The Complexity of Finding Tarski Fixed Points

Master Thesis

March 8, 2024

Nils Jensen

Advised by

Prof. Dr. Bernd Gärtner

Sebastian Hasselbacher

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:

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

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

[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.

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 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.

[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 [3]:

[3]: 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.

Polynomial Parity Argument on Directed Graphs (PPAD)

TODO

End of Potential Line (EOPL)

TODO

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 $i \in \{1, \ldots, d\}$. The name originates from TARSKI's fixed point Theorem as introduced in [4] which we remind the reader of below:

[4]: 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 litterature.

A proof of this theorem can be found in the previously mentionned work [4]. Without surprise the TARSKI problem as defined in [5], is now to find such a fixed-point. Formally we define the problem as follows:

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

Tarski
Input: A boolean circuit $S : [N]^d \to [N]^d$.
Output: Either:
- An $x \in [N]^d$ such that $S(x) = x$ (fixed point) or
- $x, y \in [N]^d$ such that $x \leq y$ and $f(x) \not\leq f(y)$ (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 [5] that Tarski lies in both PLS and $\mathsf{P}^{\mathsf{PPAD}}$. Previous work [6], showed that many-to-one reductions and Turing-reduction onto PPAD are equivalent. In particular this means that $\mathsf{P}^{\mathsf{PPAD}} = \mathsf{PPAD}$, and that Tarski also lies in PPAD.

[6]: 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.

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 page 3).

[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] 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 3).

[4] 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 page 4).

[5] 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 4, 5).

[6] 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 page 5).

Alphabetical Index