The Complexity of Finding Tarski Fixed Points

Master Thesis

May 15, 2024

Nils Jensen

Advised by

Prof. Dr. Bernd Gärtner

Sebastian Haslebacher

ETH zürich

Abstract

Insert the abstract here.

Contents

List of Figures

List of Tables

1 Introduction

Write the introduction here. This is a test.

Preliminaries 2

The aim of this chapter is to introduce the complexity class **TNFP**, and some of its subclasses, in particular **PPAD**, **PLS** and **EOPL**. We will also introduce the TARSKI problem.

2.1 Total search problems

The study of complexity classes originally works with so-called *decision-problems*, which are the question of deciding on the membership in a set — also called a *language*. Now while these problems are interesting, real world questions or problem often ask for an explicit anwser. For instance while deciding if a function has a global minimum is a decision problem, we are interrested in actually finding this minimum, which is not a decision problem.

Notable such problems include deciding on whether a boolean formula can be satified or if a k-Clique exist in a given graph.

Even though as we will see it can be transformed into one

This is where so called *search problems* come into play:

Definition 2.1 — Search Problem.
A *search problem* is given by a relation $R \subset \{0,1\}^* \times \{0,1\}^*$. For a given *instance* $I \in \{0,1\}^*$ the computational problem, to find a *solution* $s \in \{0,1\}^*$, that satisfies: $(I, s) \in R$ or output "No" if no such s exists.

The "No" case can be encoded as some special binary string.

Now of course we can view these search problems as decision problems by looking at the corresponding decision problem given by the language:

$$\mathcal{L}_R = \{I \in \{0,1\}^* \mid \exists s \in \{0,1\}^* : (I, s) \in R\}$$

We can then ask the classical complexity questions about these search problems, i.e. whether these search problems are in **P**? **NP**? whether they are **NP**-Hard? One easily observes that search problems are always at least as hard as just deciding whether a solution exist. This is because solving a search problem also solves the underlying decision problem. This leads to the natural question: what if we remove the underlying decision problem? This can be done by garanteeing that "No" is never a solution. We call these problems where every instance admits a solution *total search problems*.

Definition 2.2 — Total search problems.
A *total search problem* is a search problem given by a relation $R \subset \{0,1\}^* \times \{0,1\}^*$, such that for every given instance $I \in \{0,1\}^*$ there is a solution $s \in \{0,1\}^*$, that satisfies: $(I, s) \in R$.

The complexity class **TNFP** as introduced in [1] is simply the class of all total search problems that lie in **NP**. Examples of **TNFP** problems are:

[1]: Papadimitriou (1994), *On the complexity of the parity argument and other inefficient proofs of existence*

This means that **TNFP** can be seen as an intermediate class between **P** and **NP**, containing all search problems where a solution is guaranteed to exist, and where one can efficiently check the feasibility of a candidate solution.

- Factoring, the problem of finding the prime factors of a number,
- Nash, the problem of finding a nash equilibrium in a bimatrix game,
- Minimize, the problem of finding the global minimum of a convex function.

Similarly to decision problem we can also define reduction inside **TNFP**.

Definition 2.3 — Reduction.
For two problem $R, S \in$ **TNFP**, we say that R *reduces* (many to one) to S if there exist polynomial time computable functions $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that for $I, s \in \{0,1\}^*$: if $(f(I), s) \in S$ then $(I, g(I, s)) \in R$. This means that if s is a solution to an instance $f(I)$ in S, we can compute $g(I, s)$ a solution to an instance I in R

Saying *one can reduce R onto S* can be understood as saying *if one can solve S efficiently then I can solve R efficiently*.

2.2 An excursion into Binary Circuits

TODO

2.3 Subclasses of TNFP

Because the existence of complete **FNP**-Problems in **TNFP** would imply **NP** = **coNP**, as described in [2]. Because this is a very unexpected outcome we cannot expect to find complete problems in **TNFP**. This means that we should use other tools to study the structure of **TNFP**.

[2]: Megiddo et al. (1991), *On total functions, existence theorems and computational complexity*

One of the challenges is that **TNFP** is a so-called *semantic* class. By semantic class we mean a class for which it is difficult to check if that Turing Machine defines a language in this class. A *syntactic* class is a class for which it is easy to check that the accepted language of a Turing Machine indeed belongs to the

Examples of syntactic classes include **P** and **NP**.

class. These terms are defined in more detail in [3]. Hence we want to study syntactic subclasses of **TNFP**. One of the proposed methods [1] is to categorize total search problems with respect to the existence results which allow them to be *total*. This is what leads to the complexity classes we will discuss next.

[3]: Papadimitriou (1994), *Computational complexity*

[1]: Papadimitriou (1994), *On the complexity of the parity argument and other inefficient proofs of existence*

Polynomial Local Search (PLS)

The existence results which gives rise to **PLS** is "*every directed acyclic graph has a sink*". We can then construct the class **PLS** by defining it as all problems which reduce to finding the sink of a directed acyclic graph (DAG).

Formally we first define the problem LOCALOPT as in [4]:

[4]: Johnson et al. (1988), *How easy is local search?*

LOCALOPT
Input: Two binary circuits $P, S : [2^n] \to [2^n]$.
Output: A vertex $v \in [2^n]$ such that $P(S(v)) \geq P(v)$.

S can be seen as a proposed successor, and P as a potential. The goal is to find a local minima v of the potential.

One might ask why this is equivalent to finding the sink of a DAG? The circuit S defines a directed graph, which might contain cycles. Only keeping the edge on which the potential decreases (strictly) leads to a DAG, with as sinks exactly the v such that $P(S(v)) \geq P(v)$. Now we can define **PLS**:

Definition 2.4 — Polynomial Local Search (PLS).
The class **PLS** is the set of all **TNFP** problems that reduce to LOCALOPT.

One of the reasons we think that studying very "easy" problems such as **PLS** is that we strongly believe that there is no clever way of solving these problems without actually walking through the graph. Hence if we have a graph of exponentially large size it seems very unlikely that one can find an efficient way of solving the problem. Hence all problems in **PLS** can be though of as including the fundamental difficulty of no beeing able to do better than to walk along some graph.

By "easy" we mean that the problem can be solved by simply walking through the graph, and checking whether every vertex is a local minima.

Polynomial Parity Argument on Directed Graphs (PPAD)

Now we want to discuss the complexity class **PPAD**, introduced by Papadimitriou as one of the first syntatic subclasses of **TNFP** in [1]. The existence result giving rise to this class is that "*If a directed graph has an unbalanced vertex, then it has at least one other unbalanced vertex*". **PPAD** can be defined using the problem END-OF-LINE as introduced in [5].

[1]: Papadimitriou (1994), *On the complexity of the parity argument and other inefficient proofs of existence*

[5]: Daskalakis et al. (2009), *The Complexity of Computing a Nash Equilibrium*

END-OF-LINE
Input: Boolean circuit $S, P : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$ (0^n is a source.)
Output: An $x \in \{0,1\}^n$ such that either:
- $P(S(x)) \neq x$ (x is a sink) or
- $S(P(x)) \neq x \neq 0^n$ (x is a non non-standard source)

Here S can be thought of giving the successor of a vertex, and P as giving the predecessor of a vertex.

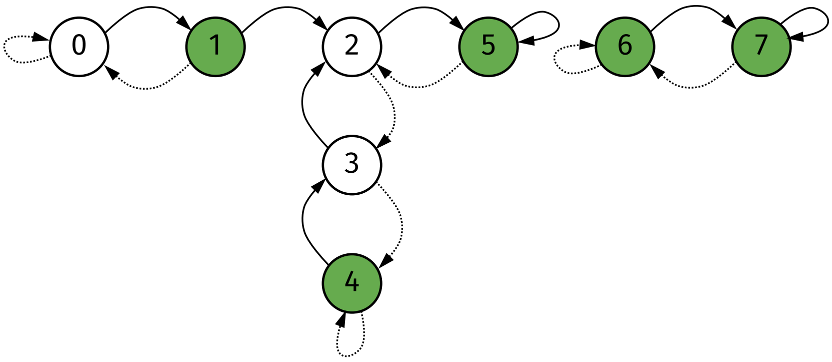

Figure 2.1: Example of an END-OF-LINE Problem with $n = 3$ (8 vertices). The circuit S is represented by solid lines and the circuit P by dashed lines. The solutions are the sinks $x = 5$, $x = 7$ and $x = 1$, aswell as the sources $x = 4$ and $x = 6$.

These boolean circuits represent a directed graph with maximal in and out degree 1, by having an edge from x to y if and only if $S(x) = y$ and $P(y) = x$. The goal is to find a sink of the graph, or another source. It can be shown that the general case of finding a second imbalanced vertex in a directed graph (a problem called IMBALANCE) can be reduced to END-OF-LINE [6]. Now we can define the complexity class **PPAD** as follows:

Notice that END-OF-LINE allows cycles, and that these do not induce solutions.

[6]: Goldberg et al. (2021), *The Hairy Ball problem is PPAD-complete*

Definition 2.5 — PPAD.
The class **PPAD** is the set of all **TNFP** problems that reduce to END-OF-LINE.

End of Potential Line (EOPL)

Next we want to discuss the complexity class **EOPL** as introduced in [7]. The existence results which gives rise to **EOPL** is that *"in a directed acyclic graph, there must be at least two unbalanced vertices"*. Similarly to **PLS** acyclicity will be enforced using a potential.

[7]: Fearnley et al. (2018), *End of Potential Line*

End of Potential Line
Input: Two boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$, and a boolean circuit $V : \{0,1\}^n \to [2^n - 1]$, such that 0^n is a source, (i.e. $P(0^n) = 0^n \neq S(0^n)$).
Output: An $x \in \{0,1\}^n$ such that either:
- $P(S(x)) \neq x$ (x is a sink)
- $S(P(x)) \neq x \neq 0^n$ (x is a *non-standard source*)
- $S(x) \neq x$, $P(S(x)) = x$ and $V(S(x)) \leq V(x)$ (violation of the monoticity of the potential)

Here S can be thought of giving the successor of a vertex, and P as giving the predecessor of a vertex. V can be though of as a potential which is suppose to be monotonously increasing along the line.

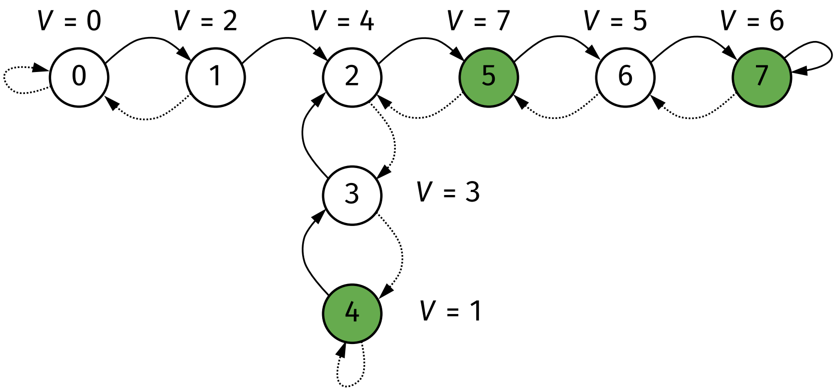

Figure 2.2: Example of an **EOPL** Problem with $n = 3$ (8 vertices). The circuit S is represented by solid lines and the circuit P by dashed lines. The solutions are the sink $x = 7$, the violation of potential at $x = 5$ and the non-standard source $x = 4$.

S and P can be though of as representing a directed line. Finding another source (a non-standard source), is a violation, as a directed line only has one source. The potential serves a garantee of acyclicity. Now we can define the complexity class **EOPL**.

Definition 2.6 — EOPL.
The class **EOPL** is the set of all **TNFP** problems that reduce to End of Potential Line.

2.4 The Tarski Problem

Next we want to introduce the Tarski Problem. Before we do this we recall that there is a partial order on the d dimensional latice $[N]^d$, given by $x \leq y$ iff $x_i \leq y_i$ for all $i \in \{1, \dots, d\}$. The name originates from Tarski's fixed point Theorem as introduced in [8] which we remind the reader of below:

[8]: Tarski (1955), *A lattice-theoretical fixpoint theorem and its applications.*

Theorem 2.1 — Tarski's fixed point Theorem.
Let $f : [N]^d \to [N]^d$ a function on the d-dimentional lattice. If f is monotonous (with respect to the previously discussed partial order), then f has a fixed point, i.e. there is an $x \in [N]^d$ such that $f(x) = x$.

This theorem is also known as the Knaster–Tarski Theorem in the literature.

A proof of this theorem can be found in the previously mentionned work [8]. Without surprise the TARSKI problem as defined in [9], is now to find such a fixed-point. Formally we define the problem as follows:

[9]: Etessami et al. (2020), *Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria*

TARSKI
Input: A boolean circuit $f : [N]^d \to [N]^d$.
Output: Either:
- An $x \in [N]^d$ such that $f(x) = x$ (fixed point) or
- $x, y \in [N]^d$ such that $x \leq y$ and $f(x) \nleq f(y)$ (violation of monoticity).

This is of course a total search problem, as there will always either be a fixed point, or a point violating monoticity. We now want to summarize where TARSKI lies inside of **TNFP**. It has been shown in [9] that TARSKI lies in both **PLS** and $\mathbf{P}^{\mathbf{PPAD}}$. Previous work [10], showed that many-to-one reductions and Turing-reduction onto **PPAD** are equivalent. In particular this means that $\mathbf{P}^{\mathbf{PPAD}} = \mathbf{PPAD}$, and that TARSKI also lies in **PPAD**.

[10]: Buss et al. (2012), *Propositional proofs and reductions between NP search problems*

2.5 Structure of PLS ∩ PPAD

Now that we have established that TARSKI lies inside **PLS** ∩ **PPAD**, we want to discuss the structure of **PLS** ∩ **PPAD** and describe recent advances in the study of this class.

Reducing Tarski to PPAD | 3

In this chapter, we explore the membership of Tarski to the complexity class **PPAD**. We begin by presenting an established proof of the reduction of this problem to Brouwer [9], focusing on a high-level overview. Subsequently, we introduce a novel problem, Tarski*, which facilitates a divide and conquer approach to solving Tarskiby leveraging the structure of the function f. This new formulation allows us to provide an alternative proof of Tarski's membership in PPAD using *Sperner's Lemma* instead of the traditional *Brouwer's Fixed Point Theorem*. This approach not only simplifies the proof but also sets the stage for further reduction of Tarski* to **EOPL** in the subsequent chapter.

[9]: Etessami et al. (2020), *Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria*

3.1 Presentation of the known reduction of Tarski to PPAD

We want to give a high level presentation of the proof of Tarski membership in **PPAD** from [9], which will help us motivate the introduction of Tarski* and the subsequent use of *Sperner's Lemma*. The proof given by Etessami et al. relies on *Brouwer's fixed point theorem*, which we introduce below.

[9]: Etessami et al. (2020), *Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria*

Theorem 3.1 — Brouwer's fixed point theorem.
Let $K \subset \mathbb{R}^d$ be a compact, convex set. Then every continuous function $f : K \to K$ has a fixed point $x^* \in K$, i.e. $f(x^*) = x^*$.

The original proof can be found in [11], a simpler proof relying on Sperner's Lemma can be found in [12]. This theorem gives rise to a total search problem which we call Brouwer:

[11]: Brouwer (1911), *Über Abbildung von Mannigfaltigkeiten*

[12]: Aigner et al. (2018), *Proofs from THE BOOK*

Brouwer
Input: A continuous function $f : K \to K$.
Output: A fixed point $x^* \in K$ such that $f(x^*) = x^*$.

We leave out the technical detail of how this function is given using boolean circuits, and how precise the output needs to be, as it is not relevant for this high level presentation.

The problem Brouwer was first introduced and shown to be **PPAD**-complete in [13]. This means that it suffices to reduce Tarski to Brouwer in order to show that Tarski is in **PPAD**. We will actually reduce Tarski to at most polynomially many instances of Brouwer, which will allow us to show that Tarski is in $\mathbf{P}^{\mathbf{PPAD}}$. This means that we will show a Turing reduction of

[13]: Papadimitriou (1994), *On the complexity of the parity argument and other inefficient proofs of existence*

TARSKI to BROUWER, which suffice as **PPAD** is closed under Turing reductions [10].

[10]: Buss et al. (2012), *Propositional proofs and reductions between NP search problems*

The idea of the the reduction is to extend the discrete function f, to a function $\tilde{f} : [0, 2^n - 1]^d \rightarrow [0, 2^n - 1]^d$, such that $\tilde{f}$ interpolates the lattice function f, is continuous and piecewise linear between lattice points, and hence continuous. This can be achieved using a simplicial decomposition of each cell of the lattice. Now we have an instance of BROUWER, and hence we can find a fixed point x^* of $\tilde{f}$. Of course, this fixed point does not need to be *integral* . The key insight is that we can use this fixed point to reduce the search area for a integral fixed point by at least half, or find a violation of monotonicity. In particular, either there is a fixed point in both $\{x \in [2^n - 1]^d : x \geq x^*\}$ and $\{x \in [2^n - 1]^d : x \leq x^*\}$, or there is a violation of monotonicity in the cell containing x^*. We can repeat this procedure always halfing the search area, which allows us to solve a TARSKI instance using at most $\mathcal{O}(d \cdot n)$ calls to BROUWER.

We call a point *integral* if it belongs to the original lattice.

3.2 Introducing TARSKI*

In the previous section, we have seen that TARSKI can be reduced to a polynomial number of BROUWER instances. We would like to study a single such reduction, in order to give an alternative proof that TARSKI is in **PPAD**. In order to do this, we introduce a new problem, TARSKI*. This problem can be thought of as a subproblem towards solving TARSKI. A standard strategy to solve TARSKI is to use a *divide and conquer* strategy, as for instance used in [9]. We want to construct a problem, which allows us to divide the TARSKI problem into two smaller problems, where solving the smaller of the two leads to a solution.

[9]: Etessami et al. (2020), *Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria*

For the sake of generality and for the proofs in the following we introduce the problem on the integer lattice $L = N_1 \times \cdots \times N_d$, such that $N_i \leq 2^n$ for all $i \in \{1, \ldots, d\}$. We propose the following problem:

TARSKI*
Input: A boolean circuit $f : L \rightarrow L$.
Output: Either:
(T*1) Two points $x, y \in L$ such that $\|x - y\|_\infty \leq 1$, $x \leq f(x)$ and $y \geq f(y)$, or
(T*2) A violation of monotonicity: Two points $x, y \in L$ such that $x \leq y$ and $f(x) \not\leq f(y)$.

We now want to show that TARSKI* can be seen as a subproblem of TARSKI.

Claim 3.1
An instance of Tarski can be solved using $\mathcal{O}(d \cdot n)$ calls to Tarski* and up to $\mathcal{O}(d)$ additional function evaluations.

Proof. We will show that we can use a single call of Tarski* to either find a violation of monotonicity, a fixpoint, or an instance of Tarski which has at most half as many points, and must contain a solution. Let x, y be the two points outputed by a Turing machine solving Tarski*on a function f. We proceed by case distinction:

Case 1: If either $f(x) = x$ or $f(y) = y$, then we are done, because we have found a fixpoint.

Case 2.1: If $x < y$ and $f(x) \nleq f(y)$, we have a violation of monotonicity, which solves the given Tarski instance.

Case 2.2: If $x < y$ and $f(x) \leq f(y)$, we claim that we can solve the Tarski instance in $\mathcal{O}\left(\|x - y\|_1\right)$ additional function calls. Notice that we have $\|x - y\|_\infty \leq 1$. Now notice that because $f(x) > x$ (if not see case 1), there is at least one dimension $i \in \{1, \dots, d\}$ such that $f(x)[i] > x[i]$. Also notice that in this dimension i if $f(y)[i] < y[i]$, then because $|x[i] - y[i]| \leq \|x[i] - y[i]\|_\infty \leq 1$, we would have a violation of the monotonicity of f in this dimension. Therefore we must have $f(y)[i] = y[i]$. The same argument shows that if in any dimension j we have $f(y)[j] < y[j]$, then $f(x)[j] = x[j]$. Therefore we know that because there must be at least one such dimension i and j we have:

$$\|f(x) - f(y)\|_\infty \leq \|x - y\|_\infty \leq 1 \quad \text{and} \quad \|f(x) - f(y)\|_1 \leq \|x - y\|_1 - 2$$

Hence we can now repeat the same argumentation with $f(x)$ and $f(y)$, and we can do this at most $\mathcal{O}\left(\|x - y\|_1\right)$ times, until we find a violation of monotonicity or a fixpoint. Because $\|x - y\|_1 \leq d$, this will take at most $\mathcal{O}(d)$ additional steps.

Case 3: If $x \nleq y$, then we can partition the set of lattice points into two sets S_x and S_y, as follows:

$$S_x = \{z \in L : z \geq x\} \quad \text{and} \quad S_y = \{z \in L : z \leq y\}.$$

These two sets are disjoint: if there was a $z \in S_x \cap S_y$, then $x \leq z \leq y$, which would imply $x \leq y$, which is a contradiction. We will show that S_x must contain a solution to the Tarski instance. If for some $z \in S_x$ we have $f(z) \notin S_x$, then we have $f(z) \ngeq f(x)$, which means that z and x form a violation of monotonicity. This means that S_x forms a new valid instance of Tarski. By the same argumentation S_y also forms a valid instance of Tarski and hence it suffices to recursively solve the smaller of the two instances. In particular because they are disjoint, one of the instances S_x or S_y contains less than half of the lattice points of L, and hence we can solve the instance in $\mathcal{O}\left(\log 2^{dn}\right) = \mathcal{O}(d \cdot n)$ calls of TARSKI*. □

Now that we know that TARSKI* is a good stepping stone towards solving TARSKI, we want to investigate why TARSKI* lies in **PPAD**.

3.3 Sperner's Lemma

The preceding discussion hinges on the assumption that TARSKI* is a total problem, implying that every instance of the problem is guaranteed a solution. In this section, we will substantiate this claim, establishing TARSKI*'s classification within **TNFP**. Rather than employing *Brouwer's fixed point Theorem* — a cornerstone of continuous topology — we pivot to its discrete analogue, *Sperner's Lemma*, a foundational result in combinatorial topology. This approach is particularly apt for two main reasons:

- We are working on a discrete lattice, and hence it seems more natural to use a discrete tool.
- Papadimitriou proved that BROUWER is **PPAD**-complete by reducing BROUWER to SPERNER [13]. Hence by reducing to BROUWER, we introduce continuity into the problem, which is not necessary, as it get removed again behind the scenes.

[13]: Papadimitriou (1994), *On the complexity of the parity argument and other inefficient proofs of existence*

Our goal is to apply *Sperner's Lemma* on the integer lattice. This is not directly possible, as *Sperner's Lemma* is defined on a simplicial decomposition of a simplex. Hence we will first introduce *Sperner's Lemma* for simplices, and then show how it can be adapted to work on an integer lattice.

Sperner's Lemma for Simplices

Before we introduce the Lemma itself, we want to define the setting of the result. We consider a d-dimensional simplex[1] with vertices $v_0, v_1, \dots, v_d$. We now consider a *simplicial subdivision* of this simplex. This means that we partition the simplex into smaller simplices. We give an example of such a partition in Figure 3.1 in the 3-dimensional case.

1: By d dimensional simplex we mean the convex Hull of these $d+1$ points in $\mathbb{R}^d$

Now we introduce a coloring c of the vertices of this subdivision with colors $\{0, 1, \dots, d\}$. We want to enforce that the vertices v_i of the large simplex are colored with color i, and that the vertices on a subsimplex $\{v_{i_0}, \dots, v_{i_k}\}$ are colored with colors $i_0, \dots, i_k$. We give an example of such a coloring in 2 dimensions in **??**.

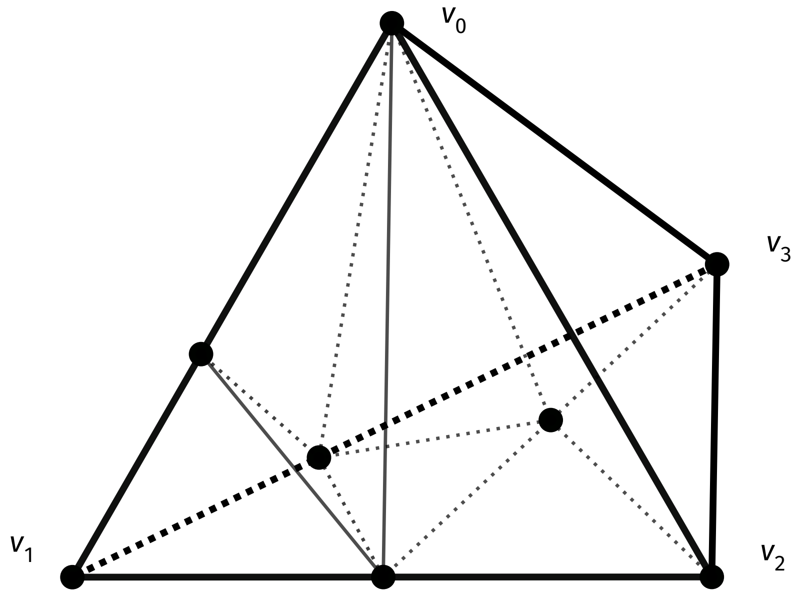

Figure 3.1: Setup for SPERNER'S LEMMA in the 3-dimensional case. The large simplex spanned by v_0, v_1, v_2, v_3 is subdivided into smaller simplices.

We now introduce Sperner's Lemma, which was first proven in [14], and for which a more modern proof can be found in [12].

[14]: Sperner (1928), *Neuer beweis für die invarianz der dimensionszahl und des gebietes*

[12]: Aigner et al. (2018), *Proofs from THE BOOK*

Theorem 3.2 — Sperner's Lemma.
Suppose that a d-dimensional simplex with vertices $v_0, \dots, v_d$ is subdivided into smaller simplices. Now color every vertex with a color $\{0, \dots, d\}$ such that v_i is colored i, and the vertices on a subsimplex $\left\{v_{i_0}, \dots, v_{i_k}\right\}$ are colored with colors $i_0, \dots, i_k$. Then there is a subsimplex, with vertices of every color.

We give an example of a 2-dimensional simplex, which is subdivided into smaller simplices, and colored according to *Sperner's Lemma* in Figure 3.2.

Sperner's Lemma for an integer lattice

Now that we have introduced *Sperner's Lemma* for a integer lattice. The motivation is to be able to find a region of a colored lattice which contains all colors under certain conditions. Instead of looking for a subsimplex, we will look for a *cell*[2] of the lattice, which contains all colors.

2: By cell we mean a unit hypercube of the integer lattice

In order to do this we proceed as follows. We take the d-dimensional lattice $L = [N_1] \times \dots \times [N_d]$, we subdivide each cell into simplices[3]. We set $v_0 = (0, \dots, 0)$, $v_1 = (N_1 - 1, 0, \dots, 0), \dots, v_d = (0, \dots, 0, N_d - 1)$. We give an example of such a subdivision in the 3-dimensional case in Figure 3.3. Notice that we can deform the lattice an we obtain an equivalent simplex, and a simplicial decomposition of this simplex.

3: How this is done is not relevant in this chapter but will be discussed in the next chapter.

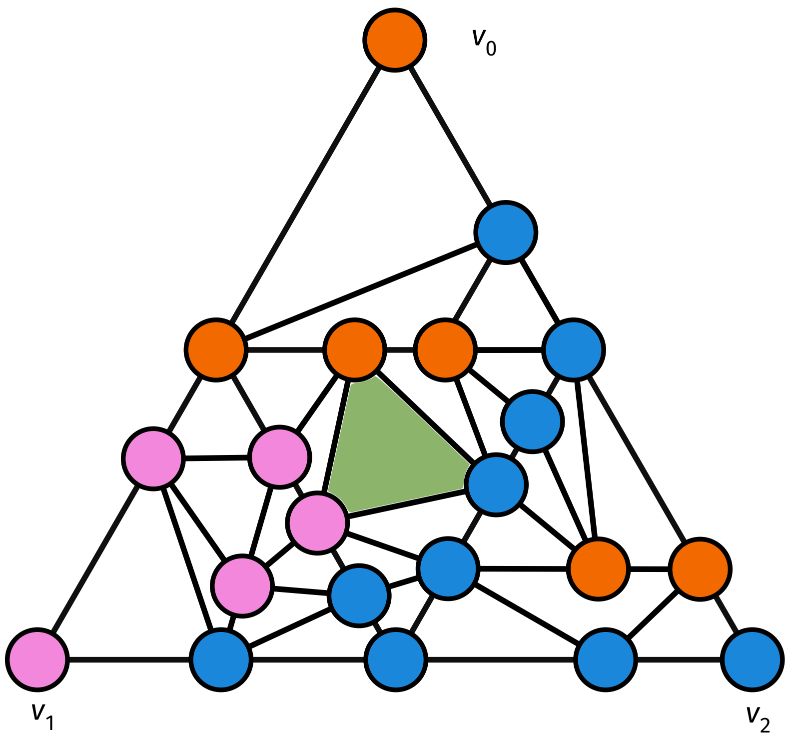

Figure 3.2: Example of SPERNERS LEMMA in the two dimensional case, with 3 colors: orange (0), purple (1) and blue (2). The subsimplex spanned by v_0 and v_1 only contains blue and purple vertices, the subsimplex spanned by v_1 and v_2 contains only purple and blue vertices and the subsimplex spanned by v_0 and v_2 contains only orange and blue vertices. *Sperner's Lemma* implies that there must be a cell (colored in green), which contains all colors.

This means that under the appropriate conditions — which we will detail next — we can apply *Sperner's Lemma* to the lattice. Assume that we color all vertices of the lattice with colors $\{0, \dots, d\}$, such that v_i is colored i, and every vertex x with $x[i] = 0$, is *not* colored i for $i \in \{1, \dots, d\}$. Then we can apply *Sperner's Lemma* to this simplical decomposition of the lattice, and we will find a simplex which contains all colors. Of course because every subsimplex is included in exactly one cell by construction, there must be a cell which contains all colors. This motivates the definition of the total problem SPERNERwhich was introduced and shown to be **PPAD**-complete in [13]. We introduce the problem for a general lattice $L = N_1 \times \cdots \times N_d$, such that $N_i \leq 2^n$.

[13]: Papadimitriou (1994), *On the complexity of the parity argument and other inefficient proofs of existence*

SPERNER
Input: A coloring $c : L \to \{0, \dots, d\}$ of the vertices of L, such that for every $i \in \{0, \dots, d\}$ the the vertices $\{x \in L : x[i] = 0\}$ are not colored i.
Output: A cell C such that for all $i \in \{0, \dots, d\}$ there is a vertex $x \in C$ such that $c(x) = i$.

Next we will use this problem to show that TARSKI* is a total search problem, and hence lies in **PPAD**.

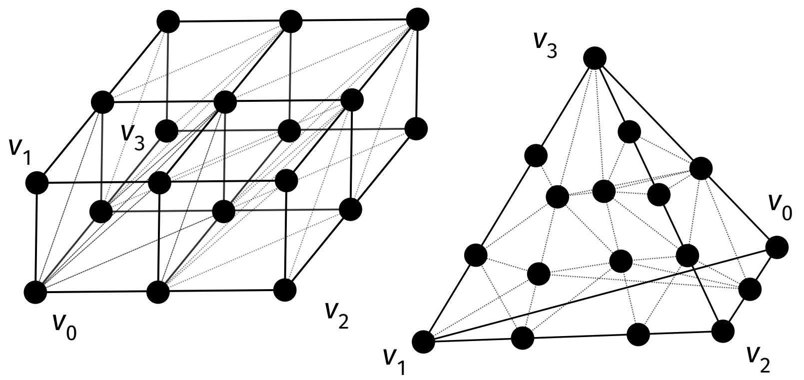

Figure 3.3: Example of the simplicial decomposition of a lattice in the 3 dimensional case on the left, and the equivalent simplicial decomposition on the right of a simplex v_0, v_1, v_2, v_3.

3.4 Reducing TARSKI* to SPERNER

For us to be able to use SPERNER's Lemma on our TARSKI* instances, we need to define a coloring of the vertices of L. We propose the following coloring $c : L \to \{0, \dots, d\}$:

A vertex colored 0 indicates that the function points *weakly forwards* in all dimensions, a vertex colored i for $i \geq 1$ indicates that the function points *backwards* in at least the i-th dimension.

$$c(x) = \begin{cases} 0 & \text{if } x \leq f(x) \\ 1 & \text{else if } x[1] > f(x)[1] \\ & \vdots \\ d & \text{else if } x[d] > f(x)[d] \end{cases}$$

We now need two results. First we need to show that a cell with all colors always exists, which will allow us to show that TARSKI* is a total search problem. Second we need to show that finding a cell with all colors, yields a solution to TARSKI*, in polynomial time.

Claim 3.2
For any TARSKI* instance, with vertices colored as above, there is always a cell with all colors.

Proof. This claim follows directly from SPERNER's Lemma, and the coloring we have defined. There can never be a vertex colored i with $x[i] = 0$, because this would imply that $f(x)[i] < x[i]$, which is a contradiction to the construction of the function. Hence by dividing each cell of the lattice into simplices, we can apply SPERNER's Lemma to show that a cell with all colors always exists. The vertices we use as the vertices of the large simplex are $\{(0, \dots, 0), (2^n - 1, 0, \dots, 0), \dots, (0, \dots, 2^n - 1)\}$. □

Claim 3.3
Finding a cell with all colors yields a solution to TARSKI*, in $\mathcal{O}(d)$ additional steps.

Proof. Assume we have found a simplex, with vertices colored $0, \dots, d$. Let us denote x_i the vertex colored i, for $i \in \{0, \dots, d\}$. Notice that all of these vertices are by construction contained in some cell (hypercube of length 1), let $\mathbf{0}$ be the smallest vertex of this hypercube and $\mathbf{1}$ the largest. In particular this means that for all i we have:

$$\mathbf{0} \le x_i \le \mathbf{1} \quad \text{and} \quad f(x_i)[i] < x_i[i] \quad \text{for } i > 0$$

We now proceed by case distinction:

Case 1: If x_0 is a fixed point, then $x = y = x_0$ is a solution to Tarski*.

Case 2: If $x_0 \neq f(x_0)$ and $x_0 = \mathbf{0}$. Then there is an i such that $f(x_0)[i] > x_0[i]$, which means that $f(x_0[i]) - x_0[i] \ge 1$. At the same time we must have $f(x_i)[i] < x_i[i]$ and $x_0[i] - x_i[i] \le 0$ because $x_0 = \mathbf{0}$, and hence $x_i[i] - f(x_i)[i] \ge 1$. Now we get:

$$f(x_0)[i] - f(x_i)[i] = \underbrace{f(x_0)[i] - x_0[i]}_{\ge 1} + \underbrace{x_0[i] - x_i[i]}_{\ge 0} + \underbrace{x_i[i] - f(x_i)[i]}_{\ge 1}$$

$$f(x_0)[i] - f(x_i)[i] \ge 2$$

This implies that $f(x_0) \nleq f(x_i)$, and hence x_0, x_i are two points witnessing a violation of monotonicity of f, which form a solution to Tarski*.

Case 3: If $x_0 \neq f(x_0)$ and $x_0 \neq \mathbf{0}$. We claim that either $f(\mathbf{0}) \le \mathbf{0}$, or we have a violation of monotonicity. Assume for the sake of contradiction that there is an i such that $f(\mathbf{0})[i] > \mathbf{0}[i]$. Then we must have $f(x_i)[i] < x_i[i]$ hence we get: $f(\mathbf{0})[i] \nleq f(x_i)[i]$, which is a violation of monotonicity. This means that either we can return $y = x_0$ and $x = \mathbf{0}$ as a solution to Tarski*, or x_i and $\mathbf{0}$ as a violation of monotonicity. □

This shows that Tarski* is a total search problem, and can be reduced to Sperner. Hence Tarski* lies in **PPAD**, and by using that $\mathbf{P}^{\mathbf{PPAD}} = \mathbf{PPAD}$ we have shown that Tarski lies in **PPAD**, without relying on Brouwer.

APPENDIX

Bibliography

[1] Christos H. Papadimitriou. 'On the complexity of the parity argument and other inefficient proofs of existence'. In: *Journal of Computer and System Sciences* 48.3 (June 1994), pp. 498–532. DOI: 10.1016/S0022-0000(05)80063-7. (Visited on 03/05/2024) (cited on pages 3, 4).

[2] Nimrod Megiddo and Christos H. Papadimitriou. 'On total functions, existence theorems and computational complexity'. In: *Theoretical Computer Science* 81.2 (Apr. 1991), pp. 317–324. DOI: 10.1016/0304-3975(91)90200-L. (Visited on 03/05/2024) (cited on page 3).

[3] Christos H. Papadimitriou. *Computational complexity*. Reading (Mass): Addison-Wesley, 1994 (cited on page 4).

[4] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. 'How easy is local search?' In: *Journal of Computer and System Sciences* 37.1 (Aug. 1988), pp. 79–100. DOI: 10.1016/0022-0000(88)90046-3. (Visited on 03/06/2024) (cited on page 4).

[5] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. 'The Complexity of Computing a Nash Equilibrium'. In: *SIAM Journal on Computing* 39.1 (Jan. 2009), pp. 195–259. DOI: 10.1137/070699652. (Visited on 03/11/2024) (cited on page 4).

[6] Paul W. Goldberg and Alexandros Hollender. 'The Hairy Ball problem is PPAD-complete'. In: *Journal of Computer and System Sciences* 122 (Dec. 2021), pp. 34–62. DOI: 10.1016/j.jcss.2021.05.004. (Visited on 03/10/2024) (cited on page 5).

[7] John Fearnley et al. *End of Potential Line*. Apr. 18, 2018. URL: http://arxiv.org/abs/1804.03450 (visited on 03/02/2024) (cited on page 5).

[8] Alfred Tarski. 'A lattice-theoretical fixpoint theorem and its applications.' In: *Pacific Journal of Mathematics* 5.2 (Jan. 1, 1955), pp. 285–309 (cited on pages 6, 7).

[9] Kousha Etessami et al. 'Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria'. In: (2020). In collab. with Thomas Vidick. Artwork Size: 19 pages, 651206 bytes ISBN: 9783959771344 Medium: application/pdf Publisher: Schloss Dagstuhl – Leibniz-Zentrum für Informatik Version Number: 1.0, 19 pages, 651206 bytes. DOI: 10.4230/LIPICS.ITCS.2020.18. (Visited on 02/24/2024) (cited on pages 7–9).

[10] Samuel R. Buss and Alan S. Johnson. 'Propositional proofs and reductions between NP search problems'. In: *Annals of Pure and Applied Logic* 163.9 (Sept. 2012), pp. 1163–1182. DOI: 10.1016/j.apal.2012.01.015. (Visited on 02/24/2024) (cited on pages 7, 9).

[11] L. E. J. Brouwer. 'Über Abbildung von Mannigfaltigkeiten'. In: *Mathematische Annalen* 71.1 (Mar. 1911), pp. 97–115. DOI: 10.1007/BF01456931. (Visited on 05/04/2024) (cited on page 8).

[12] Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. 6th ed. 2018. Berlin, Heidelberg: Springer Berlin Heidelberg : Imprint: Springer, 2018. 1 p. (cited on pages 8, 12).

[13] Christos H. Papadimitriou. 'On the complexity of the parity argument and other inefficient proofs of existence'. In: *Journal of Computer and System Sciences* 48.3 (June 1994), pp. 498–532. DOI: 10.1016/S0022-0000(05)80063-7. (Visited on 03/22/2024) (cited on pages 8, 11, 13).

[14] E. Sperner. 'Neuer beweis für die invarianz der dimensionszahl und des gebietes'. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 6.1 (Dec. 1928), pp. 265–272. DOI: 10.1007/BF02940617. (Visited on 04/18/2024) (cited on page 12).

Alphabetical Index