $\mathbb{R}^2$ and in the hyperbolic plane $\mathbb{H}^2$ is proven in Section 3.3. The first application is to prove ∃ℝ-hardness of recognizing hyperbolic unit disk graphs in Section 3.4.1, the second application being our general framework in Section 3.4.2. We finish the chapter by discussing further candidate problems for our framework.

3.1 Stretchability of Pseudolines

A *pseudoline* is an x-monotone[1] curve in $\mathbb{R}^2$. A *pseudoline arrangement* $\mathcal{A}$ is a collection of pseudolines that pairwise intersect each other at most once. We say that $\mathcal{A}$ is *simple* if each pair of pseudolines intersects exactly once and no three pseudolines have a common intersection. See Figure 3.1 for two pseudoline arrangements.



(a) A simple and stretchable arrangement of four pseudolines $\ell_1, \dots, \ell_4$.



(b) A non-simple and non-stretchable arrangement of nine pseudolines.

Figure 3.1: Two pseudoline arrangements with opposing properties.

A pseudoline arrangement $\mathcal{A}$ is *stretchable* if there exists a homeomorphism $f\colon \mathbb{R}^2 \to \mathbb{R}^2$ of the plane such that $f(\mathcal{A})$ is a line arrangement. Perhaps surprisingly, not all pseudoline arrangements are stretchable. Indeed, the arrangement shown in Figure 3.1b is not stretchable: By Pappus' Hexagon Theorem [45, page 232] the three intersection points x, y, z must be collinear, so no stretching of the dashed pseudoline can contain just two of them. Let us note that this is the smallest non-stretchable assignment, as all pseudolines arrangements with up to eight pseudolines are stretchable [75].

Definition 3.1 ((Simple)Stretchability).
Input: *A (simple) pseudoline arrangement $\mathcal{A}$.*
Question: *Is $\mathcal{A}$ stretchable, i.e., homeomorphic to a line arrangement in $\mathbb{R}^2$.*

Let us note that there are several equivalent possibilities to encode the input of (Simple)Stretchability. Section 3.1.1 below describes the one option that is used throughout this thesis.

The following theorem is due to Mnëv [115] (who actually proves a much stronger statement). Subsequently, Shor [144] and Richter-Gebert [130] gave simplified proofs.

Theorem 3.2 (Mnëv [115]). *Stretchability and SimpleStretchability are $\exists\mathbb{R}$-complete.*

1 An equivalent definition is that a pseudoline is a simple curve homeomorphic to a line.

3.1.1 Encoding Pseudoline Arrangements

In order to reduce from (Simple)Stretchability we need a finite description of a given pseudoline arrangement. We require that two pseudoline arrangements with the same description have the same intersection pattern, i.e., the same pairs of intersecting pseudolines and the same order of intersections along each pseudoline.

Let $\mathcal{A} = \{\ell_1, \dots, \ell_n\}$ be a pseudoline arrangement with n pseudolines. Throughout this thesis, we always assume that no pseudoline is vertical and that the pseudolines are labeled such that a vertical line to the left of all intersections crosses ℓ_i below ℓ_j (for $i, j \in \{1, \dots, n\}$) if and only if $i < j$. We further assume that each pseudoline $\ell \in \mathcal{A}$ is oriented (arbitrarily) and thus divides the plane $\mathbb{R}^2$ into two open half-planes ℓ^- and ℓ^+.

Definition 3.3 (Combinatorial Description)**.** *A combinatorial description $\mathcal{D}$ of an arrangement $\mathcal{A}$ of n pseudolines consists of n lists, one per pseudoline, each listing the order of intersections along it from left to right. We say that $\mathcal{A}$ realizes $\mathcal{D}$.*

For example, the list of intersections for ℓ_1 in Figure 3.1a contains (in this order) ℓ_3, ℓ_4 and ℓ_2. If $\mathcal{A}$ is simple, then each list contains exactly $n - 1$ entries. Otherwise, some lists may be shorter. If multiple pseudolines intersect in the same point, the list entry is a tuple instead of a single other pseudoline.

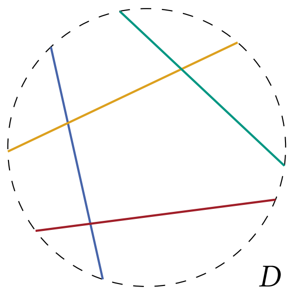
Let us note that there are several other but equivalent possibilities to define a combinatorial description in the literature. In this thesis, we just need the one from Definition 3.3.

3.2 Introduction to the Hyperbolic Plane

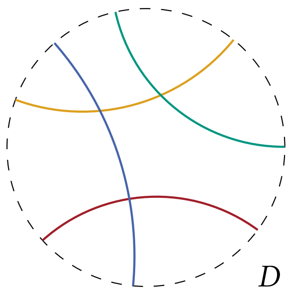
The *hyperbolic plane* $\mathbb{H}^2$ is a non-Euclidean geometry. In many ways it behaves similar to the Euclidean plane $\mathbb{R}^2$, e.g., two points define a unique line and we can measure distances and angles.

Formally, both $\mathbb{R}^2$ and $\mathbb{H}^2$ can be described by an axiomatic system (like the one from Hilbert for $\mathbb{R}^2$ [87]). In fact, axiomatic systems for $\mathbb{R}^2$ and $\mathbb{H}^2$ are nearly identical, explaining the many similarities between $\mathbb{R}^2$ and $\mathbb{H}^2$. Without going into the technical details, Hilbert's axiomatic system contains the so-called *parallel postulate* stating that for any line ℓ and point p not on ℓ in $\mathbb{R}^2$ there is at most one line through p parallel to ℓ (indeed there is exactly one). Negating this axiom turns Hilbert's axiomatic system for $\mathbb{R}^2$ into one that defines $\mathbb{H}^2$.

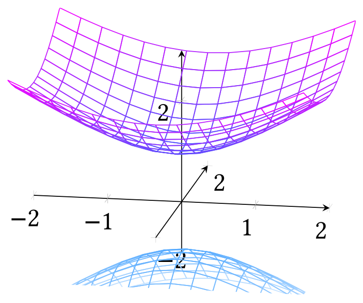
Remark 3.4 (Other Geometries)**.** There are many other (2-dimensional) geometries besides Euclidean and hyperbolic geometry, e.g., *spherical geometry* and *elliptic geometry*. Both share many properties with Euclidean and hyperbolic geometry. On the other hand, axiomatic systems for them differ significantly from the ones for Euclidean and hyperbolic geometry. ⌟



(a) Beltrami-Klein disk with hyperbolic lines.



(b) Poincaré disk with hyperbolic lines.



(c) The hyperboloid S for the hyperboloid model.

Figure 3.2: Models of the hyperbolic plane.

3.2.1 Models of the Hyperbolic Plane

When working with the hyperbolic plane $\mathbb{H}^2$, we usually avoid working with the axioms directly. Instead, we consider *models*, i.e., embeddings of $\mathbb{H}^2$ into Euclidean space. Formally, a model is a mathematical structure that defines "points" and "lines" such that the axioms of hyperbolic geometry are true. Typically, points in $\mathbb{H}^2$ are mapped to points in $\mathbb{R}^2$ and hyperbolic lines are mapped to geodesics according to some non-Euclidean metric.

Several of these models appear in the literature, some of these are used in this thesis. In the list below, we do not define these models formally, neither do we prove that they indeed model the hyperbolic plane, i.e., satisfy all axioms (the interested reader may consider [36] for the omitted details). Instead, we focus on the properties that will be used by us.

- In the *Beltrami-Klein model*, the hyperbolic plane $\mathbb{H}^2$ is mapped to the interior of a unit disk D, the so-called *Beltrami-Klein disk*. The mapping is such that the set of hyperbolic lines corresponds exactly to the set of chords of D, see Figure 3.2a. In particular, hyperbolic lines (and line segments) are mapped to Euclidean line segments. This will be helpful when we consider hyperbolic line arrangements and the SimpleStretchability problem in $\mathbb{H}^2$ in Section 3.3 below.
- In the *Poincaré disk model*, the hyperbolic plane $\mathbb{H}^2$ is again mapped to the interior of a unit disk D, now called the *Poincaré disk*. In this model, hyperbolic lines are mapped to circular arcs orthogonal to D (this includes diameters of D, which can be thought of as circular arcs orthogonal to D with infinite radius), see Figure 3.2b. The Poincaré disk model has several useful properties:
  - Circles (and circular arcs) are mapped to other circles (and circular arcs). However, the center of a hyperbolic circle is not mapped to the Euclidean center of the corresponding circle in the Poincaré disk.

  - The Poincaré disk model is *conformal*, meaning that the angles in the hyperbolic plane equal the angles in a drawing inside the Poincaré disk D.

- Lastly, in the *hyperboloid model*, the hyperbolic plane $\mathbb{H}^2$ gets embedded onto a three-dimensional hyperboloid $S := \{(x, y, z) \in \mathbb{R}^3 \mid Q(x, y, z) = 1\}$, where $Q(x, y, z) := z^2 - x^2 - y^2$ is the so-called *Minkowski quadratic form*. Actually, the hyperboloid S consists of two connected components, see Figure 3.2c. The hyperbolic plane is represented by all points on the *forward sheet* S^+ of S (the connected component with $z > 0$). The hyperbolic distance between two points $u, v \in S^+$ is

  $$d_{\mathbb{H}}(u, v) = \cosh^{-1}(B(u, v)),$$

  where $B(u, v)$ is known as the *Minkowski bilinear form*

  $$\begin{aligned} B\colon \mathbb{R}^3 \times \mathbb{R}^3 &\to \mathbb{R} \\ B(u, v) &\mapsto u_z v_z - u_x v_x - u_y v_y \end{aligned}$$

  and $\cosh^{-1}(x) := \ln\left(x + \sqrt{x^2 - 1}\right)$ is the inverse hyperbolic cosine. Note that the Minkowski bilinear form B is a polynomial.

3.3 Stretchability in the Hyperbolic Plane

In Section 3.1, we defined pseudolines as curves in $\mathbb{R}^2$ that are homeomorphic to a line in $\mathbb{R}^2$. However, the input for (Simple)Stretchability is not a set of curves but only its combinatorial description. In particular, hyperbolic pseudoline arrangements in which every two lines intersect exactly once can be described in the same manner as Euclidean ones[2]. This justifies the consideration of (Simple)Stretchability in the hyperbolic plane as well, i.e., ask whether a given pseudoline arrangement can be realized by a set of hyperbolic lines.

The main theorem of this chapter is the following easy application of the Beltrami-Klein model of the hyperbolic plane.

Theorem 3.5. *Let $\mathcal{D}$ be a combinatorial description of a pseudoline arrangement in which every two pseudolines intersect exactly once. Then there is a Euclidean line arrangement $L_{\mathbb{R}}$ realizing $\mathcal{D}$ in $\mathbb{R}^2$ if and only if there is a hyperbolic line arrangement $L_{\mathbb{H}}$ realizing $\mathcal{D}$ in $\mathbb{H}^2$.*

Proof. For an illustration of the following argument, see Figure 3.3.

Let $L_{\mathbb{R}}$ be a line arrangement in $\mathbb{R}^2$ realizing $\mathcal{D}$, and let D be a disk strictly enclosing all intersections of $L_{\mathbb{R}}$. For each line in $L_{\mathbb{R}}$, keep only its part strictly

2 In all other cases, an extended (but still finite) combinatorial description may be used to describe the relative position of non-intersecting subarrangements. One way to achieve this is to additionally store the cyclic ordering of all intersections of pseudolines with the Poincaré disk.

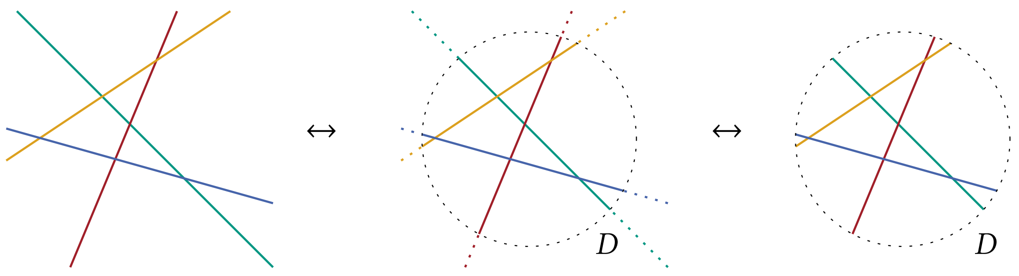


Figure 3.3: Transforming line arrangements between the Euclidean plane $\mathbb{R}^2$ (left) and the hyperbolic plane $\mathbb{H}^2$ (right, Betrami-Klein model).

inside D. We think of D as a unit disk and obtain a representation of a hyperbolic line arrangement in the Beltrami-Klein model of the hyperbolic plane.

For the other direction, let $L_{\mathbb{H}}$ be a hyperbolic line arrangement and take a representation inside the Beltrami-Klein disk D. Recall that all hyperbolic lines are represented by chords of D. Remove D and extend all chords to lines. The resulting Euclidean line arrangement has the same combinatorial description because every two pseudolines in $\mathcal{D}$ intersect exactly once. Therefore, all possible intersections between two lines are already inside the Beltrami-Klein disk D. ■

By definition, simple pseudoline arrangements fulfill the condition of Theorem 3.5, i.e., every two pseudolines intersect exactly once. Thus, we obtain the following corollary:

Corollary 3.6. *Let $\mathcal{D}$ be the combinatorial description of a simple pseudoline arrangement. Then $\mathcal{D}$ is stretchable in $\mathbb{R}^2$ if and only if it is stretchable in $\mathbb{H}^2$.*

Remark 3.7. Theorem 3.5 does not hold for general pseudoline arrangements. For example, consider three pseudolines ℓ_1, ℓ_2 and ℓ_3 such that ℓ_2 intersects ℓ_3 and both are parallel to ℓ_1, see Figure 3.4. This pseudoline arrangement is stretchable in the hyperbolic plane but not in the Euclidean plane. ⌟

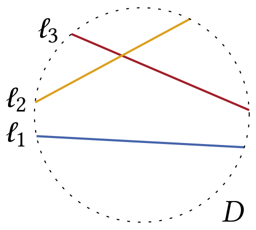


Figure 3.4: A hyperbolic line arrangement in the Beltrami-Klein disk for which no Euclidean line arrangement with the same intersection pattern exists.

3.4 Applications

We proceed by showing possible applications of Corollary 3.6. The applications in this section are relatively straightforward. A more surprising application follows in Chapter 4 below.

3.4.1 Recognition of Hyperbolic Unit-Disk Graphs

Recall that a *unit disk graph* is a graph that admits an intersection representation by unit disks in $\mathbb{R}^2$. The class of unit disk graphs, denoted by UDG, is a well-studied graph class due to its mathematical simplicity (compared to other geometric intersection graphs) and its practical relevance, e.g., in the context of sensor networks.

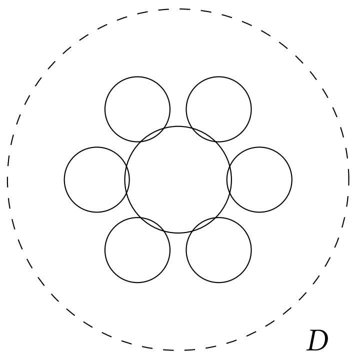
Usually, unit disk graphs are considered in the Euclidean plane $\mathbb{R}^2$. However, in the past decade, research on intersection graphs of equally sized disks in the hyperbolic plane $\mathbb{H}^2$ has gained traction (see Remark 3.9 on why we have to talk about *equally sized* disks instead of *unit* disks here). Hyperbolic geometry is well suited to represent a wide range of graph structures, including complex scale-free networks with heterogeneous degree distributions [24, 27, 78, 104, 119]. Most research on such graphs is driven by the network science community studying probabilistic network models, i.e., hyperbolic random graphs. However, when omitting the probability distribution and looking at hyperbolic unit disk graphs as a graph class, little is known so far.

We prove that recognizing hyperbolic unit disk graphs is $\exists\mathbb{R}$-complete. Let us start by formally defining the graph class and its recognition problem.

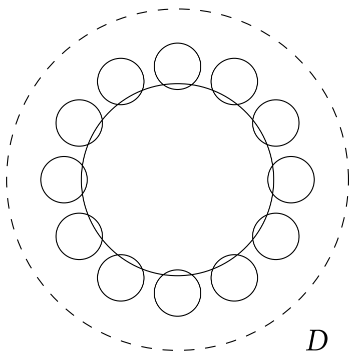
Definition 3.8 (Hyperbolic Unit Disk Graphs, HUDG). *A graph is a* hyperbolic unit disk graph *if it admits an intersection representation by equally sized disks in $\mathbb{H}^2$. We denote the class of hyperbolic unit disk graphs by* HUDG.

Let us note that there are earlier results on a related family of graph classes parameterized by the radius of the disks by Kisfaludi-Bak [100]. For a fixed $\rho > 0$, he defines the class $\mathrm{UBG}_{\mathbb{H}^2}(\rho)$ to contain all intersection graphs of disks with radius ρ in $\mathbb{H}^2$ (UBG stands for "unit ball graph"). In a sense, the class HUDG is the union over all ρ of all these classes. This subtle difference is important when considering asymptotic behavior, as it can be desirable to grow the disk size with the graph size; see [23] for a detailed discussion.

Remark 3.9 (Equally Sized Disks). Our definition of hyperbolic unit disk graphs requires an intersection representation with *equally sized* disks (instead of *unit* disks). It is indeed important that the radius of the disks can depend on the graph: For example, stars with up to five leaves are Euclidean unit disk graphs, while stars with at least six leaves do not have an intersection representation by unit (or equally sized) disks in $\mathbb{R}^2$. On the other hand, for every star S, there exists a radius r_S such that S has an intersection representation of disks with radius r_S in $\mathbb{H}^2$. However, there is no universal radius r^* such that all stars have an intersection representation of disks with radius r^* in $\mathbb{H}^2$. See Figure 3.5 for an illustration. ⌟



(a) Star with six leaves.



(b) Star with twelve leaves.

Figure 3.5: Two intersection representations of stars by *equally sized (!)* disks in the hyperbolic plane (shown in the Poincaré disk model).

The *recognition problem* for a graph class $\mathcal{G}$ is to decide whether a given graph G belongs to $\mathcal{G}$:

Definition 3.10 (Recog($\mathcal{G}$)).
Input: *A graph G.*
Question: *Is $G \in \mathcal{G}$?*

In the following, we prove that Recog(HUDG) is $\exists\mathbb{R}$-complete. We start by giving a reduction to ETR, thereby proving $\exists\mathbb{R}$-membership.

Proposition 3.11. Recog(HUDG) *is in* $\exists\mathbb{R}$.

Proof. Let $G = (V, E)$ be a graph with vertex set $V = \{v_1, \ldots, v_n\}$. We introduce the following variables: A single variable R and for each vertex $v_i \in V$ the (three-dimensional) variable $V_i = (X_i, Y_i, Z_i)$. We think of V_i as the coordinates of v_i in the hyperboloid model of the hyperbolic plane, and of R as the threshold distance determining whether two vertices should be adjacent.

By definition, G is a hyperbolic unit disk graph if and only if

$$\begin{aligned} \exists V_1 \ldots, V_n, R \in \mathbb{R}^{3n+1} \colon & \bigwedge_{\{v_i,v_j\}\in E} \cosh^{-1}(B(V_i, V_j)) \leq R \wedge \\ & \bigwedge_{\{v_i,v_j\}\notin E} \cosh^{-1}(B(V_i, V_j)) > R \end{aligned} \tag{3.1}$$

is true. Recall that $B(V_i, V_j) = Z_i Z_j - X_i X_j - Y_i Y_j$ is the Minkowski bilinear form. Sentence (3.1) is not a well-formed sentence in the $\exists$-fragment of the first-order theory of the reals because their syntax does not contain (hyperbolic) trigonometric functions like $\cosh^{-1}(\cdot)$. However, $\cosh^{-1}(\cdot)$ is a monotone function, so sentence (3.1) can be rewritten as

$$\exists V_1 \ldots, V_n, R \in \mathbb{R}^{3n+1} \colon \bigwedge_{\{v_i,v_j\}\in E} B(V_i, V_j) \leq R \wedge \bigwedge_{\{v_i,v_j\}\notin E} B(V_i, V_j) > R. \tag{3.2}$$

Sentence (3.2) is a valid ETR instance (using that the Minkowski bilinear form is a polynomial) and equivalent to $G \in$ HUDG. The reduction takes quadratic time, thus $\exists\mathbb{R}$-membership follows. ■

We now consider $\exists\mathbb{R}$-hardness. It is well-known that Recog(UDG), i.e., the recognition of Euclidean unit disk graphs, is $\exists\mathbb{R}$-complete [97, 111, 112], proven by a reduction from SimpleStretchability. Our reduction for Recog(HUDG) builds on the reduction by McDiarmid and Müller [111] (we chose the reduction from the conference version [111] over the reduction from the journal version [112] because it is conceptually simpler).

Allow us to shortly summarize their reduction: Let $\mathcal{D}$ be a combinatorial description of a simple pseudoline arrangement. McDiarmid and Müller construct a graph $G_{\mathcal{D}}$ from $\mathcal{D}$ as follows: Let n be the number of pseudolines $\ell_1, \dots, \ell_n$ and let $m = 1 + \binom{n+1}{2}$ be the number of cells $C_1, \dots, C_m$. The arrangement described by $\mathcal{D}$ has exactly this number of cells, because it is simple. We define $G_{\mathcal{D}}$ as the graph with vertex set $V = A \cup B \cup C$, where $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$ and lastly $C = \{c_1, \dots, c_m\}$. Vertex c_i corresponds to cell C_i. Regarding the edges, each of the sets A, B and C forms a clique. Furthermore, c_i (for $i \in \{1, \dots, m\}$) is connected to a_j (for $j \in \{1, \dots, n\}$) if and only if $C_i \in \ell_j^-$, and to b_j (for $j \in \{1, \dots, n\}$) if and only if $C_i \in \ell_j^+$.

McDiarmid and Müller prove that $G_{\mathcal{D}}$ is a unit disk graph if and only if $\mathcal{D}$ is stretchable. Their correctness argument is based on the following lemma:

Lemma 3.12 (adapted from [111, Lemma 1]). *Fix a unit disk intersection representation of $G_{\mathcal{D}}$ in $\mathbb{R}^2$. Then the line arrangement $L = \{\ell_1, \dots, \ell_n\}$ defined by*

$$\ell_i := \left\{p \in \mathbb{R}^2 \mid \|p - a_i\| = \|p - b_i\|\right\}$$

has combinatorial description $\mathcal{D}$ (up to isometry).

The line ℓ_i defined in Lemma 3.12 is the perpendicular bisector of the line segment between a_i and b_i. In fact, the proof by McDiarmid and Müller still holds if we consider the hyperbolic plane $\mathbb{H}^2$ and hyperbolic distances instead. As a corollary of their proof, we obtain the following hyperbolic version of Lemma 3.12:

Lemma 3.13 (based on [111, Lemma 1]). *Fix an intersection representation of $G_{\mathcal{D}}$ by equally sized disks in $\mathbb{H}^2$. Then the line arrangement $L = \{\ell_1, \dots, \ell_n\}$ defined by*

$$\ell_i := \left\{p \in \mathbb{H}^2 \mid d_{\mathbb{H}}(p, a_i) = d_{\mathbb{H}}(p, b_i)\right\}$$

has combinatorial description $\mathcal{D}$ (up to isometry).

Building on the reduction by McDiarmid and Müller sketched above, we are now ready to describe our reduction to prove $\exists\mathbb{R}$-hardness of Recog(HUDG).

Proposition 3.14. Recog(HUDG) *is $\exists\mathbb{R}$-hard.*

Proof. Let $\mathcal{D}$ be the combinatorial description of a simple pseudoline arrangement, i.e., an instance of the $\exists\mathbb{R}$-complete SimpleStretchability problem. Using the reduction from McDiarmid and Müller sketched above, we obtain a graph $G_{\mathcal{D}}$ that is a Euclidean unit disk graph if and only if $\mathcal{D}$ is stretchable in $\mathbb{R}^2$ [111]. By Corollary 3.6 this is equivalent to $\mathcal{D}$ being stretchable in $\mathbb{H}^2$.

Recently, Bläsius, Friedrich, Katzmann and Stephan proved that $\mathsf{UDG} \subseteq \mathsf{HUDG}$, i.e., every Euclidean unit disk graph is also a hyperbolic unit disk graph[3] [23]. We conclude that $G_{\mathcal{D}} \in \mathsf{HUDG}$ if $\mathcal{D}$ is stretchable in $\mathbb{H}^2$.

It remains to show the other direction, namely that $\mathcal{D}$ is stretchable if $G_{\mathcal{D}} \in \mathsf{HUDG}$. This follows directly from Lemma 3.13. ■

The following theorem follows directly from Propositions 3.11 and 3.14:

Theorem 3.15. Recog(HUDG) *is* $\exists\mathbb{R}$*-complete.*

3.4.2 A Framework for $\exists\mathbb{R}$-Hardness

We saw in Section 3.1 that SimpleStretchability is the basis for many $\exists\mathbb{R}$-hardness reductions in $\mathbb{R}^2$. In Section 3.4.1, we adapted the reduction for the Euclidean Recog(UDG) problem to the hyperbolic Recog(HUDG) problem. The goal of this section is to generalize the reduction from Section 3.4.1 to a few (hopefully) simple steps that are independent of the specific Recog(HUDG) problem. We believe that the resulting framework is applicable to other geometric problems as well, see Section 3.4.3.

Let $\Pi_{\mathbb{R}}$ be a geometric decision problem and f be a polynomial-time reduction from SimpleStretchability to $\Pi_{\mathbb{R}}$ establishing $\exists\mathbb{R}$-hardness. Further, let $\Pi_{\mathbb{H}}$ be the corresponding decision problem obtained by considering the hyperbolic plane instead of the Euclidean plane[4]. Our framework consists of the following steps. It allows us to prove $\exists\mathbb{R}$-hardness of $\Pi_{\mathbb{H}}$ by using the reduction f intended for $\Pi_{\mathbb{R}}$:

Framework 3.16 (Hyperbolic $\exists\mathbb{R}$-Hardness).

1. Let $\mathcal{D}$ be a combinatorial description of a simple pseudoline arrangement, i.e., an instance of the $\exists\mathbb{R}$-complete SimpleStretchability problem.

2. Apply the reduction f to obtain $I = f(\mathcal{D})$, i.e., an instance of the Euclidean problem $\Pi_{\mathbb{R}}$. Recall that I is a yes-instance of $\Pi_{\mathbb{R}}$ if and only if $\mathcal{D}$ is stretchable.

3. Prove that every yes-instance of $\Pi_{\mathbb{R}}$ is also a yes-instance of $\Pi_{\mathbb{H}}$.

4. Prove that a realization of a yes-instance of $\Pi_{\mathbb{H}}$ in $\mathbb{H}^2$ can be used to extract a hyperbolic line arrangement realizing $\mathcal{D}$.

3 Note that the converse is not true. Consider again Figure 3.5 for examples of hyperbolic unit disk graphs that are not Euclidean unit disk graphs.

4 You may think of $\Pi_{\mathbb{R}} = $ Recog(UDG) and $\Pi_{\mathbb{H}} = $ Recog(HUDG). In this case, the reduction f is the construction of $G_{\mathcal{D}}$ by McDiarmid and Müller [111].

In the following, we consider the steps of the framework in more detail. We argue why the framework is correct, and how a typical application of the framework looks like.

Steps 1 and 2 require no work when applying the framework.

Step 3 ensures that a stretchable instance $\mathcal{D}$ yields a yes-instance of $\Pi_{\mathbb{H}}$. This step requires coming up with a new argument. Still, we expect it to be relatively simple because locally $\mathbb{R}^2$ and $\mathbb{H}^2$ are very similar. A promising approach is to scale a Euclidean realization of I to a tiny area, and to then interpret the Euclidean polar coordinates as hyperbolic ones.

Step 4 ensures correctness. We have to show that a realization of a yes-instance of $\Pi_{\mathbb{H}}$ in $\mathbb{H}^2$ yields a hyperbolic line arrangement realizing $\mathcal{D}$. In doing so, we prove that a no-instance of SimpleStretchability is mapped to a no-instance of $\Pi_{\mathbb{H}}$. This step is non-trivial, but the Euclidean reduction f for $\Pi_{\mathbb{R}}$ might help us again here (though not as a black box as in Step 2): If we are lucky, the argument why a realization of I in $\mathbb{R}^2$ induces a Euclidean line arrangement realizing $\mathcal{D}$ only uses the axioms of *absolute geometry* (the "common subset" of Euclidean and hyperbolic geometry) and works without any adaptations for realizations in $\mathbb{H}^2$, too.

Example 3.17 (Applying the Framework to Reprove Proposition 3.14)**.** Using our framework, proving $\exists\mathbb{R}$-hardness of Recog(HUDG) becomes extremely simple:

- Step 3 follows from UDG $\subseteq$ HUDG, which is proven by Bläsius, Friedrich, Katzmann and Stephan [23]. Their approach follows exactly our sketched idea of scaling a Euclidean representation to a tiny area, to then interpret the Euclidean polar coordinates of all disks to be hyperbolic polar coordinates.
- Step 4 follows directly from Lemma 3.13. Recall that this lemma is a simple corollary of the proof of Lemma 3.12, i.e., the same lemma for the Euclidean problem Recog(UDG). Proving Lemma 3.13 just requires checking that all steps in the proof of Lemma 3.12 also work in the hyperbolic plane $\mathbb{H}^2$. ⌟

3.4.3 Discussing Further Graph Classes

In search of further applications of our framework, we discuss several other candidate graph classes below. While in principle being applicable to most of these classes, we still reach very different conclusions: As it turns out, for some classes the Euclidean and hyperbolic recognition problems are equivalent, rendering the framework unnecessary for them. In another case, the framework is applicable, but $\exists\mathbb{R}$-completeness is not (yet) proven even for the Euclidean variant. Lastly, we discuss a problem for which we would have to adapt the framework slightly.

Disk Graphs A *(hyperbolic) disk graph* is a graph that admits an intersection representation by disks in $\mathbb{R}^2$, respectively $\mathbb{H}^2$. Let us denote the classes of (hyperbolic) disk graphs by DISK and HDISK.

Indeed, Recog(DISK) is $\exists\mathbb{R}$-complete [97, 112] and our framework is applicable to prove that Recog(HDISK) is also $\exists\mathbb{R}$-complete. Steps 3 and 4 require arguments similar to the ones for (hyperbolic) unit disk graphs. We omit these arguments here, as there is an easier argument for the $\exists\mathbb{R}$-hardness:

It holds that DISK = HDISK, so Recog(DISK) and Recog(HDISK) are actually the same problem. As such, they are obviously equally difficult. To prove DISK = HDISK, consider a disk intersection representation $\mathcal{R}$ in $\mathbb{R}^2$. Enclose $\mathcal{R}$ by a large circle and interpret this circle as a Poincaré disk. As the Poincaré disk model maps hyperbolic circles to Euclidean circles, we constructed a disk representation in $\mathbb{H}^2$. The other direction works similarly by just “ignoring” the Poincaré disk.

Segment Graphs *(Hyperbolic) segment graphs* are those admitting an intersection representation by line segments in $\mathbb{R}^2$, respectively $\mathbb{H}^2$. We denote the corresponding graph classes by SEG and HSEG. $\exists\mathbb{R}$-completeness of Recog(SEG) is well-known [103, 110, 133].

The situation of (hyperbolic) segment graphs is similar to those of (hyperbolic) disk graphs: While our framework is applicable, it is unnecessary because SEG = HSEG. The proof is the same as for DISK = HDISK, just switch to the Beltrami-Klein model of the hyperbolic plane, which maps hyperbolic line segments to Euclidean line segments.

Unit Segment Graphs *(Hyperbolic) unit segment graphs* are intersection graphs of equal length[5] segments in $\mathbb{R}^2$, respectively $\mathbb{H}^2$. The corresponding graph classes are denoted by USEG and HUSEG. It is known that USEG $\subsetneq$ SEG [35], so this is indeed a sensible question.

To the best of our knowledge, it is not known whether Recog(USEG) is $\exists\mathbb{R}$-hard[6]. If $\exists\mathbb{R}$-hardness could be proven by a reduction from SimpleStretchability, then we consider Recog(HUSEG) to be a promising candidate for our framework.

Step 3 of our framework, i.e., proving that USEG $\subseteq$ HUSEG, can be done along the lines of [23, Theorem 1]. Note that we do not know if this inclusion is strict. Step 4 would of course depend on the hypothetical Euclidean reduction.

Linkage Realization Schaefer proves that the GraphRealizability problem is $\exists\mathbb{R}$-complete, i.e., deciding whether a graph has a straight line drawing in $\mathbb{R}^2$ with prescribed edge lengths [134]. He further proves that this also holds if several vertices are allowed to overlap or vertices may lie on edges. In this

5 The reason for considering *equal length* instead of *unit* segments is the same as for (hyperbolic) unit disk graphs, see Remark 3.9.

6 Hoffmann, Miltzow, Weber and Wulf claim that Recog(USEG) is $\exists\mathbb{R}$-complete (personal communication). They prove this by a reduction from SimpleStretchability to Recog(USEG), exactly what we need to apply our framework. However, a written version was not yet available at the time of submission of this thesis.

case, the problem is known as LinkageRealizability. His reduction is from SimpleStretchability.

Are these problems $\exists\mathbb{R}$-complete in $\mathbb{H}^2$ as well? Our framework is not directly applicable because Step 3 does not fit this kind of problem. Nevertheless, Schaefer's Euclidean reduction may serve as the basis for a hyperbolic reduction: The main ingredients are two gadgets that force triples of vertices to be (1) collinear and (2) in a particular order in the resulting drawing. Both gadgets are based on a mechanism known as the *Peaucellier linkage*. Kourganoff presents a hyperbolic version of the Peaucellier linkage [102]. While the details remain to be checked, we are confident that hyperbolic gadgets can be built with this mechanism.

Conjecture 1. The GraphRealizability and LinkageRealizability problems are $\exists\mathbb{R}$-complete in $\mathbb{H}^2$.

3.5 Conclusion and Open Problems

Based on Corollary 3.6, i.e., the equivalence of SimpleStretchability in $\mathbb{R}^2$ and $\mathbb{H}^2$, we proved $\exists\mathbb{R}$-completeness of Recog(HUDG) using the already known reduction for Recog(UDG). The proof strategy was independent of the concrete problem, allowing us to formulate a general framework to transform $\exists\mathbb{R}$-hardness reductions from $\mathbb{R}^2$ to $\mathbb{H}^2$.

Open Problem 1. To which other problems is our framework applicable? How can the framework be extended to be applicable to a wider range of problems? In particular, answer Conjecture 1.

We already discussed several candidates in Section 3.4.3.

In addition to Euclidean and hyperbolic geometry, one might also consider *spherical geometry*. Recall that two points in the Euclidean plane define a unique line. Similarly, two points on a sphere define a unique *great circle* (a largest possible circle through both points). Now, a *(great-)pseudocircle* is a simple closed curve, in particular homeomorphic to a (great-)circle on a sphere. A *(great-)pseudocircle arrangement* is a set of (great)-pseudocircles such that every pair is either independent or crosses exactly twice. Whether a (great-)pseudocircle arrangement is *circularizable*, i.e., homeomorphic to a (great-)circle arrangement, is $\exists\mathbb{R}$-complete [65].

Open Problem 2. Is there a similar framework in spherical geometry?

4 Lombardi Graph Drawing

This chapter is based on our paper "On the Complexity of Lombardi Graph Drawing" published at the *31st International Symposium on Graph Drawing and Network Visualization (GD 2023)* [92].

In this chapter, we present an application of the main result from the previous chapter, which states that SimpleStretchability is equivalent in $\mathbb{R}^2$ and $\mathbb{H}^2$. We study whether a graph admits a so-called *Lombardi drawing*, a well-investigated graph drawing style that embeds graphs into the Euclidean plane $\mathbb{R}^2$. Still, the key idea of the proof is to "think hyperbolically".

4.1 Lombardi Drawings of Graphs

Inspired by the work of American artist Mark Lombardi [88], a *Lombardi drawing* of a given graph G maps vertices to points and edges to circular arcs or line segments connecting their endpoints. Furthermore, each vertex v has *perfect angular resolution*, meaning that all angles between edges incident to v have an equal measure of $2\pi/\deg(v)$. Notably, planarity is not required (even for planar graphs) and the crossing angle at intersections may be arbitrary. See Figure 4.1 for Lombardi drawings of three well-known graphs.

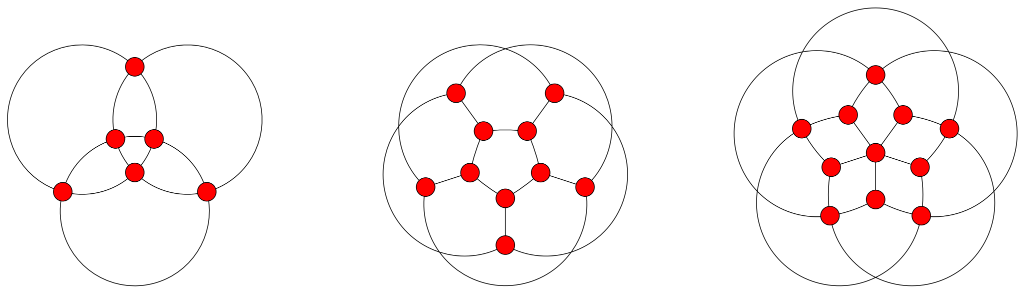

(a) Octahedron Graph **(b)** Petersen Graph **(c)** Grötzsch Graph

Figure 4.1: Three Lombardi drawings created with the *Lombardi Spirograph* by Eppstein [56].

Introduced by Duncan, Eppstein, Goodrich, Kobourov and Nöllenburg over ten years ago [56], Lombardi drawings have received a lot of attention in the graph drawing community, see the related work in Section 4.2 below. While most literature focuses on the construction of Lombardi drawings for different graph classes, the computational complexity to decide whether a Lombardi drawing exists remains largely unknown. To the best of our knowledge, NP-completeness is only known for certain regular graphs under the additional requirement that all vertices must lie on a common circle [56]. No lower or upper bounds on the complexity for general graphs are known (allowing arbitrary vertex placement).

We consider the case that the graph G comes with a fixed *rotation system* $\mathcal{R}$, i.e., a cyclic ordering of the incident edges around each vertex. To this end, we define the following decision problem:

Definition 4.1 (Lombardi).
Input: *A graph G and a rotation system $\mathcal{R}$ that fixes the cyclic ordering of the incident edges around each vertex.*
Question: *Does G admit a Lombardi drawing respecting $\mathcal{R}$.*

Our main result is to determine the exact computational complexity of Lombardi:

Theorem 4.2. Lombardi *is $\exists\mathbb{R}$-complete.*

Previous work frequently utilizes hyperbolic geometry to construct Lombardi drawings [56, 59, 62], the reason being that straight line segments in the hyperbolic plane $\mathbb{H}^2$ can be visualized by circular arcs in the Euclidean plane $\mathbb{R}^2$ (with the same crossing angles). We take a similar approach: A key ingredient of our $\exists\mathbb{R}$-hardness reduction is Corollary 3.6, i.e., that a simple pseudoline arrangement is stretchable in the Euclidean plane $\mathbb{R}^2$ if and only if it is stretchable in the hyperbolic plane $\mathbb{H}^2$ (see Sections 3.1 and 3.2 for the necessary definitions). This result allows us on the one hand to construct Lombardi drawings from hyperbolic line arrangements, and on the other hand to prove that sometimes no Lombardi drawing can exist.

Chapter Outline We start by reviewing the related work in Section 4.2, before recalling the necessary preliminaries in Section 4.3. The main part of this chapter presents the $\exists\mathbb{R}$-completeness proof in Section 4.4. There, we first consider a restricted (but easier) case in Section 4.4.1, before presenting the general case in Section 4.4.2.

4.2 Related Work

Lombardi drawings were introduced by Duncan, Eppstein, Goodrich, Kobourov and Nöllenburg [56], motivated by the network visualizations of Mark Lombardi [88]. While not all graphs admit Lombardi drawings (with or without prescribing the rotation system) [55, 56], many graph classes always admit Lombardi drawings. Among them are 2-degenerate (and some 3-degenerate) graphs [56], subclasses of 4-regular graphs [56, 99] and many classes of planar graphs that even admit planar Lombardi drawings. These include trees [57], cactus graphs [63], Halin graphs [56, 60], subcubic graphs [59], and outerpaths [55]. However, many planar graphs do not admit planar Lombardi drawings in general [55, 56, 59, 61, 99].

A user study confirmed that Lombardi drawings are considered more aesthetic than straight line drawings but do not increase the readability [124].

Many variants have been considered: In a *k-circular Lombardi drawing*, all vertices lie on one of k concentric circles [56]. Slightly relaxing the perfect angular resolution condition leads to *near Lombardi drawings* [43, 99]. Lastly, edges in *k-Lombardi drawings* may be drawn as the concatenation of up to k circular arcs [55, 99].

Not much is known regarding the computational complexity of deciding whether a given graph admits a Lombardi drawing. Proving that a graph class always admits a Lombardi drawing is usually done constructively, and this is the case for all classes mentioned above. In fact, these proofs lead to efficient algorithms (at least in the real RAM model of computation in which square roots can be computed exactly). On the other hand, it is NP-complete to decide whether d-regular graphs have a 1-circular Lombardi drawing if $d \equiv 2 \mod 4$ [56]. Containment in NP might be surprising as this is in sharp contrast to our main result showing $\exists\mathbb{R}$-hardness for general graphs and "classical" Lombardi drawings. NP-membership follows because those graphs are yes-instances if and only if they are Hamiltonian, an easily verifiable property.

4.3 Preliminaries

SimpleStretchability Our reduction to establish $\exists\mathbb{R}$-hardness is from the SimpleStretchability problem as introduced in Section 3.1. In particular, the intersection pattern of a given pseudoline arrangement $\mathcal{A}$ is given by its combinatorial description $\mathcal{D}$. To recall, this means that for each pseudoline $\ell \in \mathcal{A}$ we store an ordered list enumerating all intersections of ℓ with other pseudolines in $\mathcal{A}$.

Hyperbolic Plane $\mathbb{H}^2$ Recall Corollary 3.6, i.e., that a simple pseudoline arrangement $\mathcal{A}$ is stretchable in $\mathbb{R}^2$ if and only if it is stretchable in $\mathbb{H}^2$. In particular, we seek realizations in the Poincaré disk model because its properties lead us towards constructing Lombardi drawings: Hyperbolic lines are mapped to circular arcs, exactly what we need for the edges in a Lombardi drawing. Furthermore, these circular arcs are orthogonal to the Poincaré disk. As the Poincaré disk model is *conformal* (it preserves angles), this helps us obtain perfect angular resolution.

Circle Geometry A *circle inversion* with respect to a circle c with midpoint m and radius r swaps the interior and the exterior of c. Each point $p \in \mathbb{R}^2 \setminus \{m\}$ is mapped to another point $p' \in \mathbb{R}^2 \setminus \{m\}$ such that both lie on the same ray originating from m and such that $\|p - m\| \cdot \|p' - m\| = r^2$. See Figure 4.2 for an example. By adding a single *point at infinity* (denoted by ∞) to $\mathbb{R}^2$, we obtain the so-called *extended plane*, allowing us to extend the definition of a circle inversion to $\mathbb{R} \cup \{\infty\}$. Now m is mapped to ∞ and vice versa.

Circle inversions map circles and straight lines to other circles and straight lines. Additionally, they are conformal, i.e., they preserve the angles between crossing

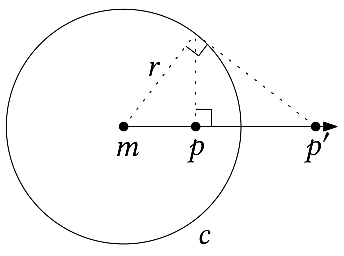


Figure 4.2: Inversion of a point p with respect to a circle c of radius r.

lines and circles. In particular, they map Lombardi drawings to other Lombardi drawings. See [140] for a more thorough introduction.

4.4 Computational Complexity of Lombardi

We prove that Lombardi is $\exists\mathbb{R}$-complete, i.e., deciding whether a graph admits a Lombardi drawing respecting a fixed rotation system. For that, we provide a polynomial-time many-one reduction from SimpleStretchability.

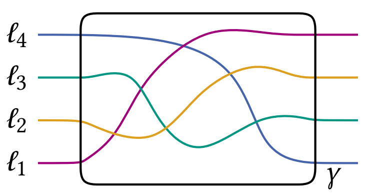
Our reduction is split into two parts: We start by transforming a combinatorial description $\mathcal{D}$ of a simple pseudoline arrangement into a graph G with a fixed rotation system $\mathcal{R}$. The reduction is such that $\mathcal{D}$ is stretchable if and only if G admits a Lombardi drawing Γ respecting $\mathcal{R}$, under the additional restriction that certain cycles in G must be drawn as circles in Γ. Only then we extend our construction so to enforce the additional restrictions "automatically" in each Lombardi drawing.

4.4.1 Restricted Lombardi Drawings

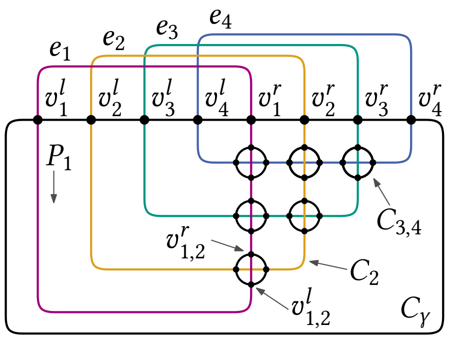
The following construction is illustrated in Figure 4.3. Let $\mathcal{D}$ be a combinatorial description of a simple arrangement $\mathcal{A} = \{\ell_1, \ldots, \ell_n\}$ of $n \geq 2$ pseudolines, i.e., an instance of the $\exists\mathbb{R}$-complete SimpleStretchability problem. As $\mathcal{A}$ is simple, each pseudoline has exactly $n-1$ intersections listed in $\mathcal{D}$, one for each other pseudoline.

The first step of the construction is to extend the pseudoline arrangement $\mathcal{A}$ by a simple closed curve γ intersecting every pseudoline in $\mathcal{D}$ exactly twice, such that γ contains all intersections of $\mathcal{A}$ in its interior, see Figure 4.3a. Let us denote the resulting arrangement by $\mathcal{A}_\gamma$. The combinatorial description $\mathcal{D}_\gamma$ of $\mathcal{A}_\gamma$ can be obtained by adding one intersection with γ to the beginning and to the end of each pseudoline's list. Similarly, $\mathcal{D}_\gamma$ contains a list of intersections for γ whose cyclic ordering is $\ell_1, \ldots, \ell_n, \ell_1, \ldots, \ell_n$.

Now let G_γ be the following graph: We start by adding two vertices v_i^l and v_i^r per pseudoline ℓ_i corresponding to the left- and rightmost intersections of ℓ_i (these are the ones with γ). These $2n$ vertices are then connected to form a cycle in the order they appear on γ. Next, we connect each pair v_i^l and v_i^r by an edge e_i. The rotation system $\mathcal{R}_\gamma$ shall be such that all edges e_i are on the same side of C_γ. Now, for each pseudoline ℓ_i, we add a path P_i from v_i^l to v_i^r by iterating through the list of its intersections from left to right. For each intersection with another pseudoline ℓ_j,



(a) An arrangement of four pseudolines enclosed by a curve γ.



(b) Graph G_γ drawn such that it respects rotation system $\mathcal{R}_\gamma$.

Figure 4.3: Example construction of G_γ and $\mathcal{R}_\gamma$ from a pseudoline arrangement $\mathcal{A}$.

we add (in this order) two new vertices $v_{i,j}^l$ and $v_{i,j}^r$ to the path. In $\mathcal{R}_\gamma$, path P_i and edge e_i should be on opposite sides of C_γ. Let us denote by C_i the cycle formed by concatenating P_i with e_i. Lastly, for each intersection of two pseudolines ℓ_i and ℓ_j with $i < j$, we connect (in this order) $v_{i,j}^l$, $v_{j,i}^l$, $v_{i,j}^r$ and $v_{j,i}^r$ into a 4-cycle $C_{i,j}$. In $\mathcal{R}_\gamma$, the circular ordering around each of the four vertices should contain alternately an edge of $C_{i,j}$ and an edge of C_i respectively C_j. See Figure 4.3b for the complete construction.

In the two lemmas below, we restrict ourselves to drawings of G_γ in which some cycles must be drawn as circles. A cycle C is said to be *drawn as a circle* c if all vertices and edges of C lie on c and the drawing is non-degenerate[1]. In particular, this fixes the ordering of the vertices and edges of C along c (the only degree of freedom is whether this ordering is clockwise or counterclockwise).

Lemma 4.3. *If $\mathcal{D}$ is stretchable, then G_γ has a Lombardi drawing Γ respecting $\mathcal{R}_\gamma$ such that C_γ, all C_i (for $i \in \{1, \dots, n\}$) and all $C_{i,j}$ (for $i, j \in \{1, \dots, n\}$ with $i < j$) are drawn as circles in Γ.*

Proof. By Corollary 3.6, we can obtain a hyperbolic line arrangement realizing $\mathcal{D}$ in the Poincaré disk model. Furthermore, each pseudoline ℓ_i is drawn as a circular arc a_i (with underlying circle c_i) inside and orthogonal to the Poincaré disk. From this, we construct a Lombardi drawing Γ of G_γ respecting $\mathcal{R}_\gamma$.

We denote by c_γ the circle representing the Poincaré disk. We draw the vertices of cycle C_γ on it, such that v_i^l and v_i^r are placed at the left and right intersection of a_i with c_γ. Next, we draw the edges e_i outside c_γ on $c_i \setminus a_i$, and the paths P_i inside c_γ on a_i. As v_i^l and v_i^r have degree four and c_i is orthogonal to c_γ, all vertices on C_γ have perfect angular resolution.

Next, for each pair of intersecting pseudolines ℓ_i and ℓ_j (with $i < j$), we place the vertices $v_{i,j}^l$ and $v_{i,j}^r$ to the left, respectively to the right, of the intersections

1 A drawing is *degenerate* if two vertices are drawn at the same point or a vertex is drawn in the interior of an edge.

on a_i (and similar $v^l_{j,i}$ and $v^r_{j,i}$ on a_j) such that they lie on a common circle $c_{i,j}$ which is orthogonal to a_i and a_j. Here, the orthogonality of $c_{i,j}$ with a_i and a_j guarantees perfect angular resolution at the four involved vertices. (We prove in Lemma 4.11 below that such a circle indeed exists.) Furthermore, we can choose $c_{i,j}$ small enough so that no two such circles intersect, touch or contain each other.

The drawing is non-degenerate, respects $\mathcal{R}_\gamma$, has C_γ, all C_i and all $C_{i,j}$ drawn as circles and perfect angular resolution, i.e., it is a Lombardi drawing. ∎

See Figure 4.4 for a Lombardi drawing of the graph shown in Figure 4.3b that is constructed as described in the proof of Lemma 4.3.

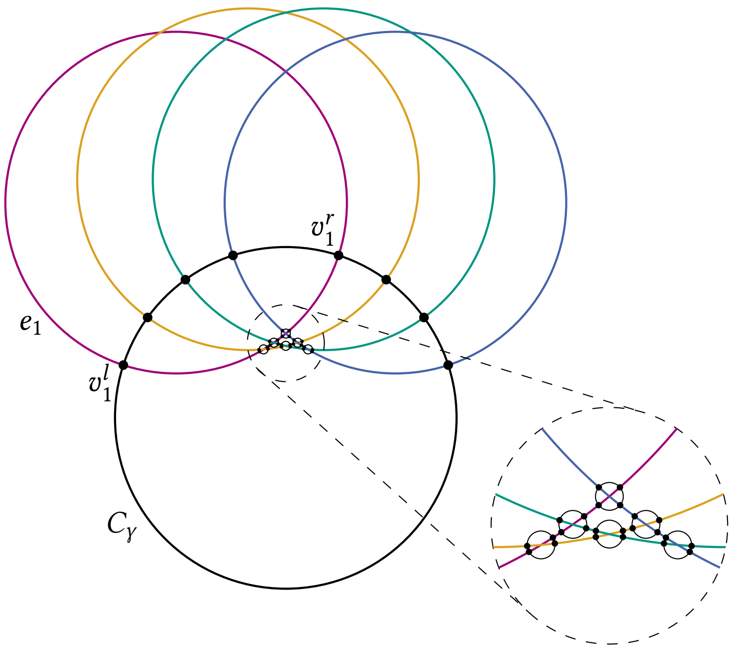


Figure 4.4: A Lombardi drawing of the graph constructed in Figure 4.3.

Lemma 4.4. *If G_γ has a Lombardi drawing Γ respecting $\mathcal{R}_\gamma$ such that C_γ, all C_i (for $i \in \{1, \dots, n\}$) and all $C_{i,j}$ (for $i, j \in \{1, \dots, n\}$ with $i < j$) are drawn as circles in Γ, then $\mathcal{D}$ is stretchable.*

Proof. Let c_γ be the circle that C_γ is drawn as. We can assume without loss of generality that all edges e_i are drawn outside c_γ and all paths P_i are drawn inside c_γ, as we can otherwise consider the drawing obtained from Γ by a circle inversion with respect to c_γ. Recall that all vertices on C_γ have degree four. Let c_i be the circles that the cycles C_i are drawn as (for $i \in \{1, \dots, n\}$). All c_i are orthogonal to c_γ because Γ has perfect angular resolution.

This allows us to interpret c_γ as a Poincaré disk, and each circular arc a_i of c_i containing the drawing of P_i in Γ as a hyperbolic line. Therefore, the interior of c_γ induces a hyperbolic line arrangement. We prove that this hyperbolic line arrangement has combinatorial description $\mathcal{D}$.

To this end, consider an arbitrary but fixed path P_i. From left to right along P_i, we encounter pairs of vertices $v^l_{i,j}$ and $v^r_{i,j}$ that, together with $v^l_{j,i}$ and $v^r_{j,i}$, form a 4-cycle corresponding to the intersection between pseudolines ℓ_i and ℓ_j (assuming $i < j$). As c_i and c_j are circles orthogonal to c_γ, their circular arcs a_i and a_j intersect at most once. Furthermore, as the vertices on $C_{i,j}$ are alternately on P_i and P_j, there must be an odd number of intersections between a_i and a_j inside the drawing of $C_{i,j}$ in Γ. It follows, that a_i and a_j intersect exactly once, and they do so between $v^l_{i,j}$ and $v^r_{i,j}$. Thus, each pseudoline intersects each other pseudoline exactly once and in the order described by $\mathcal{D}$, i.e., our Lombardi drawing Γ induces a hyperbolic line arrangement with combinatorial description $\mathcal{D}$. Because $\mathcal{D}$ is simple and by Corollary 3.6, it follows that $\mathcal{D}$ is then also stretchable in $\mathbb{R}^2$. ■

Summarizing the results so far, we see that Lemmas 4.3 and 4.4 give a reduction from SIMPLESTRETCHABILITY to a restricted form of LOMBARDI. In what follows, we see how to incorporate these restrictions into the reduction itself.

4.4.2 Enforcing the Circles

In Lemmas 4.3 and 4.4 above, we assumed that certain cycles in G_γ are drawn as circles. Below, we describe how we can omit this explicit restriction by enforcing all possible Lombardi drawings to "automatically" satisfy it.

An *arc-polygon* is a set of points $v_0, \ldots, v_k$ such that v_i and v_{i+1} (with $v_{k+1} = v_0$) are connected by a circular arc or line segment. An arc-polygon is *simple* if it does not self-touch or self-intersect. In case of two or three vertices, we speak of a *bigon* and an *arc-triangle*, respectively. We utilize a lemma by Eppstein, Frishberg and Osegueda [63] who characterize simple arc-triangles. Following their notation, the vertices v_0, v_1 and v_2 are numbered in clockwise order such that the interior of the arc-triangle is to the right when going from v_i to $v_{(i+1) \bmod 3}$. The vertices enclose internal angles θ_0, θ_1 and θ_2. If the vertices do not lie on a common line, then they define a unique circle c. In this case, we denote by ϕ_i the internal angle of the bigon enclosed by c and the circular arc a_i between $v_{(i-1) \bmod 3}$ and $v_{(i+1) \bmod 3}$. Negative (positive) values of ϕ_i mean that a_i is outside (inside) of c, and $\phi_i = 0$ means that a_i is on c. See Figure 4.5 for an illustration.

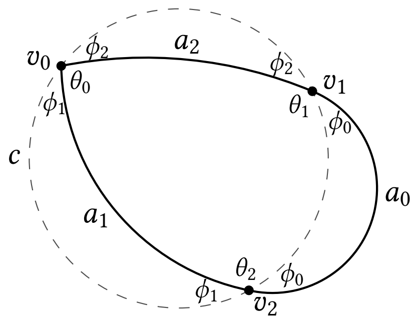


Figure 4.5: A simple arc-triangle illustrating the used notation.

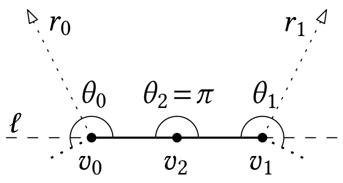


Figure 4.6: Ruling out the collinear case in Lemma 4.6.

Lemma 4.5 ([63, Lemma 4 and Corollary 5]). *Let v_0, v_1 and v_2 be a simple arc-triangle as above, not on a common line. Then for $\psi = \left(\pi - \sum_{i=0}^{2} \theta_i\right)/2$ it holds that $\phi_i = \psi + \theta_i$.*

We use Lemma 4.5 to prove that prescribing the interior angles of an arc-triangle to certain values is enough to guarantee that one of its vertices lies on the underlying circle of the circular arc connecting the other two vertices:

Lemma 4.6. *Let v_0, v_1 and v_2 be a simple arc-triangle with $\theta_0 = \theta_1 \in [\pi, 3\pi/2)$ and $\theta_2 = \pi$. Further, the edge v_0v_2 is drawn as a circular arc a_1 (and not as a line segment) with an underlying circle c_1. Then v_0, v_1 and v_2 do not lie on a common line, and c_1 is the unique circle through them, with v_1 on $c_1 \setminus a_1$.*

Proof. We first rule out the case that all three vertices lie on a common line ℓ, see Figure 4.6: As the internal angle at v_2 has size π, vertices v_0 and v_1 must be on opposite sides of ℓ. The internal angle θ_0 at v_0 defines a ray r_0 that must contain the center of the underlying circle of a_2 (the circular arc connecting v_0 and v_1). Similarly, θ_1 defines another ray r_1 that must contain the center of a_2. However, $r_0 \cap r_1 = \emptyset$, because $\theta_0 = \theta_1 \in [\pi, 3\pi/2)$. We conclude that the three vertices cannot lie on a common line. Thus, they lie on a unique circle c.

We use Lemma 4.5 to compute the internal angle of the bigon enclosed by a_1 and c: We get that $\psi = (\pi - \sum_{i=0}^{2} \theta_i)/2 = -\theta_1$ and with that $\phi_1 = -\theta_1 + \theta_1 = 0$, i.e., a_1 must lie on c and in particular $c_1 = c$. As simple arc-triangles do not self-intersect or self-touch, it follows that v_1 lies on $c_1 \setminus a$. ■

Continuing with our reduction, we extend G_γ and $\mathcal{R}_\gamma$ by new vertices and edges to obtain a graph $G_\mathcal{D}$ with rotation system $\mathcal{R}_\mathcal{D}$. We will identify several arc-triangles in $G_\mathcal{D}$ that fulfill the conditions of Lemma 4.6. Iteratively applying this lemma allows us to prove that C_γ as well as all C_i (for $i \in \{1, \ldots, n\}$) and all $C_{i,j}$ (for $i, j \in \{1, \ldots, n\}$ with $i < j$) must be drawn as circles in any Lombardi drawing of $G_\mathcal{D}$ respecting $\mathcal{R}_\mathcal{D}$.

Let us introduce some notation for cycles $C = (v_1, \ldots, v_k)$. We denote by e_i the edge of the cycle connecting v_i and v_{i+1} (where $v_{k+1} = v_1$). If each v_i (for $i \in \{1, \ldots, k\}$) has degree 4, then the incident edges form four angles between them, which all have size $\pi/2$ because of perfect angular resolution. We call these the *quadrants* of v_i and label them by $q_i^1, \ldots, q_i^4$ in counterclockwise order. In doing so, we label such that q_i^1 is left of e_i when traversing it from v_i to v_{i+1}. A *half-edge* is an edge incident to a vertex whose other endpoint is not yet specified.

A *circle gadget* for a cycle $C = (v_1, \ldots, v_k)$ as above together with $k-3$ additional half-edges in each quadrant of all $v \in V(C)$ is the following set of edges: For

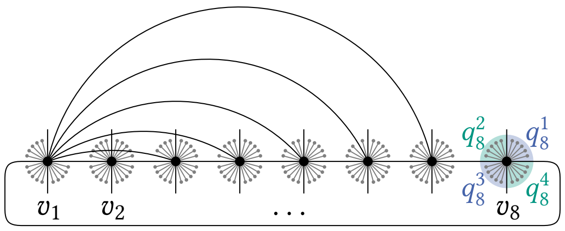


Figure 4.7: Circle gadget for cycle $C = (v_1, \dots, v_8)$. The four quadrants at v_8 are labeled.

$j \in \{1, \dots, k-3\}$, the j-th half-edge of q_1^1 in clockwise order is joined with the j-th half-edge of q_{k-j}^2 in counterclockwise order, see Figure 4.7. The following lemma shows that these edges enforce that C is drawn as a circle.

Lemma 4.7. *Let G be a graph containing a cycle $C = (v_1, \dots, v_k)$, extended with the edges of a circle gadget as described above. Then in every Lombardi drawing Γ that maps the edge e_k to a circular arc a (i.e., not to a line segment), all vertices and edges of C are drawn onto the underlying circle c of a.*

Proof. First, note that each vertex has equally many incident (half-)edges in each of its four quadrants, so by the perfect angular resolution of Γ, each quadrant spans an angle of $\pi/2$. In particular, the angle between any two consecutive cycle edges is π. Now consider the three cycle vertices v_1, v_{k-1} and v_k which form a simple arc-triangle whose internal angles satisfy the conditions of Lemma 4.6. It follows that v_{k-1} and e_{k-1} are drawn onto $c \setminus a$.

As we now know that the path from v_1 counterclockwise via v_k to v_{k-1} follows a single circular arc a' in Γ, the same argument can be repeated for the simple arc-triangle formed by the points v_1, v_{k-2} and v_{k-1}. It follows that v_{k-2} and e_{k-2} lie on c. Iterating the argument until we reach the simple arc-triangle formed by v_1, v_2 and v_3 proves the statement. ■

With the circle gadget at hand, we can finally construct $G_\mathcal{D}$ and $\mathcal{R}_\mathcal{D}$. Recall that G_γ is 4-regular. We add $2n-3$ half-edges into each quadrant of every vertex $v \in V(G_\gamma)$ and then the following circle gadgets:

- For the cycle $C_\gamma = (v_1^l, \dots, v_n^l, v_1^r, \dots, v_n^r)$ on $2n$ vertices. Here v_1^l takes the role of v_1 and v_n^r takes the role of v_k in the circle gadget.
- For each cycle C_i on $2n$ vertices (for $i \in \{1, \dots, n\}$). Here v_i^r takes the role of v_1 and v_i^l takes the role of v_k in the circle gadget.
- For each cycle $C_{i,j}$ (for $i, j \in \{1, \dots, n\}$ with $i < j$). Here $v_1 = v_{i,j}^l$, $v_2 = v_{j,i}^l$, $v_3 = v_{i,j}^r$ and $v_4 = v_{j,i}^r$.

Note that several vertices are involved in multiple circle gadgets in our construction. This is not a problem because we carefully placed the circle gadgets such that no two circle gadgets operate in the same quadrant on each affected vertex. See Figure 4.8 for a visualization (for simplicity, just C_1 and one C_i are drawn): Green half-edges

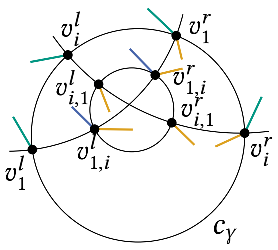


Figure 4.8: How to combine different circle gadgets.

show which quadrants are used by the circle gadget for C_γ. Orange half-edges belong to the circle gadgets of C_1 and C_i. Lastly, blue half-edges belong to the circle gadget of $C_{1,i}$.

As the last step of the reduction, all remaining half-edges are terminated with a new vertex of degree one. The resulting graph and rotation system are $G_\mathcal{D}$ and $\mathcal{R}_\mathcal{D}$.

Lemma 4.8. *If $\mathcal{D}$ is stretchable, then $G_\mathcal{D}$ has a Lombardi drawing respecting $\mathcal{R}_\mathcal{D}$.*

Proof. We start by applying Lemma 4.3 to obtain a Lombardi drawing Γ_γ of the subgraph G_γ of $G_\mathcal{D}$ respecting $\mathcal{R}_\gamma$ and in which C_γ, all C_i (for $i \in \{1, \dots, n\}$) and all $C_{i,j}$ (for $i, j \in \{1, \dots, n\}$ with $i < j$) are drawn as circles.

It remains to draw the vertices and edges added by the circle gadgets. Recall that equally many edges were added into each quadrant of all vertices $v \in V(G_\gamma)$. Thus, the angles between edges in $E(G_\gamma)$ remain unchanged. New edges must be drawn with equal angles between them into their quadrants to obtain perfect angular resolution.

Let e be an edge of a circle gadget with endpoints u and v, and let c be the circle that u and v lie on in Γ_γ. By construction, e was obtained by joining the j-th half-edge in clockwise order in the first quadrant of u with the j-th half-edge in counterclockwise order in the second quadrant of v for some j. Thus, the two angles between c and the arc representing e at u and v are equal and in $[0, \pi/2)$. There is exactly one circular arc that e can be drawn onto [56, Property 1].

Lastly, we need to make sure that the vertices of degree 1 that resulted from unjoined half-edges are drawn such that they do not lie on any other edge. This can be achieved by drawing the half-edges sufficiently short. ■

Lemma 4.9. *If $G_\mathcal{D}$ has a Lombardi drawing Γ respecting $\mathcal{R}_\mathcal{D}$, then $\mathcal{D}$ is stretchable.*

Proof. Recall that edges of $G_\mathcal{D}$ are mapped to circular arcs or line segments in Γ and that each vertex has perfect angular resolution. We begin by analyzing how the cycle $C_\gamma = (v_1^l, \dots, v_n^l, v_1^r, \dots, v_n^r)$ in $G_\mathcal{D}$ must be drawn in Γ. We can assume that v_1^l, v_n^r and v_{n-1}^r do not lie on a common line, and that the edge between v_1^l and v_n^r is mapped to a circular arc a (by a suitable circle inversion). Then, by Lemma 4.7, all vertices and edges of C_γ lie on the underlying circle c_γ of a.

Next, we consider the circles C_i for $i \in \{1, \dots, n\}$. By applying a suitable circle inversion with respect to c_γ (if necessary), we can assume that the paths P_i are drawn inside c_γ. Conversely, the edges e_i are drawn outside c_γ. Therefore, they are drawn as circular arcs (because a line segment would be inside c_γ). Applying Lemma 4.7 to each C_i guarantees that it is drawn as a circle c_i in Γ.

It remains to consider the circles $C_{i,j}$ for $i, j \in \{1, \dots, n\}$ with $i < j$. If they are not already drawn as circles $c_{i,j}$, we can replace their drawing by a circle with sufficiently small radius by Lemma 4.11.

Now it follows from Lemma 4.4 that $\mathcal{D}$ is stretchable. ■

4.4.3 Establishing ∃ℝ-Completeness

The previous sections establish ∃ℝ-hardness of Lombardi. It remains to prove ∃ℝ-membership as well.

Lemma 4.10. Lombardi $\in \exists\mathbb{R}$.

Proof. We describe a polynomial-time verification algorithm for a real RAM machine. Recall that the existence of such an algorithm proves ∃ℝ-membership (see Section 2.2 and [64]).

Let G with rotation system $\mathcal{R}$ be a yes-instance of the Lombardi problem. The obvious witness is a Lombardi drawing Γ mapping each vertex to a point and each edge to a circular arc or line segment. Given real-valued coordinates for each vertex and a description of each edge, we have to check the following:

- No edge contains a vertex, except for its two endpoints.
- No two edges share more than one point.
- No two vertices are mapped to the same point.
- Each vertex has perfect angular resolution.
- The rotation system of Γ is $\mathcal{R}$.

∃ℝ-membership follows because all of the above checks can easily be done in polynomial time on a real RAM machine. ■

Proof of Theorem 4.2 (∃ℝ-Completeness of Lombardi). Let $\mathcal{D}$ be a combinatorial description of a simple pseudoline arrangement. Construct $G_{\mathcal{D}}$ and $\mathcal{R}_{\mathcal{D}}$ as described above. By Lemmas 4.8 and 4.9, $\mathcal{D}$ is stretchable if and only if $G_{\mathcal{D}}$ admits a Lombardi drawing respecting $\mathcal{R}_{\mathcal{D}}$, proving ∃ℝ-hardness. ∃ℝ-membership is proven in Lemma 4.10. Together, this yields the ∃ℝ-completeness of Lombardi. ■

4.4.4 Omitted Details

In the proof of Lemma 4.3, we construct a Lombardi drawing Γ_γ of G_γ. There, for each pseudoline ℓ_i, the corresponding path P_i is drawn along a single circular arc. Given a pair ℓ_i, ℓ_j of two pseudolines (with $i < j$), their corresponding circular arcs a_i and a_j cross, and we draw a sufficiently small circle $c_{i,j}$ enclosing their intersection. Then the vertices $v^l_{i,j}$, $v^l_{j,i}$, $v^r_{i,j}$ and $v^r_{j,i}$ are placed on the intersection of $c_{i,j}$ with a_i and a_j. To achieve perfect angular resolution, $c_{i,j}$ must be orthogonal to both circular arcs. See again Figure 4.4. The following lemma guarantees the existence of such a circle:

Lemma 4.11. *Let a_1 and a_2 be two circular arcs with a unique proper intersection p (i.e., not a touching). There is a sufficiently small circle c orthogonal to a_1 and a_2 enclosing p.*

Proof. For $i \in \{1, 2\}$, let c_i be the underlying circle of a_i with center (x_i, y_i) and radius r_i. Without loss of generality, we assume that $r_1 \geq r_2$. We denote by d the Euclidean distance between the centers of c_1 and c_2. By the assumption that a_1 and a_2 have a proper intersection, c_1 and c_2 must have two intersections, which is the case if and only if $d < r_1 + r_2$ and $d > r_1 - r_2$.

We shall find a circle c orthogonal to c_1 and c_2 with a tiny but fixed radius r (whose exact value is to be determined later). Circle c is orthogonal to c_i if and only if $r_i^2 + r^2 = d_i^2$, where d_i denotes the distance between the center of c_i and c [140]. Thus, the center of c must be at distance $d_i = (r_i^2 + r^2)^{1/2}$ from the center of c_i. In particular, for $i \in \{1, 2\}$, the center of c must be on the circle c_i' with center (x_i, y_i) and radius d_i.

It remains to choose r such that c_1' and c_2' have non-empty intersection. This is the case if and only if $d \leq d_1 + d_2$ and $d \geq |d_1 - d_2| = d_1 - d_2$ (where the last step follows from $r_1 \geq r_2$). The first inequality holds for any choice of $r > 0$ because $d < r_1 + r_2 < d_1 + d_2$ (where the first inequality follows from c_1 having two intersections with c_2). For the second inequality, we know that $d > r_1 - r_2$ (again, because c_1 and c_2 intersect twice), so there is an $\varepsilon > 0$ such that $d = r_1 - r_2 + \varepsilon$. Any $r > 0$ such that $|d_i - r_i| < \varepsilon/2$ works for us: Then

$$d = r_1 - r_2 + \varepsilon > \left(d_1 - \frac{\varepsilon}{2}\right) - \left(d_2 + \frac{\varepsilon}{2}\right) + \varepsilon = d_1 - d_2$$

as desired. Thus, for sufficiently small $r > 0$, there are exactly two possible centers for circles that are orthogonal to c_1 and c_2 and enclose one of their intersections each. Choose the one corresponding to the intersection between a_1 and a_2. ■

4.5 Conclusion and Open Problems

We proved that Lombardi is $\exists\mathbb{R}$-complete, i.e., deciding whether a given graph G admits a Lombardi drawing respecting a fixed rotation system $\mathcal{R}$. To the best of our knowledge, this is the first result on the complexity of Lombardi drawing for general graphs.

In fact, Lombardi drawing is just a special case of the more general problem, where we want to draw a graph with circular arc edges and angles of prescribed size. Our reduction immediately proves $\exists\mathbb{R}$-hardness for this problem as well:

Corollary 4.12. *Let G be a graph with a rotation system $\mathcal{R}$, and let Θ be an angle assignment prescribing the size of all angles in $\mathcal{R}$. Then it is $\exists\mathbb{R}$-complete to decide* *whether G admits a drawing with edges as circular arcs or line segments respecting $\mathcal{R}$ and Θ.*

On the other hand, several interesting questions remain open: Our reduction heavily relies on fixing the rotation system $\mathcal{R}$. By the perfect angular resolution requirement, this fixes all angles in every Lombardi drawing. We wonder whether the problem remains $\exists\mathbb{R}$-complete without fixing $\mathcal{R}$:

Open Problem 3. What is the computational complexity of deciding whether a graph admits any Lombardi drawing (without fixing a rotation system $\mathcal{R}$)?

Concerning Open Problem 3, let us note that an extension of Lemma 4.10 for $\exists\mathbb{R}$-membership is trivial. Thus, we actually ask for a stronger $\exists\mathbb{R}$-hardness reduction.

Given a planar graph, one usually asks for a planar Lombardi drawing. The graphs constructed in our reduction are in general not planar. In fact, they contain arbitrarily large clique minors. This motivates our second open problem:

Open Problem 4. What is the complexity of deciding whether a planar graph admits a planar Lombardi drawing (with or without fixing a rotation system $\mathcal{R}$)?

Concerning Open Problem 4, let us note that a crossing gadget can easily be constructed: At each intersection of two circular arcs a_1 and a_2, we add four vertices that form a cycle C around it, just like the cycles $C_{i,j}$ in our reduction. Inside C, one of the circular arcs, say a_1, can be "cut through", thereby eliminating the intersection. Complete the two loose ends with new vertices of degree one. The intact arc a_2 then acts just like a circle gadget, forcing C to be drawn as a circle. Perfect angular resolution and the fixed rotation system guarantee that the two halves of a_1 must lie inside C.

The existence of a crossing gadget does not yet solve Open Problem 4, because we do not know in advance which pairs of edges intersect (this is particularly obtuse for the edges of multiple overlapping circle gadgets).

5 Training Neural Networks

This chapter is based on our paper “Training Fully Connected Neural Networks is ∃ℝ-Complete” published in *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)* [18]. It is joint work with Daniel Bertschinger, Christoph Hertrich, Tillmann Miltzow and Simon Weber.

This chapter considers our second example problem from Chapter 1, i.e., the neural network training problem. For this, we shift our focus away from SimpleStretchability and towards ETR itself, more precisely to its ∃ℝ-complete special case ETR-Inv.

While the general training problem, usually denoted by EmpiricalRiskMinimization, is best described as a combinatorial optimization problem, our hardness proof already holds for networks with an extremely simple architecture. As we are going to see, this simple architecture allows us to study the problem from a geometric perspective, and therefore to benefit from the ideas used in previous hardness reductions for other geometric ∃ℝ-complete problems.

5.1 Introduction

The usage of neural networks in modern computer science is ubiquitous. They are arguably the most powerful tool at our hands in machine learning [74]. EmpiricalRiskMinimization, i.e., the neural network training problem, is one of the most fundamental algorithmic questions in the field. It is natural to ask for its algorithmic complexity, and indeed, a wide body of literature deals with this question for many special cases (see Section 5.5).

Abrahamsen, Kleist and Miltzow [4] prove that the problem is ∃ℝ-complete already for certain two-layer neural networks and linear activation functions. Their result has one major downside, namely that the network architecture is *adversarial*: The hardness inherently relies on choosing a network architecture that is particularly difficult to train. The instances by Abrahamsen, Kleist and Miltzow could be trained trivially if they were fully connected. This stems from the fact that they use the identity function as the activation function, which reduces the problem to matrix factorization. While intricate network architectures such as convolutional and residual neural networks, pooling, autoencoders and generative adversarial neural networks are common in practice, they are usually designed in a way that facilitates training rather than making it difficult [74]. We strengthen the result in [4] by showing ∃ℝ-completeness for *fully connected* two-layer neural networks.

This shows that ∃ℝ-hardness does not stem from one specifically chosen worst-case architecture but is inherent in the neural network training problem itself.

Although in practice a host of different architectures are used, fully connected two-layer neural networks are arguably the most basic ones and they are often part of more complicated network architectures [74]. We show hardness even for the case of fully connected two-layer ReLU neural networks with exactly two input and output neurons.

Remarkably, with only one instead of two output neurons, the problem is in NP. This follows from a celebrated combinatorial search algorithm by Arora, Basu, Mianjy and Mukherjee [12]. Our result explains why their algorithm was never successfully generalized to more complex network architectures: Adding only a second output neuron significantly increases the computational complexity of the problem, from being contained in NP to being complete for ∃ℝ.

To achieve our result, our reduction follows a completely novel approach compared to the reduction in [4]. Instead of encoding polynomial inequalities into an adversarial network architecture, we make use of the underlying geometry of the functions computed by two-layer neural networks and utilize the fact that their different output dimensions have nonlinear dependencies.

Chapter Outline We start by formally introducing neural networks and their training problem in Section 5.2. Thereafter, we present our main results in Section 5.3. Section 5.4 presents different perspectives on how to interpret our findings, including an in-depth discussion of the strengths and limitations. We cover related work in Section 5.5. A simplified overview of the proof ideas is in Section 5.6. The detailed proofs of ∃ℝ-membership and ∃ℝ-hardness follow in Section 5.7 and Section 5.8, respectively.

5.2 Preliminaries

We introduce the necessary definitions and concepts related to neural network training.

All neural networks considered in this thesis have a very simple architecture. Therefore, for ease of presentation, we do not define neural networks and their training problem in full generality here. Instead, we restrict ourselves to the simple architectures we need.

Definition 5.1 (Fully Connected Two-Layer Neural Network). *A fully connected two-layer neural network $N = (S \cup H \cup T, E)$ is a directed acyclic graph (the architecture) with real-valued edge weights. The vertices, called neurons, are partitioned into the inputs S, the hidden neurons H and the outputs T. Exactly all possible edges from S to H as well as all possible edges from H to T are present. Additionally, each hidden neuron has a real-valued bias and an activation function $(\mathbb{R} \to \mathbb{R})$.*

The probably most commonly used activation function [12, 69, 74] is the *rectified linear unit (ReLU)* defined as

$$\begin{aligned} \mathrm{ReLU}\colon \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto \max\{0, x\}. \end{aligned}$$

See Figure 5.1 for a small fully connected two-layer ReLU neural network.

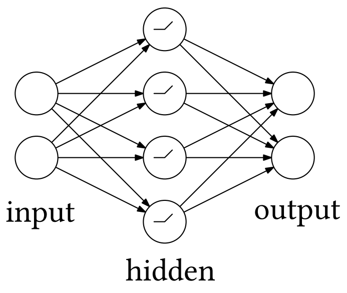


Figure 5.1: A fully connected two-layer neural network as studied in this thesis. The symbol inside the hidden neurons denotes the ReLU activation function.

Given a neural network architecture N as defined above, let us fix an arbitrary ordering on S and T. Then N realizes a function $f(\cdot, \Theta)\colon \mathbb{R}^{|S|} \longrightarrow \mathbb{R}^{|T|}$, where Θ denotes the weights and biases that parameterize the function. For $x \in \mathbb{R}^{|S|}$ we define $f(\cdot, \Theta)$ inductively: The i-th input neuron forwards the i-th component of x to all its outgoing neighbors. Each hidden neuron forms the weighted sum over all incoming values, adds its bias, applies the activation function to this sum, and forwards it to all outgoing neighbors. An output neuron also forms the weighted sum over all incoming values but does not add a bias and does not apply any activation function.

For our purposes, we define the following special case of EmpiricalRiskMinimization, i.e., a restriction of the general neural network training problem.

Definition 5.2 (TrainNN).
Input: *A 5-tuple $(N, \varphi, D, \gamma, c)$:*

- *$N = (S \cup H \cup T, E)$ is the fully connected two-layer network architecture.*
- *$\varphi\colon \mathbb{R} \longrightarrow \mathbb{R}$ is the activation function for hidden neurons.*
- *$D \subseteq \mathbb{Q}^{|S|} \times \mathbb{Q}^{|T|}$ is the* training data*, i.e., a set of n* data points *of the form $(x; y)$. Here, y is called a* label.
- *$\gamma \in \mathbb{Q}_{\geq 0}$ is the* target error.
- *$c\colon \mathbb{R}^{|T|} \times \mathbb{R}^{|T|} \longrightarrow \mathbb{R}_{\geq 0}$ is a* loss function*. We require that it is computable on a real RAM machine in polynomial time.*

Question: *Are there weights and biases Θ such that*

$$\sum_{(x;y)\in D} c\big(f(x, \Theta), y\big) \leq \gamma?$$

The data points D and the target error γ are part of the input, which has finite length (in some chosen encoding). We restrict them to be rational numbers because these are straightforward to encode by noting their numerator and denominator in binary, in contrast to much more complicated encodings required for arbitrary (algebraic) real numbers.

Additionally, we require that the loss function is *honest*, meaning that it returns zero if and only if the data is fit exactly. We will see that this is the only requirement on the loss function to prove $\exists\mathbb{R}$-hardness of the zero-error case ($\gamma = 0$).

5.3 Results

Our main result is the following theorem establishing $\exists\mathbb{R}$-completeness of TrainNN even in very restricted cases:

Theorem 5.3. TrainNN *is* $\exists\mathbb{R}$*-complete, even if*

- *there are only two input neurons,*
- *there are only two output neurons,*
- *the number of data points is linear in the number of hidden neurons,*
- *the data has only* 13 *different labels,*
- *the target error is* $\gamma = 0$ *and*
- *the* ReLU *activation function is used.*

Let us note that the combination of all restrictions in Theorem 5.3 describes a special case of EmpiricalRiskMinimization. Proving that TrainNN is $\exists\mathbb{R}$-hard also implies $\exists\mathbb{R}$-hardness for more general cases like EmpiricalRiskMinimization, for example with more input/output neurons, more data points, more labels, arbitrary $\gamma \geq 0$, and possibly different activation functions.

Additionally, our $\exists\mathbb{R}$-hardness reduction implies *algebraic universality*:

Theorem 5.4. *Let* $\alpha \in \mathbb{R}$ *be an algebraic number. Then there exists an instance of* TrainNN*, which has a solution with weights and biases* Θ *from* $\mathbb{Q}[\alpha]$*, but no solution when the weights and biases* Θ *are restricted to a field* $\mathbb{F}$ *not containing* α*.*

Here, $\mathbb{Q}[\alpha]$ is the smallest field extension of $\mathbb{Q}$ containing α. This means that there are training instances for which all global optima require irrational weights or biases, even if all data points are integral.

Algebraic universality is known to hold for various $\exists\mathbb{R}$-complete problems [6], however this is not an automatism: Algebraic universality cannot occur in problems where the solution space is open, for example for Recog(UDG) [112]. On the other hand, reductions to prove universality do not need to be in polynomial time. Therefore, universality can be shown without proving $\exists\mathbb{R}$-hardness at the same time.

5.4 Discussion

There is already a wide body of literature about the computational hardness of EmpiricalRiskMinimization, see Section 5.5 below. In order to clarify our contribution, we use this section to discuss our results from various perspectives, pointing out strengths and limitations.

Number of Input Neurons In practice, neural networks are often trained on high dimensional data, thus having only two input neurons is even more restrictive than the practical setting. Note that we easily obtain hardness for higher input dimensions by simply placing all data points of our reduction into a two-dimensional subspace. The precise complexity of training fully connected two-layer neural networks with only one-dimensional input and multi-dimensional output remains unknown. While this setting does not have practical relevance, we are still curious about this open question from a purely mathematical perspective.

Number of Output Neurons As discussed earlier, if there is only one output neuron instead of two, then the problem is known to be NP-complete [12, 66]. Our reduction can easily be extended to the case with more than two output neurons by padding all output vectors with zeros. Thus, the complexity classification is complete with respect to the number of output neurons.

Number of Hidden Neurons Consider a situation where the number m of hidden neurons is larger than the number n of data points. If there are no two contradicting data points $(x_1; y_1)$ and $(x_2; y_2)$ with $x_1 = x_2$ but $y_1 \neq y_2$, then we can always fit all data points exactly [156]. Thus, we need at least a linear number of data points in terms of m for $\exists\mathbb{R}$-hardness. Our result is (asymptotically) tight in this aspect. Note that by adding additional data points, the ratio between n and m can be made arbitrarily large. Thus, our reduction holds also for all settings in which m is (asymptotically) much smaller than n.

Number of Output Labels The number of labels used in our reduction is just 13. Requiring only a small constant number of different labels shows the relevance of our result to practice, where the number of data points often largely exceeds the number of labels, for instance, in classification tasks.

If all labels are contained in a one-dimensional affine subspace, then the problem is in NP, as they can be projected down to one-dimensional labels and the problem can be solved with the algorithm by Arora, Basu, Mianjy and Mukherjee [12]. As any two labels span a one-dimensional affine subspace, the problem can only be $\exists\mathbb{R}$-hard for at least three affinely independent output labels.

We think it is not particularly interesting to close the gap between 13 and 3 output labels, but it would be interesting to investigate the complexity of the problem when output labels have more structure. For example, in classification tasks one often uses *one-hot encodings*, where the output dimension equals the number of classes and all labels have the form $(0, \ldots, 0, 1, 0, \ldots, 0)$. Note that in this case, at least three output dimensions are needed to obtain three different labels.

Target Error For simplicity, we only prove hardness for the case with target error $\gamma = 0$. However, it is generally not required to fit the data exactly in real-world applications. It is not too difficult to see that we can modify the value of γ by adding inconsistent data points that can only be fit best in exactly one way. The precise choice of these inconsistent data points heavily depends on the loss function. In conclusion, for different values of γ, the decision problem does not get easier.

Activation Function The ReLU activation function is currently still the most commonly used activation function in practice [12, 69, 74]. Our methods are probably easily adaptable to other piecewise linear activation functions, such as *leaky* ReLU*s*. Having said that, our methods are *not* applicable to other types of activation functions, such as *Sigmoid*, *soft* ReLU or step functions. We want to point out that TrainNN (and even EmpiricalRiskMinimization) is in NP if a step function is used as the activation function [98]. Concerning the Sigmoid and soft ReLU function, it is not even clear whether the EmpiricalRiskMinimization is decidable, as trigonometric functions and exponential functions are not computable on the real RAM [64, 129].

Other Architectures We consider fully connected two-layer networks as the most important case, but we are also interested in $\exists\mathbb{R}$-hardness results for other network architectures. Specifically, fully connected three-layer neural networks and convolutional neural networks are interesting. While it is hard to imagine that more complicated architectures are easier to train, a formal proof of this intuition would strengthen our result and show that $\exists\mathbb{R}$-completeness is a robust phenomenon, in other words, independent of a choice of a specific network type.

Training in Practice Fitting the training data exactly is not required in practice. Quite the contrary, allowing small errors significantly simplifies the training because it permits the use of (meta-)heuristics. One meta-heuristic that often yields “good” solutions for $\exists\mathbb{R}$-complete problems is *gradient descent*. Actually, training neural networks might be the most prominent example in this context: A variety of different gradient descent variants powered by *backpropagation* are nowadays capable of training neural networks containing millions of neurons. In general, we do not get any approximation or runtime guarantees when using gradient descent, but under the right additional assumptions such guarantees are sometimes possible [33].

From a complexity perspective, rounding the weights and biases Θ to the first “few” digits after the comma might allow placing the problem of *approximate* neural network training in NP. Yet, we are not aware of such a proof and we consider it an interesting open question to establish this fact thoroughly. Related to this, Bienstock, Muñoz and Pokutta [21] use the above idea to discretize the weights and biases to show that arbitrary architectures can be trained to approximate optimality via linear programs with size linear in the size of the data set, but exponential in the architecture size. While being an important insight, let us emphasize that this does not imply NP-membership of an approximate version of neural network training.

Connection to Learning Theory We purely focus on the computational complexity of EmpiricalRiskMinimization, that is, minimizing the *training error*. In the practice, one usually desires to achieve low a *generalization error*, i.e., the error on unseen test samples.

To formalize the concept of the generalization error, one needs to combine the computational aspect with a statistical one. There are various models to do so in the literature, the most famous one being *probably approximately correct (PAC) learnability* [142, 149]. While EmpiricalRiskMinimization and learning are two different questions, they are strongly intertwined; see Section 5.5 for related work in this context. Despite the close connections, to the best of our knowledge, the $\exists\mathbb{R}$-hardness of EmpiricalRiskMinimization has no direct implications on the complexity of learning. Still, since EmpiricalRiskMinimization is the most common learning paradigm in practice, our work is arguably also interesting in the context of learning theory.

Lipschitz Continuity The set of data points created in the proof of Theorem 5.3 is intuitively very tame. Formally, this is captured by proving that for yes-instances there exists a Θ such $f(\cdot, \Theta)$ is Lipschitz continuous for a small Lipschitz constant L. We prove in Remark 5.21 that this holds for the instances constructed by our reduction. Lipschitz continuity is also related to *overfitting* and *regularization* [76], the purpose of the latter being to prefer simpler functions over more complicated ones. Being Lipschitz continuous with a small Lipschitz constant essentially means that the function is relatively flat. It is particularly remarkable that we can show hardness even for small Lipschitz constants, since Lipschitz continuity has been a crucial assumption in several recent results about training *and* learning ReLU networks, for example in [21, 42, 70].

5.5 Related Work

Complexity of Neural Network Training It is well-known that minimizing the training error of a neural network is a computationally difficult problem for a large variety of activation functions and architectures [142].

Closest to our work is the recent $\exists\mathbb{R}$-completeness result by Abrahamsen, Kleist and Miltzow for two-layer neural networks [4]. In contrast to our work, they use the identity activation function and rely on particularly difficult to train architectures, both qualities being uncommon in practice. Zhang [158] sketched a similar result already in 1992: Training neural networks with *real-valued* data points is $\mathrm{NP}_{\mathbb{R}}$-complete, again with the identity activation function and with an adversarial architecture. $\mathrm{NP}_{\mathbb{R}}$ is a complexity class in the BSS-model of computation [26], but a suitable discretization (considering the so-called *constant-free Boolean part*) yields ∃ℝ-completeness in today's language.

For ReLU networks, NP-hardness, parameterized hardness and inapproximability results have been established even for the simplest possible architecture consisting of only a single ReLU neuron [29, 52, 67, 72]. While all these results require non-constant input-dimension, Froese and Hertrich show that it is also NP-hard to train a two-layer ReLU network with two input neurons and one output neuron [66]. On the positive side, the seminal algorithm by Arora, Basu, Mianjy and Mukherjee [12] solves EmpiricalRiskMinimization for two-layer ReLU networks with one output neuron to global optimality, placing the problem in NP. It was later extended to a more general class of loss functions by Froese, Hertrich and Niedermeier [67]. The running time is exponential in the number of neurons in the hidden layer and in the input dimension, but polynomial in the number of data points if the former two parameters are considered to be constant. This NP-membership of EmpiricalRiskMinimization with one-dimensional output is in sharp contrast to our ∃ℝ-completeness result for TrainNN with two-dimensional outputs.

While minimizing training and generalization errors are different problems, the hardness of the former also imposes challenges on the latter. Strategies to circumvent hardness from the perspective of learning theory include allowing improper learners, restricting the type of weights allowed in a neural network, or imposing assumptions on the underlying distribution. For example, Chen, Klivans and Meka [42] show fixed-parameter tractability of learning a ReLU network under several assumptions, including Gaussian data and Lipschitz continuity of the network. We refer to [15, 41, 53, 70, 73, 71] as a non-exhaustive list of other results about (non-)learnability of ReLU networks in different settings.

Expressivity of ReLU Networks It is essential for our reduction to understand the classes of functions representable by certain ReLU network architectures. So-called *universal approximation theorems* state that a single hidden layer (with arbitrary width) is already sufficient to approximate every continuous function on a bounded domain with arbitrary precision [46, 90]. However, deeper networks require much fewer neurons to reach the same expressive power, yielding a potential theoretical explanation of the dominance of deep networks in practice [12, 58, 80, 82, 106, 120, 125, 132, 147, 154]. Other related work includes counting and bounding the number of linear regions [81, 116, 117, 122, 125, 141], classifying the set of functions *exactly* representable by different architectures [12, 50, 79, 85, 86, 118, 157], or analyzing the memorization capacity of ReLU networks [150, 155, 156]. Huchette, Muñoz, Serra and Tsay [91] provide a survey on the interactions of neural networks and polyhedral geometry, including implications on training, verification, and expressivity.

5.6 Proof Idea

To prove ∃ℝ-hardness, we reduce from ETR-Inv, a restricted version of ETR introduced in Section 2.3. To recall, an instance of ETR-Inv consists of real variables and a conjunction of constraints in these variables. Each constraint is either an addition constraint $X + Y = Z$, or an inversion constraint $X \cdot Y = 1$. Given such an instance, we construct a TrainNN instance that models the variables and whose training corresponds to satisfying all addition and inversion constraints.

Variables A natural candidate for encoding variables are the weights and biases of the neural network. However, those did not prove to be suitable for our purposes. The main problem with using the parameters of the neural network as variables is that the same function can be computed by many neural networks with different combinations of these parameters. We are not aware of an easy way to normalize the parameters.

To circumvent this issue, we work with the functions representable by fully connected two-layer neural networks directly. We frequently make use of the geometry of their graphs. For now, it is only important to understand that each hidden ReLU neuron encodes a continuous piecewise linear function with exactly two pieces (both of constant gradient). These two pieces are separated by a so-called *breakline*. Now, if we have m hidden neurons, their individually encoded functions add up such that the function computed by the whole neural network is a continuous piecewise linear function with at most m breaklines. Between these breaklines all pieces of the function have constant gradient.

To keep things simple for now, let us first consider a neural network with only one input and one output neuron. We place a series of data points $(x_i; y_i) \in \mathbb{R}^2$ as seen in Figure 5.2. All continuous piecewise linear functions $f(\cdot, \Theta)$ computed by a neural network with only four hidden neurons (i.e., only four breaklines and therefore at most five pieces) that fit these data points exactly must be very similar. In fact, they can only differ in one degree of freedom, namely the slope of the piece going through the data point p. In our construction, this slope represents the value of a variable. The whole set of data points enforcing this configuration is called a *variable gadget*.

Linear Dependencies The key insight for encoding constraints between variables is that we can relate the values of several variable gadgets by a data point: By placing a data point p at a location where several variable gadgets overlap, each of the variable gadgets contributes its part towards fitting p. The exact contribution of each variable gadget depends on its slope. Consequently, if one variable gadget contributes more, the others have to lower their contribution by the same amount. This enforces linear dependencies between different variable gadgets and can be used to design *addition* and *copy gadgets*.

We need a second input dimension in order to intersect multiple variable gadgets. This extends each variable gadget into a *stripe* in $\mathbb{R}^2$, with Figure 5.2 showing only

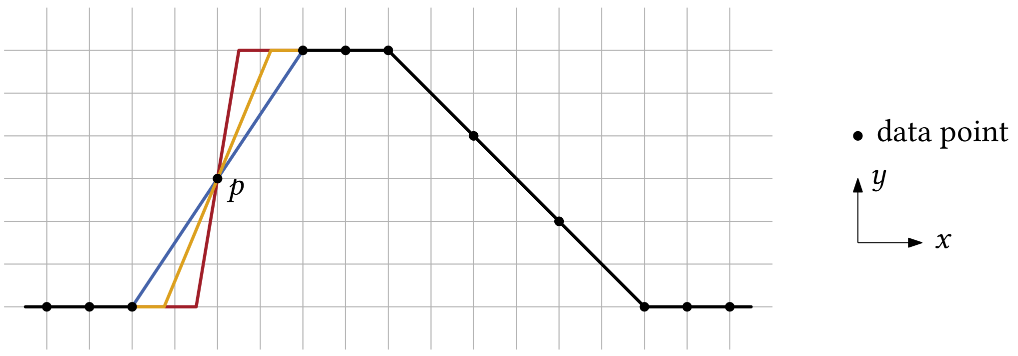


Figure 5.2: The value of $f(\cdot,\Theta)$ is fixed (black part), except for the segment through data point p. The red, orange and blue segments are just three out of uncountably many possibilities. Its slope can be used to encode a real-valued variable.

an orthogonal cross-section of this stripe. See Figure 5.3 for two intersecting variable gadgets. Much of the technical difficulties lie in the subtleties to enforce the presence of multiple (possibly intersecting) gadgets using a finite number of data points.

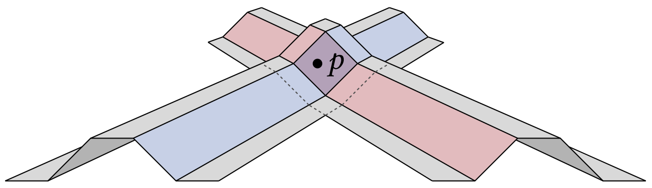


Figure 5.3: Two intersecting variable gadgets. The slopes of the blue and the red region encode the values. Point p lies in the intersection of both and can encode a linear relationship between them.

An addition gadget encodes a linear relation between three variables. In $\mathbb{R}^2$, three lines (or stripes) usually do not intersect in a single point. Thus, we have to carefully place the gadgets to guarantee such a single intersection point. To this end, we create copies of the involved variable gadgets (copying is a linear relation between just two variables, thus "easy"). These copies can then be positioned (almost) freely.

Inversion We are not able to encode nonlinear constraints within only a single output dimension [12]. By adding a second output dimension, the neural network now represents two functions $f^1(\cdot,\Theta)$ and $f^2(\cdot,\Theta)$. Consequently, we are allowed to use data points with two different output labels, one for each output dimension.

One important observation is that the locations of the breaklines of $f = f(\cdot,\Theta) = (f_1(\cdot,\Theta), f_2(\cdot,\Theta))$ are independent of the weights of the edges in the second layer of the neural network. Thus, both functions f^1 and f^2 have the same breaklines. Still, setting some weights to zero may *erase* a breakline in one of the functions.

We define an *inversion gadget* (realizing the constraint $X \cdot Y = 1$), which also corresponds to a stripe in $\mathbb{R}^2$. For simplicity, we only show a cross-section here, see Figure 5.4. In each output dimension individually, the inversion gadget looks exactly like a variable gadget. The inversion gadget can therefore be understood as a variable gadget that carries two values.

We prove that by allowing only five breaklines in total, a function f can only fit all data points exactly if f^1 and f^2 share three of their four breaklines (while both having one "exclusive" breakline each, which is erased in the other dimension). This enforces a nonlinear dependency between the slopes of f^1 and f^2. By choosing the right parameters, this nonlinear relation models exactly an inversion constraint.

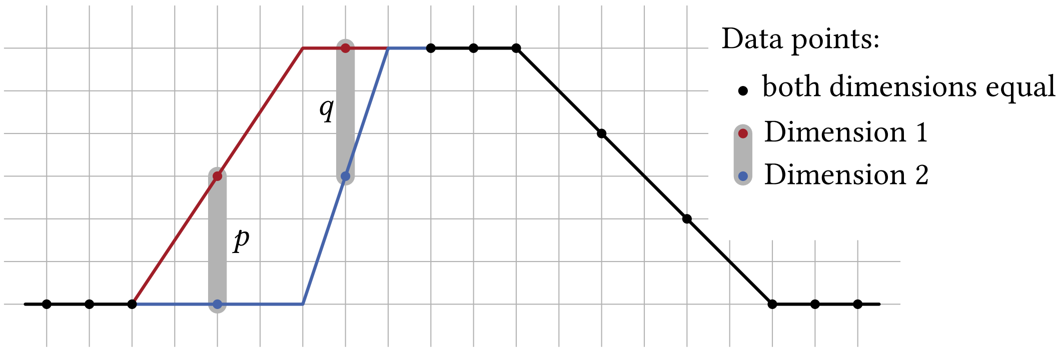


Figure 5.4: Data points p and q have different labels in the two output dimensions, enforcing that the slopes of the red and the blue pieces are related via a nonlinear dependency.

Reduction Let us illustrate the reduction by giving a simple example. Note that this is not yet the complete picture. We start with an ETR-INV instance, for example, deciding whether the following sentence

$$\exists X_1, X_2, X_3, X_4 \in \mathbb{R}\colon\ (X_1 + X_2 = X_3) \land (X_1 + X_3 = X_4) \land (X_1 \cdot X_4 = 1) \land (X_4 \cdot X_3 = 1)$$

is true. This instance has four variables X_1, X_2, X_3, X_4 and four constraints: two additions and two inversions. Recall that every gadget corresponds to a stripe in the input space $\mathbb{R}^2$. See Figure 5.5 for the following construction (the stripes are drawn as lines for better readability).

- We add a variable gadget for each of the variables. All of these are placed such that their corresponding stripes are parallel and do not overlap, see the horizontal lines in Figure 5.5.
- We introduce three more variable gadgets for each addition constraint, one per involved variable. These are placed such that they have a common intersection point while also intersecting their corresponding variable gadget. In Figure 5.5, see the two bundles to the left. A data point at the triple intersection enforces the addition constraint, while data points labelled $\bullet_=$ encode that the values of the two intersecting variable gadgets are equal.

- Lastly, we add an inversion gadget for each inversion constraint and place it such that it intersects the variable gadgets of the two involved variables. See the two dashed lines in Figure 5.5. Data points labelled $\bullet^{=_1}$ ($\bullet^{=_2}$) enforce equality only in the first (second) output dimension.

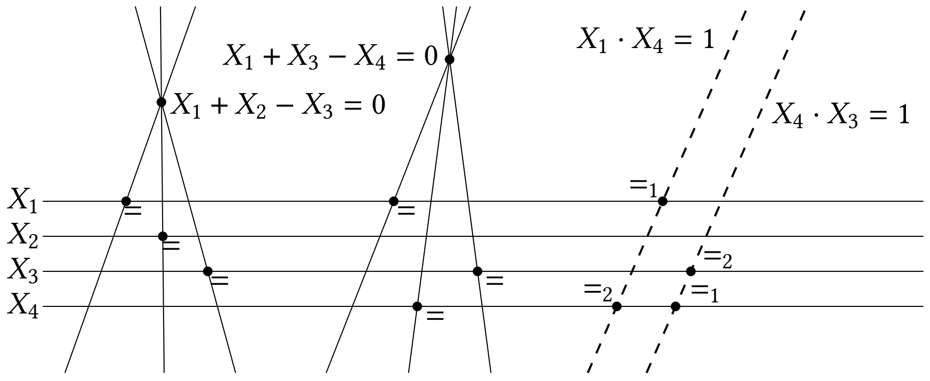


Figure 5.5: Overview of the global arrangement of the gadgets.

To see that the above reduction is correct, assume first that the ETR-Inv instance is true. Then there exist real values for the four variables and these values can be used as the slopes of their corresponding variable gadgets. By the correctness of the individual gadgets (that we prove below), it follows that each data point is fit exactly.

Conversely, if all data points are fit exactly, then the correctness of the gadgets implies that the slopes of the variable gadgets give a solution to the ETR-Inv instance. Intuitively, this holds because the addition and inversion constraints are encoded exactly by the gadgets of our construction.

5.7 $\exists\mathbb{R}$-Membership

$\exists\mathbb{R}$-membership is already proven by Abrahamsen, Kleist and Miltzow:

Proposition 5.5 ([4, Section 2]). TrainNN $\in \exists\mathbb{R}$.

For the sake of completeness, while not being too repetitive, we shortly summarize their argument: $\exists\mathbb{R}$-membership is shown by describing a polynomial-time real verification algorithm (see Section 2.3 and [64]). The input of such an algorithm is a TrainNN instance I, as well as a witness Θ consisting of real-valued weights and biases. Instance I consists of a network architecture, data points D and a target error γ. The algorithm has to verify that the neural network parameterized by Θ fits all data points in D with a total error at most γ.

To achieve this, it loops over all data points in D and evaluates the function realized by the neural network for each of them individually. As each hidden neuron uses the ReLU activation function, each such evaluation takes only linear time in the size of the network. Proposition 5.5 follows, since, by Definition 5.2, the loss function can be computed in polynomial time on a real RAM.

5.8 ∃ℝ-Hardness

This section is devoted to proving ∃ℝ-hardness of TrainNN. Our reduction is mostly geometric, so we start by reviewing the underlying geometry of two-layer neural networks in Section 5.8.1. This is followed by a high-level overview of the reduction in Section 5.8.2 before we describe the gadgets in detail in Section 5.8.3. Finally, in Section 5.8.4, we combine the gadgets into the proof of Theorem 5.3.

5.8.1 Geometry of Two-Layer Neural Networks

Our reduction constructs a neural network that has just two input neurons, two output neurons, and m hidden neurons. Thus, for given weights and biases Θ, it realizes a function $f(\cdot,\Theta)\colon \mathbb{R}^2 \to \mathbb{R}^2$. In this section, we build a geometric understanding of $f(\cdot,\Theta)$, in particular, we study the geometry of the graph of $f(\cdot,\Theta)$. For further results in this direction, we point the interested reader to [12, 50, 85, 118, 157] that investigate the set of functions exactly represented by different architectures of ReLU networks.

The i-th hidden ReLU neuron v_i realizes a function

$$\begin{aligned} f_i\colon \mathbb{R}^2 &\to \mathbb{R} \\ (x_1,x_2) &\mapsto \mathrm{ReLU}(a_{1,i}x_1 + a_{2,i}x_2 + b_i), \end{aligned}$$

where $a_{1,i}$ and $a_{2,i}$ are the edge weights from the first and second input neuron to v_i and b_i is its bias. Note that f_i is a continuous piecewise linear function: If $a_{1,i} = a_{2,i} = 0$, then f_i is constant, $f_i = \mathrm{ReLU}(b_i) = \max\{b_i, 0\}$. Otherwise, the domain $\mathbb{R}^2$ is partitioned into two half-planes, touching along a so-called *breakline* given by the equation $a_{1,i}x_1 + a_{2,i}x_2 + b_i = 0$. The two half-planes are (see Figure 5.6)

- the *inactive region* $\{(x_1,x_2) \subseteq \mathbb{R}^2 \mid a_{1,i}x_1 + a_{2,i}x_2 + b_i \leq 0\}$, in which f_i is constantly 0, and
- the *active region* $\{(x_1,x_2) \subseteq \mathbb{R}^2 \mid a_{1,i}x_1 + a_{2,i}x_2 + b_i > 0\}$, in which f_i is positive and has a constant gradient.

Now let $c_{i,1}$ and $c_{i,2}$ be the weights of the edges connecting v_i with the first and second output neuron, and let $f(\cdot,\Theta) = (f^1(\cdot,\Theta), f^2(\cdot,\Theta))$. For $j \in \{1,2\}$, the function $f^j(\cdot,\Theta) = \sum_{i=1}^{m} c_{i,j}\cdot f_i(\cdot,\Theta)$ is a weighted linear combination of the functions computed at the hidden neurons. We make three observations:

- Each function computed by a hidden ReLU neuron has at most one breakline. Thus, the domain of $f^j(\cdot,\Theta)$ is partitioned into the cells of a line arrangement containing at most m breaklines. Apart from that, $f^j(\cdot,\Theta)$ has a constant gradient inside each cell.
- Let b_i be the breakline produced by a hidden neuron v_i in $f^j(\cdot,\Theta)$. Its position is solely determined by $a_{\cdot,i}$ and b_i. In particular, it is independent of $c_{i,j}$. Thus, the sets of breaklines of $f^1(\cdot,\Theta)$ and $f^2(\cdot,\Theta)$ are both subsets of the same set of (at most m) breaklines determined by the hidden neurons.

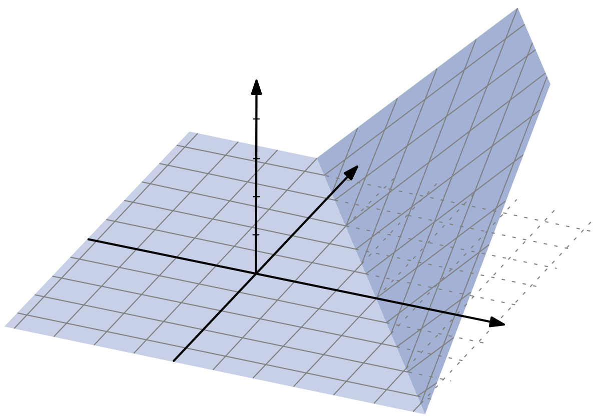


Figure 5.6: A continuous piecewise linear function computed by a hidden ReLU neuron. It has exactly one breakline; the flat part is the inactive region, whereas the sloped part is the active region.

- Even if all $f_i(\cdot, \Theta)$ have a breakline, their sum $f^j(\cdot, \Theta)$ at each output neuron might have fewer breaklines: It is possible for a breakline to be *erased* by setting $c_{i,j} = 0$. Other possibilities are that several breaklines contributed by different hidden neurons cancel each other (producing no breakline) or lie on top of each other (combining multiple breaklines into one). In our reduction we deliberately erase some breaklines in some output dimensions, i.e, we make use of the $c_{i,j} = 0$ trick. However, we avoid the other two cases of breaklines combining/canceling.

Combining above observations yields a stronger statement: For each output neuron and breakline ℓ, the change of gradient of $f^j(\cdot, \Theta)$ along ℓ is constant (see also [50]). Based on this, we distinguish two types of breaklines:

Definition 5.6. *A breakline ℓ is concave (convex) in $f^j(\cdot, \Theta)$ if the restriction of $f^j(\cdot, \Theta)$ to any two cells separated by ℓ is concave (convex).*

The type of a breakline is a tuple $(t_1, t_2) \in \{\wedge, 0, \vee\}^2$ describing whether the breakline is concave $(\wedge)$, erased (0), or convex $(\vee)$ in $f^1(\cdot, \Theta)$ and $f^2(\cdot, \Theta)$, respectively.

By now, we gained a geometric understanding of $f(\cdot, \Theta)$, the continuous piecewise linear function computed by a ReLU neural network with two input and two output neurons. However, not every continuous piecewise linear function can be computed by such a neural network. For the correctness of our reduction, we need a sufficient condition for this:

Lemma 5.7. *A continuous piecewise linear function $f \colon \mathbb{R}^2 \to \mathbb{R}^2$ whose breaklines form a line arrangement $\mathcal{L}$ with m lines can be realized by a fully connected two-layer neural network with m hidden neurons if the following two conditions hold:*

- *In at least one cell of $\mathcal{L}$ the value of f is constantly $(0, 0)$.*
- *For each breakline $\ell \in \mathcal{L}$, the change of the gradient of f along ℓ is constant in both output dimensions.*

Proof. We can use the following construction: Add one hidden neuron per breakline, oriented such that its inactive region is the halfplane containing the $(0, 0)$-cell. The position solely depends on the weights of the first layer and the bias. The weights of the second layer are then chosen to produce the right change of gradient in each output dimension. It is easy to see that the sum of all these neurons computes f. ■

We refer to [50] for a precise characterization of the functions representable by two-layer neural networks with m hidden neurons.

5.8.2 Preparing the Reduction

We establish $\exists\mathbb{R}$-hardness of TrainNN by giving a polynomial-time many-one reduction from ETR-Inv to TrainNN. As mentioned in Section 2.3, ETR-Inv is a variant of ETR that is frequently used as a starting point for $\exists\mathbb{R}$-hardness reductions in the literature, for example in [3, 4, 107, 54].

Recall that ETR-Inv is a special case of ETR in which the quantifier-free part φ of the input sentence $\Phi :\equiv \exists X_1, \ldots, X_n \in \mathbb{R} \colon \varphi(X_1, \ldots, X_n)$ is a conjunction of constraints, each of which is either of the form $X + Y = Z$ or $X \cdot Y = 1$. Additionally, we are promised that Φ either has no solution or one with all values in $\left[\frac{1}{2}, 2\right]$. $\exists\mathbb{R}$-completeness of ETR-Inv is proven in [3]. Furthermore, ETR-Inv exhibits the same algebraic universality that we seek for TrainNN:

Theorem 5.8 ([6]). *Let α be an algebraic number. Then there exists an instance of* ETR-Inv, *which has a solution in $\mathbb{Q}[\alpha]$, but no solution when the variables are restricted to a field $\mathbb{F}$ that does not contain α.*

The reduction starts with an ETR-Inv instance Φ and outputs an integer m and a set of n data points such that there is a fully connected two-layer ReLU neural network N with m hidden neurons exactly fitting all data points ($\gamma = 0$) if and only if Φ is true. Recall that the neural network N defines a continuous piecewise linear function $f(\cdot, \Theta) \colon \mathbb{R}^2 \to \mathbb{R}^2$.

We define several *gadgets* representing the variables as well as the linear and inversion constraints of the ETR-Inv instance Φ. Strictly speaking, a gadget is defined by a set of data points that need to be fit exactly. These data points serve two tasks: Firstly, most of the data points are used to enforce that $f(\cdot, \Theta)$ has m breaklines with predefined orientations and at almost predefined positions. Secondly, the remaining data points enforce relationships between the exact positions of different breaklines.

Globally, our construction yields $f(x, \Theta) = (0, 0)$ for "most" $x \in \mathbb{R}^2$. Each gadget consists of a constant number of parallel breaklines (enforced by data points) that lie in a *stripe* of constant width in $\mathbb{R}^2$. The value of $f(\cdot, \Theta)$ may be non-zero only within these stripes. The "meaning" of a gadget[1] is fully determined by the distances between its parallel breaklines. Thus, each gadget can be translated and rotated arbitrarily without affecting its meaning.

1 For a variable gadget, its "meaning" is the real number represented by it.

Abstractions Describing all gadgets purely by their data points is tedious and obscures the relatively simple geometry enforced by these data points. We therefore introduce two additional constructs, namely *data lines* and *weak data points*, that simplify the presentation. In particular, data lines impose breaklines, which in turn are needed to define gadgets. Weak data points allow us to have features that are only active in one output dimension. How these constructs can be realized with carefully placed data points is deferred to Sections 5.8.3.6 and 5.8.3.7, after we have introduced all other gadgets.

- A *data line* $(\ell; y)$ consists of a line $\ell \subseteq \mathbb{R}^2$ and a label $y \in \mathbb{R}^2$. We say that a data line is fit if $f(\ell, \Theta) = \{y\}$, i.e., the neural network maps every point on it to y.

  As soon as we consider several gadgets, their corresponding stripes in $\mathbb{R}^2$ might intersect. We do not require that the data lines are fit correctly inside these intersections. As we are going to see below, each data line is realized by finitely many data points on it. We make sure that their coordinates do not lie in any of the intersections.

- A *weak data point* relaxes the notion of a regular data point and prescribes only a lower bound on the label. For example, we denote by $(x; y_1, \geq y_2)$ that $f^1(x, \Theta) = y_1$ and $f^2(x, \Theta) \geq y_2$. Weak data points can have such an inequality label in the first, the second or both output dimensions.

5.8.3 Gadgets and Constraints

We describe all gadgets in isolation first, the interaction of two or more gadgets is considered only where it is necessary. In particular, we assume that $f(x, \Theta)$ is constantly zero for $x \in \mathbb{R}^2$ outside the outermost breaklines enforced by each gadget. After all gadgets have been introduced, we describe the global arrangement of the gadgets in Section 5.8.4. Recall that, since each gadget can be freely translated and rotated, we can describe the positions of all its data lines and (weak) data points relative to each other.

Not all gadgets make use of the two output dimensions. Some gadgets have the same labels in both output dimensions for all of their data lines, and thus look the same in both output dimensions. For these gadgets, we simplify the usual notation of $(y_1, y_2) \in \mathbb{R}^2$ to single-valued labels $y \in \mathbb{R}$. In our figures, data points and functions looking the same in both output dimensions are drawn in black, while features only occurring in one dimension are drawn in different colors to distinguish them from each other.

Let $(\ell_1; y_i), \ldots, (\ell_k; y_k)$ be parallel data lines. Further, let $\ell \subseteq \mathbb{R}^2$ be an oriented line intersecting all ℓ_i. Without loss of generality, we assume that ℓ intersects ℓ_i before ℓ_j if and only if $i < j$. A *cross-section* through $(\ell_1; y_i), \ldots, (\ell_k; y_k)$ is defined as follows: For each data line $(\ell_i; y_i)$, the cross-section contains a data point $p_i = (x_i; y_i) \in \mathbb{R} \times \mathbb{R}^2$, where x_i is the oriented distance between the intersections of ℓ_1 and ℓ_i on ℓ. Two data points p_i and p_j in the cross-section are *consecutive* if $|i - j| = 1$.

If ℓ is perpendicular to all ℓ_i, then the cross-section is *orthogonal*. The intersection of a breakline with ℓ is a *breakpoint*.

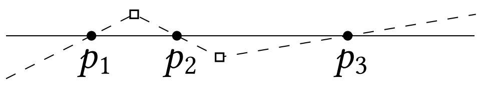
We draw cross-sections by projecting a data point $(x_i; y_i) \in \mathbb{R} \times \mathbb{R}^2$ into a two-dimensional coordinate system, marking x_i along the abscissa and y_i along the ordinate. If a y_i behaves differently in the two output dimensions, then we draw it twice and distinguish the two dimensions by color.

Observation 5.9. *Let f be a continuous piecewise linear function interpolating three consecutive data points p_i, p_{i+1} and p_{i+2} in a cross-section of a gadget. The following holds for each output dimension:*

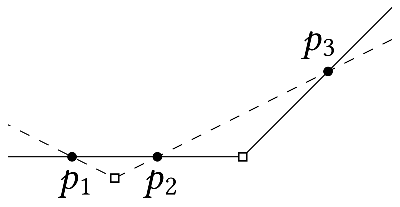
(i) If they are collinear, then f has either no breakpoint strictly between p_i and p_{i+2} or at least two.

(ii) If they are not collinear, then f has a breakpoint b strictly between p_i and p_{i+2}. Furthermore, if p_{i+2} is left (right) of the ray from p_i through p_{i+1}, then b is convex (concave).

Observation 5.9 is the key to prove that data lines enforce breaklines of a certain type, with a prescribed orientation and (almost) fixed position. It is illustrated in Figure 5.7.



(a) If the points are collinear, then there is either no breakpoint or there are at least two.



(b) If the points are not collinear, then we need a breakpoint of a certain type, here convex.

Figure 5.7: Three consecutive points p_1, p_2 and p_3 in a cross-section and possible interpolations through them (solid and dashed).

5.8.3.1 Variable Gadget

A *variable gadget* consists of twelve parallel data lines $\ell_1, \ldots, \ell_{12}$, without loss of generality vertical and numbered from left to right. Additionally, there is a weak data point q between ℓ_3 and ℓ_4. For all of these, the following table lists their relative distance to ℓ_1 and their label (note that both output dimensions have the same label):

	ℓ_1	ℓ_2	ℓ_3	q	ℓ_4	ℓ_5	ℓ_6	ℓ_7	ℓ_8	ℓ_9	ℓ_{10}	ℓ_{11}	ℓ_{12}
distance to ℓ_1	0	1	2	$3+\frac{2}{3}$	4	6	7	8	10	12	14	15	16
label	0	0	0	≥ 2	3	6	6	6	4	2	0	0	0

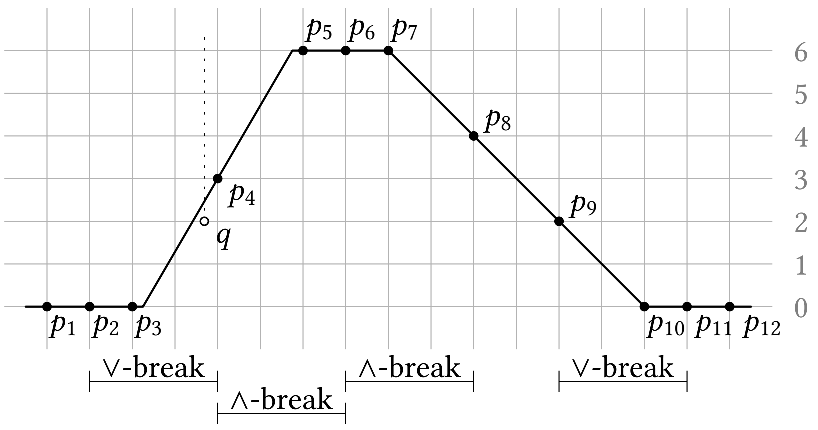


Figure 5.8: Orthogonal cross-section of a variable gadget. The bars below the cross-section indicate non-collinear triples used in the proof of Lemma 5.10. For example, there needs to be a convex breakpoint between p_2 and p_4.

See Figure 5.8 for an orthogonal cross-section through a variable gadget.

Lemma 5.10. *A continuous piecewise linear function f that fits $\ell_1, \ldots, \ell_{12}$ and q exactly must have at least four breaklines. If it has exactly four breaklines, then they must all be parallel to the data lines. In this case, let b_1, b_2, b_3 and b_4 be the breaklines, numbered from left to right. It holds in both output dimensions that:*

- *f is constantly 0 to the left of b_1 and to the right of b_4.*
- *f is constantly 6 between b_2 and b_3.*
- *b_3 lies on ℓ_7 and b_4 lies on ℓ_{10}.*
- *The slope of the variable gadget, i.e., the norm of the gradient between b_1 and b_2, is at least $\frac{3}{2}$ and at most 3.*

Before we prove Lemma 5.10, let us describe the functionality of a variable gadget: The slope s_X of a variable gadget for a variable X is in $\left[\frac{3}{2}, 3\right]$. In order to represent values in $\left[\frac{1}{2}, 2\right]$, we say that a slope s_X encodes the value $X = s_X - 1$.

Proof of Lemma 5.10. We first prove that four breaklines are indeed necessary to fit all data lines exactly. Every orthogonal cross-section contains four non-collinear triples: (p_2, p_3, p_4), (p_4, p_5, p_6), (p_6, p_7, p_8) and (p_9, p_{10}, p_{11}). They pairwise share at most one point, so by Observation 5.9(ii), four breakpoints are indeed required. Since the data lines are parallel to each other, all orthogonal cross-sections look the same, and each breakpoint corresponds to a breakline that is parallel to the data lines. For the rest of the proof, we denote the breaklines by b_1, b_2, b_3 and b_4, numbered from left to right.

In the following, we consider each of the non-collinear triples individually, to further locate the positions of the breaklines. The following observations are all due to Observation 5.9:

- The non-collinear triple (p_2, p_3, p_4) implies that b_1 must be between ℓ_2 and ℓ_4. The collinear triple (p_1, p_2, p_3) enforces that b_1 is to the right of ℓ_3 and that f is constantly 0 to the left of b_1.
- The non-collinear triple (p_4, p_5, p_6) implies that b_2 must be between p_4 and p_6. The collinear triple (p_5, p_6, p_7) enforces that b_2 is to the left of ℓ_5 and that f is constantly 6 to the right of b_2.
- The non-collinear triple (p_6, p_7, p_8) implies that b_3 must be between ℓ_6 and ℓ_8. The collinear triples (p_5, p_6, p_7) and (p_7, p_8, p_9) leave ℓ_7 as the only remaining position for b_3. The collinear triple (p_5, p_6, p_7) implies that f must be constantly 6 to the left of b_3.
- The non-collinear triple (p_9, p_{10}, p_{11}) implies that b_4 must be between ℓ_9 and ℓ_{11}. The collinear triples (p_8, p_9, p_{10}) and (p_{10}, p_{11}, p_{12}) leave ℓ_{10} as the only remaining position for b_4. The collinear triple (p_{10}, p_{11}, p_{12}) implies that f must be constantly 0 to the right of b_4.

As b_1 must be on ℓ_3 or to its right and b_2 must be on p_5 or to its left, the slope is at most 3. Lastly, weak data point q enforces that the slope is at least $\frac{3}{2}$. ■

5.8.3.2 Measuring a Value from a Variable Gadget

Consider a variable gadget for a variable X with slope s_X. We call the two parallel lines with distance 1 to ℓ_4 its *measuring lines*. More precisely, we distinguish between the *lower measuring line* (the one towards ℓ_3) and the *upper measuring line* (the one towards ℓ_5). Since the slope of the variable gadget is in $\left[\frac{3}{2}, 3\right]$, both measuring lines are always inside or at the boundary of the sloped part (in other words, between breaklines b_1 and b_2).

Assume that the variable gadget is fit exactly. Then, at any point p on ℓ_4, the variable gadget contributes 3 to $f(p, \Theta)$. It follows that a point p_u on the upper measuring line contributes $3 + s_X$ to $f(p_u, \Theta)$. Similarly, a point p_l on the lower measuring line contributes $3 - s_X$ to $f(p_l, \Theta)$. See Figure 5.9 for a visualization.

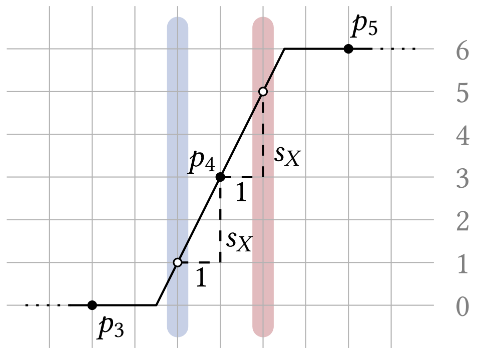


Figure 5.9: Partial cross-section of a variable gadget with slope $s_X = 2$, so $X = 1$. The lower and upper measuring lines are drawn in blue and red. This variable gadget contributes $3 - s_X$ to the lower and $3 + s_X$ to the upper measuring line.

5.8.3.3 Linear Constraints: Addition and Copying

Until this point, we considered individual gadgets separately from each other. As soon as we have two or more gadgets, their corresponding stripes may intersect, leading to interference of the gadgets inside these intersections. We exploit this to encode linear constraints.

Let $\mathcal{A}$ and $\mathcal{B}$ be subsets of the variables. We can enforce a linear constraint of the form $\sum_{A\in\mathcal{A}} A = \sum_{B\in\mathcal{B}} B$ using just one additional data point p. In particular, we care about the following two special cases:

- To copy a value from one variable X to another variable Y, we model $X = Y$ by $\mathcal{A} = \{X\}$ and $\mathcal{B} = \{Y\}$.
- To encode the addition $X + Y = Z$, we set $\mathcal{A} = \{X, Y\}$ and $\mathcal{B} = \{Z\}$.

For all variables in $\mathcal{A}$, the data point p must be on the upper measuring line of their corresponding variable gadget. Similarly, for variables in $\mathcal{B}$, the data point p must be on the lower measuring line. This requires a placement of the gadgets such that all required measuring lines intersect in a single point, where we can place p. This is trivial for $|\mathcal{A}| + |\mathcal{B}| = 2$, as we only require the involved variable gadgets to be non-parallel, see Figure 5.10. For $|\mathcal{A}| + |\mathcal{B}| \geq 3$, this is more involved. We can use the equality constraint $X = Y$ to copy the value of a variable onto additional variable gadgets, which can then be positioned freely to obtain the required intersections, see Figure 5.11. We discuss the global layout to achieve this in more detail in Section 5.8.4 below.

Lemma 5.11. *Let $\mathcal{A}$ and $\mathcal{B}$ be subsets of the variables. A data point p with label $4|\mathcal{A}|+2|\mathcal{B}|$ placed on the upper measuring line of each $A \in \mathcal{A}$ and the lower measuring line of each $B \in \mathcal{B}$ enforces the linear constraint $\sum_{A\in\mathcal{A}} A = \sum_{B\in\mathcal{B}} B$.*

Proof. Let s_X be the slope of the variable gadget for variable X. For each $A \in \mathcal{A}$, the data point p is placed on the upper measuring line of A's variable gadget, so A contributes $3 + s_A$ to $f(p, \Theta)$. Similarly, for each variable $B \in \mathcal{B}$, the data point p is placed on the lower measuring line of B's variable gadget, so B contributes $3 - s_B$ to $f(p, \Theta)$.

The overall contribution of all involved variables adds up to

$$\begin{aligned} f(p,\Theta) &= \sum_{A\in\mathcal{A}} (3 + s_A) + \sum_{B\in\mathcal{B}} (3 - s_B) \\ &= \sum_{A\in\mathcal{A}} (4 + A) + \sum_{B\in\mathcal{B}} (2 - B) \\ &= 4|\mathcal{A}| + 2|\mathcal{B}| + \sum_{A\in\mathcal{A}} A - \sum_{B\in\mathcal{B}} B, \end{aligned}$$

where we used that $X = s_X - 1$ to get from the first to the second line. Setting the label of p to $4|\mathcal{A}| + 2|\mathcal{B}|$, yields that p is fit exactly if and only if $\sum_{A\in\mathcal{A}} A = \sum_{B\in\mathcal{B}} B$. ∎

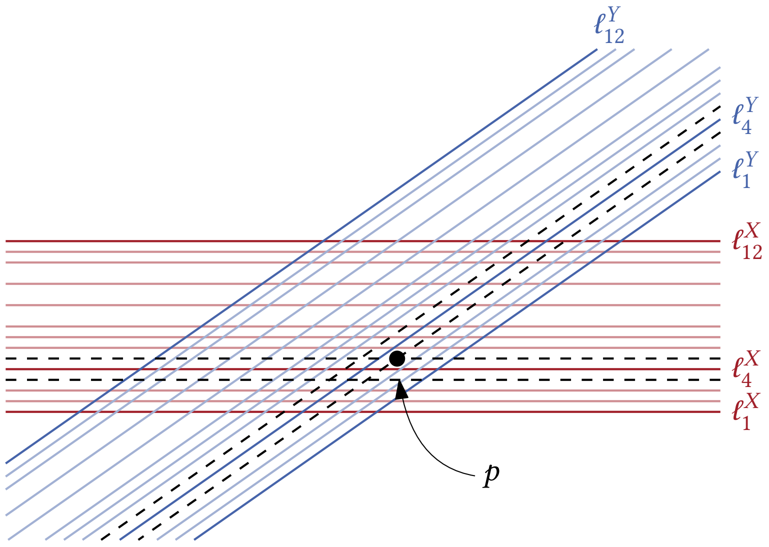


Figure 5.10: Top-down view on the intersection of two variable gadgets corresponding to two variables X (red) and Y (blue). The dashed lines are their measuring lines. The point p is placed at the intersection of the upper measuring line for X and lower measuring line for Y, and receives label 6 to enforce the constraint $X = Y$.

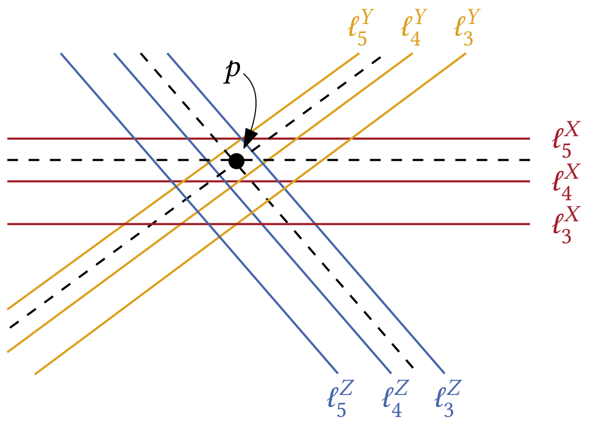


Figure 5.11: Top-down view of the intersection of three variable gadgets corresponding to variables X (red), Y (orange), and Z (blue). The dashed lines are the upper measuring lines for X and Y and the lower measuring line for Z, intersecting in a single point p with label 10. This realizes the constraint $X + Y = Z$.

As already mentioned above, we do not require Lemma 5.11 in its full generality, but only two special cases: The only linear constraint in an ETR-INV instance is the addition $X + Y = Z$. Additionally, we also need the ability to copy values in our reduction, i.e., constraints of the form $X = Y$.

Corollary 5.12. *To encode the addition constraint $X + Y = Z$ of* ETR-INV*, data point p has label* 10*. For the copy constraint $X = Y$, the data point has label* 6*.*

Until now, we never distinguished between the two output dimensions of a variable gadget. Let us note, that the data point p above may also be a weak data point, as long as its label is fixed in one output dimension. This holds, because the breaklines of a variable gadget are the same in both output dimensions, therefore also representing the same value in both dimensions.

5.8.3.4 Inversion Gadget

In essence, an *inversion gadget* is the superposition of two variable gadgets. By using data lines with different labels in the two output dimensions, it can represent two real variables at once. However, their values have a non-linear dependency.

Formally, the inversion gadget consists of 13 data lines $\ell_1, \dots, \ell_{13}$, without loss of generality vertical and numbered from left to right. The following table lists their relative distance to ℓ_1 and their labels:

	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5	ℓ_6	ℓ_7	ℓ_8	ℓ_9	ℓ_{10}	ℓ_{11}	ℓ_{12}	ℓ_{13}
distance to ℓ_1	0	1	2	4	7	9	10	11	13	15	17	18	19
label in dim. 1	0	0	0	3	6	6	6	6	4	2	0	0	0
label in dim. 2	0	0	0	0	3	6	6	6	4	2	0	0	0

See Figure 5.12 for an orthogonal cross-section through an inversion gadget.

Lemma 5.13. *A continuous piecewise linear function that fits $\ell_1, \dots, \ell_{13}$ exactly must have at least five breaklines. If it has exactly five breaklines, then they must all be parallel to the data lines. In this case, let b_1, b_2, b_3, b_4 and b_5 be the breaklines, numbered from left to right. It holds that:*

- *f is constantly $(0, 0)$ to the left of b_1 and to the right of b_5.*
- *In output dimension* 1*, f is constantly* 6 *between b_2 and b_4.*
- *In output dimension* 2*, f is constantly* 0 *to the left of b_2 and constantly* 6 *between b_3 and b_4.*
- *b_4 is on ℓ_8 and b_5 is on ℓ_{11}.*
- *The inversion gadget has two slopes s_X and s_Y.*
  - *In dimension* 1*, slope s_X is the norm of the gradient between b_1 and b_2.*
  - *In dimension* 2*, slope s_Y is the norm of the gradient between b_2 and b_3.*

  It holds that $s_X s_Y = s_X + s_Y$.

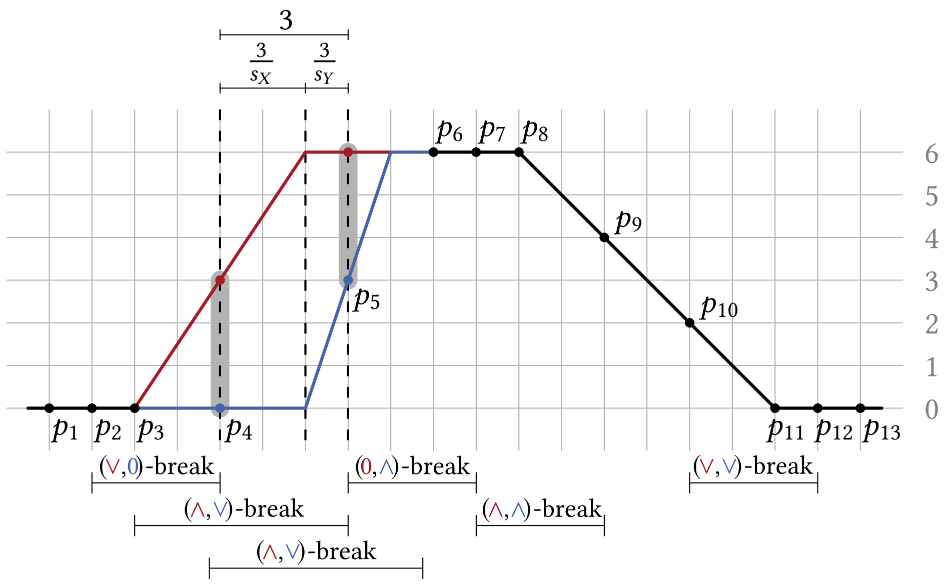


Figure 5.12: Cross-section of the inversion gadget. Data points p_4 and p_5 have different labels in the first (red) and second (blue) output dimension.

Before we prove Lemma 5.13, let us describe the functionality of an inversion gadget: As mentioned above, an inversion gadget is the superposition of two variable gadgets. Only four of the five breaklines are "visible" in each output dimension, the fifth being erased. It has a slope in each dimension: s_X in the first, s_Y in the second. Therefore, it encodes two values $X = s_X - 1$ and $Y = s_Y - 1$. It holds that

$$XY = (s_X - 1)(s_Y - 1) = s_X s_Y - s_X - s_Y + 1 \overset{(*)}{=} s_X + s_Y - s_X - s_Y + 1 = 1,$$

where the equality labeled $(*)$ follows from the $s_X s_Y = s_X + s_Y$ condition provided by the lemma. We conclude, that the non-linear relation between the two slopes exactly models the inversion constraint $XY = 1$ of an ETR-Inv instance.

Proof of Lemma 5.13. As in the proof of Lemma 5.10, we use non-collinear triples to argue that each orthogonal cross-section requires exactly five breakpoints. Again, because all data lines are parallel to each other, all orthogonal cross-sections look the same, and all breakpoints correspond to a breakline that is parallel to the data lines.

All the following observations rely on Observation 5.9:

- The non-collinear triple (p_2, p_3, p_4) enforces a breakline of type $(\vee, 0)$ strictly between ℓ_2 and ℓ_4. We call this breakline b_1. As (p_1, p_2, p_3) is collinear in both dimensions, b_1 must be on or to the right of ℓ_3 and f must be constantly $(0, 0)$ to the left of b_1.
- The non-collinear triple (p_5, p_6, p_7) enforces a breakline of type $(0, \wedge)$ strictly between ℓ_5 and ℓ_7. We call this breakline b_3. As (p_6, p_7, p_8) is collinear in both dimensions, b_3 must be on or to the left of ℓ_6 and f must be constantly $(6, 6)$ to the right of b_3.

- The non-collinear triple (p_7, p_8, p_9) enforces a breakline of type $(\wedge, \wedge)$ strictly between ℓ_7 and ℓ_9. We call this breakline b_4. As (p_6, p_7, p_8) and (p_8, p_9, p_{10}) are collinear in both dimensions, b_4 must lie on ℓ_8.
- The non-collinear triple (p_{10}, p_{11}, p_{12}) enforces a breakline of type $(\vee, \vee)$ strictly between ℓ_{10} and ℓ_{12}. We call this breakline b_5. The triples (p_9, p_{10}, p_{11}) and (p_{11}, p_{12}, p_{13}) are collinear in both dimensions, so b_5 must lie on ℓ_{11}.

The triples enforcing the breaklines b_1, b_3, b_4 and b_5 are pairwise disjoint, so all of these breaklines are indeed necessary.

An orthogonal cross-section of an inversion gadget contains two more non-collinear triples: (p_3, p_4, p_5) and (p_4, p_5, p_6). Both enforce a breakline of type $(\wedge, \vee)$. Note, that all other non-collinear triples intersecting them have a different type. Therefore, none of the previous four breaklines is compatible, and at least one more breakline is indeed necessary. Assuming that $\ell_1, \ldots, \ell_{13}$ must be fit with just five breaklines, we call this breakline b_2, and conclude that it must be on or between ℓ_4 and ℓ_5. Furthermore, f must be constantly 0 in dimension 2 to the left of b_2.

It remains to prove that $s_X s_Y = s_X + s_Y$. To this end, we derive the exact position of b_2 between ℓ_4 and ℓ_5. In dimension 1, it holds that $f^1(\ell_4, \Theta) = 3$ and $f^1(b_2, \Theta) = 6$. The distance between ℓ_4 and b_2 is $\frac{6-3}{s_X}$. In dimension 2, it holds that $f^2(\ell_5, \Theta) = 3$ and $f^2(b_2, \Theta) = 0$. The distance between ℓ_5 and b_2 is $\frac{3-0}{s_Y}$. We conclude that $3 = \frac{3}{s_X} + \frac{3}{s_Y}$, which is equivalent to $s_X + s_Y = s_X s_Y$ for all $s_X, s_Y \neq 0$. ■

5.8.3.5 Applying the Inversion Gadget

An inversion gadget has two pairs of measuring lines, one in each dimension. The lower and upper measuring lines in dimension 1 have distance 1 to ℓ_4. Similar, in dimension 2, they have distance 1 to ℓ_5.

To encode an $XY = 1$ constraint of an ETR-INV instance, we first identify two normal variable gadgets carrying the variables X and Y. Then the inversion gadget is placed such that its measuring lines intersect the measuring lines of the variable gadgets. We copy X to the first dimension of the inversion gadget at the intersection with the variable gadget for X. Similarly, we copy Y to the second dimension of the inversion gadget at the intersection with the variable gadget for Y. This copying can be done as described in Section 5.8.3.3 using weak data points. See Figure 5.13 for a top-down view. Technically, the inversion gadget enforces the inversion constraint only in one dimension of each involved variable gadget. However, this is sufficient because variable gadgets always carry the same value in both output dimensions.

5.8.3.6 Realizing Weak Data Points: Lower Bound Gadgets

So far, we used weak data points, i.e., data points whose label is just a lower bound on $f(\cdot, \Theta)$. Weak data points are just a concept meant to simplify the description of the gadgets; a TRAINNN instance cannot contain weak data points. For this reason, we introduce a *lower bound gadget* that simulates a weak data point using only ordinary data points, i.e., data points with constant labels.

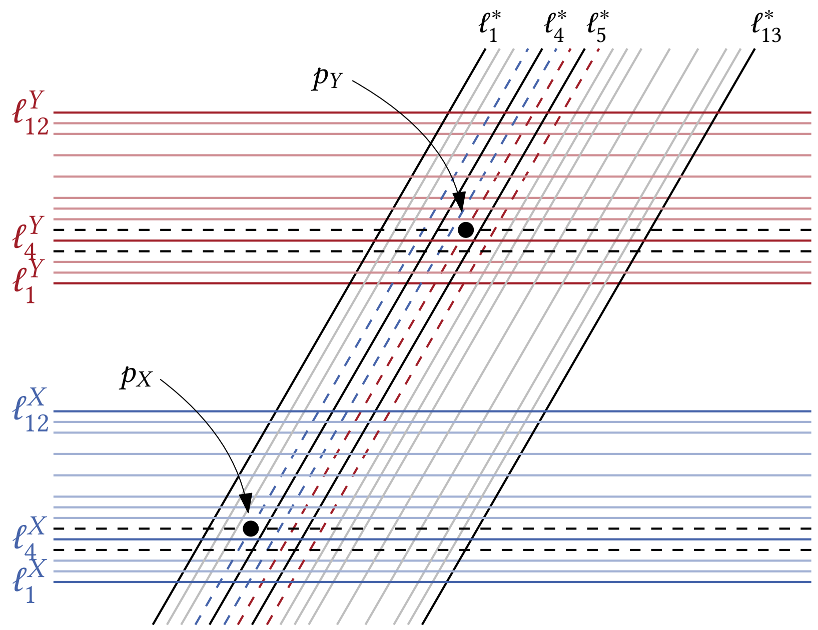
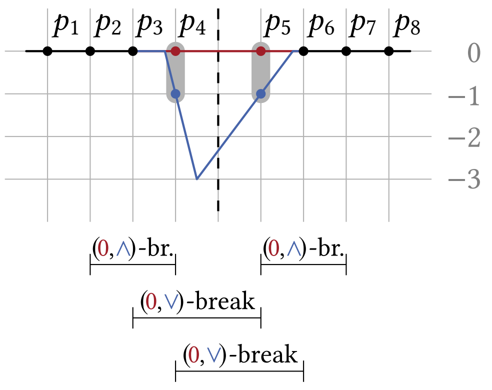
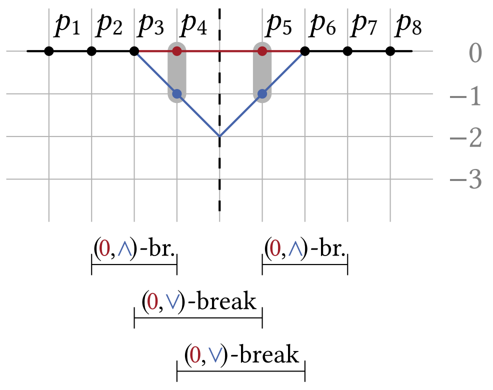


Figure 5.13: Top-down view on two variable gadgets (horizontal) for variables X (blue) and Y (red) (the data lines are solid, the measuring lines are dashed). The sloped gadget is an inversion gadget. Two weak data points p_X and p_Y copy X and Y to the first and second dimension of the inversion gadget, respectively.



(a) The lower bound gadget can be asymmetric.



(b) A lower bound gadget has a maximum contribution to the weak data point of −2.

Figure 5.14: Cross-sections of a lower bound gadget which is inactive in the first (red) dimension and active in the second (blue) dimension. It is used to simulate a weak data point in the active dimension (blue) that lies on the dashed vertical line. It does not contribute to $f(\cdot, \Theta)$ in the inactive dimension (red), i.e., all breaklines are erased (type 0).

Recall that a weak data point can have a lower bound label in either one or in both dimensions. For this reason, a lower bound gadget can be either *active* or *inactive* in each dimension. If the lower bound gadget is active in some dimension, its breaklines form a $\vee$-shape of (almost) arbitrary depth in that dimension. On the other hand, if the lower bound gadget is inactive in some output dimension, then it is constantly 0 in this dimension.

Formally, a lower bound gadget consists of eight data lines $\ell_1, \dots, \ell_8$, without loss of generality vertical and numbered from left to right. The following table lists their relative distance to ℓ_1 and their labels:

	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5	ℓ_6	ℓ_7	ℓ_8
distance to ℓ_1	0	1	2	3	5	6	7	8
label in active dimension(s)	0	0	0	-1	-1	0	0	0
label in inactive dimension(s)	0	0	0	0	0	0	0	0

See Figure 5.14 for orthogonal cross-sections through a lower bound gadget. We always assume that a lower bound gadget has at least one active dimension, as it is otherwise unnecessary.

Lemma 5.14. *A continuous piecewise linear function f that fits $\ell_1, \dots, \ell_8$ exactly must have at least three breaklines. If it has exactly three breaklines, then they must all be parallel to the data lines. In this case, let b_1, b_2 and b_3 be the breaklines. In an inactive dimension f is constantly 0, in an active dimension it holds that:*

- *f is constantly 0 to the left of b_1 and to the right of b_3.*
- *$f(p, \Theta) \leq -2$ for all points p with equal distance to ℓ_4 and ℓ_5.*

Proof. As in the proof of Lemma 5.10, we use non-collinear triples to argue that at least three breaklines are necessary for a lower bound gadget with at least one active dimension. Again, because all data lines are parallel to each other, all orthogonal cross-sections look the same, and all breakpoints correspond to a breakline that is parallel to the data lines.

The following observations rely on Observation 5.9. In an inactive dimension, all triples are collinear, so f must be constantly 0 everywhere. From now on, we focus purely on the active dimension.

- The non-collinear triple (p_2, p_3, p_4) enforces a $\wedge$-type breakline strictly between ℓ_2 and ℓ_4. We call this breakline b_1. By the collinear triple (p_1, p_2, p_3), b_1 must be on or to the right of ℓ_3. Furthermore, f must be constantly 0 to the left of b_1.
- The non-collinear triple (p_5, p_6, p_7) enforces a $\wedge$-type breakline strictly between ℓ_5 and ℓ_7. We call this breakline b_3. By the collinear triple (p_6, p_7, p_8), b_3 must be on or to the left of ℓ_6. Furthermore, f must be constantly 0 to the right of b_3.

Breaklines b_1 and b_3 are both necessary, as the non-collinear triples enforcing them are disjoint. The lower bound gadget contains two more non-collinear triples: (p_3, p_4, p_5) and (p_4, p_5, p_6), both of type $\vee$. Assuming that all data lines are fit with just three breaklines, a third breakline called b_2 must lie on or between p_4 and p_5. The value of $f(b_2, \Theta)$ can be arbitrarily small. However, in the extreme case, it holds that $b_1 = \ell_4$ and $b_3 = \ell_5$, in which case $f(b_2, \Theta) = -2$. No larger values are possible. ■

For a lower bound gadget with two active dimensions, we can make the following observation. It holds because the breaklines must be at the exact same positions in both output dimensions.

Observation 5.15. *A lower bound gadget that is active in both dimensions contributes the same amount in both dimensions.*

We need one lower bound gadget per weak data point. It gets placed such that the weak data point is equidistant to ℓ_4 and ℓ_5. The weak data point with label $\geq y$ is converted into an ordinary data point with label $y - 2$.

By Lemma 5.14, the lower bound gadget can contribute any value $c \in (-\infty, -2]$ to the new data point. Thus, the data point can be fit perfectly if and only if the other gadgets contribute at least a value of y to the data point, that is, the intended lower bound constraint is met.

5.8.3.7 Realizing Data Lines using Data Points

We previously assumed that our gadgets are defined by data *lines*, while in reality, we are only allowed to use data *points*. In this section, we argue that a set of data lines can be simulated by replacing each data line by three data points. This allows us to define the gadgets described throughout previous sections solely using data points.

This section is devoted to showing the following lemma, which captures this transformation formally. Note that our replacement of data lines by data points does not work in full generality, but we show it for all the gadgets that we constructed.

Lemma 5.16. *Assume we are given a set of variable, inversion and lower bound gadgets that in total require at least m breaklines (four, five, and three per variable, inversion and lower bound gadget, respectively). Furthermore, assume that the gadgets are placed in $\mathbb{R}^2$ such that no two parallel gadgets overlap. Then each data line can be replaced by three data points, such that a continuous piecewise linear function with at most m breaklines fits the data points if and only if it fits the data lines.*

For the proof, consider the line arrangement induced by the data lines. We introduce three vertical lines v_1, v_2, v_3 to the right of all intersections. The vertical lines are placed at unit distance to one another. In our construction in Section 5.8.4, we make sure that no data line is vertical. Thus, each data line intersects each of the vertical lines exactly once. We place one data point on each intersection between a vertical line and a data line. The new data point inherits the label of the underlying data line. Furthermore, on each vertical line, we ensure that the minimum distance α between any two data points belonging to different gadgets is larger than the maximum distance w between data points belonging to the same gadget. This can be achieved by placing the v_1, v_2 and v_3 far enough to the right and by ensuring a minimum distance between parallel gadgets. See Figure 5.15 for an illustration.

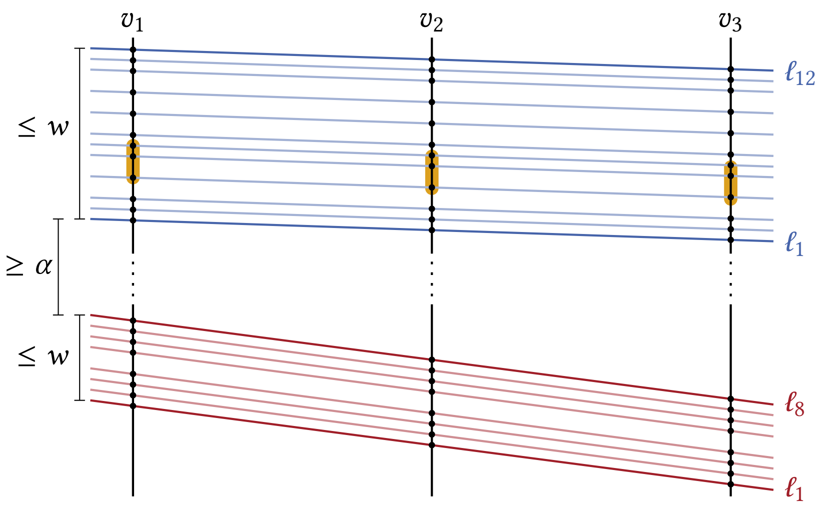


Figure 5.15: Data lines defining a variable gadget (blue) and a lower bound gadget (red), and their intersections with the vertical lines v_1, v_2, v_3. We add a data point at each intersection. The values α and w describe the minimal distance between data lines of different gadgets, and the maximal distance between data lines of the same gadget, respectively. In orange, we highlighted three matching breakpoint intervals (forcing a $\wedge$-breakpoint between ℓ_4 and ℓ_6 of the variable gadget).

Along each of the three vertical lines, the data points form cross-sections of all the gadgets, similar to the cross-sections shown in Figures 5.8, 5.12 and 5.14 (but here, the cross-sections are not orthogonal). We have previously analyzed cross-sections of individual gadgets in the proofs of Lemmas 5.10, 5.13 and 5.14. There, we identified certain intervals between some of the data lines that need to contain a *breakpoint* (the intersection of a breakline and the cross section). We refer to these intervals as *breakpoint intervals* along the vertical lines. Note that a breakpoint interval may degenerate to just one point. By our placement of the vertical lines, the cross-sections (and thus also the breakpoint intervals) of different gadgets do not overlap.

Any two data lines bounding a breakpoint interval on v_1 also bound a breakpoint interval on v_2 and v_3. We call the three breakpoint intervals on v_1, v_2 and v_3 which are bounded by the same data lines *matching* breakpoint intervals.

In total, there are $3m$ breakpoint intervals. We show that the only way to *stab* each of them exactly once using m breaklines is if each breakline stabs exactly three matching breakpoint intervals. The first observation towards this is that each breakline can only stab a single breakline interval per vertical line because all breakline intervals are pairwise disjoint. Thus, having m breakpoint intervals on each vertical line, each of the m breaklines has to stab exactly three intervals, one per vertical line. In a first step, we show that each breakline has to stab three breakpoint intervals belonging to the same gadget.

Claim 5.17. *Each breakline has to stab three breakpoint intervals of the same gadget.*

Proof of Claim. The proof is by induction on the number of gadgets. For a single gadget, the claim holds trivially. For the inductive step, we consider the lowest gadget g (on v_1, v_2 and v_3) and assume for the sake of contradiction that there is a breakline b stabbing a breakpoint interval of g on v_2 and a breakpoint interval of a different gadget g' above g on v_1. By construction, the minimum distance α between different gadgets is larger than the maximum width w of any gadget on all three vertical lines. Thus, the distance of any breakpoint interval of g' to any breakpoint interval of g on v_1 is larger than the width of g on v_3. Therefore, we know that the breakline b intersects v_3 below any breakpoint intervals of g, which is the lowest gadget on v_3. Thus, it stabs at most two breakpoint intervals in total, and therefore not all intervals can be stabbed. The same reasoning holds if the roles of v_1 and v_3 are flipped. All breaklines stabbing breakpoint intervals of g on v_2 must therefore also stab breakpoint intervals of g on v_1 and v_3. Applying the induction hypothesis on the remaining gadgets, it follows that each breakline only stabs breakpoint intervals of the same gadget. ◀

We can therefore analyze the situation for each gadget in isolation. The main underlying idea is to use the type of the required breakline. Each breakline must stab three breakpoint intervals of the same type. Let us summarize the findings about required breakline locations and types from the proofs of Lemmas 5.10, 5.13 and 5.14 in Table 5.1.

Table 5.1: Location and type of the breaklines in variable gadgets, inversion gadgets, and lower bound gadgets.

	Location	Type
b_1	$[\ell_3, \ell_4)$	$(\vee, \vee)$
b_2	$(\ell_4, \ell_5]$	$(\wedge, \wedge)$
b_3	on ℓ_7	$(\wedge, \wedge)$
b_4	on ℓ_{10}	$(\vee, \vee)$

(a) Variable gadget.

	Location	Type
b_1	$[\ell_3, \ell_4)$	$(\vee, 0)$
b_2	(ℓ_4, ℓ_5)	$(\wedge, \vee)$
b_3	$(\ell_5, \ell_6]$	$(0, \wedge)$
b_4	on ℓ_8	$(\wedge, \wedge)$
b_5	on ℓ_{11}	$(\vee, \vee)$

(b) Inversion gadget.

	Location	Type
b_1	$[\ell_3, \ell_4)$	$(0, \wedge)$
b_2	(ℓ_4, ℓ_5)	$(0, \vee)$
b_3	$(\ell_5, \ell_6]$	$(0, \wedge)$

(c) Lower bound gadget.

Claim 5.18. *To stab all breakpoint intervals of a variable gadget with only four breaklines, each of them has to stab three matching breakpoint intervals.*

Proof of Claim. See Table 5.1a. On the three vertical lines, there are six breakpoint intervals for breaklines of type $(\vee, \vee)$ in total. If only two breaklines should stab these six breakpoint intervals, one breakline needs to stab at least two of the single-point intervals. If a breakline goes through two of the single points, it also goes through the third point, and can thus not go through the proper intervals. Therefore, one breakline must stab the single-point intervals, and the other one stabs the proper breakpoint intervals.

The same argument can be made for the breakpoint intervals of type $(\wedge, \wedge)$, and thus each breakline stabs three matching breakpoint intervals. ◀

Claim 5.19. *To stab all breakpoint intervals of an inversion gadget with only five breaklines, each of them has to stab three matching breakpoint intervals.*

Proof of Claim. See Table 5.1b. All five sets of three matching breakpoint intervals have a different type of required breakline, thus each breakline stabs three matching breakpoint intervals. ◀

Claim 5.20. *To stab all breakpoint intervals of a lower bound gadget with only three breaklines, each of them has to stab three matching breakpoint intervals.*

Proof of Claim. See Table 5.1c. There is only one set of three breakpoint intervals for a breakline of type $(0, \vee)$, so it is trivially matched correctly.

We can see that the breakpoint intervals for a breakline of type $(0, \wedge)$ have distance 2 from each other, each having width 1. If the two breaklines of this type would not stab three matching breakpoint intervals, one of them would need to stab two matching intervals and one non-matching interval. As the distance between the vertical lines is equal, and the breakpoint intervals are further apart from each other than their width, there is no way for a breakline to lie in this way. We conclude that all breaklines stab three matching breakpoint intervals. ◀

It also follows from Claims 5.18 to 5.20 that no two breaklines can cross each other between the vertical lines. Neither can a breakline cross a data line. Together with Claim 5.17, we can finally prove Lemma 5.16.

Proof of Lemma 5.16. By Claim 5.17, every breakline must stab three breakpoint intervals of the same gadget. By Claims 5.18 to 5.20, each breakline must stab three matching breakpoint intervals, and therefore the breaklines do not cross any data lines between the three vertical lines.

It remains to show that the data points already ensure that each breakline b is parallel to the two parallel data lines d and d' enclosing it. To this end, consider the parallelogram defined by d, d', v_1, v_3 (see Figure 5.16) and let j be an output dimension in which b is not erased. Since no other breakline intersects this parallelogram, we obtain that f^j has exactly two linear pieces within the parallelogram, which are separated by b. Moreover, since b stabs matching breakpoint intervals, the three data points on d must belong to one of the pieces. Since these points have the same label, it follows that the gradient of this piece in output dimension j must be orthogonal to d (and, thus, to d' as well). Applying the same argument on the data points on d', we obtain that the gradient of the other piece must be orthogonal to d and d' as well. This implies that also the difference of the gradients of the two pieces is orthogonal to d and d'. Finally, since b must be orthogonal to this difference of gradients, we obtain that it is parallel to d and d'. ■

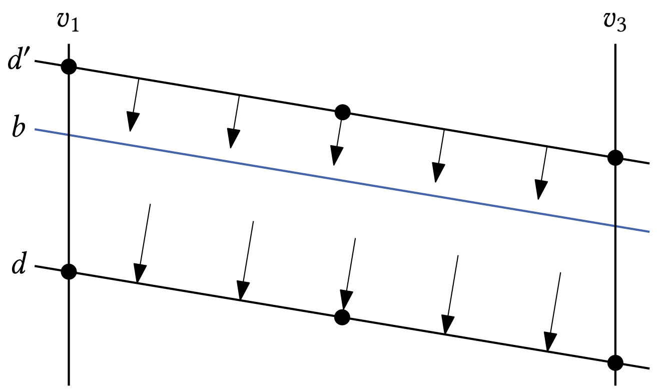


Figure 5.16: The parallelogram enclosed by the two data lines d, d' and the vertical lines v_1, v_3. The three data points (black) on each data line enforce the gradient in both cells to be orthogonal to the data lines. As a consequence, the breakline b (blue) separating the cells has to be parallel to the data lines.

5.8.4 Global Construction

As a last step in our $\exists\mathbb{R}$-hardness proof of TrainNN we describe the global arrangement of the different gadgets. To this end, fix an arbitrary ETR-Inv instance. See Figure 5.17 for a visualization.

Variables For each variable X, we build a horizontal variable gadget carrying the value of this variable. We say that this is the *canonical* variable gadget for X. Lemma 5.10 already ensures that $X \in \left[\frac{1}{2}, 2\right]$.

In order to realize the weak data points of the variable gadgets, we add one lower bound gadget each. They are placed parallel to each other, not parallel to the variable gadgets, and such that their stripes do not intersect any other data points.

Addition The following is done for each addition constraint $X + Y = Z$, next to each other: For each involved variable, we copy the value from its canonical variable gadget to a new variable gadget. To this end, a data point with label 6 is placed on the intersection of the upper measuring line of the canonical variable gadget and the lower measuring line of the new variable gadget (Corollary 5.12).

The three new variable gadgets are positioned such that the correct measuring lines intersect in a common intersection above all horizontal variable gadgets (upper for X, Y and lower for Z). A data point with label 10 at the intersection enforces the addition constraint (Corollary 5.12).

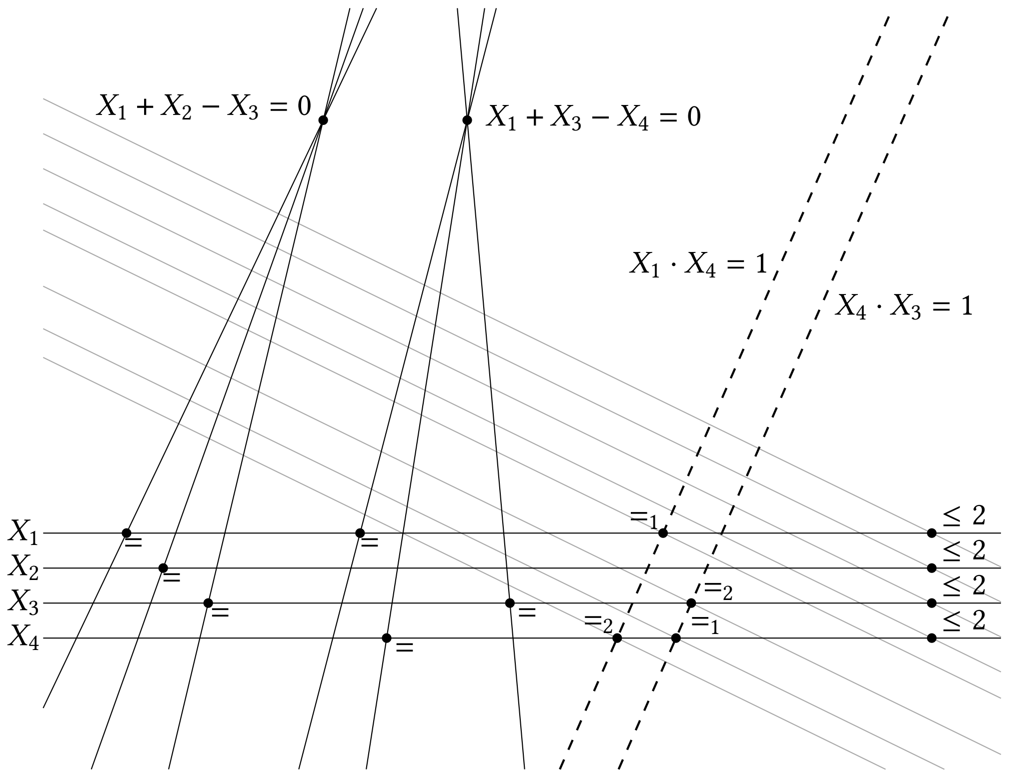


Figure 5.17: The layout of all gadgets and additional data points for the complete reduction. Each gadget is simplified to a single line for clarity. Solid: Variable gadgets. Dashed: Inversion gadgets. Gray: Lower bound Gadgets. A point with label $=_i$ indicates a copy that is only active in output dimension i.

Inversion The following is done for each inversion constraint $XY = 1$, next to each other: We add an inversion gadget that intersects all canonical variable gadgets. Using weak data points with label 6 in the respective dimensions, we copy X to the first dimension of the inversion gadget, and Y to its second dimension (Corollary 5.12). The inversion constraint itself is then enforced by the non-linear relation of the two slopes of the inversion gadget (Lemma 5.13).

All inversion gadgets are placed parallel to each other.

The involved weak data points are realized by two lower bound gadgets per inversion gadget. We place them parallel to the lower bound gadgets for the variables, again such that their stripes do not intersect any other data points.

Based on the global arrangement of the gadgets, we can finally prove our main theorem of this chapter, i.e., the $\exists\mathbb{R}$-completeness of TrainNN:

Proof of Theorem 5.3. For $\exists\mathbb{R}$-membership we refer to [4] and Section 5.7.

For $\exists\mathbb{R}$-hardness, we reduce the $\exists\mathbb{R}$-complete problem ETR-Inv to TrainNN. Given an instance of ETR-Inv, we construct an instance of TrainNN as described in the previous paragraphs. We set the target error to $\gamma = 0$.

Let m be the minimum number of breaklines needed to realize all gadgets of the above construction: We need four breaklines per variable gadget (Lemma 5.10), five breaklines per inversion gadget (Lemma 5.13) and three breaklines per lower bound gadget (Lemma 5.14). By Lemma 5.16, no breakline can contribute to different gadgets, so we need exactly that many.

In the remainder of the proof, we show equivalence of the following statements.

(1) The ETR-INV instance is a yes-instance, i.e., there exists a satisfying assignment of the variables.

(2) There exists a continuous piecewise linear function with m breaklines that fits all data points of the constructed TRAINNN instance. Furthermore, it fulfills the conditions of Lemma 5.7.

(3) The TRAINNN instance is a yes-instance, i.e., there exists a fully connected two-layer neural network with m hidden ReLU neurons exactly fitting all the data points.

To see that (1) implies (2), assume that there is a satisfying assignment of the ETR-INV instance with all variables in $\left[\frac{1}{2}, 2\right]$. For each variable X, we use $s_X = X + 1$ as the slope of all corresponding variable gadgets and inversion gadgets. The superposition of all these gadgets yields the desired continuous piecewise linear function. It satisfies Lemma 5.7 because, first, the gadgets are built in such a way that functions fitting all data points are constantly zero everywhere except for within the gadgets, and second, the gradient condition is satisfied for each gadget separately and, hence, also for the whole function.

For the other direction, i.e., that (2) implies (1), assume that such a continuous piecewise linear function exists. By Lemmas 5.14 and 5.16, the data points enforce exactly the same continuous piecewise linear function as the conceptual data lines and weak data points would. Then, by Lemmas 5.10 and 5.13, this continuous piecewise linear function has the shape of the gadgets. The fact that all data points are fit implies that the slopes of the variable and inversion gadgets indeed correspond to a satisfying assignment of ETR-INV.

Lemma 5.7 yields that (2) implies (3).

It remains to prove that (3) implies (2). To this end, first note that the function realized by a fully connected two-layer neural network with m hidden ReLU neurons is always a continuous piecewise linear function with at most m breaklines that additionally satisfies the gradient condition by Lemma 5.7. The existence of a $(0, 0)$-cell follows from the fact, that all gadgets are constantly 0 outside their stripes.

The TRAINNN instance can be constructed in polynomial time, as the gadgets can be arranged in such a way that all data points (residing on intersections of lines) have coordinates which can be encoded in polynomial length.

The number of hidden neurons m is linear in the number of variables and the number of constraints of the ETR-INV instance. The number of data points can be bounded by $10m$, thus the number of hidden neurons is linear in the number of data points.

As can be gathered from Lemmas 5.10, 5.13 and 5.14 and Corollary 5.12, the set of used labels has cardinality 13 as claimed. ■

Remark 5.21. Note that if the ETR-INV instance is satisfiable, then each variable gadget and inversion gadget in a corresponding solution Θ to the constructed TRAINNN instance has a slope of at most 3 in each dimension. Furthermore, no lower bound gadget needs to contribute less than -12 to satisfy its corresponding weak data point. Thus, there must also be a solution Θ', where each lower bound gadget is symmetric, and thus the function $f(\cdot, \Theta')$ is Lipschitz continuous with a low Lipschitz constant L, which in particular does not depend on the given ETR-INV instance. Checking all the different ways how our gadgets intersect, one can verify that $L = 25$ is sufficient. ⌟

5.9 Algebraic Universality

It remains to prove algebraic universality of TRAINNN. Intuitively, it suffices to show that a solution of an ETR-INV instance can be transformed into a solution of the corresponding TRAINNN instance (and vice versa) using only basic field arithmetic, that is, addition, subtraction, multiplication, and division.

For the following lemma, let Φ be an instance of ETR-INV with k variables, and let N be an instance of TRAINNN with a total of ℓ weights and biases. Furthermore, N was constructed from Φ via our reduction. We denote by $V(\Phi) \subseteq \mathbb{R}^k$ the set of all satisfying variable assignments. Similarly, $V(N) \subseteq \mathbb{R}^\ell$ contains all weight-bias-combinations that fit all data points.

Lemma 5.22. *For any field extension $\mathbb{F}$ of $\mathbb{Q}$ it holds that*

$$V(\Phi) \cap \mathbb{F}^k \neq \emptyset \iff V(N) \cap \mathbb{F}^\ell \neq \emptyset.$$

Proof. "$\Longrightarrow$:" Let $X_1, \ldots, X_n \in V(\Phi) \cap \mathbb{F}^k$ be a satisfying variable assignment for Φ. In our reduction, we place our data points on rational coordinates, and thus all implied data lines can be described by equations with rational coefficients. There exists a unique continuous piecewise linear function f which fits these data points, corresponds to the solution $X_1, \ldots, X_n$, and in which all lower bound gadgets are symmetric. This function can be realized by a fully connected two-layer neural network by Lemma 5.7. The gradients of all cells in this function can be obtained through elementary operations from the values $X_1, \ldots, X_n$ and rational numbers. Furthermore, all breaklines can be described by equations with coefficients derivable from these same numbers. Thus, there exist weights and biases $\Theta \in \mathbb{F}^\ell$ for the neural network which realize function f, showing that $\Theta \in V(N) \cap \mathbb{F}^\ell \neq \emptyset$.

"$\Longleftarrow$:" Let $\Theta \in V(N) \cap \mathbb{F}^\ell$ be a set of weights and biases fitting all data points of the TRAINNN instance N. For each variable X of Φ, there is a canonical variable gadget corresponding to X whose slope s_X satisfies $X = s_X - 1$. There is a unique hidden neuron v_i contributing the first breakline of that variable gadget. Using the notation from Section 5.8.1, the slope of this variable gadget is $a_{2,i} \cdot c_{i,1}$, because the variable gadget is horizontal (implying that $a_{1,i} = 0$) and its output is equal in both output dimensions (implying $c_{i,1} = c_{i,2}$). Thus, $X = a_{2,i} \cdot c_{i,1} - 1$, which is clearly in $\mathbb{F}$. The same holds for all other variables, thus $V(\Phi) \cap \mathbb{F}^k \neq \emptyset$.

■

Now Theorem 5.4, i.e., the algebraic universality of TrainNN, follows directly from the algebraic universality of ETR-Inv (Theorem 5.8) combined with Lemma 5.22.

5.10 Conclusion and Open Problems

We proved $\exists\mathbb{R}$-completeness of TrainNN, and therefore also for the general neural network training problem known as EmpiricalRiskMinimization. Previous work shows this only for specifically designed adversary network architectures [4].

Our result fits well into the existing body of literature about the hardness of training even small neural networks. It improves the more than twenty-year-old NP-hardness result by Blum and Rivest [25], contemplating it with algebraic universality. At the same time, our result renders any generalization of the celebrated NP-membership argument for single-output neural networks by Arora, Basu, Mianjy and Mukherjee impossible [12].

Apart from that, $\exists\mathbb{R}$-hardness has the "usual" implications, see Section 1.3. In particular, our result gives theoretical evidence on why no better algorithms than backpropagation are available to train neural networks.

Nevertheless, several problems remain open, some of which are highlighted below:

Open Problem 5. What is the complexity to train (two-layer) neural networks with a single input neuron?

Open Problem 5 asks about neural networks with an even simpler architecture than the ones considered above. We proved $\exists\mathbb{R}$-hardness for two input and two output neurons. It is known that a single output neuron leads to NP-membership [12], but the case for a single input neuron is open. A possible approach to prove NP-membership would be to first guess the so-called *activation pattern* of each ReLU neuron non-deterministically ("Which side is the active region?"), and then to use a linear program to compute the weights and biases. However, the details of this approach would need to be worked out. Note that training two-layer neural networks with a (large enough) constant number of inputs is already NP-hard [66].

Open Problem 6. What is the complexity of training neural networks with two or more hidden layers?

Having just one hidden layer is the simplest possible case, and we proved $\exists\mathbb{R}$-hardness for it. However, our reduction heavily relies on the geometry of continuous piecewise linear functions realized by neural networks with a single hidden layer. We are not aware of any approach to generalize our reduction to the more complicated geometry of neural networks with more hidden layers. On the other hand, we are also not aware of any reason why the problem should become computationally simpler.

Open Problem 7. What is the complexity of "approximately" training neural networks?

Open Problem 7 is deliberately not well-defined. In practice, one does not need large precision for the weights and biases to achieve low training and generalization error. Even custom hardware like *tensor processing units* (TPUs) with particularly low precision but in exchange high performance is used for training large neural networks.

$\exists\mathbb{R}$-hardness and high precision often go hand in hand, and our result does not generalize to approximate training[2]. In particular, it does not rule out efficient algorithms for well-defined approximate versions of EmpiricalRiskMinimization. Progress in this direction would be of high significance to the machine learning community.

2 Actually, we can prove $\exists\mathbb{R}$-hardness for a very weak form of approximate training, namely such that each data point is fit up to an error of at most ε, where ε is doubly exponentially small in the instance size. This follows from standard techniques based on Theorem 6.13 below. See also [49]. We consider this to only be of theoretical interest, and not yet as a promising approach towards solving Open Problem 7.

6 The Hausdorff Distance and ∀∃ℝ

This chapter is based on our paper "The Complexity of the Hausdorff Distance" published in *Discrete & Computational Geometry* [95]. Preliminary versions were presented at the *38th European Workshop on Computational Geometry (EuroCG 2022)* [93] and published at the *38th International Symposium on Computational Geometry (SoCG 2022)* [94]. It is joint work with Linda Kleist and Tillmann Miltzow.

So far, in Chapters 3 to 5, the computational complexity of all considered problems was captured exactly by the complexity class ∃ℝ. In other words, all these problems are equally difficult as deciding ETR, the existential theory of the reals. Now, we consider a problem that is presumably even more difficult, namely the problem to compute the *Hausdorff distance* between two semi-algebraic sets. As it turns out, this problem is ∀∃ℝ-complete, where $\exists\mathbb{R} \subseteq \forall\exists\mathbb{R}$.

Recall that we can think of ∃ℝ as a "real analogue" of NP. Together, NP and coNP form the first level of the *polynomial hierarchy* (PH). Similarly, one can define the *real polynomial hierarchy* having ∃ℝ and $\forall\mathbb{R} = \text{co}\exists\mathbb{R}$ at its first level. The complexity class ∀∃ℝ is on the second level of this hierarchy and can be understood as the real analogue of Π_2^{P} in the (classical) polynomial hierarchy.

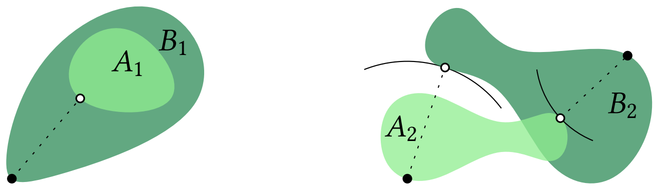
Explicitly, the study of ∀∃ℝ was initiated recently by Dobbins, Kleist, Miltzow and Rzążewski [54], but similar concepts appeared earlier [34]. Still, not much is known about this class, and standard tools to prove ∀∃ℝ-hardness have yet to be developed. Our result about the ∀∃ℝ-completeness of computing the Hausdorff distance is the first ∀∃ℝ-completeness result for a natural problem.

Chapter Outline In Section 6.1, we introduce the Hausdorff distance, motivate its importance, highlight its practical difficulty, and discuss possible solution strategies. The main results and a high-level proof sketch are presented in Section 6.2, followed by the related work in Section 6.3. In Section 6.4, we discuss the necessary tools from real algebraic geometry. Section 6.5 contains our key technical contribution, which is crucial to prove ∀∃ℝ-hardness in Section 6.6.

Next, we discuss ∀∃ℝ-membership: First, in Section 6.7, under consideration of a recent paper that followed our initial publication. Second, in Section 6.8, using our original approach. We finish this chapter with an excursion about exotic quantifiers and their applications to the Hausdorff problem in Section 6.9.

6.1 Introduction

The question "How similar are two given objects?" occurs in numerous settings. For three concrete examples, consider Figure 6.1. A typical tool to quantify their similarity is the *Hausdorff distance*: Two sets have small Hausdorff distance if every point of one set is close to some point of the other set and vice versa.



(a) $d_{\mathrm{H}}(A_1, B_1)$ **(b)** $d_{\mathrm{H}}(A_2, B_2)$ **(c)** $d_{\mathrm{H}}(A_3, B_3)$

Figure 6.1: How similar are these sets?

Formally, the *directed Hausdorff distance* between two non-empty sets $A, B \subseteq \mathbb{R}^n$ is defined as

$$\vec{d}_{\mathrm{H}}(A, B) := \sup_{a\in A} \inf_{b\in B} \|a - b\|.$$

The directed Hausdorff distance between A and B can be interpreted as the smallest value $t \geq 0$ such that the (closed) t-neighborhood of B contains A. Hence, it nicely captures the intuition of how much B has to be expanded uniformly in all directions to contain A. Note that this definition is not symmetric, so $\vec{d}_{\mathrm{H}}(A, B)$ and $\vec{d}_{\mathrm{H}}(B, A)$ may differ. For an example, consider Figure 6.1a; while $A_1 \subseteq B_1$ and thus $\vec{d}_{\mathrm{H}}(A_1, B_1) = 0$, it holds that $\vec{d}_{\mathrm{H}}(B_1, A_1) > 0$. In contrast, the (undirected) *Hausdorff distance* is symmetric and defined as

$$d_{\mathrm{H}}(A, B) := \max\left\{\vec{d}_{\mathrm{H}}(A, B), \vec{d}_{\mathrm{H}}(B, A)\right\}.$$

In this thesis, we investigate the computational complexity of deciding whether the Hausdorff distance of two sets is at most a given threshold.

The Hausdorff distance appears in many branches of science: In mathematics, the Hausdorff distance provides a metric on sets and henceforth also a topology. This topology can be used to discuss continuous transformations of one set to another [30]. In computer vision and geographical information science, the Hausdorff distance is used to measure the similarity between spacial objects [114, 131] for example the quality of quadrangulations of complex 3D models [151].

6.1.1 Semi-Algebraic Sets

The algorithmic complexity of computing the Hausdorff distance clearly depends on the type of the underlying sets: If both sets consist of finitely many points, then their Hausdorff distance can easily be computed in polynomial time by checking all pairs of points. However, one often considers infinite sets, such as collections of disks in the plane, cubic splines or surfaces in three (or more) dimensions.

Here, we consider semi-algebraic sets. Formally, a *semi-algebraic set* is the finite union of so-called basic semi-algebraic sets. A *basic semi-algebraic set* S is specified by two finite families of polynomials $\mathcal{P}$ and $\mathcal{Q}$ such that

$$S = \left\{x \in \mathbb{R}^n \mid \bigwedge_{P \in \mathcal{P}} P(x) \leq 0 \land \bigwedge_{Q \in \mathcal{Q}} Q(x) < 0\right\}.$$

Semi-algebraic sets clearly cover the vast majority of practical cases. Simultaneously, even in supposedly simple cases, i.e., when considering circles, ellipses or cubic splines, one has to use polynomial equations to describe the sets.

Remark 6.1 (Integer Coefficients)**.** In real algebraic geometry, one usually allows real-valued coefficients, i.e., $\mathcal{P}, \mathcal{Q} \subseteq \mathbb{R}[X_1, \ldots, X_n]$. However, we consider the computational complexity of computing the Hausdorff distance in the standard bit-model of computation. Here, the polynomials describing the two sets A and B are part of the input and therefore given with finite precision, usually in some binary encoding. In this thesis, we always assume that the coefficients are integers encoded as described in Section 2.1.3. ⌟

Having said that, it is still possible to encode semi-algebraic sets with arbitrary coefficients from $\mathbb{R}_{\text{alg}}$ by introducing auxiliary variables:

Example 6.2 (Algebraic Coefficients)**.** Assume that we are interested in $d_{\text{H}}(A, B)$ for two semi-algebraic sets

$$A := \left\{x \in \mathbb{R} \mid \sqrt{2} \cdot x^2 = 1\right\} \quad \text{and}$$
$$B := \left\{x \in \mathbb{R} \mid x = 1\right\}.$$

The coefficient $\sqrt{2}$ in the definition of A is obviously not an integer and thus does not fulfill above restriction. However, it is possible to define two new sets

$$A' := \left\{(x, u) \in \mathbb{R}^2 \mid u > 0 \land u^2 = 2 \land u \cdot x^2 = 1\right\} \quad \text{and}$$
$$B' := \left\{(x, u) \in \mathbb{R}^2 \mid u > 0 \land u^2 = 2 \land x = 1\right\}$$

that circumnavigate this problem: The trick is to add a new variable u whose value is forced to $u = \sqrt{2}$ by the additional constraints. Note that all constraints have integer coefficients only. It holds that $d_{\text{H}}(A, B) = d_{\text{H}}(A', B')$. Thus, modeling algebraic coefficients is indeed possible with a small overhead in the description complexity of the considered sets. ⌟

6.1.2 An Algorithmic Approach

In order to demonstrate how difficult it is in practice to compute the Hausdorff distance even between two simple curves in $\mathbb{R}^2$, let us consider an example. Our presented approach uses the convexity and continuity of the two curves. In particular, it does not easily generalize to arbitrary semi-algebraic sets. In the next paragraph, we present a slower, but more general method.

Example 6.3 (Bernd Sturmfels, 2019). The polynomials $f, g \in \mathbb{Z}[x, y]$, given by

$$f(x, y) := x^4 + y^4 + 12x^3 + 2y^3 - 3xy + 11 \quad \text{and}$$
$$g(x, y) := 7x^4 + 8y^4 - 1,$$

define sets $A = \{(x, y) \in \mathbb{R}^2 \mid f(x, y) = 0\}$ and $B = \{(x, y) \in \mathbb{R}^2 \mid g(x, y) = 0\}$. For an illustration, see the blue and green curve in Figure 6.2, respectively.

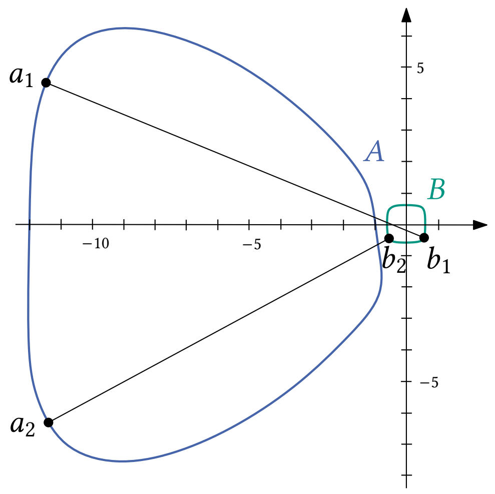


Figure 6.2: The Hausdorff distance between the compact semi-algebraic sets A (blue) and B (green) is attained at points (a_2, b_2) such that the segment a_2b_2 is orthogonal to the tangents at a_2 and b_2. The segment a_1b_1 is longer than a_2b_2, but it crosses both A and B. Therefore, (a_1, b_1) does not realize the Hausdorff distance.

Both curves are convex and smooth, so their Hausdorff distance is attained at points $a \in A$ and $b \in B$ such that the segment ab is orthogonal to the tangents at a and b. This can be formulated by a set of polynomial equations in four variables. The system has 240 complex solutions, eight of which are real. These 240 solutions can be computed using computer algebra systems based on Gröbner bases. For some real solutions (a, b), the segment ab crosses A and B, for example a_1b_1 in Figure 6.2. These solutions can be discarded. Among the remaining solutions, the points $a_2 \approx (-11.48362, -6.1760)$ and $b_2 \approx (-0.56460, -0.43583)$ realize the Hausdorff distance of approximately 12.33591. ⌟

6.1.3 A Decision Algorithm and Computational Complexity

Assume we are given two semi-algebraic sets A and B, and a threshold $t \in \mathbb{N}$. The statement $\vec{d}_{\mathrm{H}}(A, B) \leq t$ can be encoded as a first-order sentence of the form[1]

$$\forall \varepsilon > 0, a \in A \,.\, \exists b \in B \colon \|a - b\| \leq t + \varepsilon, \tag{6.1}$$

1 An extension to the undirected case $d_{\mathrm{H}}(A, B) \leq t$ is straightforward, see Section 6.8, and was already done by Dobbins, Kleist, Miltzow and Rzążewski [54].

where the ε is needed to also consider points in the closure of B. Recall that such a sentence is decidable (Section 2.2) by applying sophisticated algorithms from real algebraic geometry that can deal with two blocks of quantifiers [16, Chapter 14]. However, these algorithms are so slow that they are impractical, even for small instances like the one in Example 6.3.

Despite being unable to decide the truth of a sentence like (6.1) efficiently in practice, it already provides an upper bound on the complexity of the directed Hausdorff distance: Deciding whether $\vec{d}_{\mathrm{H}}(A,B) \leq t$ is at most as difficult as deciding the *∀∃-fragment* of the first-order theory of the reals.

To make this precise, let us recall the definition from Section 2.2: Let φ be a quantifier-free formula in the first-order theory of the reals with variables $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_m)$. The decision problem UNIVERSAL EXISTENTIAL THEORY OF THE REALS (UETR) asks whether a sentence of the form

$$\forall X \in \mathbb{R}^n \, . \, \exists Y \in \mathbb{R}^m \colon \varphi(X, Y)$$

is true. We denote the special case of UETR in which φ does not contain negations (no "$\neg$") and all atoms are strict inequalities (only "$<$", "$>$" or "$\neq$") by STRICT-UETR. Of course, STRICT-UETR is at most as difficult as UETR.

The complexity classes $\forall\exists\mathbb{R}$ and $\forall\exists_{<}\mathbb{R}$ contain exactly all decision problems that polynomial-time many-one reduce to UETR and STRICT-UETR, respectively. To the best of our knowledge, $\forall\exists\mathbb{R}$ was first introduced by Bürgisser and Cucker [34, Section 9] under the name $\mathrm{BP}^0(\forall\exists)$ (in the constant-free Boolean part of the BSS model [26]). The notation $\forall\exists\mathbb{R}$ emerged later in [54], extending the notation from Schaefer and Štefankovič [138].

6.2 Problem and Results

We now have all ingredients to state our problem and main results:

Definition 6.4 (HAUSDORFF).
Input: *An integer $n \in \mathbb{N}$, two quantifier-free formulas $\varphi_A(X)$ and $\varphi_B(X)$ with n free variables $X = (X_1, \ldots, X_n)$, and an integer $t \in \mathbb{N}$.*
Question: *For $A := \{x \in \mathbb{R}^n \mid \varphi_A(x)\}$ and $B := \{x \in \mathbb{R}^n \mid \varphi_B(x)\}$, is $d_{\mathrm{H}}(A,B) \leq t$?*

In the same manner, we denote by DIRECTEDHAUSDORFF the problem to determine whether $\vec{d}_{\mathrm{H}}(A,B) \leq t$.

Note that n, the dimension of the ambient space of A and B, is part of the input. In fact, there is a polynomial-time algorithm for every fixed n, see the related work in Section 6.3. Our main result determines the algorithmic complexity:

Theorem 6.5. HAUSDORFF *and* DIRECTEDHAUSDORFF *are $\forall\exists_{<}\mathbb{R}$-complete.*

Prior to our result, it was not even known whether computing the Hausdorff distance is NP-hard. As $\forall\exists_{<}\mathbb{R}$ contains the complexity classes NP, coNP, $\exists\mathbb{R}$ and $\forall\mathbb{R}$, our result implies hardness for these classes. Theorem 6.5 answers an open question posed by Dobbins, Kleist, Miltzow and Rzążewski [54].

Remark 6.6 ($\forall\exists_<\mathbb{R} = \forall\exists\mathbb{R}$). When we published Theorem 6.13 [93, 94, 95], it remained an open question whether $\forall\exists_<\mathbb{R} \subseteq \forall\exists\mathbb{R}$ was a strict inclusion. This question was answered recently by Schaefer and Štefankovič, who prove that $\forall\exists_<\mathbb{R} = \forall\exists\mathbb{R}$ [139], implying $\forall\exists\mathbb{R}$-completeness of Hausdorff and DirectedHausdorff. We discuss the implications of their result on our reduction in Section 6.7 below. ⌟

Our $\forall\exists_<\mathbb{R}$-hardness reduction for Hausdorff creates instances with some additional properties. First, our reduction is a gap reduction. In particular, the Hausdorff distance of the obtained instance is either below the threshold t or at least $t \cdot 2^{2^{\Omega(n)}}$. Thus, our result also yields the following inapproximability result:

Corollary 6.7. *Let A and B be two semi-algebraic sets in $\mathbb{R}^n$ and $f(n) = 2^{2^{o(n)}}$. There is no polynomial-time $f(n)$-approximation algorithm to compute $d_H(A, B)$, unless $\mathsf{P} = \forall\exists_<\mathbb{R}$.*

Second, our reduction can be modified slightly to obtain a Hausdorff instance in which A and B are described by *structurally* simple formulas: All atoms are polynomial equations of bounded degree and the formula is a conjunction of atoms. Of course, this comes at an expense, namely an increased number of variables. The following corollary states that this structural simplicity does not make the Hausdorff problem simpler, instead it remains equally difficult:

Corollary 6.8. *The Hausdorff problem remains $\forall\exists_<\mathbb{R}$-complete, even if the two sets A and B are both described by either*

(i) a conjunction of quadratic polynomial equations, or

(ii) a single polynomial equation of degree at most four.

6.2.1 Techniques and Proof Idea

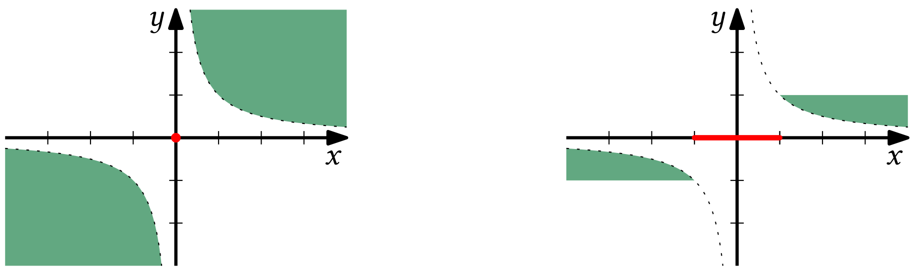
In this section, we present the general idea behind the $\forall\exists_<\mathbb{R}$-hardness reduction for the Hausdorff problem. The goal is to convey the intuition and to motivate the technical intermediate steps needed. The sketched reduction is oversimplified and thus neither in polynomial time nor fully correct. We point out both of these issues and give first ideas on how to solve them.

Let $\Phi :\equiv \forall X \in \mathbb{R}^n . \exists Y \in \mathbb{R}^m \colon \varphi(X, Y)$ be a Strict-UETR instance. We define two sets

$$A := \left\{x \in \mathbb{R}^n \mid \exists Y \in \mathbb{R}^m \colon \varphi(x, Y)\right\} \quad \text{and}$$
$$B := \mathbb{R}^n$$

and ask whether $d_H(A, B) = 0$. If Φ is true, then $A = \mathbb{R}^n$ and we have $d_H(A, B) = 0$ because both sets are equal. Otherwise, if Φ is false, then there exists some $x \in \mathbb{R}^n$ for which there is no $y \in \mathbb{R}^m$ satisfying $\varphi(x, y)$ and we conclude that $A \subsetneq \mathbb{R}^n$. In general, we call the set of all $x \in \mathbb{R}^n$ for which there is no $y \in \mathbb{R}^m$ satisfying $\varphi(x, y)$ the *counterexamples* $\perp(\Phi)$ of Φ. One might hope that $\perp(\Phi) \neq \emptyset$ is enough to obtain that $d_H(A, B) > 0$. However, this is not the case. To this end, consider the formula $\Psi :\equiv \forall X \in \mathbb{R} \,.\, \exists Y \in \mathbb{R} \colon XY > 1$, which is false. The set $\bot(\Psi) = \{0\}$ contains only a single element, so we have $A = \mathbb{R} \setminus \{0\}$ and $B = \mathbb{R}$. Sets A and B have the same closure, so their Hausdorff distance evaluates to $d_H(A, B) = 0$. We conclude that above reduction does not (yet) work, because it may map no-instances of Strict-UETR to yes-instances of Hausdorff.

We solve this issue by a preprocessing step that expands the set of counterexamples. Specifically, Theorem 6.20 below establishes a polynomial-time algorithm to transform a Strict-UETR instance Φ into an equivalent instance Φ' such that the set of counterexamples is either empty (if Φ' is true) or contains an open ball of positive radius (if Φ' is false). The radius of the ball serves as a lower bound on the Hausdorff distance $d_H(A, B)$. Thus, a reduction starting from Φ' is correct. A key tool for this step is that we can restrict the variable ranges from $\mathbb{R}^n$ and $\mathbb{R}^m$ to small and compact intervals. Figure 6.3 presents an example on how such a range restriction may enlarge the set of counterexamples from a single point to an interval.



(a) Points $(x, y) \in \mathbb{R}^2$ in the green open region satisfy $xy > 1$. Only for $x = 0$ (in red) no suitable $y \in \mathbb{R}$ exists.

(b) If we restrict Y to $[-1, 1]$, then for no $x \in [-1, 1]$ (in red) a suitable $y \in [-1, 1]$ with $xy > 1$ exists.

Figure 6.3: Expanding the set of counterexamples of $\forall X \in \mathbb{R} \,.\, \exists Y \in \mathbb{R} \colon XY > 1$.

We emphasize that it is unclear, whether it is possible to restrict the variable ranges for general UETR instances. However, we exploit a special property of Strict-UETR instances, namely that they are ∀-strict: A negation-free and implication-free formula is *∀-strict* if each atom containing universally quantified variables is a strict inequality. Being ∀-strict is a key property of many of the formulas considered throughout this chapter, and our proofs crucially rely on it.

Another challenge hides in the definition of set A. While the description complexity of B depends only on n, the definition of A contains an existential quantifier. This is troublesome because our definition of the Hausdorff problem requires *quantifier-free* formulas as its input, and in general there is no equivalent quantifier-free formula of polynomial length which describes the set A [48]. We overcome this issue by taking the existentially quantified variables as additional dimensions into account. We scale them to a range doubly exponentially smaller than the range of the universally quantified variables, so that their influence on the Hausdorff distance becomes negligible. Therefore, instead of the above, we work (in Section 6.6) with sets similar to

$$\begin{aligned} A &:= \left\{(x,y) \mid x \in [-C,C]^n \wedge y \in [-1,1]^m \wedge \varphi(x,y)\right\} \quad \text{and} \\ B &:= [-C,C]^n \times \{0\}^m \end{aligned}$$

for some value C that is doubly exponentially large in $|\Phi|$. This definition of A and B introduces the new issue that, even if Φ is true, the Hausdorff distance $d_{\mathrm{H}}(A,B)$ might be strictly positive. However, we manage to identify a threshold $t \in \mathbb{N}$, such that $d_{\mathrm{H}}(A,B) \leq t$ if and only if Φ is true.

6.3 Related Work

The notion of the Hausdorff distance was introduced by Hausdorff in 1914 [84]. Many early works focused on the Hausdorff distance for finite point sets. For a set of a points and another set of b points in any fixed dimension, the Hausdorff distance can be computed by checking all pairs, i.e., in time $O(ab)$. In the plane, the runtime can be improved to $O((a+b)\log(a+b))$ using Voronoi diagrams [9]. In fact, this method can be extended to sets consisting of pairwise non-crossing line segments in the plane, e.g., simple polygons and polygonal chains fulfill this property. If the polygons are convex, their Hausdorff distance can even be computed in linear time [13].

More generally, the Hausdorff distance can be computed in polynomial time whenever the two sets can be described by a simplicial complex of fixed dimension. Alt, Braß, Godau, Knauer and Wenk [10, Theorem 3.3] show how to compute the directed Hausdorff distance between two sets in $\mathbb{R}^n$ consisting of a and b k-dimensional simplices in time $O(ab^{k+2})$ (assuming n is constant). Using a Las Vegas algorithm for computing the vertices of the lower envelope, similar ideas yield an approach with randomized expected time in $O(ab^{k+\varepsilon})$ for $k > 1$ and every $\varepsilon > 0$ [10, Theorem 3.4]. They additionally present algorithms with better randomized expected running times for sets of triangles in $\mathbb{R}^3$ and point sets in $\mathbb{R}^n$.

Given two semi-algebraic sets $A, B \subseteq \mathbb{R}^n$ and a threshold value $t \in \mathbb{N}$, the Hausdorff decision problem can be encoded as a UETR sentence Φ, thereby proving $\forall\exists\mathbb{R}$-membership. Recall that we have done this already for the DirectedHausdorff problem in sentence (6.1), based on a UETR formula for Hausdorff by Dobbins, Kleist, Miltzow and Rzążewski [54]. Such a sentence can be decided in time $s^{(n+1)^2} d^{O(n)^2}$ by Theorem 2.5, i.e., by using an algorithm to decide general sentences from the first-order theory of the reals. (Here d denotes the maximum degree of any polynomial of Φ and s denotes the number of atoms.)

Independently (and using different terminology), Bürgisser and Cucker proved $\forall\exists\mathbb{R}$-membership and $\forall^*\exists\mathbb{R}$-hardness for DirectedHausdorff [34]. The $\forall^*$ is a so-called "exotic" quantifier. We discuss the connection between this complexity class and $\forall\exists_<\mathbb{R}$ in Section 6.9.

In other contexts, the two sets are allowed to undergo certain transformations (e.g., translations) such that their Hausdorff distance gets minimized [32]. See for example the survey by Alt and Guibas [11].

6.4 Preliminaries

This section introduces the basic tools for our reduction. We start by recalling the necessary complexity classes and discuss their relations among one another. Then we review two important results from real algebraic geometry that will allow us to restrict the ranges of the variables. For the definitions of first-order formulas and their encoding, see Section 2.1.

6.4.1 Complexity Classes

Several complexity classes appear in this chapter. Here, we discuss their relations among one another. We make use of a helpful lemma from the literature. It allows us to replace a quantifier-free formula φ by a structurally simpler one (at the cost of adding additional variables).

Concerning notation, we express the existence of a polynomial $p \in \mathbb{Z}[Y_1, \ldots, Y_k]$ such that $x \leq p(y_1, \ldots, y_k)$ by $x \leq \mathrm{poly}(y_1, \ldots, y_k)$. Furthermore, we write QFF for the set of all quantifier-free first-order formulas.

Lemma 6.9 ([138, Lemma 3.2]). *Let $\varphi(X) \in \mathrm{QFF}$ be a formula with n free variables X. Then we can construct either of the following in polynomial time:*

(i) Integers $\ell, m \leq \mathrm{poly}(|\varphi|)$, and for $i \in \{1, \ldots, m\}$ a polynomial $F_i \colon \mathbb{R}^{n+\ell} \to \mathbb{R}$ with integer coefficients of degree at most 2 such that

$$\{x \in \mathbb{R}^n \mid \varphi(x)\} = \{x \in \mathbb{R}^n \mid \exists Y \in \mathbb{R}^\ell \colon \bigwedge_{i=1}^{m} F_i(x, Y) = 0\}.$$

(ii) An integer $k \leq poly(|\varphi|)$, and a polynomial $F \colon \mathbb{R}^{n+k} \to \mathbb{R}$ with integer coefficients of degree at most 4 such that

$$\{x \in \mathbb{R}^n \mid \varphi(x)\} = \{x \in \mathbb{R}^n \mid \exists Y \in \mathbb{R}^k \colon F(x, Y) = 0\}.$$

For any fixed $\circ \in \{<, \leq\}$, the subset $\mathrm{QFF}_\circ \subseteq \mathrm{QFF}$ contains all quantifier-free *and negation-free* first-order formulas in which each atom uses $\circ$. Further, we denote by $\forall\exists_\circ\mathbb{R}$ the subset of $\forall\exists\mathbb{R}$ containing all decision problems that polynomial-time many-one reduce to a UETR-instance whose quantifier-free parts are contained in $\mathrm{QFF}_\circ$. Similarly, we denote the corresponding subsets of $\exists\mathbb{R}$ and $\forall\mathbb{R}$ by $\exists_\circ\mathbb{R}$ and $\forall_\circ\mathbb{R}$, respectively. The following lemma summarizes what we know about the relation between the complexity classes $\forall\exists_<\mathbb{R}$, $\forall\exists_\leq\mathbb{R}$ and $\forall\exists\mathbb{R}$ as well as their relation to the well-studied classes NP, coNP, $\exists\mathbb{R}$, $\forall\mathbb{R}$, and PSPACE.

Lemma 6.10. *The following inclusions hold:*

$$\begin{tikzcd}
\mathrm{NP} \arrow[r, phantom, "\subseteq"] & \exists\mathbb{R} \arrow[dr, phantom, "\subsetneq"] & & & & & & \\
& & \forall\exists_<\mathbb{R} \arrow[r, phantom, "\subseteq"] & \forall\exists_\leq\mathbb{R} \arrow[r, phantom, "="] & \forall\exists\mathbb{R} \arrow[r, phantom, "\subseteq"] & \mathrm{PSPACE} \\
\mathrm{coNP} \arrow[r, phantom, "\subseteq"] & \forall\mathbb{R} \arrow[ur, phantom, "\subsetneq"] & & & &
\end{tikzcd}$$

Proof. The inclusion $\mathrm{NP} \subseteq \exists\mathbb{R}$ was first presented by Shor [144]. This directly implies $\mathrm{coNP} \subseteq \forall\mathbb{R}$ (because $\forall\mathbb{R} = \mathrm{co}\exists\mathbb{R}$). For $\circ \in \{<, \leq\}$ the inclusion $\forall\exists_\circ\mathbb{R} \subseteq \forall\exists\mathbb{R}$ follows by definition because the left-hand side is just a special case of the right-hand side. Using that $\exists_<\mathbb{R} = \exists\mathbb{R}$ [138, Theorem 4.1], the same argument can be used for $\exists\mathbb{R} \subseteq \forall\exists_<\mathbb{R}$. Canny first established $\forall\exists\mathbb{R} \subseteq \mathrm{PSPACE}$ [37].

To show that $\forall\exists\mathbb{R} \subseteq \forall\exists_\leq\mathbb{R}$, consider a UETR instance

$$\Phi \;:\equiv\; \forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m \colon \varphi(X, Y).$$

We apply Lemma 6.9 to φ and obtain in polynomial time an integer $k \leq \mathrm{poly}(|\varphi|)$ and a polynomial $F\colon \mathbb{R}^{n+m+k} \to \mathbb{R}$, such that

$$\Psi \;:\equiv\; \forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^{m+k} \colon F(X, Y) \leq 0 \wedge -F(X, Y) \leq 0,$$

is equivalent to Φ. Note that both atoms use $\leq$.

Lastly, let us consider the inclusion $\forall\mathbb{R} \subseteq \forall\exists_<\mathbb{R}$. Note that $\forall\mathbb{R} = \forall_<\mathbb{R}$ (because two complexity classes are equal whenever their complement classes are equal and $\exists\mathbb{R} = \exists_\leq\mathbb{R}$ is known [138]). Then $\forall_<\mathbb{R} \subseteq \forall\exists_<\mathbb{R}$ follows by definition. ∎

6.4.2 Tools from Real Algebraic Geometry

We review two sophisticated results from real algebraic geometry. The first is singly exponential *quantifier elimination*, i.e., an algorithm to transform a first-order formula in prenex normal form into an equivalent quantifier-free formula. The currently most efficient versions are presented in a series of articles by Renegar [126, 127, 128].

Theorem 6.11 ([16, Theorem 14.16]). *Let $X_1, \ldots, X_k, Y$ be blocks of real variables where X_i has length n_i, Y has length m, formula $\varphi(X_1, \ldots, X_k, Y) \in \mathrm{QFF}$ has s atoms and $Q_i \in \{\exists, \forall\}$ is a quantifier for all $i \in \{1, \ldots, k\}$. Furthermore, let d be the maximum total degree of any polynomial of $\varphi(X_1, \ldots, X_k, Y)$. Then for any formula*

$$\Phi(Y) \;:\equiv\; Q_1 X_1 \in \mathbb{R}^{n_1} \ldots Q_k X_k \in \mathbb{R}^{n_k} \colon \varphi(X_1, \ldots, X_k, Y)$$

there is an equivalent quantifier-free formula of size at most

$$s^{(n_1+1)\cdots(n_k+1)(m+1)} d^{O(n_1)\cdots O(n_k)O(m)}.$$

Throughout this chapter we use the following corollary of Theorem 6.11 which is already stated by D'Costa, Lefaucheux, Neumann, Ouaknine and Worrel [47]. It is weaker but easier to work with.

Corollary 6.12 ([47]). *Let $\Phi(Y)$ be as in Theorem 6.11, and let $L = |\varphi(X_1, \ldots, X_k, Y)|$ be the length of its matrix. Then for some constant $\alpha \in \mathbb{R}$ independent of Φ, there exists an equivalent quantifier-free formula of size at most*

$$L^{\alpha^{k+1} \cdot (n_1+1) \cdot \ldots \cdot (n_k+1) \cdot (m+1)}.$$

The second tool is the so-called *Ball Theorem*. It guarantees that every non-empty semi-algebraic set contains an element not too far from the origin. The Ball Theorem was first proven by Vorob'ev [152] as well as by Grigor'ev and Vorobjov [77][2]. Explicit bounds on the distance are given by Basu and Roy [17]. We use a version from Schaefer and Štefankovič [138]:

Theorem 6.13 (*Ball Theorem* [138, Corollary 3.1]). *Every non-empty semi-algebraic set in $\mathbb{R}^n$ of complexity at most $L \geq 4$ contains a point of distance at most $2^{L^{8n}}$ from the origin.*

Recall that for any quantifier-free formula $\varphi(X)$ with free variables $X \in \mathbb{R}^n$, the set $S := \{x \in \mathbb{R}^n \mid \varphi(X)\}$ is semi-algebraic. It follows from Theorem 6.13 that $\exists X \in \mathbb{R}^n \colon \varphi(X)$ is equivalent to $\exists X \in [-C, C]^n \colon \varphi(X)$ where $C := 2^{L^{8n}}$. This is how we are going to use Theorem 6.13.

Below, we apply Corollary 6.12 and Theorem 6.13 to prove a lemma that is stated in [47, Lemma 14] for two quantifiers only. We are interested in the general case with k quantifiers. The proof goes along the same lines as the proof for two quantifiers. A qualitatively equivalent statement also follows from (the proof of) [34, Theorem 9.2].

Lemma 6.14. *Let $X_1, \dots, X_k$ be blocks of variables where X_i has length $n_i \geq 1$ and let $\varphi(\varepsilon, X_1, \dots, X_k) \in \mathsf{QFF}$ with $L := |\varphi|$. For $\mathrm{Q}_i \in \{\exists, \forall\}$ consider the set*

$$S := \{\varepsilon > 0 \mid \mathrm{Q}_1 X_1 \in \mathbb{R}^{n_1} \dots \mathrm{Q}_k X_k \in \mathbb{R}^{n_k} \colon \varphi(\varepsilon, X_1, \dots, X_k)\}.$$

If S is non-empty, then there is an $\varepsilon^* \in S$ such that for some constant $\beta \in \mathbb{R}$ we have

$$\varepsilon^* \geq 2^{-L^{\beta^{k+2}(n_1+1)\cdots(n_k+1)}}.$$

Proof. Let $\Phi(\varepsilon)$ be the subformula $\mathrm{Q}_1 X_1 \in \mathbb{R}^{n_1} \dots \mathrm{Q}_k X_k \in \mathbb{R}^{n_k} \colon \varphi(\varepsilon, X_1, \dots, X_k)$. By Corollary 6.12, there is a constant $\alpha \in \mathbb{R}$ and a quantifier-free formula $\phi(\varepsilon)$ of length

$$|\phi(\varepsilon)| \leq L^{2\alpha^{k+1}(n_1+1)\cdots(n_k+1)}$$

such that $S = \{\varepsilon > 0 \mid \phi(\varepsilon)\}$. Let d be the maximum degree of any polynomial in ϕ and δ be a new variable. We replace each atom $P(\varepsilon) \circ 0$ (where $\circ \in \{<, \leq\}$) of ϕ by $\delta^d P\left(\frac{1}{\delta}\right) \circ 0$ and denote the new formula by $\psi(\delta)$. Then for $\varepsilon > 0$ it follows that $\phi(\varepsilon)$ is true if and only if for $\delta = \frac{1}{\varepsilon}$ the sentence $\psi(\delta)$ is true. We get

$$\exists \varepsilon > 0 \colon \phi(\varepsilon) \;\equiv\; \exists \delta > 0 \colon \psi(\delta).$$

To obtain an upper bound on $|\psi(\delta)|$, note that the length of each atom increases by a factor of at most d, which is obviously at most $|\phi(\varepsilon)|$. We conclude that

$$|\psi(\delta)| \leq |\phi(\varepsilon)| \cdot d \leq |\phi(\varepsilon)|^2.$$

2 Vorob'ev and Vorobjov are two different transcriptions of the same name from the Cyrillic to the Latin alphabet.

If S is non-empty, then $\exists \delta > 0\colon \psi(\delta)$ is true. By Theorem 6.13, there is some δ^* such that $\psi(\delta^*)$ is true and $\delta^* \leq 2^{|\psi(\delta)|^8}$. We get that

$$\delta^* \leq 2^{|\psi(\delta)|^8} \leq 2^{|\phi(\varepsilon)|^{16}} \leq 2^{L^{32\alpha^{k+1}(n_1+1)\cdots(n_k+1)}} \leq 2^{L^{\beta^{k+2}(n_1+1)\cdots(n_k+1)}},$$

where $\beta := \max\{32, \alpha\}$ is a real constant independent of the input. The result follows for $\varepsilon^* := \frac{1}{\delta^*}$. ■

Lastly, let us state a lemma that we will use frequently to scale (some dimensions of) semi-algebraic sets. For some $N \in \mathbb{N}$ and $N+1$ variables $U = (U_0, \ldots, U_N)$, consider the following formula:

$$\chi(U) \;:\equiv\; (2 \cdot U_0 = 1) \wedge \bigwedge_{i=1}^{N} (U_i = U_{i-1}^2) \tag{6.2}$$

Lemma 6.15. *For $u \in [-1,1]^{N+1}$ formula $\chi(u)$ is true if and only if $u_i = 2^{-2^i}$.*

Proof. The if-part is trivial. The only-if-part follows from a simple induction. ■

6.4.3 Bounding the Ranges of the Quantifiers

In the following, we show how to restrict the ranges of the variables. This was first done by D'Costa et al. [47] in the context of their $\exists\forall_{\leq}\mathbb{R}$-complete escape problem. Lemmas 6.16 and 6.18 below are stated in our setting, but their proofs directly follow the ideas in [47].

As a first step, we restrict the universally quantified variables. This works for general UETR instances without any further requirements on the formula.

Lemma 6.16. *Let X and Y be blocks of variables with $n := |X|$ and $m := |Y|$, let $\varphi(X,Y) \in$ QFF, and let*

$$\Phi \;:\equiv\; \forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m \colon \varphi(X,Y).$$

Then there exists an integer $N \leq \operatorname{poly}(n, m, |\varphi|)$, such that for $C := 2^{2^N}$ the sentence

$$\Psi \;:\equiv\; \forall X \in [-C, C]^n \,.\, \exists Y \in \mathbb{R}^m \colon \varphi(X,Y)$$

is equivalent to Φ.

Proof. We rewrite Φ via a double negation to get

$$\Phi \;\equiv\; \neg\big(\exists X \in \mathbb{R}^n \,.\, \forall Y \in \mathbb{R}^m \colon \neg\varphi(X,Y)\big)$$

and let $L := |\neg\varphi|$ denote the length of the quantifier-free part. By Corollary 6.12 there is a constant $\alpha \in \mathbb{R}$ and a quantifier-free formula $\psi(X)$ such that Φ is equivalent to $\neg\big(\exists X \in \mathbb{R}^n \colon \psi(X)\big)$, where

$$|\psi| \leq L^{\alpha^2(n+1)(m+1)} = 2^{\alpha^2 \log(L)(n+1)(m+1)}.$$

Assuming that $\{x \in \mathbb{R}^n \mid \psi(x)\}$ is non-empty, Theorem 6.13 yields that it contains a point of distance at most

$$D := 2^{|\psi|^{8n}} \leq 2^{(2^{\alpha^2 \log(L)(n+1)(m+1)})^{8n}} = 2^{2^{8\alpha^2 \log(L) n(n+1)(m+1)}}$$

from the origin. Let $N = \lceil 8\alpha^2 \log(L) n(n+1)(m+1) \rceil \leq \text{poly}(n, m, \log(L)) \leq \text{poly}(n, m, |\varphi|)$. Then it holds that $C := 2^{2^N} \geq D$. It follows, that

$$\begin{aligned} \neg\Phi &\equiv \neg(\exists X \in \mathbb{R}^n \colon \psi(X)) \\ &\equiv \neg(\exists X \in [-C, C]^n \colon \psi(X)) \\ &\equiv \neg(\exists X \in [-C, C]^n \,.\, \forall Y \in \mathbb{R}^m \colon \varphi(X, Y)) \\ &\equiv \neg\Psi \end{aligned}$$

and therefore $\Phi \equiv \Psi$. ■

In a second step, we additionally restrict the existentially quantified variables. Before we do so, we show that this may be impossible in general (without changing its true/false value):

Example 6.17. Consider the following sentence:

$$\forall X \in \mathbb{R} \,.\, \exists Y \in \mathbb{R} \colon X = 0 \vee XY = 1$$

It is clearly true, as either $X = 0$ or, if $X \neq 0$, we may choose Y to equal $1/X$. This remains true if we restrict the range of X, e.g., to $[-1, 1]$. However, note that the absolute value of $1/X$ may be arbitrarily large, even if $X \in [-1, 1]$. Consequently, we cannot restrict the range of Y to any interval. ⌟

In the following, we show how the ranges can be restricted in case of $\forall$-strict formulas. Requiring the formula to be $\forall$-strict is a slight generalization of the corresponding statement shown in [47] (where the formula is required to be strict). This more general case is crucial for our proofs in Sections 6.8 and 6.9.

Lemma 6.18. *Let X and Y be blocks of variables with $n := |X|$ and $m := |Y|$ and $\varphi(X, Y) \in \text{QFF}$. Furthermore, let N be an integer and $C := 2^{2^N}$. Then for a $\forall$-strict sentence*

$$\Phi :\equiv \forall X \in [-C, C]^n \,.\, \exists Y \in \mathbb{R}^m \colon \varphi(X, Y)$$

there is an integer $M \leq \text{poly}(n, m, N, |\varphi|)$ such that for $D := 2^{2^M}$ the sentence

$$\Psi :\equiv \forall X \in [-C, C]^n \,.\, \exists Y \in [-D, D]^m \colon \varphi(X, Y)$$

is equivalent to Φ.

Proof. If Φ is false, then there exists an $x \in [-C, C]^n$ such that no $y \in \mathbb{R}^m$ satisfies $\varphi(x, y)$. In particular, no $y \in [-D, D]^m \subseteq \mathbb{R}^m$ satisfies $\varphi(x, y)$. Thus, Ψ is also false.

In the remainder of the proof, we assume that Φ is true. The proof consists of two steps. First, we show that an upper bound D for the existentially quantified variables indeed exists. In a second step, we use the Ball Theorem to actually compute an upper bound for D.

For the first step, let $S := [-C, C]^n$. Sentence Φ being true implies that for each $x \in S$ there is a $y(x) \in \mathbb{R}^m$ such that $\varphi(x, y(x))$ is true. Even stronger, as φ is $\forall$-strict, we even find an $\varepsilon(x) > 0$, such that for all $\widetilde{x} \in S$ with $\|x - \widetilde{x}\| < \varepsilon(x)$ we get that $\varphi(\widetilde{x}, y(x))$ is true. Recall that we denote by $B_n(x, r) = \{\widetilde{x} \in \mathbb{R}^n \mid \|\widetilde{x} - x\| < r\}$ the open ball with center x and radius r in $\mathbb{R}^n$. Then $\{B_n(x, \varepsilon(x)) \mid x \in S\}$ is an open cover of S. As S is compact, it has a finite subcover $B_n(x_1, \varepsilon(x_1)), \dots, B_n(x_s, \varepsilon(x_s))$. Now, given some $x \in S$, there is an $i \in \{1, \dots, s\}$, such that $\varphi(x, y(x_i))$ is true. We define $y_{\max} := \max\{\|y(x_1)\|_\infty, \dots, \|y(x_s)\|_\infty\}$. Then, for all $D \geq y_{\max}$ formula Φ implies

$$\exists D > 0 \,.\, \forall X \in [-C, C]^n \,.\, \exists Y \in \mathbb{R}^m \colon \bigwedge_{i=1}^{m} |Y_i| \leq D \wedge \varphi(X, Y),$$

proving the existence of an upper bound D for the existentially quantified variables.

The second step is to obtain a bound on D. We first need to construct $C = 2^{2^N}$ inside the formula. For this, let $U = (U_0, \dots, U_N)$ be $N+1$ new variables and $\chi(U)$ be the formula (6.2). Recall that, by Lemma 6.15, $\chi(u)$ is true if and only if $u_i = 2^{-2^i}$. Furthermore, each U_i can be trivially restricted to be in $[-1, 1]$. Using $\chi(U)$, we can rewrite above sentence as

$$\exists D > 0 \,.\, \forall X \in \mathbb{R}^n, U \in [-1, 1]^{N+1} \,.\, \exists Y \in \mathbb{R}^m \colon$$
$$\Big(\chi(U) \wedge \bigwedge_{i=1}^{n} |X_i| U_N \leq 1\Big) \implies \bigwedge_{i=1}^{m} |Y_i| \leq D \wedge \varphi(X, Y).$$

From here on, bounding D is a straightforward application of the Ball Theorem: Let L be the length of the subformula behind the existential quantification of D. By Corollary 6.12 there is a constant $\alpha \in \mathbb{R}$ and a quantifier-free formula $\psi(D)$, such that above sentence is equivalent to $\exists D > 0 \colon \psi(D)$ where $|\psi| \leq L^{2\alpha^3(n+N+2)(m+1)}$. Then Theorem 6.13 yields the following upper bound for D:

$$D \leq 2^{|\psi(D)|} \leq 2^{(L^{2\alpha^3(n+N+2)(m+1)})^{8n}} = 2^{2^{16\alpha^3 \log(L) n(n+N+2)(m+1)}}.$$

Lastly, we choose $M \leq \lceil 16\alpha^3 \log(L) n(n + N + 2)(m + 1) \rceil$ to be the smallest integer such that $D \leq 2^{2^M}$. Note that $M \leq \operatorname{poly}(n, m, N, \log L) \leq \operatorname{poly}(n, m, N, |\varphi|)$ as required. ■

6.5 Counterexamples of Strict-UETR

Let us recall the definition of *counterexamples* which we motivated in Section 6.2.1 above. Given a sentence $\Phi :\equiv \forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m \colon \varphi(X, Y)$, we call

$$\bot(\Phi) := \{x \in \mathbb{R}^n \mid \forall Y \in \mathbb{R}^m \colon \neg\varphi(x, Y)\}$$

its counterexamples. The counterexamples of Φ are exactly the values $x \in \mathbb{R}^n$ for which there is no $y \in \mathbb{R}^m$ such that $\varphi(x, y)$ is true. The main result of this section, Theorem 6.20, is that we can transform a Strict-UETR instance Φ into an equivalent formula Ψ for which $\bot(\Psi)$ is either empty or contains an open ball. The main tools for this are the range restrictions from Section 6.4.3 and the following lemma from calculus.

Lemma 6.19. *Let S and T be compact sets and $f \colon S \times T \to \mathbb{R}$ be a continuous function. Then $g \colon S \to \mathbb{R}, x \mapsto \min_{y \in T}\{f(x, y)\}$ is continuous over S.*

Proof. We first observe that the compactness of S and T implies that their Cartesian product $S \times T$ is compact as well. Thus, because f is continuous on $S \times T$, it is even uniformly continuous, i.e., for every $\varepsilon > 0$ there is a $\delta > 0$, such that for every two points $(x, y), (\widetilde{x}, \widetilde{y}) \in S \times T$ we have $|f(x, y) - f(\widetilde{x}, \widetilde{y})| < \varepsilon$ whenever $\|(x, y) - (\widetilde{x}, \widetilde{y})\| < \delta$.

Now consider $x, \widetilde{x} \in S$ with $\|x - \widetilde{x}\| < \delta$. We have

$$\begin{aligned} g(\widetilde{x}) - g(x) &= g(\widetilde{x}) - f(x, y) && \text{(for some } y \in T\text{)} \\ &< g(\widetilde{x}) - (f(\widetilde{x}, y) - \varepsilon) && \text{(by uniform continuity)} \\ &\le g(\widetilde{x}) - (g(\widetilde{x}) - \varepsilon) && \text{(by definition of } g\text{)} \\ &= \varepsilon. \end{aligned}$$

By exchanging the role of x and $\widetilde{x}$, we get $g(x) - g(\widetilde{x}) < \varepsilon$. Combined, we obtain that $|g(x) - g(\widetilde{x})| < \varepsilon$ for all $x, \widetilde{x} \in S$ with $\|x - \widetilde{x}\| < \delta$. It follows that g is continuous on S. ■

With these tools at hand, we are able to tackle the main result of this section:

Theorem 6.20. *Given a* Strict-UETR *instance Φ, we can construct in polynomial time an equivalent* UETR *instance Ψ of the form*

$$\forall X \in [-1, 1]^n \,.\, \exists Y \in [-1, 1]^\ell \colon \psi(X, Y),$$

such that $\bot(\Psi)$ is either empty or contains an n-dimensional open ball.

Before proving Theorem 6.20 in full generality, let us illustrate the key idea with an example for a very simple Strict-UETR instance Φ having only a single atom. This already shows how bounding the ranges of the quantifiers expands the set of counterexamples while not yet requiring the technical calculus arguments that are required to handle more than one atom.

Example 6.21 (Single Atom)**.** We consider the Strict-UETR instance

$$\Phi \;:\equiv\; \forall X \in \mathbb{R} \,.\, \exists Y \in \mathbb{R} \colon XY < 0.$$

Note that Φ is a no-instance with $\bot(\Phi) = \{0\}$, i.e., there is only a single counterexample. We start by introducing a new existentially quantified variable Z and rewrite the single atom $P < 0$ as $\exists Z \in \mathbb{R} \colon Z^2 \cdot P + 1 < 0$ to get

$$\Phi_1 \;:\equiv\; \forall X \in \mathbb{R} \,.\, \exists Y \in \mathbb{R}, Z \in \mathbb{R} \colon Z^2 \cdot XY + 1 < 0.$$

It holds that $\Phi \equiv \Phi_1$ and even stronger that $\bot(\Phi) = \bot(\Phi_1) = \{0\}$. While this transformation may look innocent, it is very powerful. The key insight is as follows: Once we bound the range of Z to some compact interval $[-D, D]$, this requires $XY < -\frac{1}{D^2}$ in order to satisfy the atom. This is stronger than just requiring $XY < 0$ and expands the set of counterexamples. To see this in action, we apply Lemmas 6.16 and 6.18 and get integer constants C and D such that Φ_1 is equivalent to

$$\Phi_2 \;:\equiv\; \forall X \in [-C, C] \,.\, \exists Y \in [-D, D], Z \in [-D, D] \colon Z^2 \cdot XY + 1 < 0$$

and, furthermore, $\bot(\Phi_2) \supseteq \bot(\Phi_1) \cap [-C, C]$. It remains to argue that $\bot(\Phi_2)$ indeed contains an open ball.

Consider some $x \in (0, C]$. For which $y, z \in [-D, D]$ does $z^2 \cdot xy + 1 < 0$ hold? Neither of y, z may be zero and y must be negative (as z^2 is always positive). Then, $z^2 \cdot xy + 1 < 0$ is equivalent to $xy < -\frac{1}{z^2}$. We get

$$-xD \le xy < -\frac{1}{z^2} \le -\frac{1}{D^2}.$$

It follows that $-xD < -\frac{1}{D^2}$ and thus $x > \frac{1}{D^3}$. This means that $\left(0, \frac{1}{D^3}\right] \subseteq \bot(\Phi_2)$ and this interval contains an open ball. ⌟

Proof of Theorem 6.20. The proof is split into two parts. First, we construct Ψ from Φ. Afterward, we show that $\bot(\Psi)$ has the desired properties.

Construction of Ψ Each atom of the sentence

$$\Phi \;:\equiv\; \forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m \colon \varphi_<(X, Y)$$

with $\varphi_< \in \mathsf{QFF}_<$ is of the form $P < 0$, where $P \in \mathbb{Z}[X, Y]$ is a polynomial. Individually, each atom $P < 0$ is equivalent to the formula $\exists Z \in \mathbb{R} \colon Z^2 P + 1 < 0$. Note that the new variable Z is monotone: If the formula is true for one particular choice of Z, it is also true for all values with smaller absolute value. Thus, a single new variable Z can be used to rewrite all atoms of Φ to obtain an equivalent sentence

$$\Phi_1 \;:\equiv\; \forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m, Z \in \mathbb{R} \colon \varphi'_<(X, Y, Z)$$

in prenex normal form. Here, $\varphi'_<$ is obtained from $\varphi_<$ by above transformation, in particular, they have exactly the same logical structure (their only difference lies in the transformed atoms). The length increases only by a constant amount per atom, so $|\Phi_1|$ is linear in $|\Phi|$. Furthermore, we have $\bot(\Phi_1) = \bot(\Phi)$ by construction.

We can apply Lemma 6.16 to restrict the ranges of the universally quantified variables and obtain an integer $N \leq \mathrm{poly}(|\Phi_1|)$ such that for $C := 2^{2^N}$ the sentence Φ_1 is equivalent to

$$\Phi_2 \;:\equiv\; \forall X \in [-C, C]^n \,.\, \exists Y \in \mathbb{R}^m, Z \in \mathbb{R} \colon \varphi'_{<}(X, Y, Z).$$

It holds that $\bot(\Phi_2) \subseteq \bot(\Phi_1)$ and that $\bot(\Phi_2) = \bot(\Phi_1) \cap [-C, C]^n$.

Each atom in Φ_2 is a strict inequality. Thus, we can use Lemma 6.18 to also restrict the ranges of the existentially quantified variables. We obtain another integer $M \leq \mathrm{poly}(N, |\Phi_2|)$ such that for $D := 2^{2^M}$ above sentence Φ_2 is equivalent to

$$\Phi_3 \;:\equiv\; \forall X \in [-C, C]^n \,.\, \exists Y \in [-D, D]^m, Z \in [-D, D] \colon \varphi'_{<}(X, Y, Z).$$

Regarding the counterexamples, we have $\bot(\Phi_3) \supseteq \bot(\Phi_2)$.

The last step is to scale the ranges over which the variables are quantified to the interval $[-1, 1]$. To this end, define $K := \max\{N, M\}$, let $U := (U_0, \ldots, U_K)$ be $K + 1$ new variables, and let $\chi(U)$ be formula (6.2). Recall that by Lemma 6.15, for $u \in [-1, 1]^{K+1}$ we have $\chi(u)$ if and only if $u_i = 2^{-2^i}$. Let d be the maximum degree of any polynomial in $\varphi'_{<}$. We define

$$\begin{aligned}\Psi \;:\equiv\; &\forall X \in [-1, 1]^n \,.\, \exists Y \in [-1, 1]^m, Z \in [-1, 1], U \in [-1, 1]^K \colon \\ &\quad \chi(U) \wedge U_K^d \cdot \varphi'_{<}\Big(\frac{X}{U_N}, \frac{Y}{U_M}, \frac{Z}{U_M}\Big),\end{aligned}$$

where X/U_N expresses that every X_i is replaced by X_i/U_N (and likewise for Y/U_M). The multiplication of $\varphi'_{<}$ with U_K^d denotes that both sides of each atom are multiplied by U_K^d. This restores the requirement that each atom is a polynomial inequality. (Strictly speaking, the obtained formula contains the divisions by U_N and U_M. However, because $K \geq N$, we can replace any $U_K \cdot (X_i/U_N)$ by $U_{K-N}X_i$, which does not contain divisions. Likewise, we handle Y/U_M.) As this last step just scales variables, we conclude that Ψ is equivalent to Φ_3 and therefore also to Φ. Furthermore, Ψ has the form required by the statement of the theorem.

Properties of $\bot(\Psi)$ It remains to show that $\bot(\Psi)$ is either empty (if Ψ is true) or contains an n-dimensional open ball (if Ψ is false). Note that scaling variables (as done to get from Φ_3 to Ψ) also scales the counterexamples; thus, an open ball in $\bot(\Phi_3)$ is mapped to an open ball in $\bot(\Psi)$. It therefore suffices to prove that $\bot(\Phi_3)$ contains an open ball. As Φ_3 is the simpler formula, we analyze $\bot(\Phi_3)$ below.

By construction, Φ and Φ_3 are equivalent. Thus, Φ is true if and only Φ_3 is true. In particular, $\bot(\Phi) = \emptyset$ implies that $\bot(\Phi_3) = \emptyset$. From now on, we assume that Φ_3 is false. Let $x^* \in \bot(\Phi_2)$ be a counterexample of Φ_2, fixed until the end of the proof. We know that $x^* \in [-C, C]^n$ (by Lemma 6.16) and also that $x^* \in \bot(\Phi_3)$ (by construction of Φ_3).

We prove below that for some $r > 0$, all $x \in [-C, C]^n$ with $\|x^* - x\| < r$ are counterexamples of Φ_3 as well. If $B_n(x^*, r) \subseteq [-C, C]^n$, then x^* is the center of our desired open ball of counterexamples. If $B_n(x^*, r)$ is not completely contained in $[-C, C]^n$, then any $x' \in B_n(x^*, r) \cap (-C, C)^n$ can be used instead as the center of a smaller (but still open) ball of counterexamples.

To simplify the following argument, we further assume that $\varphi_<$ (in Φ) is in disjunctive normal form (DNF), i.e., a disjunction of conjunctions of atoms. By construction, $\varphi'_<$ is then also in DNF and has exactly the same logical structure. This is justified, as the set of counterexamples is invariant under applications of the distributive law on the matrix. Thus, $\varphi_<$ and $\varphi'_<$ have exactly the same counterexamples as their DNFs. Assuming that $\varphi_<$ is in DNF allows us to consider the conjunctive clauses independently, as our counterexample x^* is indeed a counterexample for each of them. We are going to prove that a false clause remains false, even for x sufficiently close to x^*. As this holds for clauses, and all clauses are false, we may just consider a single clause from now on.

Let $C(X, Y) := \left(\bigwedge_{i=1}^{s} P_i(X, Y) < 0\right)$ be an arbitrary conjunctive clauses of (the DNF of) $\varphi_<(X, Y)$. For our fixed counterexample $x^* \in \bot(\Phi_2)$, every conjunctive clause of $\varphi_<(x^*, Y)$ evaluates to false independently of Y. We get that for all $y \in \mathbb{R}^m$ and thus in particular for all $y \in [-D, D]^m$ that $C(x^*, y)$ is false and that

$$\bigvee_{i=1}^{s} \left(P_i(x^*, y) \geq 0\right) \tag{6.3}$$

is true. Let us point out that for different choices of $y \in [-D, D]^m$, different subsets of the polynomials $P_i(x^*, y)$ may evaluate to non-negative values. We only know that for every y, at least one of the polynomials is non-negative (here it is important that $\varphi_<(X, Y)$ is in DNF). To overcome this, we combine the polynomials into a single function.

Each of the $P_i \in \mathbb{Z}[X, Y]$, $i \in \{1, \ldots, s\}$, is a polynomial and thus continuous. The maximum over a finite number of continuous functions is again continuous, so

$$\begin{aligned} P_{\max}\colon\ [-C, C]^n \times [-D, D]^m &\to \mathbb{R} \\ (x, y) &\mapsto \max_{i=1,\ldots,s} \{P_i(x, y)\} \end{aligned}$$

is continuous. For our fixed counterexample x^* and all $y \in [-D, D]^m$ it follows by (6.3) that

$$P_{\max}(x^*, y) \geq 0. \tag{6.4}$$

We want to argue about the value of $P_{\max}$ at points x in a small neighborhood around x^*. To this end, we consider the function

$$\begin{aligned} P^*\colon\ [-C, C]^n &\to \mathbb{R} \\ x &\mapsto \min_{y \in [-D, D]^m} P_{\max}(x, y), \end{aligned}$$

which eliminates the dependency on y. The sets $[-C, C]^n$ and $[-D, D]^m$ are compact, so, by Lemma 6.19, the function P^* is again continuous. From (6.4), we get for our fixed counterexample x^* that

$$P^*(x^*) \geq 0.$$

By the continuity of P^*, for every $\varepsilon > 0$ there exists a $\delta > 0$ such that for all $x \in [-C, C]^n$ with $\|x^* - x\| < \delta$ we have $|P^*(x) - P^*(x^*)| < \varepsilon$. We choose $\varepsilon < 1/D^2$ and conclude that for a sufficiently small $\delta > 0$ and all $x \in [-C, C]^n$ with $\|x^* - x\| < \delta$, it holds that

$$P^*(x) > -\frac{1}{D^2}.$$

Fix one such x. Going backwards through our chain of defined functions, it follows for all $y \in [-D, D]^m$ that $P_{\max}(x, y) > -1/D^2$ and moreover that

$$\bigvee_{i=1}^{s} P_i(x, y) > -\frac{1}{D^2}. \tag{6.5}$$

Now also fix an arbitrary $y \in [-D, D]^m$ and choose $j \in \{1, \ldots, s\}$ such that $P_j(x, y) > -1/D^2$. Because $A := (P_j(X, Y) < 0)$ is an atom in the DNF of $\varphi_<(X, Y)$, there is a corresponding atom $A' := (Z^2 P_j(X, Y) + 1 < 0)$ in the DNF of $\varphi'_<(X, Y, Z)$. Note that A' can never be true for $Z = 0$. For $Z \neq 0$, the atom A' can be rewritten as $P_j(X, Y) < -1/Z^2$. From $Z \in [-D, D]$, we get that $Z^2 \leq D^2$ and therefore our considered atom A' can only ever be satisfied, if $P_j(X, Y) < -1/Z^2 \leq -1/D^2$. However, by the choice of j and (6.5), we know that $P_j(x, y) > -1/D^2$. Thus, because y was fixed arbitrarily, x must be a counterexample of Φ_3. Additionally, because $x \in \mathbb{R}^n$ with $\|x^* - x\| < \delta$ was chosen arbitrarily, we conclude that all such x are counterexamples of Φ_3, forming an n-dimensional open ball. ■

6.6 $\forall\exists_<\mathbb{R}$-Hardness of Hausdorff

We are now able to prove $\forall\exists_<\mathbb{R}$-hardness.

Theorem 6.22. Hausdorff *and* DirectedHausdorff *are* $\forall\exists_<\mathbb{R}$*-hard.*

Proof. Let Φ be an instance of Strict-UETR. We give a polynomial-time many-one reduction to an equivalent Hausdorff instance. The proof is split into two parts: In the first part, we transform Φ into an equivalent UETR instance Ψ whose counterexamples $\perp(\Psi)$ contain an open ball (if there are any). Sentence Ψ is then used to construct a Hausdorff instance (A, B, t). The second part proves that Φ and (A, B, t) are indeed equivalent.

Constructing Hausdorff instance (A, B, t) The first step is to apply Theorem 6.20 to Φ and to obtain in polynomial time an equivalent UETR instance

$$\Psi' :\equiv \forall X \in [-1,1]^n \,.\, \exists Y \in [-1,1]^m \colon \psi'(X, Y),$$

where $\psi' \in$ QFF. We know that either $\bot(\Psi') = \emptyset$ (if Ψ' is true) or that $\bot(\Psi')$ contains an n-dimensional open ball (if Ψ' is false). Based on Ψ' we define

$$\begin{aligned} \psi(X, Y) &:\equiv \psi'(X, Y) \lor \bigwedge_{i=1}^{n} X_i = 0 \quad \text{and} \\ \Psi &:\equiv \forall X \in [-1,1]^n \,.\, \exists Y \in [-1,1]^m \colon \psi(X, Y). \end{aligned}$$

Note that Ψ' and Ψ are equivalent: If Ψ' is true, then obviously Ψ is also true because the new condition is added using a logical "or". If Ψ' is false, then $\bot(\Psi) = \bot(\Psi') \setminus \{\vec{0}\}$. Since $\bot(\Psi')$ contains an open ball, it follows that $\bot(\Psi)$ also contains an open ball. The key idea behind the definition of Ψ is that $\bot(\Psi)$ is guaranteed to be a strict subset of $\mathbb{R}^n$. This will be important below to make sure that set A is non-empty.

If Ψ is false, then there is an $x \in \bot(\Psi) \subseteq [-1,1]^n$, such that $B_n(x, r) \subseteq \bot(\Psi)$ for some $r > 0$. Expressed as a sentence in the first-order theory of the reals, we get

$$\exists r > 0, X \in [-1,1]^n \,.\, \forall \widetilde{X} \in [-1,1]^n, Y \in [-1,1]^m \colon \|X - \widetilde{X}\|^2 < r^2 \implies \neg\psi(\widetilde{X}, Y).$$

Let us denote by L the length of the matrix of this sentence. We see that L is clearly polynomial in $|\Psi|$ which by construction is polynomial in $|\Phi|$. Above sentence has the form required by Lemma 6.14, and we get a constant $\beta \in \mathbb{R}$ such that the following lower bound for r can be assumed:

$$r \geq 2^{-L^{\beta^4 (n+1)(n+m+1)}} \tag{6.6}$$

Let $N \leq \lceil \beta^4 (n+1)(n+m+1) \rceil$ be the smallest integer, such that

$$r \cdot 2^{2^N} > m. \tag{6.7}$$

By Equation (6.6), it holds that $N \leq \text{poly}(n, m, \log(L)) \leq \text{poly}(|\Phi|)$. Define $C := 2^{2^N}$.

The idea now is to scale the universally quantified variables by a factor of C (so that they are from the interval $[-C, C]$). This then also scales the set of counterexamples $\bot(\Psi)$ and in particular the radius of the open ball in $\bot(\Psi)$ by C. Let $U = (U_0, \ldots, U_N) \in [-1,1]$ be $N+1$ new variables and $\chi(U)$ be formula (6.2). Recall that by Lemma 6.15, for $u \in [-1,1]^{N+1}$ we have $\chi(u)$ if and only if $u_i = 2^{-2^i}$. With this, we define

$$\phi(X, Y, U) :\equiv \chi(U) \land \psi(U_N X, Y),$$

where $U_N X$ means that every occurrence of X_i in ψ is replaced by $U_N X_i$. Finally, we are ready to define our desired Hausdorff instance:

$$\begin{aligned} A &:= \left\{ (x, y, u) \in [-C, C]^n \times [-1,1]^m \times \{2^{-2^0}\} \times \ldots \times \{2^{-2^N}\} \mid \phi(x, y, u) \right\} \\ B &:= [-C, C]^n \times \{0\}^m \times \{2^{-2^0}\} \times \ldots \times \{2^{-2^N}\} \\ t &:= m \end{aligned}$$

Note that this is well-defined, because both sets A and B are non-empty. While this is trivial for B, it holds for A by our construction of ϕ from Φ: It always holds that

$$\emptyset \neq \{0\}^n \times [-1,1]^m \times \{2^{-2^0}\} \times \ldots \times \{2^{-2^N}\} \subseteq A.$$

Equivalence of Φ and (A, B, t) To see that Φ and (A, B, t) are equivalent, assume first that Φ is true. For every point $a := (x, y, u) \in A$, it holds that $u_i = 2^{-2^i}$ as this is necessary to satisfy $\chi(u)$. Consider the point $b := (x, \{0\}^n, u) \in B$. We get that

$$\|a - b\| = \|(x, y, u) - (x, \{0\}^m, u)\| = \|y - \vec{0}\| \leq \sqrt{\textstyle\sum_{i=1}^m 1} = \sqrt{m} \leq m = t.$$

As a was chosen arbitrarily, we get an upper bound for the directed Hausdorff distance $\vec{d}_{\mathrm{H}}(A, B) \leq t$. On the other hand, let $b := (x, \{0\}^m, u)$ be an arbitrary point in B. Because Φ (and therefore Ψ) is true, there is some $y \in [-1, 1]^m$ such that there is a point $a := (x, y, u) \in A$. By the same calculation as above, we get $\vec{d}_{\mathrm{H}}(B, A) \leq t$ and thus

$$d_{\mathrm{H}}(A, B) \leq t. \tag{6.8}$$

Now assume that Φ and Ψ are false. Then there is some $x \in [-1, 1]^n$ such that there is an n-dimensional open ball $B_n(x, r) \subseteq \bot(\Psi)$ (the r here is the one from (6.6)). By the construction of A, this corresponds to an open ball of radius $C \cdot r$ in $\mathbb{R}^n \setminus A$. Let x^* be the center of this open ball in $\mathbb{R}^n \setminus A$. Then for $b := (x^*, \{0\}^m, u) \in B$ all points $a \in A$ have

$$\|a - b\| \geq C \cdot r > m = t.$$

It follows that

$$d_{\mathrm{H}}(A, B) \geq \vec{d}_{\mathrm{H}}(B, A) \geq \|a - b\| > t. \tag{6.9}$$

Equations (6.8) and (6.9) prove that $d_{\mathrm{H}}(A, B) \leq t$ (and also $\vec{d}_{\mathrm{H}}(B, A) \leq t$) if and only if Φ is true. ■

In the proof of Theorem 6.22, we could choose $N' := N + 1$ instead of N in Equation (6.7). Then in the case that Φ is false, the Hausdorff distance $d_{\mathrm{H}}(A, B)$ is at least

$$2^{2^{N+1}} r > 2^{2^{N+1} - 2^N} m = 2^{2^N} m = 2^{2^N} t.$$

Note that the number of free variables in the formulas describing the resulting sets A and B equals $n + m + N' + 1 = \Theta(N)$. We created a gap of size $2^{2^{\Theta(N)}}$. This implies the following inapproximability result.

Corollary 6.7. *Let A and B be two semi-algebraic sets in $\mathbb{R}^n$ and $f(n) = 2^{2^{o(n)}}$. There is no polynomial-time $f(n)$-approximation algorithm to compute $d_{\mathrm{H}}(A, B)$, unless* $\mathsf{P} = \forall\exists_{<}\mathbb{R}$.

Another interesting observation is that we can restrict the sets A and B to be described by *structurally* simple formulas. We can express both formulas as a conjunction of (down to one) polynomial equation(s) with bounded degree at the expense of adding a polynomial number of new variables.

Corollary 6.8. *The* Hausdorff *problem remains $\forall\exists_<\mathbb{R}$-complete, even if the two sets A and B are both described by either*

(i) a conjunction of quadratic polynomial equations, or
(ii) a single polynomial equation of degree at most four.

Proof. Taking the formula ψ in the proof of Theorem 6.22, we apply Lemma 6.9(i) to obtain an equivalent new formula ψ' (with additional existentially quantified variables) which is a conjunction of quadratic polynomial equations. Then A can be defined using ψ' instead of ψ. Similarly, B can be trivially described in the desired form. This shows statement (i).

For (ii), we modify the above procedure by applying Lemma 6.9(ii) to ψ to obtain an equivalent formula, which is a single polynomial of degree at most four. ■

6.7 Interlude: $\forall\exists_<\mathbb{R} = \forall\exists\mathbb{R}$

The main result of the previous section, Theorem 6.22, gives a lower bound on the computational complexity of the Hausdorff problem. We obtained $\forall\exists_<\mathbb{R}$-hardness by a reduction from Strict-UETR. Regarding an upper bound, sentence (6.1) shows that the DirectedHausdorff problem is in $\forall\exists\mathbb{R}$; an extension of (6.1) to the Hausdorff problem is trivial (and done in Section 6.8 below). Does one of these classes capture the computational complexity of the Hausdorff problem exactly? Do both?

In Section 6.8 below, we demonstrate how to transform a Hausdorff instance into an equivalent Strict-UETR instance, thereby proving that Hausdorff is indeed in $\forall\exists_<\mathbb{R}$. The reduction is laborious and involves several technical ad-hoc arguments. While this settles the question about the computational complexity of Hausdorff, it is still an intriguing open question whether $\forall\exists_<\mathbb{R}$ is really a strict subset of $\forall\exists\mathbb{R}$, or whether these two classes coincide. In the journal article that this chapter builds upon [95], we pose the following open problem:

Problem 6.23. Is $\forall\exists_<\mathbb{R} = \forall\exists\mathbb{R}$?

Motivated by our $\forall\exists_<\mathbb{R}$-completeness result and the $\exists\forall_\leq\mathbb{R}$-completeness result by D'Costa et al. [47], Schaefer and Štefankovič very recently studied the impact of strictness to first-order formulas. In their recent paper [139], they answer Problem 6.23 affirmatively. In fact, they prove the general case with k blocks of quantifiers: Let us denote by $\Pi_k\mathbb{R}$ the complexity class containing all problems that are polynomial-time many-one reducible to deciding a sentence in the $\forall\exists\ldots$-fragment ($k-1$ quantifier alternations) of the first-order theory of the reals. Furthermore, $\Pi_k^<\mathbb{R} \subseteq \Pi_k\mathbb{R}$ is obtained by restricting to strict inequalities (and forbidding negations). Similarly, $\Sigma_k\mathbb{R}$ and $\Sigma_k^<\mathbb{R}$ denote the classes obtained by starting with an $\exists$-quantifier and possibly restricting to strict inequalities.

Theorem 6.24 ([139, Theorem 1.1]). *We have* $\Sigma_k\mathbb{R} = \Sigma_k^{<}\mathbb{R}$ *and* $\Pi_k\mathbb{R} = \Pi_k^{<}\mathbb{R}$ *for all* $k \geq 1$.

With Theorem 6.24 at hand, we now know that $\forall\exists_<\mathbb{R} = \forall\exists\mathbb{R}$ and can therefore conclude that Hausdorff is actually $\forall\exists\mathbb{R}$- and "not just" $\forall\exists_<\mathbb{R}$-hard. This is good news, because we already know that $\forall\exists\mathbb{R}$-membership can be shown easily. Still, the proof of Theorem 6.24 is quite complicated and requires sophisticated results from real algebraic geometry, in particular an effective Łojasiewicz inequality by Solernó [145]. For the sake of self-containedness, we continue by presenting our original $\forall\exists_<\mathbb{R}$-membership proof. It is rather technical, but requires only basic transformations.

6.8 $\forall\exists_<\mathbb{R}$-Membership of Hausdorff

This section is devoted to proving the following theorem.

Theorem 6.25. Hausdorff *and* DirectedHausdorff *are contained in* $\forall\exists_<\mathbb{R}$.

Let (A, B, t) be a Hausdorff instance, where the two sets $A = \{x \in \mathbb{R}^n \mid \varphi_A(x)\}$ and $B = \{x \in \mathbb{R}^n \mid \varphi_B(x)\}$ are described by quantifier-free formulas φ_A and φ_B with n free variables each. For simplicity, we only consider the directed Hausdorff distance here, namely whether

$$\vec{d}_{\mathrm{H}}(A, B) := \sup_{a\in A} \inf_{b\in B} \|a - b\| \overset{?}{\leq} t.$$

It is obvious, that $d_{\mathrm{H}}(A, B) \leq t$ if and only if $\vec{d}_{\mathrm{H}}(A, B) \leq t$ and $\vec{d}_{\mathrm{H}}(B, A) \leq t$. So if we can formulate the decision problem for the directed Hausdorff distance as a Strict-UETR instance, their conjunction is a formula for the general Hausdorff problem. Assuming that no variable name appears in both operands of this conjunction, this formula can be converted into prenex normal form by just moving the quantifiers to the front.

Recall (see also formula (6.1)) that $\vec{d}_{\mathrm{H}}(A, B) \leq t$ is equivalent to

$$\forall \varepsilon > 0, a \in A \,.\, \exists b \in B \colon \|a - b\|^2 < t^2 + \varepsilon. \tag{6.10}$$

Let us remark that introducing the real variable ε is necessary to also consider the points in the closure of B. Moreover, we work with the squared distance between a and b, because $\|a - b\|$ is the square root of a polynomial.

Below we transform formula (6.10) in multiple technical steps into a form that allows us to apply a recent theorem by D'Costa et al. [47] such that $\forall\exists_<\mathbb{R}$-membership follows. Before we do so, we state a few helpful lemmas. These allow us to consider some intermediate steps in isolation, thereby simplifying the needed notation. Also, Lemma 6.28 below is used again in Section 6.9.

The first lemma allows us to transform a UETR instance of special structure into an equivalent Strict-UETR instance:

Lemma 6.26. *Given a* UETR *instance*

$$\Phi :\equiv \forall X \in [-1,1]^n \,.\, \exists Y \in [-1,1]^m \colon \varphi_<(X,Y) \vee H(X,Y) = 0,$$

where $\varphi_<(X,Y) \in \mathsf{QFF}_<$ and $H\colon [-1,1]^{n+m} \to \mathbb{R}$ is a polynomial. Then we can compute in polynomial time an equivalent Strict-UETR *instance.*

Proof. We first prove that there exists an integer $N \leq \operatorname{poly}(|\Phi|)$, such that the Strict-UETR instance

$$\Psi :\equiv \forall X \in [-1,1]^n \,.\, \exists Y \in [-1,1]^m \colon \varphi_<(X,Y) \vee H(X,Y)^2 < 2^{-2^N}$$

is equivalent to Φ. In a second step, we construct 2^{-2^N} inside the formula.

The direction $\Phi \implies \Psi$ is trivially true for any $N \in \mathbb{N}$. To prove the other direction, we show its contraposition $\neg\Phi \implies \neg\Psi$. Assume that

$$\neg\Phi \equiv \exists X \in [-1,1]^n \,.\, \forall Y \in [-1,1]^m \colon \neg\varphi_<(X,Y) \wedge H(X,Y)^2 > 0$$

is true. Hence, for at least one fixed $x \in [-1,1]^n$, we obtain a polynomial $H(x,Y)^2$ that is positive everywhere on $[-1,1]^m$ (the fixed x values are real coefficients for the variables Y). Because $[-1,1]^m$ is compact and because polynomials are continuous, $H(x,Y)^2$ attains its minimum over $[-1,1]^m$ and it follows that

$$\exists \varepsilon > 0 \,.\, \exists X \in [-1,1]^n \,.\, \forall Y \in [-1,1]^m \colon \neg\varphi_<(X,Y) \wedge H(X,Y)^2 \geq \varepsilon \tag{6.11}$$

is true. Let L be the length of the matrix in (6.11). By Lemma 6.14, there is a constant $\beta \in \mathbb{R}$ such that $\exists \varepsilon > 0$ in (6.11) can be strengthened to $\exists \varepsilon \geq 2^{-L^{\beta^4(n+1)(m+1)}}$. Now choose $N \leq \lceil \beta^4(n+1)(m+1) \rceil$ to be the smallest integer satisfying $2^{-2^N} < 2^{-L^{\beta^4(n+1)(m+1)}}$. Note that $N \leq \operatorname{poly}(n, m, \log L)$, so it is polynomial in the input size. Plugging in the lower bound on ε, we get that $\neg\Phi$ is equivalent to

$$\exists X \in [-1,1]^n \,.\, \forall Y \in [-1,1]^m \exists \neg\varphi_<(X,Y) \wedge H(X,Y)^2 \geq 2^{-2^N},$$

which is exactly $\neg\Psi$. We conclude that $\Phi \equiv \Psi$ as claimed.

To construct a Strict-UETR instance from Ψ, we need to express 2^{-2^N} inside the formula. To this end, introduce $N+1$ new variables $U = (U_0, \ldots, U_N) \in [-1,1]^{N+1}$ and let $\chi(U)$ be formula (6.2). Recall that, by Lemma 6.15, $\chi(u)$ is true if and only if $u_i = 2^{-2^i}$. Including $\chi(U)$ into our formula, we conclude that

$$\forall X, U \in [-1,1]^{n+N+1} \,.\, \exists Y \in [-1,1]^m \colon \neg\chi(U) \vee \neg\varphi_<(X,Y) \vee H(X,Y)^2 < U_N \tag{6.12}$$

is equivalent to Φ.

We arrived at a sentence where all variables are restricted to $[-1,1]$ and in which all atoms are strict. At this point, we use a recent result by D'Costa et al. [47]. They show that it is $\exists\forall_{\leq}\mathbb{R}$-complete to decide a sentence of the form

$$\exists X \in [-1,1]^n \,.\, \forall Y \in [-1,1]^m \colon \varphi_{\leq}(X,Y)$$

with $\varphi_{\leq} \in \mathrm{QFF}_{\leq}$. Because $\exists\forall_{\leq}\mathbb{R} = \mathrm{co}\forall\exists_{<}\mathbb{R}$, deciding the complements of these sentences, i.e., sentences of the form

$$\forall X \in [-1,1]^n \,.\, \exists Y \in [-1,1]^m \colon \varphi_{<}(X,Y)$$

with $\varphi_{<} \in \mathrm{QFF}_{<}$ is $\forall\exists_{<}\mathbb{R}$-complete. Sentence (6.12) is of this form. Thus, there is a polynomial-time reduction to an equivalent Strict-UETR instance. ■

The next lemma establishes an upper bound on the value of a polynomial over a compact domain.

Lemma 6.27. *Let $P\colon \mathbb{R}^n \to \mathbb{R}$ be a polynomial, N be an integer and $C := 2^{2^N}$. Then we can compute in polynomial time an integer $K \leq \mathrm{poly}(|P|, N, n)$ such that for $E := 2^{2^K}$ and all $x \in [-C,C]^n$ it holds that $|P(x)| \leq E$.*

Proof. Because P is a polynomial, $|P|$ is continuous, and therefore $|P|$ attains its maximum over any compact domain. We conclude that

$$\exists E \in \mathbb{R} \,.\, \forall X \in [-C,C]^n \colon |P(X)| \leq E$$

is true. Note that, strictly speaking, we may not use $|\cdot|$ inside the formula. However, $|P(X)| \leq E$ is equivalent to $P(X) \leq E \wedge -P(X) \leq E$.

To obtain an upper bound on E, we first need to encode C inside the formula. We introduce $N+1$ new variables $U = \{U_0, \dots, U_N\}$ and let $\chi(U)$ be formula (6.2). Recall that, by Lemma 6.15, $\chi(u)$ is true if and only if $u_i = 2^{-2^i}$. Now we can rewrite the above formula equivalently as

$$\exists E \in \mathbb{R} \,.\, \forall X \in \mathbb{R}^n \,.\, \exists U \in \mathbb{R}^{N+1} \colon \bigwedge_{i=1}^{n} |X_i U_N| \leq 1 \implies |P(X)| \leq E.$$

Let $\varphi(E)$ be the subformula following the quantification of E (starting from $\forall$) and $L := |\varphi(E)|$. Applying quantifier elimination (Corollary 6.12) to $\varphi(E)$, we obtain a constant $\alpha \in \mathbb{R}$ and an equivalent, quantifier-free formula $\psi(E)$ of length

$$|\psi(E)| \leq L^{2\alpha^3(n+1)(N+2)}.$$

The Ball Theorem (Theorem 6.13) applied to $\exists E \in \mathbb{R} \colon \psi(E)$ now yields an upper bound for E:

$$E \leq 2^{|\psi|^8} \leq 2^{L^{16\alpha^3(n+1)(N+2)}} = 2^{2^{16\log(|\psi|)\alpha^3(n+1)(N+2)}}$$

Choose $K = \lceil 16\log(|\psi|)\alpha^3(n+1)(N+2) \rceil$. Obviously, $K \leq \mathrm{poly}(\log|\psi|, N, n)$. Since $|\psi| \leq \mathrm{poly}(|P|, n, N)$, the claim follows. ■

Above lemma is used to prove the following lemma that allows us to transform some more general UETR instances into equivalent Strict-UETR instances.

Lemma 6.28. *Given a* UETR *instance*

$$\forall \varepsilon > 0, X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m \colon F(X)^2 > 0 \vee \big(G(Y) = 0 \wedge P(X,Y) < \varepsilon\big),$$

where $F\colon \mathbb{R}^n \to \mathbb{R}$, $G\colon \mathbb{R}^m \to \mathbb{R}$ and $P\colon \mathbb{R}^{n+m} \to \mathbb{R}$ are polynomials. Then we can compute in polynomial time an equivalent Strict-UETR *instance.*

Proof. Via a series of manipulations, we transform the given sentence into an equivalent UETR instance that has the form required by Lemma 6.26. The first step is to move "$\varepsilon > 0$"-condition into the formula. We obtain an equivalent sentence

$$\begin{aligned}&\forall \varepsilon \in \mathbb{R}, X \in \mathbb{R}^n \colon\ (\varepsilon > 0) \implies \\ &\qquad \big(\exists Y \in \mathbb{R}^n \colon F(X)^2 > 0 \vee (G(Y) = 0 \wedge P(X,Y) < \varepsilon)\big).\end{aligned}$$

Now we observe that $\varepsilon > 0$ is equivalent to $\exists \delta \in \mathbb{R} \colon \delta^2\varepsilon - 1 = 0$. Incorporating this yields an equivalent sentence

$$\begin{aligned}&\forall \varepsilon \in \mathbb{R}, X \in \mathbb{R}^n \colon\ (\exists \delta \in \mathbb{R} \colon \delta^2\varepsilon - 1 = 0) \implies \\ &\qquad \big(\exists Y \in \mathbb{R}^m \colon F(X)^2 > 0 \vee (G(Y) = 0 \wedge P(X,Y) < \varepsilon)\big).\end{aligned}$$

Rewriting the implication $A \implies B$ as $\neg A \vee B$ turns the existential quantifier in front of δ into a universal quantifier. Furthermore, we replace $\neg(\delta^2\varepsilon - 1 = 0)$ by the equivalent $(\delta^2\varepsilon - 1)^2 > 0$. We get an equivalent sentence

$$\begin{aligned}&\forall \varepsilon \in \mathbb{R}, X \in \mathbb{R}^n \colon\ \big(\forall \delta \in \mathbb{R} \colon (\delta^2\varepsilon - 1)^2 > 0\big) \vee \\ &\qquad \big(\exists Y \in \mathbb{R}^m \colon F(X)^2 > 0 \vee (G(Y) = 0 \wedge P(X,Y) < \varepsilon)\big).\end{aligned}$$

Moving all quantifiers to the front turns this into an equivalent prenex normal form

$$\begin{aligned}&\forall \varepsilon \in \mathbb{R}, \delta \in \mathbb{R}, X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m \colon \\ &\qquad (\delta^2\varepsilon + 1)^2 > 0 \vee F(X)^2 > 0 \vee \big(G(Y) = 0 \wedge P(X,Y) < \varepsilon\big).\end{aligned}$$

This sentence is $\forall$-strict, so Lemmas 6.16 and 6.18 are applicable. Thus, there are two integers N, M bounded by a polynomial in the length of the sentence, such that for $C := 2^{2^N}$ and $D := 2^{2^M}$ all universally quantified variables can be restricted to $[-C, C]$ and all existentially quantified variables can be restricted to $[-D, D]$. We obtain another equivalent sentence

$$\begin{aligned}&\forall \varepsilon \in [-C, C], \delta \in [-C, C], X \in [-C, C]^n \,.\, \exists Y \in [-D, D]^m \colon \\ &\qquad (\delta^2\varepsilon + 1)^2 > 0 \vee F(X)^2 > 0 \vee \big(G(Y) = 0 \wedge P(X,Y) < \varepsilon\big).\end{aligned}$$

In the next step, we replace the strict inequality $P(X,Y) < \varepsilon$ by the non-strict inequality $P(X,Y) \leq \varepsilon$. For this step, we exploit the fact that a continuous function over a compact domain attains its minimum and maximum. Consequently, the sentence $\forall \varepsilon > 0, X \in [-C, C] \,.\, \exists Y \in [-D, D] \colon P(X,Y) < \varepsilon$ is (true if and only if $\max_{X\in[-C,C]}\min_{Y\in[-D,D]} P(X,Y) \leq 0$ and thus) equivalent to $\forall\varepsilon > 0, X \in [-C,C] \,.\, \exists Y \in [-D,D] \colon P(X,Y) \leq \varepsilon$. We obtain the equivalent sentence

$$\begin{aligned}&\forall\varepsilon \in [-C,C], \delta \in [-C,C], X \in [-C,C]^n \,.\, \exists Y \in [-D,D]^m \colon\\ &\qquad (\delta^2\varepsilon+1)^2 > 0 \lor F(X)^2 > 0 \lor \big(G(Y) = 0 \land P(X,Y) \leq \varepsilon\big).\end{aligned}$$

Going one step further, we now want to express $P(X,Y) \leq \varepsilon$ as a polynomial equation. We replace it by the equivalent formula $\exists B \in \mathbb{R} \colon P(X,Y) - \varepsilon + B^2 = 0$. By Lemma 6.27, we can also bound the range over which B is quantified: We can compute in polynomial time an integer $K \leq \operatorname{poly}(|P|, \max\{N,M\}, n+m+1)$ such that $|B| \leq E := 2^{2^K}$. We get another equivalent sentence

$$\begin{aligned}&\forall\varepsilon \in [-C,C], \delta \in [-C,C], X \in [-C,C]^n \,.\, \exists Y \in [-D,D]^m, B \in [E,E] \colon\\ &\qquad (\delta^2\varepsilon+1)^2 > 0 \lor F(X)^2 > 0 \lor \big(G(Y) = 0 \land P(X,Y) - \varepsilon + B^2 = 0\big).\end{aligned}$$

At this point, we define

$$\begin{aligned}\varphi_<(\varepsilon,\delta,X) > 0 &:= (\delta^2\varepsilon+1)^2 \lor F(X)^2 > 0 \quad \text{and}\\ H(X,Y,\varepsilon,B) &:= G(Y)^2 + (P(X,Y) - \varepsilon + B^2)^2.\end{aligned}$$

Note that $\varphi_< \in \mathsf{QFF}_<$. We use these to get the equivalent sentence

$$\begin{aligned}&\forall\varepsilon \in [-C,C], \delta \in [-C,C], X \in [-C,C]^n \,.\, \exists Y \in [-D,D]^m, B \in [E,E] \colon\\ &\qquad \varphi_<(\varepsilon,\delta,X) \lor H(X,Y,\varepsilon,B) = 0.\end{aligned}$$

The last step is to scale all variables to be in the interval $[-1,1]$. For this, let $S := \max\{N,M,K\}$ and introduce $S+1$ new variables $U = \{U_0,\dots,U_S\}$. Furthermore, let $\chi(U)$ be formula (6.2). Recall that, by Lemma 6.15, $\chi(u)$ is true if and only if $u_i = 2^{-2^i}$. We can rewrite our sentence to the equivalent sentence

$$\begin{aligned}&\forall\varepsilon,\delta,X,U \in [-1,1]^{1+1+n+S+1} \,.\, \exists Y,B \in [-1,1]^{m+1} \colon\\ &\qquad \neg\chi(U) \lor U_S^d \cdot \varphi_<\left(\frac{\varepsilon}{U_S},\frac{\delta}{U_S},\frac{X}{U_S}\right) \lor U_S^d \cdot H\left(\frac{X}{U_S},\frac{Y}{U_S},\frac{\varepsilon}{U_S},\frac{B}{U_S}\right) = 0.\end{aligned}$$

Here d is the maximum degree of any polynomial in $\varphi_<$ and H. By X/U_S, we denote that every variable X_i is replaced by X_i/U_S. Multiplying by U_S^d makes sure that each atom remains a polynomial. What we obtained is a UETR instance that has the form required by Lemma 6.26 (note that $\neg\chi(U)$ is strict). Therefore, we can transform this into an equivalent Strict-UETR instance in polynomial time. ■

Finally, we have all the needed tools to prove Theorem 6.25 which states that the Hausdorff problem is contained in $\forall\exists_<\mathbb{R}$. We do this by transforming formula (6.10) into the form required by Lemma 6.28. This then yields an equivalent Strict-UETR instance, and therefore proves $\forall\exists_<\mathbb{R}$-membership.

Proof of Theorem 6.25. Recall that $\vec{d}_{\mathrm{H}}(A,B) \le t$ is equivalent to (6.10), which is

$$\forall \varepsilon > 0, a \in A \,.\, \exists b \in B \colon \|a-b\|^2 < t^2 + \varepsilon.$$

In a first step, we resolve the shorthand notations $a \in A$ and $b \in B$ and we obtain

$$\forall \varepsilon > 0, a \in \mathbb{R}^n \colon \varphi_A(a) \implies \big(\exists b \in \mathbb{R}^n \colon \varphi_B(b) \wedge \|a-b\|^2 < t^2 + \varepsilon\big).$$

Next, we consider the (quantifier-free) formulas $\varphi_A(a)$ and $\varphi_B(b)$. Using Lemma 6.9, we obtain in polynomial time two integers k, ℓ and two polynomials $F_A \colon \mathbb{R}^{n+k} \to \mathbb{R}$ and $F_B \colon \mathbb{R}^{n+\ell} \to \mathbb{R}$, such that $\varphi_A(a)$ is equivalent to $\exists U_a \in \mathbb{R}^k \colon F_A(a, U_a) = 0$ and similarly $\varphi_B(b)$ is equivalent to $\exists U_b \in \mathbb{R}^\ell \colon F_B(b, U_b) = 0$. This yields the equivalent sentence

$$\begin{aligned}&\forall \varepsilon > 0, a \in \mathbb{R}^n \colon \; \big(\exists U_a \in \mathbb{R}^k \colon F_A(a, U_a) = 0\big) \implies \\ &\qquad \big(\exists b \in \mathbb{R}^n, U_b \in \mathbb{R}^\ell \colon F_B(b, U_b) = 0 \wedge \|a-b\|^2 < t^2 + \varepsilon\big).\end{aligned}$$

Rewriting the implication $X \implies Y$ as $\neg X \vee Y$ changes the existential quantifier in front of U_a into a universal quantifier, which we can move to the front. Also, the equation gets negated. Substituting $\neg(F(a, U_a) = 0)$ by $F(a, U_a)^2 > 0$, we get the equivalent sentence

$$\begin{aligned}&\forall \varepsilon > 0, a \in \mathbb{R}^n, U_a \in \mathbb{R}^k \colon F_A(a, U_a)^2 > 0 \vee \\ &\qquad \big(\exists b \in \mathbb{R}^n, U_b \in \mathbb{R}^\ell \colon F_B(b, U_b) = 0 \wedge \|a-b\|^2 < t^2 + \varepsilon\big).\end{aligned}$$

Lastly, we move the existential quantifier after the universal one and get an equivalent sentence

$$\begin{aligned}&\forall \varepsilon > 0, a \in \mathbb{R}^n, U_a \in \mathbb{R}^k \,.\, \exists b \in \mathbb{R}^n, U_b \in \mathbb{R}^\ell \colon \\ &\qquad F_A(a, U_a)^2 > 0 \vee \big(F_B(b, U_b) = 0 \wedge \|a-b\|^2 < t^2 + \varepsilon\big)\end{aligned}$$

in prenex normal form. This sentence has the form required by Lemma 6.26, concluding the proof. ∎

6.9 The Hausdorff Distance and Exotic Quantifiers

By now, we have determined the exact computational complexity of Hausdorff. The main tool was Theorem 6.20 that allowed us to obtain a Strict-UETR instance which, if false, had an open ball of counterexamples. Thus, we managed to get from a syntactical restriction (strictness) to a topological one (having an open ball of counterexamples).

In this section, we want to study these topological restrictions. Bürgisser and Cucker consider the computational complexity of several problems related to properties of semi-algebraic sets [34]. They notice that the computational complexity of several problems defies to be classified into levels of the real polynomial hierarchy. Recall that we had a similar situation with the Hausdorff problem, as did D'Costa et al. with their "escape problem". In both cases, the solution was to consider complexity classes (concretely $\forall\exists_<\mathbb{R}$ and $\exists\forall_\leq\mathbb{R}$) obtained by putting syntactic restrictions to the formulas. Bürgisser and Cucker take a different approach and define three additional *exotic* quantifiers besides $\exists$ and $\forall$[3]. Two of them are highly related to our work:

$$\begin{aligned}\forall^* X \in \mathbb{R}^n : \varphi(X) &:\equiv \forall \varepsilon > 0, X \in \mathbb{R}^n \,.\, \exists \widetilde{X} \in \mathbb{R}^n : \|X - \widetilde{X}\|^2 < \varepsilon \wedge \varphi(\widetilde{X})\\ \exists^* X \in \mathbb{R}^n : \varphi(X) &:\equiv \exists \varepsilon > 0, X \in \mathbb{R}^n \,.\, \forall \widetilde{X} \in \mathbb{R}^n : \|X - \widetilde{X}\|^2 < \varepsilon \implies \varphi(\widetilde{X})\end{aligned}$$

Intuitively, $\forall^* X \in \mathbb{R}^n : \varphi(X)$ expresses that $\varphi(x)$ does not need to be true for all $x \in \mathbb{R}^n$ but just for all $x \in D$, where D is a dense subset of $\mathbb{R}^n$. Conversely, $\exists^* X \in \mathbb{R}^n : \varphi(X)$ expresses that there must be an $x \in \mathbb{R}^n$ and some radius $r > 0$, such that for all $\widetilde{x} \in B_n(x, r)$ it holds that $\varphi(\widetilde{x})$, where $B_n(x, r)$ denotes the n-dimensional open ball of radius r centered at x.

As one would expect, it holds that

$$\begin{aligned}\neg\forall^* X \in \mathbb{R}^n : \varphi(X) &\equiv \exists^* X \in \mathbb{R}^n : \neg\varphi(X) \quad \text{and}\\ \neg\exists^* X \in \mathbb{R}^n : \varphi(X) &\equiv \forall^* X \in \mathbb{R}^n : \neg\varphi(X).\end{aligned}$$

Theorem 6.29, the main theorem of this section, establishes a relation between the two approaches. As it turns out, the topological restrictions on the formulas by Bürgisser and Cucker are equivalent to the syntactical restrictions made in this thesis. For this purpose, let Exotic-UETR denote the decision problem whether a sentence of the form

$$\forall^* X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m : \varphi(X, Y)$$

with quantifier-free φ is true. The complexity class $\forall^*\exists\mathbb{R}$ shall contain all problems that polynomial-time many-one reduce to Exotic-UETR. We show that the complexity classes $\forall^*\exists\mathbb{R}$ and $\forall\exists_<\mathbb{R}$ coincide.

Theorem 6.29. Exotic-UETR *is* $\forall\exists_<\mathbb{R}$*-complete. Thus* $\forall^*\exists\mathbb{R} = \forall\exists_<\mathbb{R}$.

Proof. The $\forall\exists_<\mathbb{R}$-hardness of Exotic-UETR follows from Theorem 6.20: For a given Strict-UETR instance

$$\Phi :\equiv \forall X \in \mathbb{R}^n \,.\, \exists Y \in \mathbb{R}^m : \varphi_<(X, Y)$$

with $\varphi_< \in \mathrm{QFF}_<$, Theorem 6.20 allows us to compute an equivalent UETR instance

$$\Psi :\equiv \forall X \in \mathbb{R}^k \,.\, \exists Y \in \mathbb{R}^\ell : \psi(X, Y)$$

in polynomial time, where $\psi \in \mathrm{QFF}$. Recall that on the one hand, if Ψ is false, then the set of counterexamples $\perp(\Psi)$ contains an open ball. On the other hand, if Ψ is

3 While they mainly work in the BSS-model, they also consider some problems in the bit-model of computation. The complexity classes $\exists\mathbb{R}$, $\forall\mathbb{R}$ and $\forall\exists\mathbb{R}$ appear under the names $\mathrm{BP}^0(\exists)$, $\mathrm{BP}^0(\forall)$ and $\mathrm{BP}^0(\forall\exists)$, respectively. (Here the BP stands for "Boolean part" and the superscript "0" denotes that there are no machine constants in the BSS machine.)

true, then $\perp(\Psi) = \emptyset$. Therefore, Ψ is true if and only if it is true for a dense subset of $\mathbb{R}^k$. It follows that the $\forall$-quantifier can be replaced by the exotic $\forall^*$-quantifier in Ψ and we get

$$\Phi \;\equiv\; \Psi \;\equiv\; \forall^* X \in \mathbb{R}^k . \exists Y \in \mathbb{R}^\ell \colon \psi(X, Y).$$

Consequently, Exotic-UETR is $\forall\exists_<\mathbb{R}$-hard.

To prove that Exotic-UETR is contained in $\forall\exists_<\mathbb{R}$, we transform

$$\forall^* X \in \mathbb{R}^n . \exists Y \in \mathbb{R}^m \colon \varphi(X, Y)$$

in polynomial time into an equivalent sentence of the form required by Lemma 6.28. This lemma allows us to construct an equivalent Strict-UETR instance in polynomial time, thereby proving $\forall\exists_<\mathbb{R}$-membership. We start by expressing the exotic quantifier $\forall^*$ in terms of classical quantifiers $\forall$ and $\exists$, obtaining an equivalent sentence

$$\forall \varepsilon > 0, X \in \mathbb{R}^n . \exists X_0 \in \mathbb{R}^n, Y \in \mathbb{R}^m \colon \|X - X_0\|^2 < \varepsilon \wedge \varphi(X_0, Y).$$

By Lemma 6.9, we can compute in polynomial time an integer k and a polynomial $G\colon \mathbb{R}^{n+m+k} \to \mathbb{R}$ such that $\varphi(X_0, Y)$ is equivalent to $\exists U \in \mathbb{R}^k \colon G(X_0, Y, U) = 0$. Plugging this into above sentence, we get another equivalent sentence

$$\forall \varepsilon > 0, X \in \mathbb{R}^n . \exists X_0 \in \mathbb{R}^n, Y \in \mathbb{R}^m, U \in \mathbb{R}^k \colon \|X - X_0\|^2 < \varepsilon \wedge G(X_0, Y, U) = 0.$$

This has the form required by Lemma 6.28. Hence, $\forall\exists_<\mathbb{R}$-membership follows. ■

Remark 6.30 (Recent Advances)**.** Again, taking into account the recent result by Schaefer and Štefankovič, we can further say that $\forall^*\exists\mathbb{R} = \forall\exists\mathbb{R}$ [139], i.e., the definition of $\forall\exists\mathbb{R}$ is *robust* under replacing the $\forall$ quantifier with a $\forall^*$ quantifier.

Furthermore, the recent bachelor's thesis of Junginger [96] also studies the real polynomial hierarchy, proving its robustness under exotic quantifiers up to the second level. ⌟

6.9.1 Application to the Hausdorff Problem

We now use above insights to establish the exact computational complexity of EuclideanRelativeDenseness (ERD) which was left as an open problem by Bürgisser and Cucker [34]. In ERD, we are given two semi-algebraic sets A and B, and ask whether A is contained in $\overline{B}$, i.e., the closure of B. Note that ERD is equivalent to deciding whether $\vec{d}_{\mathrm{H}}(A, B) = 0$. Bürgisser and Cucker show:

Theorem 6.31 ([34, Corollary 5.6])**.** ERD *is in* $\forall\exists\mathbb{R}$ *and* $\forall^*\exists\mathbb{R}$*-hard.*

They prove this in the BSS-model, but the same proof also works in the bit-model. Building on our results from above, we are able to determine the exact computational complexity of ERD in the bit-model of computation.

Theorem 6.32. ERD *is* $\forall\exists_<\mathbb{R}$*-complete.*

Proof. By Theorem 6.29 and the $\forall^*\exists\mathbb{R}$-hardness from Theorem 6.31, it follows that ERD is $\forall\exists_<\mathbb{R}$-hard. Furthermore, Theorem 6.25 implies that ERD is contained in $\forall\exists_<\mathbb{R}$. Consequently, ERD is $\forall\exists_<\mathbb{R}$-complete. ∎

Remark 6.33 (Hausdorff distance 0)**.** The $\forall\exists_<\mathbb{R}$-hardness of ERD implies $\forall\exists_<\mathbb{R}$-hardness of the DirectedHausdorff problem (for any distance $t \geq 0$): Given an instance $A, B \subseteq \mathbb{R}^n$ of ERD, we define $A' := (A, 0) \subseteq \mathbb{R}^{n+1}$ and $B' := (B, 1) \subseteq \mathbb{R}^{n+1}$. Then $\vec{d}_{\mathrm{H}}(A, B) = 0$ if and only if $\vec{d}_{\mathrm{H}}(A', B') \leq 1$. ⌟

6.10 Conclusion and Open Problems

We proved that computing the Hausdorff distance between two semi-algebraic sets is $\forall\exists_<\mathbb{R}$-complete, which is equivalent to being $\forall\exists\mathbb{R}$-complete by a recent result from Schaefer and Štefankovič stating that $\forall\exists_<\mathbb{R} = \forall\exists\mathbb{R}$ [139]. This settles the computational complexity of a very important problem in computational geometry. Furthermore, we contribute the first $\forall\exists\mathbb{R}$-completeness proof of a natural problem.

We wonder about the complexity of different variants of the Hausdorff problem. For example, deciding whether the Hausdorff distance between two semi-algebraic sets is *exactly* 0, is $\forall\mathbb{R}$-complete, so supposedly easier than the general case [95]. The opposite might be true for the Hausdorff distance under translation, i.e., the Hausdorff distance after applying an arbitrary translation to one of the two sets.

Open Problem 8. What is the computational complexity of computing the Hausdorff distance under translation?

Deciding whether the Hausdorff distance under translation is at most d can be formulated as a first-order sentence of the form

$$\exists t \in \mathbb{R}^n\colon \Big(\big(\forall a \in A, \varepsilon > 0 \,.\, \exists b \in B\colon \|(a+t) - b\| < d + \varepsilon\big) \land \big(\forall b \in B, \varepsilon > 0 \,.\, \exists a \in A\colon \|(a+t) - b\| < d + \varepsilon\big) \Big),$$

proving that it is in $\exists\forall\exists\mathbb{R}$. Is it also hard for this class?

Complementing our results, we also establish a connection to exotic quantifiers, and present first results on their impact on the complexity classes. The effect of exotic quantifiers is not yet fully understood and has been the topic of other publications following our initial results [96, 139]. The following question is again highlighted in Chapter 7 below.

Open Problem 9. What is the effect of exotic quantifiers on the complexity classes $\exists\mathbb{R}$, $\forall\exists\mathbb{R}$, …?

7 Conclusion and Outlook

We explored the complexity classes $\exists\mathbb{R}$ and $\forall\exists\mathbb{R}$. As outlined in Chapter 1, our goal was to learn more about the classes themselves and the problems therein. Our findings allowed us to prove $\exists\mathbb{R}$- and $\forall\exists\mathbb{R}$-completeness of several well-known problems whose complexity status remained unclear up to this point. For each of these results, a new idea was necessary to make the reduction possible. Below, we summarize our results, highlighting the key techniques used. Some of these techniques have already been adapted to other problems, confirming their usefulness. We hope that they continue to serve for future reductions, and thereby advance the study of $\exists\mathbb{R}$ beyond the results presented in this thesis. To finish, we state two open-ended questions to stimulate further research, and on which we hope to see progress in the future.

Summary In Chapter 3, we employed the Beltrami-Klein model to prove that SimpleStretchability is equivalent in the Euclidean plane $\mathbb{R}^2$ and the hyperbolic plane $\mathbb{H}^2$. This led to a general framework to translate $\exists\mathbb{R}$-hardness results from $\mathbb{R}^2$ to $\mathbb{H}^2$. More importantly, these hyperbolic insights are the key idea to prove $\exists\mathbb{R}$-completeness of Lombardi graph drawing in Chapter 4, a widely studied graph drawing style. Let us remind the reader that Lombardi drawability is a "purely Euclidean" drawing style. The detour through hyperbolic geometry allowed us to utilize the commonalities of Lombardi drawing and the Poincaré disk model, ultimately leading to the first complexity result for general graphs.

Chapter 5 considered the training of artificial neural networks. Since 1992, this is known to be NP-hard [25], even for small networks with just three neurons. However, a matching upper bound remained elusive. We were able to close this gap by proving $\exists\mathbb{R}$-completeness. In particular, we lifted the lower bound, and thereby proved that it is supposedly even more difficult. Our main technical contribution is to understand and utilize the underlying geometry of small neural networks. Building on the shape of the ReLU activation function ($x \mapsto \max\{0, x\}$), we constructed gadgets that could not just store real numbers, but also add and inverse them. In turn, this allowed us to encode an arbitrary system of polynomial equations into a neural network training instance. In fact, our method to geometrically construct the gadgets proved flexible enough, that it was already applied by others for an NP-hardness reduction in a similar setting [66].

Lastly, in Chapter 6, we proved $\forall\exists_{<}\mathbb{R}$-completeness of computing the Hausdorff distance of two semi-algebraic sets. Many algorithms for simpler versions of this problem have been developed, but the complexity of the general problem remained open. Actually, it was explicitly asked for in [54]. The missing piece for a reduction was a method to enlarge the set of counterexamples of an UETR instance. Recall that intuitively, a counterexample witnesses the falseness of a UETR instance. This witness might be an isolated point in $\mathbb{R}^n$, but the reduction requires an open ball. We accomplished this enlargement for Strict-UETR instances, leading to $\forall\exists_<\mathbb{R}$-completeness. Independently, this led to a proof that $\forall\exists_<\mathbb{R} = \forall^*\exists\mathbb{R}$, establishing a connection between classical and exotic quantifiers. Alongside the results in [47, 54], this triggered subsequent work answering the $\forall\exists_<\mathbb{R} = \forall\exists\mathbb{R}$ question left open by us [139].

Outlook Chapters 3 to 6 each end with a short list of open problems, which we consider to be interesting follow-up questions. While some are stated precisely, others are more open-ended. Answering them might involve finding a sensible and precise statement first. To contextualize these questions, we also provided some first ideas on how to tackle them, or discussed why our current approaches do not seem strong enough.

We finish this thesis about the complexity classes $\exists\mathbb{R}$, $\forall\exists\mathbb{R}$ and alike by highlighting two directions for future research. The first one naturally arises from our results on the complexity of computing the Hausdorff distance in Chapter 6. The second is independent of our results. Still, in our eyes, it is among the most important questions in this field.

The Real Polynomial Hierarchy In Chapter 6, we studied the class $\forall\exists\mathbb{R}$. Recall that $\exists\mathbb{R}$ and $\forall\exists\mathbb{R}$ can be understood as being real counterparts of NP and Π_2^P in the polynomial hierarchy PH. Accordingly, one can define the *real polynomial hierarchy*: On its k-th level, the classes $\Sigma_k\mathbb{R}$ and $\Pi_k\mathbb{R}$ capture the complexity of deciding a first-order sentence in prenex normal form with exactly $k-1$ quantifier alternations.

While the number of known $\exists\mathbb{R}$-complete problems has increased rapidly over the last years, there are still only a handful of problems known to be complete for the second level, i.e., for $\forall\exists\mathbb{R} = \Pi_2\mathbb{R}$ or $\exists\forall\mathbb{R} = \Sigma_2\mathbb{R}$. Interestingly, there is a similar phenomenon for NP and higher levels of the "classical" polynomial hierarchy. One reason for this discrepancy might be the natural increase of technical difficulty concomitant with an increasing number of quantifiers.

Open Problem 10. Identify more problems that are complete for the second (or even third, fourth, …) level of the real polynomial hierarchy.

Concerning Open Problem 10, several candidates are discussed in [54].

An interesting structural question concerns the *robustness* of the real polynomial hierarchy, i.e., whether the complexity classes $\Sigma_k\mathbb{R}$ and $\Pi_k\mathbb{R}$ are invariant under small changes of the syntax. A first notable result is by Schaefer and Štefankovič, who prove robustness under restrictions to the set of comparison operators [139]. In particular, one obtains the same classes when restricting to negation-free formulas in which all atoms are strict inequalities (using only "$<$", "$>$" or "$\neq$").

In contrast, the effect of the exotic quantifiers $\exists^*$ and $\forall^*$ (see Chapter 6) is not yet fully understood [34]: It is conjectured that $\exists^*$ and $\forall^*$ can always be replaced by their non-starred "classical" counterparts. For example, our Theorem 6.29 states that $\forall^*\exists\mathbb{R} = \forall\exists_<\mathbb{R}$, which itself equals $\forall\exists\mathbb{R}$ [139]. Currently, this is only known for the first level ($\exists^*\mathbb{R} = \exists\mathbb{R}$ and $\forall^*\mathbb{R} = \forall\mathbb{R}$), with some partial results for the second and higher levels, see Section 6.9 and [96, 139].

Open Problem 11. Is the real polynomial hierarchy robust under exotic quantifiers?

(Conditional) Class Separations We know that $\mathsf{NP} \subseteq \exists\mathbb{R} \subseteq \mathsf{PSPACE}$ [37, 144], but neither inclusion is known to be strict. In fact, proving either of them to be strict seems out of reach, as this would trivially imply $\mathsf{NP} \subsetneq \mathsf{PSPACE}$.

Having said that, it might be easier to prove a *conditional* separation, i.e., to prove $\mathsf{NP} \subsetneq \exists\mathbb{R}$ under an additional assumption that is unproven but widely believed by the community. For example, one such unproven assumption is that the polynomial hierarchy PH does *not* collapse, i.e., that the inclusions $\Sigma_k^{\mathsf{P}} \subsetneq \Sigma_{k+1}^{\mathsf{P}}$ and $\Pi_k^{\mathsf{P}} \subsetneq \Pi_{k+1}^{\mathsf{P}}$ are strict for all $k \in \mathbb{N}$. Now, if the hypothetical inclusion $\exists\mathbb{R} \subseteq \mathsf{NP}$ would imply a collapse of the polynomial hierarchy, then this would be strong evidence for $\mathsf{NP} \subsetneq \exists\mathbb{R}$. (Let us note that we do not have any evidence in this direction.)

Open Problem 12. Find (conditional) class separations for $\mathsf{NP} \subseteq \exists\mathbb{R}$ and/or $\exists\mathbb{R} \subseteq \mathsf{PSPACE}$.

Another interesting result would be an $\exists\mathbb{R}$-version of the so-called *relativization barrier* [14]. In its usual form, it states that there are oracles A and B such that $\mathsf{P}^A = \mathsf{NP}^A$ but $\mathsf{P}^B \neq \mathsf{NP}^B$. This is highly relevant in computational complexity, as it rules out certain proof techniques to answer the famous P vs. NP question. In particular, $P \neq \mathsf{NP}$ cannot be a proven by a proof that is based purely on diagonalization.

Open Problem 13. Are there oracles A and B such that $\mathsf{NP}^A = \exists\mathbb{R}^A$ but $\mathsf{NP}^B \neq \exists\mathbb{R}^B$? How about $\exists\mathbb{R}$ and PSPACE?

Note that the notation $\exists\mathbb{R}^O$ for an oracle O has not yet been defined formally in the literature.

Bibliography

[1] Zachary Abel, Erik Demaine, Martin Demaine, Sarah Eisenstat, Jayson Lynch and Tao Schardl. “Who Needs Crossings? Hardness of Plane Graph Rigidity”. In: *32nd International Symposium on Computational Geometry (SoCG 2016)*. Ed. by Sándor Fekete and Anna Lubiw. Vol. 51. Leibniz International Proceedings in Informatics (LIPIcs). 2016, 3:1–3:15. DOI: 10.4230/LIPIcs.SoCG.2016.3.

[2] Mikkel Abrahamsen. “Covering Polygons is Even Harder”. In: *62nd Annual Symposium on Foundations of Computer Science (FOCS 2021)*. 2022, pp. 375–386. DOI: 10.1109/FOCS52979.2021.00045.

[3] Mikkel Abrahamsen, Anna Adamaszek and Tillmann Miltzow. “The Art Gallery Problem is $\exists\mathbb{R}$-complete”. In: *Journal of the ACM* 69.1 (2022), pp. 1–70. DOI: 10.1145/3486220.

[4] Mikkel Abrahamsen, Linda Kleist and Tillmann Miltzow. “Training Neural Networks is ER-complete”. In: *Advances in Neural Information Processing Systems (NeurIPS 2021)*. Ed. by Marc A. Ranzato, Alina Beygelzimer, Yann Dauphin, Percy S. Liang and Jennifer W. Vaughan. Vol. 34. 2021, pp. 18293–18306. URL: https://proceedings.neurips.cc/paper_files/paper/2021/hash/9813b270ed0288e7c0388f0fd4ec68f5-Abstract.html.

[5] Mikkel Abrahamsen, Linda Kleist and Tillmann Miltzow. “Geometric Embeddability of Complexes is $\exists\mathbb{R}$-complete”. In: *39th International Symposium on Computational Geometry (SoCG 2023)*. Ed. by Erin W. Chambers and Joachim Gudmundsson. Vol. 258. Leibniz International Proceedings in Informatics (LIPIcs). 2023, 1:1–1:19. DOI: 10.4230/LIPIcs.SoCG.2023.1.

[6] Mikkel Abrahamsen and Tillmann Miltzow. “Dynamic Toolbox for ETRINV”. arXiv preprint. 2019. DOI: 10.48550/arXiv.1912.08674.

[7] Mikkel Abrahamsen, Tillmann Miltzow and Nadja Seiferth. “Framework for ER-Completeness of Two-Dimensional Packing Problems”. In: *61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*. 2020, pp. 1014–1021. DOI: 10.1109/FOCS46700.2020.00098.

[8] Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen Kobourov and Mingwei Li. “Multicriteria Scalable Graph Drawing via Stochastic Gradient Descent, $(SGD)^2$”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.6 (2022), pp. 2388–2399. DOI: 10.1109/TVCG.2022.3155564.

[9] Helmut Alt, Bernd Behrends and Johannes Blömer. "Approximate Matching of Polygonal Shapes". In: *Annals of Mathematics and Artificial Intelligence* 13.3 (1995), pp. 251–265. DOI: 10.1007/BF01530830.

[10] Helmut Alt, Peter Braß, Michael Godau, Christian Knauer and Carola Wenk. "Computing the Hausdorff Distance of Geometric Patterns and Shapes". In: *Discrete and Computational Geometry: The Goodman-Pollack Festschrift.* Ed. by Boris Aronov, Saugata Basu, János Pach and Micha Sharir. Vol. 25. Algorithms and Combinatorics. Berlin, Heidelberg: Springer, 2003, pp. 65–76. DOI: 10.1007/978-3-642-55566-4_4.

[11] Helmut Alt and Leonidas J. Guibas. "Discrete Geometric Shapes: Matching, Interpolation, and Approximation". In: *Handbook of Computational Geometry.* Ed. by Jörg-Rüdiger Sack and Jorge Urrutia. Amsterdam: Elsevier, 2000, pp. 121–153. DOI: B978-044482537-7/50004-8.

[12] Raman Arora, Amitabh Basu, Poorya Mianjy and Anirbit Mukherjee. "Understanding Deep Neural Networks with Rectified Linear Units". In: *International Conference on Learning Representations (ICLR 2018).* 2018. URL: https://openreview.net/forum?id=B1J_rgWRW.

[13] Mikhail J. Atallah. "A Linear Time Algorithm for the Hausdorff Distance Between Convex Polygons". In: *Information Processing Letters* 17.4 (1983), pp. 207–209. DOI: 10.1016/0020-0190(83)90042-X.

[14] Theodore Baker, John Gill and Robert Solovay. "Relativizations of the P = NP Question". In: *SIAM Journal on Computing* 4.4 (1975), pp. 431–442. DOI: 10.1137/0204037.

[15] Ainesh Bakshi, Rajesh Jayaram and David P. Woodruff. "Learning Two Layer Rectified Neural Networks in Polynomial Time". In: *Proceedings of the Thirty-Second Conference on Learning Theory (COLT 2019).* Ed. by Alina Beygelzimer and Daniel Hsu. Vol. 99. Proceedings of Machine Learning Research. 2019, pp. 195–268. URL: https://proceedings.mlr.press/v99/bakshi19a.html.

[16] Sauguta Basu, Richard Pollack and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry.* 2nd ed. Vol. 10. Algorithms and Computation in Mathematics. Springer, 2006. ISBN: 3-540-33098-4. DOI: 10.1007/3-540-33099-2.

[17] Sauguta Basu and Marie-Françoise Roy. "Bounding the radii of balls meeting every connected component of semi-algebraic sets". In: *Journal of Symbolic Computation* 45.12 (2010), pp. 1270–1279. DOI: 10.1016/j.jsc.2010.06.009.

[18] Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow and Simon Weber. "Training Fully Connected Neural Networks is $\exists\mathbb{R}$-Complete". In: *Advances in Neural Information Processing Systems (NeurIPS 2023).* Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt and Sergey Levine. Vol. 36. 2023, pp. 36222–36237. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/71c31ebf577ffdad5f4a74156daad518-Paper-Conference.pdf.

[19] Nicholas Bieker, Thomas Bläsius, Emil Dohse and Paul Jungeblut. “Recognizing Unit Disk Graphs in Hyperbolic Geometry is ∃ℝ-Complete”. In: *Proceedings of the 39th European Workshop on Computational Geometry (EuroCG 2023)*. Ed. by Clemens Huemer and Carlos Seara. 2023, 35:1–35:8. DOI: 10.48550/arXiv.2301.05550.

[20] Daniel Bienstock. “Some Provably Hard Crossing Number Problems”. In: *Discrete & Computational Geometry* 6.3 (1991), pp. 443–459. DOI: 10.1007/BF02574701.

[21] Daniel Bienstock, Gonzalo Muñoz and Sebastian Pokutta. “Principled Deep Neural Network Training through Linear Programming”. In: *Discrete Optimization* 49 (2023). DOI: 10.1016/j.disopt.2023.100795.

[22] Vittorio Bilò and Marios Mavronicolas. “∃ℝ-complete Decision Problems about (Symmetric) Nash Equilibria in (Symmetric) Multi-player Games”. In: *ACM Transactions on Economics and Computation* 9.3 (2021), 14:1–14:25. DOI: 10.1145/3456758.

[23] Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann and Daniel Stephan. “Strongly Hyperbolic Unit Disk Graphs”. In: *40th International Symposium on Theoretical Aspects of Computer Science (STACS 2023)*. Ed. by Petra Berenbrink, Patricia Bouyer, Anuj Dawar and Mamadou Moustapha Kanté. Vol. 254. Leibniz International Proceedings in Informatics (LIPIcs). 2023, 13:1–13:17. DOI: 10.4230/LIPIcs.STACS.2023.13.

[24] Thomas Bläsius, Tobias Friedrich and Anton Krohmer. “Cliques in Hyperbolic Random Graphs”. In: *Algorithmica* 80.8 (2018), pp. 2324–2344. DOI: 10.1007/s00453-017-0323-3.

[25] Avrim L. Blum and Ronald L. Rivest. “Training a 3-Node Neural Network is NP-Complete”. In: *Neural Networks* 5.1 (1992), pp. 117–127. DOI: 10.1016/S0893-6080(05)80010-3.

[26] Lenore Blum, Mike Shub and Steve Smale. “On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness, Recursive Functions and Universal Machines”. In: *Bulletin of the American Mathematical Society* 21 (1989), pp. 1–46. DOI: 10.1090/S0273-0979-1989-15750-9.

[27] Michel Bode, Nikolaos Fountoulakis and Tobias Müller. “On the Largest Component of a Hyperbolic Model of Complex Networks”. In: *Electronic Journal of Combinatorics* 22.3 (2015), P3.24:1–P3.24:52. DOI: 10.37236/4958.

[28] Tobias Boege. “The Gaussian conditional independence inference problem”. PhD thesis. tto-von-Guericke-Universität Magdeburg, 2022. DOI: 10.25673/86275.

[29] Digvijay Boob, Santanu S. Dey and Guanghui Lan. “Complexity of Training ReLU Neural Network”. In: *Discrete Optimization* 44.1 (2022). DOI: 10.1016/j.disopt.2020.100620.

[30] Glen E. Bredon. *Topology and Geometry.* 1st ed. Vol. 139. Graduate Texts in Mathematics. New York: Springer, 2013. ISBN: 978-0-387-97926-7. DOI: 10.1007/978-1-4757-6848-0.

[31] Heinz Breu and David G. Kirkpatrick. “Unit disk graph recognition is NP-hard”. In: *Computational Geometry* 9.1–2 (1998), pp. 3–24. DOI: 10.1016/S0925-7721(97)00014-X.

[32] Karl Bringmann and André Nusser. “Translating Hausdorff Is Hard: Fine-Grained Lower Bounds for Hausdorff Distance Under Translation”. In: *Journal of Computational Geometry* 13.2 (2022), pp. 30–50. DOI: 10.20382/jocg.v13i2a3.

[33] Alon Brutzkus and Amir Globerson. “Globally Optimal Gradient Descent for a ConvNet with Gaussian Inputs”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML 2017).* Ed. by Diona Precup and Yee W. Teh. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 605–614. URL: https://proceedings.mlr.press/v70/brutzkus17a.html.

[34] Peter Bürgisser and Felipe Cucker. “Exotic Quantifiers, Complexity Classes, and Complete Problems”. In: *Foundations of Computational Mathematics* 9.2 (2009), pp. 135–170. DOI: 10.1007/s10208-007-9006-9.

[35] Sergio Cabello and Miha Jejčič. “Refining the Hierarchies of Classes of Geometric Intersection Graphs”. In: *Electronic Journal of Combinatorics* 24.1 (2017), P1.33:1–P1.33:19. DOI: 10.37236/6040.

[36] James W. Cannon, William J. Floyd, Richard Kenyon and Walter R. Parry. “Hyperbolic Geometry”. In: *Flavors of Geometry.* Ed. by Silvio Levy. Vol. 31. MSRI Publications. Cambridge University Press, 1977, pp. 59–115. URL: http://library.msri.org/books/Book31/index.html.

[37] John Canny. “Some Algebraic and Geometric Computations in PSPACE”. In: *STOC ’88: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing.* 1988, pp. 460–467. DOI: 10.1145/62212.62257.

[38] Jean Cardianl, Stefan Felsner, Tillmann Miltzow, Casey Tompkins and Birgit Vogtenhuber. “Intersection Graphs of Rays and Grounded Segments”. In: *Journal of Graph Algorithms and Applications* 22.2 (2018), pp. 273–295. DOI: 10.7155/jgaa.00470.

[39] Jean Cardinal and Udo Hoffmann. “Recognition and Complexity of Point Visibility Graphs”. In: *Discrete & Computational Geometry* 57.1 (2017), pp. 164–178. DOI: 10.1007/s00454-016-9831-1.

[40] Jean Cardinal and Vincent Kusters. “The Complexity of Simultaneous Geometric Graph Embedding”. In: *Journal of Graph Algorithms and Applications* 19.1 (2015), pp. 259–272. DOI: 10.7155/jgaa.00356.

[41] Sitan Chen, Aravind Gollakota, Adam Klivans and Raghu Meka. "Hardness of Noise-Free Learning for Two-Hidden-Layer Neural Networks". In: *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. Ed. by Sanmi Koyejo, Shakir Mohamed, Alekh Agarwal, Danielle Belgrave, Kyunghyun Cho and Alice Oh. Vol. 35. 2022, pp. 10709–10724. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/45a7ca247462d9e465ee88c8a302ca70-Paper-Conference.pdf.

[42] Sitan Chen, Adam R. Klivans and Raghu Meka. "Learning Deep ReLU Networks Is Fixed-Parameter Tractable". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 696–707. DOI: 10.1109/FOCS52979.2021.00073.

[43] Roman Chernobelskiy, Kathryn I. Cunningham, Michael T. Goodrich, Stephen G. Kobourov and Lowell Trott. "Force-Directed Lombardi-Style Graph Drawing". In: *Graph Drawing (GD 2011)*. Ed. by Marc van Krefeld and Bettina Speckmann. Vol. 7034. Lecture Notes in Computer Science. 2012, pp. 320–331. DOI: 10.1007/978-3-642-25878-7_31.

[44] George E. Collins. "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition". In: *2nd GI Conference on Automata Theory and Formal Languages*. Ed. by Gerhard Goos and Juris Hartmanis. Vol. 33. Lecture Notes in Computer Science. 1975, pp. 134–183. DOI: 10.1007/3-540-07407-4_17.

[45] Harold S. M. Coxeter. *Introduction to Geometry*. 2nd ed. Wiley Classics Library. Wiley, 1989. ISBN: 978-0-471-50458-0. URL: https://www.wiley.com/en-ie/Introduction+to+Geometry,+2nd+Edition-p-9780471504580.

[46] George Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274.

[47] Julian D'Costa, Engel Lefaucheux, Eike Neumann, Joël Ouaknine and James Worrel. "On the Complexity of the Escape Problem for Linear Dynamical Systems over Compact Semialgebraic Sets". In: *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*. Ed. by Filippo Bonchi and Simon J. Puglisi. Vol. 202. Leibniz International Proceedings in Informatics (LIPIcs). 2021, 33:1–33:21. DOI: 10.4230/LIPIcs.MFCS.2021.33.

[48] James H. Davenport and Joos Heintz. "Real Quantifier Elimination is Doubly Exponential". In: *Journal of Symbolic Computation* 5.1–2 (1988), pp. 29–35. DOI: 10.1016/S0747-7171(88)80004-X.

[49] Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos and Paul G. Spirakis. "Approximating the Existential Theory of the Reals". In: *Journal of Computer and System Sciences* 125 (2022), pp. 106–128. DOI: 10.1016/j.jcss.2021.11.002.

[50] Steffen Dereich and Sebastian Kassing. “On Minimal Representations of Shallow ReLU Networks”. In: *Neural Networks* 148 (2022), pp. 121–128. DOI: 10.1016/j.neunet.2022.01.006.

[51] Pedro J. de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller and Davi C. Tozoni. “Engineering Art Galleries”. In: *Algorithm Engineering: Selected Results and Surveys.* Ed. by Lasse Kliemann and Peter Sanders. 1st ed. Vol. 9220. Lecture Notes in Computer Science. Springer, 2016, pp. 379–417. DOI: 10.1007/978-3-319-49487-6.

[52] Santanu S. Dey, Guany Wang and Yao Xie. “Approximation Algorithms for Training One-Node ReLU Neural Networks”. In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 6696–6706. DOI: 10.1109/TSP.2020.3039360.

[53] Ilias Diakonikolas, Surbhi Goel, Sushrut Karmalkar, Adam R. Klivans and Mahdi Soltanolkotabi. “Approximation Schemes for ReLU Regression”. In: *Proceedings of Thirty Third Conference on Learning Theory (COLT 2020).* Ed. by Jacob Abernethy and Shivani Agarwal. Vol. 125. Proceedings of Machine Learning Research. 2020, pp. 1452–1485. URL: https://proceedings.mlr.press/v125/diakonikolas20b.html.

[54] Michael G. Dobbins, Linda Kleist, Tillmann Miltzow and Paweł Rzążewski. “Completeness for the Complexity Class $\forall\exists\mathbb{R}$ and Area-Universality”. In: *Discrete & Computational Geometry* (2022). DOI: 10.1007/s00454-022-00381-0.

[55] Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, Maarten Löffler and Martin Nöllenburg. “Planar and Poly-Arc Lombardi Drawings”. In: *Journal of Computational Geometry* 9.1 (2018), pp. 328–355. DOI: 10.20382/jocg.v9i1a11.

[56] Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov and Martin Nöllenburg. “Lombardi Drawings of Graphs”. In: *Journal of Graph Algorithms and Applications* 16.1 (2012), pp. 85–108. DOI: 10.7155/jgaa.00251.

[57] Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov and Martin Nöllenburg. “Drawing Trees with Perfect Angular Resolution and Polynomial Area”. In: *Discrete & Computational Geometry* 49.2 (2013), pp. 157–182. DOI: 10.1007/s00454-012-9472-y.

[58] Ronen Eldan and Ohad Shamir. “The Power of Depth for Feedforward Neural Networks”. In: *29th Annual Conference on Learning Theory (COLT 2016).* Ed. by Vitaly Feldman, Alexander Rakhlin and Ohad Shamir. Vol. 49. Proceedings of Machine Learning Research. 2016, pp. 907–940. URL: https://proceedings.mlr.press/v49/eldan16.html.

[59] David Eppstein. “A Möbius-Invariant Power Diagram and Its Applications to Soap Bubbles and Planar Lombardi Drawing”. In: *Discrete & Computational Geometry* 52.3 (2014), pp. 515–550. DOI: 10.1007/s00454-014-9627-0.

[60] David Eppstein. “Simple Recognition of Halin Graphs and Their Generalizations”. In: *Journal of Graph Algorithms and Applications* 20.2 (2016), pp. 323–346. DOI: 10.7155/jgaa.00395.

[61] David Eppstein. “Bipartite and Series-Parallel Graphs Without Planar Lombardi Drawings”. In: *Journal of Graph Algorithms and Applications* 25.1 (2021), pp. 549–562. DOI: 10.7155/jgaa.00571.

[62] David Eppstein. “Limitations on Realistic Hyperbolic Graph Drawing”. In: *Graph Drawing and Network Visualization (GD 2021)*. Ed. by Helen C. Purchase and Ignaz Rutter. Vol. 12868. Lecture Notes in Computer Science. 2021, pp. 343–357. DOI: 10.1007/978-3-030-92931-2_25.

[63] David Eppstein, Daniel Frishberg and Martha C. Osegueda. “Angles of Arc-Polygons and Lombardi Drawings of Cacti”. In: *Computational Geometry* 112 (2023). DOI: 10.1016/j.comgeo.2023.101982.

[64] Jeff Erickson, Ivor van der Hoog and Tillmann Miltzow. “Smoothing the Gap Between NP and ER”. In: *SIAM Journal on Computing* (2022), FOCS20-102–FOCS20-138. DOI: 10.1137/20M1385287.

[65] Stefan Felsner and Manfred Scheucher. “Arrangements of Pseudocircles: On Circularizability”. In: *Discrete & Computational Geometry* 64.3 (2019). Ricky Pollack Memorial Issue, pp. 776–813. DOI: 10.1007/s00454-019-00077-y.

[66] Vincent Froese and Christoph Hertrich. “Training Neural Networks is NP-Hard in Fixed Dimension”. In: *Advances in Neural Information Processing Systems (NeurIPS 2023)*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt and Sergey Levine. Vol. 36. 2023, pp. 44039–44049. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/8948a8d039ed52d1031db6c7c2373378-Paper-Conference.pdf.

[67] Vincent Froese, Christoph Hertrich and Rolf Niedermeier. “The Computational Complexity of ReLU Network Training Parameterized by Data Dimensionality”. In: *Journal of Artificial Intelligence Research* 74 (2022). DOI: 10.1613/jair.1.13547.

[68] Jugal Garg, Ruta Mehta, Vijay V. Vazirani and Sadra Yazdanbod. “$\exists\mathbb{R}$-Completeness for Decision Versions of Multi-Player (Symmetric) Nash Equilibria”. In: *ACM Transactions on Economics and Computation* 6.1 (2018), 1:1–1:23. DOI: 10.1145/3175494.

[69] Xavier Glorot, Antoine Bordes and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*. Ed. by Geoffrey Gordon, David Dunson and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. 2011, pp. 315–323. URL: https://proceedings.mlr.press/v15/glorot11a.html.

[70] Surbhi Goel, Varun Kanade, Adam R. Klivans and Justin Thaler. “Reliably Learning the ReLU in Polynomial Time”. In: *Proceedings of the 2017 Conference on Learning Theory (COLT 2017)*. Ed. by Satyen Kale and Ohad Shamir. Vol. 65. Proceedings of Machine Learning Research. 2017, pp. 1004–1042. URL: https://proceedings.mlr.press/v65/goel17a.html.

[71] Surbhi Goel and Adam R. Klivans. “Learning Neural Networks with Two Nonlinear Layers in Polynomial Time”. In: *Proceedings of the Thirty-Second Conference on Learning Theory (COLT 2019)*. Ed. by Alina Beygelzimer and Daniel Hsu. Vol. 99. Proceedings of Machine Learning Research. 2019, pp. 1470–1499. URL: https://proceedings.mlr.press/v99/goel19b.html.

[72] Surbhi Goel, Adam R. Klivans, Pasin Manurangsi and Daniel Reichman. “Tight Hardness Results for Training Depth-2 ReLU Networks”. In: *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Ed. by James R. Lee. Vol. 185. Leibniz International Proceedings in Informatics (LIPIcs). 2021, 22:1–22:14. DOI: 10.4230/LIPIcs.ITCS.2021.22.

[73] Surbhi Goel, Adam R. Klivans and Raghu Meka. “Learning One Convolutional Layer with Overlapping Patches”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 1783–1791. URL: https://proceedings.mlr.press/v80/goel18a.html.

[74] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613. URL: http://www.deeplearningbook.org.

[75] Jacob E. Goodman and Richard Pollack. “Proof of Grünbaum’s Conjecture on the Stretchability of Certain Arrangements of Pseudolines”. In: *Journal of Combinatorial Theory, Series A* 29.3 (1980), pp. 385–390. DOI: 10.1016/0097-3165(80)90038-2.

[76] Henry Gouk, Eibe Frank, Bernhard Pfahringer and Michael J. Cree. “Regularisation of neural networks by enforcing Lipschitz continuity”. In: *Machine Learning* 110.2 (2021), pp. 393–416. DOI: 10.1007/s10994-020-05929-w.

[77] Dmitrii Y. Grigor’ev and Nicolai N. Vorobjov. “Solving Systems of Polynomial Inequalities in Subexponential Time”. In: *Journal of Symbolic Computation* 5.1–2 (1988), pp. 37–64. DOI: 10.1016/S0747-7171(88)80005-1.

[78] Luca Gugelmann, Konstantinos Panagiotou and Ueli Peter. “Random Hyperbolic Graphs: Degree Sequence and Clustering”. In: *International Colloquium on Automata, Languages, and Programming (ICALP 2012)*. Ed. by Artur Czumaj, Kurt Mehlhorn, Andrew Pitts and Roger Wattenhofer. Vol. 7392. Lecture Notes in Computer Science. 2012, pp. 573–585. DOI: 10.1007/978-3-642-31585-5_51.

[79] Christian Haase, Christoph Hertrich and Georg Loho. “Lower Bounds on the Depth of Integral ReLU Neural Networks via Lattice Polytopes”. In: *The Eleventh International Conference on Learning Representations (ICLR 2023)*. 2023. URL: https://openreview.net/forum?id=2mvALOAWaxY.

[80] Boris Hanin. "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations". In: *Mathematics* 7.10 (2019), 992:1–9. DOI: 10.3390/math7100992.

[81] Boris Hanin and David Rolnick. "Complexity of Linear Regions in Deep Networks". In: *Proceedings of the 36th International Conference on Machine Learning (ICML 2019).* Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 2596–2604. URL: https://proceedings.mlr.press/v97/hanin19a.html.

[82] Boris Hanin and Mark Sellke. "Approximating Continuous Functions by ReLU Nets of Minimal Width". arXiv preprint. 2018. DOI: 10.48550/arXiv.1710.11278.

[83] Teemu Hankala, Miika Hannula, Juha Kontinen and Jonni Virtema. "Complexity of Neural Network Training and ETR: Extensions with Effectively Continuous Functions". arXiv preprint. 2023. DOI: 10.48550/arXiv.2305.11833.

[84] Felix Hausdorff. *Grundzüge der Mengenlehre.* Leipzig: Von Veit & Company, 1914.

[85] Christoph Hertrich, Amitabh Basu, Marco Di Summa and Martin Skutella. "Towards Lower Bounds on the Depth of ReLU Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS 2021).* Ed. by Marc A. Ranzato, Alina Beygelzimer, Yann Dauphin, Percy Liang and Jennifer W. Vaughan. Vol. 34. 2021. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/1b9812b99fe2672af746cefda86be5f9-Paper.pdf.

[86] Christoph Hertrich and Leon Sering. "ReLU Neural Networks of Polynomial Size for Exact Maximum Flow Computation". In: *Integer Programming and Combinatorial Optimization (IPCO 2023).* Ed. by Alberto Del Pia and Volker Kaibel. Vol. 13904. Lecture Notes in Computer Science. 2023, pp. 187–202. DOI: 10.1007/978-3-031-32726-1_14.

[87] David Hilbert. *Grundlagen der Geometrie.* 13th ed. Teubner, 1968. DOI: 10.1007/978-3-322-92726-2.

[88] Robert Hobbs. *Mark Lombardi: Global Networks.* Independent Curators International, 2003.

[89] Udo Hoffmann. "On the Complexity of the Planar Slope Number Problem". In: *Journal of Graph Algorithms and Applications* 21.2 (2017), pp. 183–193. DOI: 10.7155/jgaa.00411.

[90] Kurt Hornik. "Approximation Capabilities of Multilayer Feedforward Networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. DOI: 10.1016/0893-6080(91)90009-T.

[91] Joey Huchette, Gonzalo Muñoz, Tiago Serra and Calvin Tsay. “When Deep Learning Meets Polyhedral Theory: A Survey”. arXiv preprint. 2023. DOI: 10.48550/arXiv.2305.00241.

[92] Paul Jungeblut. “On the Complexity of Lombardi Graph Drawing”. In: *Graph Drawing and Network Visualization (GD 2023)*. Ed. by Michael A. Bekos and Markus Chimani. Vol. 14465. Lecture Notes in Computer Science. 2024, pp. 180–194. DOI: 10.1007/978-3-031-49272-3_13.

[93] Paul Jungeblut, Linda Kleist and Tillmann Miltzow. “The Complexity of the Hausdorff Distance”. In: *Proceedings of the 38th European Workshop on Computational Geometry (EuroCG 2022)*. Ed. by Fabrizio Di Giacomo Emilio Montecchiani. 2022, 1:1–1:7. URL: https://eurocg2022.unipg.it/booklet/EuroCG2022-Booklet.pdf.

[94] Paul Jungeblut, Linda Kleist and Tillmann Miltzow. “The Complexity of the Hausdorff Distance”. In: *38th International Symposium on Computational Geometry (SoCG 2022)*. Ed. by Xavier Goaoc and Michael Kerber. Vol. 224. Leibniz International Proceedings in Informatics (LIPIcs). 2022, 48:1–48:17. DOI: 10.4230/LIPIcs.SoCG.2022.48.

[95] Paul Jungeblut, Linda Kleist and Tillmann Miltzow. “The Complexity of the Hausdorff Distance”. In: *Discrete & Computational Geometry* 71.1 (2024), pp. 177–213. DOI: 10.1007/s00454-023-00562-5.

[96] Tim Junginger. “Robustness of the Discrete Real Polynomial Hierarchy”. Bachelor’s thesis. Karlsruhe Institute of Technology, 2023. URL: https://i11www.iti.kit.edu/_media/teaching/theses/ba-junginger-23.pdf.

[97] Ross J. Kang and Tobias Müller. “Sphere and Dot Product Representations of Graphs”. In: *Discrete & Computational Geometry* 47.3 (2012), pp. 548–568. DOI: 10.1007/s00454-012-9394-8.

[98] Sammy Khalife, Hongyu Cheng and Amitabh Basu. “Neural networks with linear threshold activations: structure and algorithms”. In: *Mathematical Programming* (2023). DOI: 10.1007/s10107-023-02016-5.

[99] Philipp Kindermann, Stephen G. Kobourov, Maarten Löffler, Martin Nöllenburg, André Schulz and Birgit Vogtenhuber. “Lombardi Drawings of Knots and Links”. In: *Journal of Computational Geometry* 10.1 (2018), pp. 444–476. DOI: 10.20382/jocg.v10i1a15.

[100] Sándor Kisfaludi-Bak. “Hyperbolic Intersection Graphs and (Quasi)-Polynomial Time”. In: *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2020, pp. 1621–1638. DOI: 10.1137/1.9781611975994.100.

[101] Pascal Koiran. “Hilbert’s Nullstellensatz Is in the Polynomial Hierarchy”. In: *Journal of Complexity* 12.4 (1996), pp. 273–286. DOI: 10.1006/jcom.1996.0019.

[102] Mickaël Kourganoff. “Universality theorems for linkages in homogeneous surfaces”. In: *Geometriae Dedicata* 185.1 (2016), pp. 35–85. DOI: 10.1007/s10711-016-0168-y.

[103] Jan Kratochvíl and Jiří Matoušek. “Intersection Graphs of Segments”. In: *Journal of Combinatorial Theory, Series B* 62.2 (1994), pp. 289–315. DOI: 10.1006/jctb.1994.1071.

[104] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat and Marián Boguñá. “Hyperbolic Geometry of Complex Networks”. In: *Physical Review E* 82.3 (2010), 036106:1–036106:18. DOI: 10.1103/PhysRevE.82.036106.

[105] Jan Kynčl. “Simple Realizability of Complete Abstract Topological Graphs in *P*”. In: *Discrete & Computational Geometry* 45.3 (2011), pp. 383–399. DOI: 10.1007/s00454-010-9320-x.

[106] Shiyu Liang and Rayadurgam Srikant. “Why Deep Neural Networks for Function Approximation?” In: *International Conference on Learning Representations (ICLR 2017)*. 2017. URL: https://openreview.net/forum?id=SkpSlKIel.

[107] Anna Lubiw, Tillmann Miltzow and Debajyoti Mondal. “The Complexity of Drawing a Graph in a Polygonal Region”. In: *Journal of Graph Algorithms and Applications* 26.4 (202), pp. 387–401. DOI: 10.7155/jgaa.00602.

[108] Harry G. Mairson and Jorge Stolfi. “Reporting and Counting Intersections Between Two Sets of Line Segments”. In: *Theoretical Foundations of Computer Graphics and CAD*. Ed. by Rae A. Earnshaw. Vol. 40. NATO ASI Subseries F. 1988, pp. 307–325. DOI: 10.1007/978-3-642-83539-1_11.

[109] Yuri V. Matiyasevich. “Enumerable Sets are Diophantine”. In: *Doklady Akademii Nauk SSSR* 191.2 (1970), pp. 279–282.

[110] Jiří Matoušek. “Intersection graphs of segments and $\exists\mathbb{R}$”. arXiv preprint. 2014. DOI: 10.48550/arXiv.1406.2636.

[111] Colin McDiarmid and Tobias Müller. “The Number of Bits Needed to Represent a Unit Disk Graph”. In: *Graph Theoretic Concepts in Computer Science (WG 2010)*. Ed. by Dimitrios M. Thilikos. Vol. 6410. Lecture Notes in Computer Science. 2010, pp. 315–323. DOI: 10.1007/978-3-642-16926-7_29.

[112] Colin McDiarmid and Tobias Müller. “Integer realizations of disk and segment graphs”. In: *Journal of Combinatorial Theory, Series B* 103.1 (2013), pp. 114–143. DOI: 10.1016/j.jctb.2012.09.004.

[113] Tillmann Miltzow and Reinier F. Schmiermann. “On Classifying Continuous Constraint Satisfaction Problems”. In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 781–791. DOI: 10.1109/FOCS52979.2021.00081.

[114] Deng Min, Li Zhilin and Chen Xiaoyong. “Extended Hausdorff distance for spatial objects in GIS”. In: *International Journal of Geographical Information Science* 21.4 (2007), pp. 459–475. DOI: 10.1080/13658810601073315.

[115] Nikolai E. Mnëv. “The Universality Theorems on the Classification Problem of Configuration Varieties and Convex Polytopes Varieties”. In: *Topology and Geometry — Rohlin Seminar*. Ed. by Oleg Y. Viro and Anatoly M. Vershik. Vol. 1346. Lecture Notes in Mathematics. Berlin, Heidelberg: Springer, 1988, pp. 527–543. DOI: 10.1007/BFb0082792.

[116] Guido Montúfar, Yue Ren and Leon Zhang. “Sharp Bounds for the Number of Regions of Maxout Networks and Vertices of Minkowski Sums”. In: *SIAM Journal on Applied Algebra and Geometry* 6.4 (2022), pp. 618–649. DOI: 10.1137/21M1413699.

[117] Guido F. Montúfar, Razvan Pascanu, Kyunghyun Cho and Yoshua Bengio. “On the Number of Linear Regions of Deep Neural Networks”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems (NeurIPS 2014)*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil Lawrence and Weinberger Kilian Q. Vol. 27. 2014, pp. 2924–2932. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/109d2dd3608f669ca17920c511c2a41e-Paper.pdf.

[118] Anirbit Mukherjee and Amitabh Basu. “Lower bounds over Boolean inputs for deep neural networks with ReLU gates”. arXiv preprint. 2017. DOI: 10.48550/arXiv.1711.03073.

[119] Tobias Müller and Merlijn Staps. “The Diameter of KPKVB Random Graphs”. In: *Advances in Applied Probability* 51.2 (2019), pp. 358–377. DOI: 10.1017/apr.2019.23.

[120] Quynh Nguyen, Mahesh Chandra Mukkamala and Matthias Hein. “Neural Networks Should Be Wide Enough to Learn Disconnected Decision Regions”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 3740–3749. URL: https://proceedings.mlr.press/v80/nguyen18b.html.

[121] Yoshio Okamoto, Alexander Ravsky and Alexander Wolff. “Variants of the Segment Number of a Graph”. In: *Graph Drawing and Network Visualization (GD 2019)*. Ed. by Daniel Archambault and Csaba D. Tóth. Vol. 11904. Lecture Notes in Computer Science. 2019, pp. 430–443. DOI: 10.1007/978-3-030-35802-0_33.

[122] Razvan Pascanu, Guido Montúfar and Yoshua Bengio. “On the number of response regions of deep feed forward networks with piece-wise linear activations”. In: *International Conference on Learning Representations (ICLR 2014)*. 2014. URL: https://openreview.net/forum?id=bSaT4mmQt84Lx.

[123] Grant O. Passmore and Paul B. Jackson. “Combined Decision Techniques for the Existential Theory of the Reals”. In: *International Conference on Intelligent Computer Mathematics (CICM 2009)*. Vol. 5625. Lecture Notes in Computer Science. 2009, pp. 122–137. DOI: 10.1007/978-3-642-02614-0_14.

[124] Helen Purchase, John Hamer, Martin Nöllenburg and Stephen G. Kobourov. “On the Usability of Lombardi Graph Drawings”. In: *Graph Drawing (GD 2012)*. Ed. by Walter Didimo and Maurizio Patrignani. Vol. 7704. Lecture Notes in Computer Science. 2013, pp. 451–462. DOI: 10.1007/978-3-642-36763-2_40.

[125] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli and Jascha Sohl Dickstein. “On the Expressive Power of Deep Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 2847–2854. URL: https://proceedings.mlr.press/v70/raghu17a.html.

[126] James Renegar. “On the Computational Complexity and Geometry of the First-order Theory of the Reals. Part I: Introduction. Preliminaries. The Geometry of Semi-algebraic Sets. The Decision Problem for the Existential Theory of the Reals”. In: *Journal of Symbolic Computation* 13.3 (1992), pp. 255–299. DOI: 10.1016/S0747-7171(10)80003-3.

[127] James Renegar. “On the Computational Complexity and Geometry of the First-order Theory of the Reals. Part II: The General Decision Problem. Preliminaries for Quantifier Elimination”. In: *Journal of Symbolic Computation* 13.3 (1992), pp. 301–327. DOI: 10.1016/S0747-7171(10)80004-5.

[128] James Renegar. “On the Computational Complexity and Geometry of the First-order Theory of the Reals. Part III: Quantifier Elimination”. In: *Journal of Symbolic Computation* 13.3 (1992), pp. 329–352. DOI: 10.1016/S0747-7171(10)80005-7.

[129] Daniel Richardson. “Some Undecidable Problems Involving Elementary Functions of a Real Variable”. In: *The Journal of Symbolic Logic* 33.4 (1969), pp. 514–520. DOI: 10.2307/2271358.

[130] Jürgen Richter-Gebert. “Mnëv’s Universality Theorem revisited”. 2002. URL: https://geo.ma.tum.de/_Resources/Persistent/3/e/a/2/3ea2ad59228a1a24a67d1e994fa77266a599e73a/15_MnevsUniversalityhTheorem.pdf.

[131] William Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance*. 1st ed. Vol. 1173. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1996. ISBN: 978-3-540-61993-2. DOI: 10.1007/BFb0015091.

[132] Itay Safran and Ohad Shamir. "Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 2979–2987. URL: https://proceedings.mlr.press/v70/safran17a.html.

[133] Marcus Schaefer. "Complexity of Some Geometric and Topological Problems". In: *Graph Drawing (GD 2009)*. Ed. by David Eppstein and Emden R. Gansner. Vol. 5849. Lecture Notes in Computer Science. 2010, pp. 334–344. DOI: 10.1007/978-3-642-11805-0_32.

[134] Marcus Schaefer. "Realizability of Graphs and Linkages". In: *Thirty Essays on Geometric Graph Theory*. Ed. by János Pach. Springer, 2013, pp. 461–482. DOI: 10.1007/978-1-4614-0110-0_24.

[135] Marcus Schaefer. "Complexity of Geometric k-Planarity for Fixed k". In: *Journal of Graph Algorithms and Applications* 25.1 (2021), pp. 29–41. DOI: 10.7155/jgaa.00548.

[136] Marcus Schaefer. "On the Complexity of Some Geometric Problems With Fixed Parameters". In: *Journal of Graph Algorithms and Applications* 25.1 (2021), pp. 195–218. DOI: 10.7155/jgaa.00557.

[137] Marcus Schaefer. "RAC-Drawability is $\exists\mathbb{R}$-complete and Related Results". In: *Journal of Graph Algorithms and Applications* 27.9 (2023), pp. 803–841. DOI: 10.7155/jgaa.00646.

[138] Marcus Schaefer and Daniel Štefankovič. "Fixed Points, Nash Equilibria, and the Existential Theory of the Reals". In: *Theory of Computing Systems* 60 (2017), pp. 172–193. DOI: 10.1007/s00224-015-9662-0.

[139] Marcus Schaefer and Daniel Štefankovič. "Beyond the Existential Theory of the Reals". In: *Theory of Computing Systems* 68.2 (2023), pp. 195–226. DOI: 10.1007/s00224-023-10151-x.

[140] Hans Schwerdtfeger. *Geometry of Complex Numbers*. Dover Publications, Inc., 1979. ISBN: 978-0-486-63830-0.

[141] Thiago Serra, Christian Tjandraatmadja and Srikumar Ramalingam. "Bounding and Counting Linear Regions of Deep Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 4558–4566. URL: https://proceedings.mlr.press/v80/serra18b.html.

[142] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107298019.

[143] Yaroslav Shitov. "A Universality Theorem for Nonnegative Matrix Factorizations". arXiv preprint. 2018. DOI: 10.48550/arXiv.1606.09068.

[144] Peter W. Shor. “Stretchability of Pseudolines is NP-Hard.” In: *Applied Geometry And Discrete Mathematics, Proceedings of a DIMACS Workshop, Providence, Rhode Island, USA, September 18, 1990.* Ed. by Peter Gritzmann and Bernd Sturmfels. Vol. 4. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1991, pp. 531–554. DOI: 10.1090/dimacs/004/41.

[145] Pablo Solernó. “Effective Łojasiewicz Inequalities in Semialgebraic Geometry”. In: *Applicable Algebra in Engineering, Communication and Computing* 2.1 (1991), pp. 1–14. DOI: 10.1007/BF01810850.

[146] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry.* 2nd ed. University of California Press, 1951.

[147] Matus Telgarsky. “Benefits of depth in neural networks”. In: *29th Annual Conference on Learning Theory (COLT 2016).* Ed. by Vitaly Feldman, Alexander Rakhlin and Ohad Shamir. Vol. 49. Proceedings of Machine Learning Research. 2016, pp. 1517–1539. URL: https://proceedings.mlr.press/v49/telgarsky16.html.

[148] Balder ten Cate, Phokion G. Kolaitis and Walied Othman. “Data Exchange with Arithmetic Operations”. In: *Proceedings of the 16th International Conference on Extending Database Technology (EDBT 2013).* Ed. by Norman W. Paton, Giovanna Guerrini, Barbara Catania, Malu Castellanos, Paolo Atzeni, Piero Fraternali and Anastasios Gounaris. 2013, pp. 537–548. DOI: 10.1145/2452376.2452439.

[149] Leslie G. Valiant. “A Theory of the Learnable”. In: *Communications of the ACM* 27.11 (1984), pp. 1134–1142. DOI: 10.1145/1968.1972.

[150] Gal Vardi, Gilad Yehudai and Ohad Shamir. “On the Optimal Memorization Power of ReLU Neural Networks”. In: *International Conference on Learning Representations (ICLR 2022).* 2022. URL: https://openreview.net/forum?id=MkTPtnjeYTV.

[151] Floor Verhoeven, Amir Vaxman, Tim Hoffmann and Olga Sorkine-Hornung. “Dev2PQ: Planar Quadrilateral Strip Remeshing of Developable Surfaces”. In: *ACM Transactions on Graphics* 41.3 (2022), 29:1–29:18. DOI: 10.1145/3510002.

[152] Nicolai N. Vorob’ev. “Estimates of Real Roots of a System of Algebraic Equations”. In: *Journal of Soviet Mathematics* 34 (1986), pp. 1754–1762. DOI: 10.1007/BF01095637.

[153] Adrian Wurm. “Complexity of Reachability Problems in Neural Networks”. In: *Reachability Problems (RP 2023).* Ed. by Olivier Bournez, Enrico Formenti and Igor Potapov. Vol. 14235. Lecture Notes in Computer Science. 2023, pp. 15–27. DOI: 10.1007/978-3-031-45286-4_2.

[154] Dmitry Yarotsky. “Error bounds for approximations with deep ReLU networks”. In: *Neural Networks* 94 (2017), pp. 103–114. DOI: 10.1016/j.neunet.2017.07.002.

[155] Chulhee Yun, Suvrit Sra and Ali Jadbabaie. “Small ReLU networks are powerful memorizers: a tight analysis of memorization capacity”. In: *Advances in Neural Information Processing Systems (NeurIPS 2019)*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily Fox and Roman Garnett. Vol. 32. 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/dbea3d0e2a17c170c412c74273778159-Paper.pdf.

[156] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht and Oriol Vinyals. “Understanding Deep Learning (Still) Requires Rethinking Generalization”. In: *Communications of the ACM* 64.3 (2021), pp. 107–115. DOI: 10.1145/3446776.

[157] Liwen Zhang, Gregory Naitzat and Lek-Heng Lim. “Tropical Geometry of Deep Neural Networks”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 5824–5832. URL: https://proceedings.mlr.press/v80/zhang18i.html.

[158] Xiao-Dong Zhang. “Complexity Of Neural Network Learning In The Real Number Model”. In: *Workshop on Physics and Computation*. 1992, pp. 146–150. DOI: 10.1109/PHYCMP.1992.615511.