Electronic Colloquium on Computational Complexity, Report No. 69 (2025)

Provably Total Functions in the Polynomial Hierarchy

Noah Fleming
Memorial University

Deniz Imrek
UT Austin

Christophe Marciot
Memorial University

May 30, 2025

Abstract

TFNP studies the complexity of total, verifiable search problems, and represents the first layer of the total function polynomial hierarchy (TFPH). Recently, problems in higher levels of the TFPH have gained significant attention, partly due to their close connection to circuit lower bounds. However, very little is known about the relationships between problems in levels of the hierarchy beyond TFNP.

Connections to proof complexity have had an outsized impact on our understanding of the relationships between subclasses of TFNP in the black-box model. Subclasses are characterized by provability in certain proof systems, which has allowed for tools from proof complexity to be applied in order to separate TFNP problems. In this work we begin a systematic study of the relationship between subclasses of total search problems in the polynomial hierarchy and proof systems. We show that, akin to TFNP, reductions to a problem in $\mathsf{TF}\Sigma_d$ are equivalent to proofs of the formulas expressing the totality of the problems in some Σ_d-proof system. Having established this general correspondence, we examine important subclasses of TFPH. We show that reductions to the StrongAvoid problem are equivalent to proofs in a Σ_2-variant of the (unary) Sherali-Adams proof system. As well, we explore the TFPH classes which result from well-studied proof systems, introducing a number of new $\mathsf{TF}\Sigma_2$ classes which characterize variants of DNF resolution, as well as $\mathsf{TF}\Sigma_d$ classes capturing levels of Σ_d-bounded-depth Frege.

Contents

ISSN 1433-8092

1 Introduction

The class TFNP consists of the *total* search problems whose solutions are verifiable in polynomial time . It has received considerable attention since it captures fundamental problems in a broad range of areas whose lack of efficient algorithms is not readily explained by the theory of NP-completeness. Famous examples include NASH: output a Nash equilibria of a given bimatrix game; and FACTORING: output a prime divisor of a given integer. TFNP itself is not believed to admit complete problems [Pud15], and as a consequence much of the work on TFNP has focused on studying subclasses which do. However, we are limited to proving conditional or oracle separations, as a separation between any TFNP subclasses would imply $\mathsf{P} \neq \mathsf{NP}$.

A flurry of recent results have established a *complete* picture of the relationships between the major TFNP subclasses in the *black-box* setting, where the input is presented as a black-box oracle which can be queried [BCE+98, GHJ+22b, GHJ+22a, FGPR24, GKRS19]. These results exploited a deep connection between black-box TFNP — denoted TFNP^{dt} — and proof complexity, an area which studies efficient provability in certain propositional logics, known as proof systems. The connection of proof complexity to TFNP^{dt} can be summarized as follows: A reduction between two total search problems is a proof that the first is total, assuming the totality of the second. By employing this lens, it has been shown that many important TFNP^{dt} subclasses are *characterized* by provability in certain well-studied proof systems in the sense that there is a simple proof of the totality of a search problem if and only if there is an efficient reduction of that search problem to the complete problem for that subclass [GKRS19,GHJ+22b,BFI23,LPR24,DR23]. This connection has been highly impactful for the study of TFNP^{dt}, allowing for the rich set of tools in proof complexity to be leveraged in order to provide separations between the major TFNP^{dt} subclasses.

$\mathsf{TFNP} = \mathsf{TF}\Sigma_1$ is the first level of the *total function polynomial hierarchy* $\mathsf{TFPH} = \bigcup_i \mathsf{TF}\Sigma_i$ [KKMP21]. Recently, problems in higher levels of the polynomial hierarchy have received considerable attention, in part due to their close connection to circuit lower bounds. Indeed, consider the task of finding (the truth table of) a function which does not have circuits of size s. Using a standard encoding, any circuit of size s can be represented uniquely by $k = \text{poly}(s)$-many bits. Consider the map $T : \{0,1\}^k \to \{0,1\}^n$ which maps circuits of size s to truth tables of the function that they compute. Finding a truth table of a function with high circuit complexity is equivalent to finding a string which is not in the range of T. This is an instance of the RANGEAVOIDANCE problem.

Definition 1.1. RANGEAVOIDANCE (or simply AVOID) is the following search problem: Given a function $f : \{0,1\}^n \to \{0,1\}^{n+1}$, find a $y \in \{0,1\}^{n+1}$ such that for all x, $f(x) \neq y$.

Observe that any solution y to AVOID can be checked by a coNP verifier — check that for every $x \in \{0,1\}^n$, $f(x) \neq y$. This means that AVOID belongs to the class $\mathsf{TF}\Sigma_2$. If there is an algorithm for solving AVOID which belongs to a class $\mathcal{C}$ then this implies the existence of a function in $\mathcal{C}$ which does not have small circuits — a circuit lower bound against $\mathcal{C}$! This approach led to the recent breakthrough circuit lower bounds against symmetric exponential time [Li24,CHR24,KP24]. Hence, understanding the complexity of $\mathsf{TF}\Sigma_2$ is important for understanding circuit lower bounds. Indeed, the current best upper bound puts AVOID in the class of problems reducible to LOP — the $\mathsf{TF}\Sigma_2$ problem of finding a minimum element in a total order.

$\mathsf{TF}\Sigma_2$ contains numerous important problems beyond those connected to circuit lower bounds. For example, AVOID is the complete problem for the class APEPP which also captures the complexity of finding pseudo-random number generators, randomness extractors, and rigid matrices [Kor21]. We can restrict AVOID to only have one more element in its range than in its domain to obtain the problem STRONGAVOID.

Definition 1.2. STRONGRANGEAVOIDANCE (or simply STRONGAVOID) is the following search problem: Given a function $f : \{0,1\}^n \setminus \{0\} \to \{0,1\}^n$, find an empty hole $y \in \{0,1\}^n$, i.e., such that for all $x \in \{0,1\}^n \setminus \{0\}$, $f(x) \neq y$.

STRONGAVOID is the complete problem for the class PEPP which captures the complexity of finding objects whose existence is guaranteed by the union bound, including all of APEPP [KKMP21]. Important problems have also been identified in higher levels of the polynomial hierarchy, such as those related to finding sets of large VC dimension [KKMP21].

Despite the importance of problems in levels of the polynomial hierarchy beyond TFNP, there has been little structural exploration into how they relate. Indeed, [KP24] provide the first black-box separation, showing that STRONGAVOID is not reducible to any problem in $\mathsf{TF}\Sigma_2$ with a unique solution (in fact, they show that it cannot be solved with non-adaptive oracle calls to any language in Σ_2^P). Proof complexity has had an outsized impact on proving black-box separations for TFNP. To facilitate further structural exploration of TFPH, we would like to explore to what degree proof complexity tools can be used to provide separations between classes within higher levels of the black-box total function polynomial hierarchy (denoted TFPH^{dt}).

Our Results

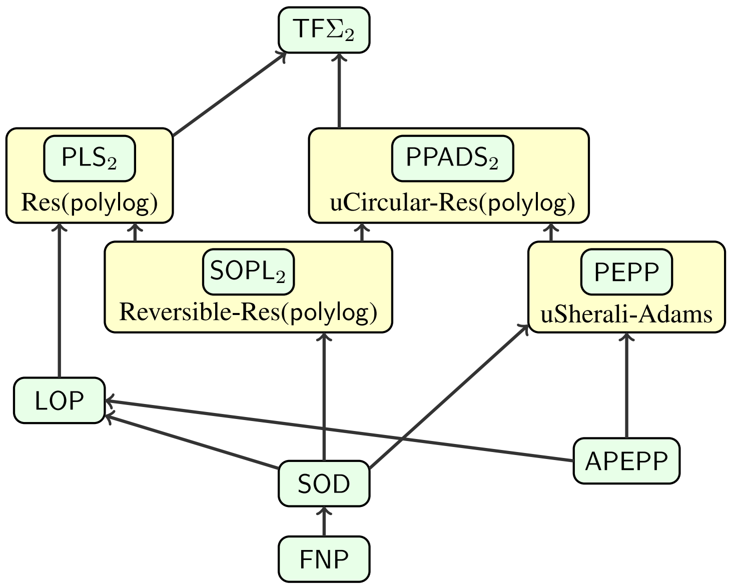

Figure 1: Relationships and characterizations of $\mathsf{TF}\Sigma_2$ classes studied. An arrow indicates containment.

In this paper we begin a systematic study of the connections between the total function polynomial hierarchy in the black-box model and propositional proof complexity. First, we identify the form that proof systems which characterize $\mathsf{TF}\Sigma_d^{dt}$ subclasses take. In order to characterize $\mathsf{TF}\Sigma_d$ subclasses, these proof systems must be able to prove the validity of depth-$(d+1)$ propositional formulas. However, they cannot be Cook-Reckhow proof systems (NP-verifiers) in general unless $\mathsf{NP} = \mathsf{coNP}$ as there are syntactic subclasses of $\mathsf{TF}\Sigma_2^{dt}$ which contain all of TFNP^{dt}; a characterization of which would imply a polynomialy-bounded proof system. Recall that a class is syntactic if it admits a complete problem. We show that in order to characterize $\mathsf{TF}\Sigma_d^{dt}$ subclasses, it suffices to augment Cook-Reckhow proof systems P with a Σ_d-*weakening rule* which generalizes the resolution weakening rule to Σ_d formulas; we call the resulting proof system Σ_d-P (see Definition 3.5).

To begin, we explore the limits of these characterizations, verifying that this is indeed the correct definition of a proof system for $\mathsf{TF}\Sigma_d^{dt}$. A syntactic class $\mathcal{C} \subseteq \mathsf{TF}\Sigma_d^{dt}$ is *uniform* is there is a polynomial-time algorithm which given n outputs the n^{th} instance of the complete problem for $\mathcal{C}$.

Theorem 1.3 (Informal)**.** *The following hold:*

1. *For every uniform $\mathcal{C} \subseteq \mathsf{TF}\Sigma_d^{dt}$ there is a Σ_d-proof system P such that $R \in \mathcal{C}$ if and only if P efficiently proves that R is total.*
2. *For every* well-behaved *Σ_d-proof system P there is a uniform $\mathsf{TF}\Sigma_d^{dt}$ subclass $\mathcal{C}$ such that $R \in \mathcal{C}$ if and only if P proves that R is total.*

Having established this scaffolding result, we begin to explore characterizations of specific $\mathsf{TF}\Sigma_d^{dt}$ subclasses; our results for $\mathsf{TF}\Sigma_2$ can be seen in Figure 1. First, we show that PEPP^{dt} is characterized by the Σ_2-variant of the Sherali-Adams proof system.

Theorem 1.4 (Informal)**.** *$R \in \mathsf{PEPP}^{dt}$ iff there is an efficient Σ_2-Sherali-Adams proof that R is total.*

This allows one to use an extension to the pseudo-expectation technique in order to exclude total search problems from PEPP, and hence also APEPP. Currently, no such exclusions are known.

We also consider several variants of the *DNF-resolution* proof system: DNF Resolution ($\mathsf{Res(polylog)}$), Circular DNF resolution ($\mathsf{uCircRes(polylog)}$), and Reversible DNF resolution ($\mathsf{RevRes(polylog)}$). We introduce new $\mathsf{TF}\Sigma_2^{dt}$ classes which characterize them.

Theorem 1.5 (Informal)**.** Σ_2-$\mathsf{Res(polylog)}$, Σ_2-$\mathsf{uCircRes(polylog)}$, Σ_2-$\mathsf{RevRes(polylog)}$ *are characterized by the* $\mathsf{TF}\Sigma_2^{dt}$ *subclasses* PLS_2, SOL_2, SOPL_2, *respectively.*

We explore how these new classes relate to natural $\mathsf{TF}\Sigma_2$ classes, which can be seen in Figure 1. In doing so, we introduce a natural $\mathsf{TF}\Sigma_2$ class SOD, of problems reducible to finding a source in a DAG given a sink, which we believe may be of independent interest.

Finally, we show that our characterization of DNF resolution can be extended to characterize bounded-depth Frege in higher levels of TFPH. The depth-d Frege system allows one to cut on depth-d propositional formulas.

Theorem 1.6 (informal)**.** Σ_d*-Depth d-Frege is characterized by the* $\mathsf{TF}\Sigma_d^{dt}$ *class* PLS_d^{dt}.

This result is inspired by the work of Beckmann and Buss, who characterize PE_d and GI_d in bounded arithmetic [PT12]. It is also the $\mathsf{TF}\Sigma_d$ analogue of Thapen's recent TFNP characterization of bounded-depth Frege [Tha24].

Comparison with Bounded Arithmetic. Characterizations of TFPH classes have been studied in the *uniform* setting by theories of bounded arithmetic. Beckmann and Buss [BB09] showed that Σ_d^b-definable functions of T_2^d correspond to the class $\mathsf{PLS}^{\Sigma_{d-1}^p}$, which is defined by replacing the polynomial-time predicates and functions of the complete problem for the TFNP subclass PLS with predicates and functions from $\mathsf{P}^{\Sigma_{d-1}^p}$. This results in the *generalized polynomial local search problem* GPLS_d of [PT12]. However, these correspondences do not stray outside of proof systems which correspond to bounded-depth Frege systems.

Comparison with [Tha24]. Recently and independently, Thapen [Tha24] considered reductions between black-box total search problems in the polynomial hierarchy under the notion of *counter example reducibility*, in order to define new TFNP subclasses. Taking the set of TFNP problems which are counter-example reducible to a $\mathsf{TF}\Sigma_2$ problem, characterizing a class $C \subseteq \mathsf{TF}\Sigma_2$, essentially acts as a projection of C to TFNP. He uses these in order to obtain TFNP^{dt} characterizations. In comparison, we are interested in reductions between and characterizations of $\mathsf{TF}\Sigma_d^{dt}$ problems. As well, he shows that versions of the game induction problems GI_d [ST11] form TFNP^{dt} subclasses which characterize bounded-depth Frege proofs of CNF formulas.

Open Problems. In this paper we provide the framework for characterizations between total search problems in the polynomial hierarchy, leaving open many natural questions.

1. We study decision-tree reductions, as these are the query analogue of polynomial-time reductions. However, it is natural also to consider more powerful reductions, such as P^{NP}-reductions. What characterizations does one obtain under such reductions?
2. There are several studied classes for which we do not yet have characterizations, such as APEPP and LOP. Due to the connection between STRONGAVOID and Sherali-Adams, it would appear that PEPP should correspond to a variant of Sherali-Adams which produces a large negative value, rather than -1. However, we cannot maintain this under decision-tree reductions.
3. $\mathsf{TF}\Sigma_2$ problems with unique solutions play a critical role in the recent circuit lower bounds [Li24, CHR24, KP24]. What properties do proof systems which characterize such problems possess?

2 Preliminaries on the Total Function Polynomial Hierarchy

Subclasses of TFPH are typically defined by a simple existence principle to which everything in the class reduces. For example, any total order must have a minimal element. These existence principles naturally give rise to total search problems. Continuing the example:

Definition 2.1. The *Linear Ordering Principle* (LOP) asks, given $\prec$: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, to find:

- A minimal element: $x \in \{0,1\}^n$ such that $\forall y \neq x$, $x \prec y$.
- A violation of the total order: either (i) $x \in \{0,1\}^n$ such that $x \prec x$, (ii) $x \neq y$ such that $x \not\prec y$ and $y \not\prec x$, or (iii) $x \prec y$ and $y \prec z$ and $x \not\prec z$.

To make these problems non-trivial, the input is presented as a circuit C so that the search space is exponential in the number of input bits n. Formally, for any $x, y \in \{0,1\}^n$, $C(x,y) =\prec (x,y)$. We call C a *white-box* encoding of the problem. Unfortunately, a separation between any pair of total search problems in the white-box model is hard to achieve, as it would imply $\mathsf{P} \neq \mathsf{NP}$.

Instead, we can gain intuition for the relationships between these classes by exploring their *black-box* variants. In this setting C is given as an oracle which can be queried, but we no longer have access to the description of C. A major benefit of considering the black-box model is that we can now prove *unconditional* separations between classes without having to resolve P versus NP. These separations imply oracle separation in the white-box setting.

A *query search problem* is a sequence of relations $R_n \subseteq \{0,1\}^n \times \mathcal{O}_n$, one for each $n \in \mathbb{N}$. It is *total* if for every $x \in \{0,1\}^n$ there is an $o \in \mathcal{O}_n$ such that $(x,o) \in R_n$. We think of $x \in \{0,1\}^n$ as a bit string which can be accessed by querying individual bits, and we will measure the complexity of solving R_n as the number of bits that must be queried. Hence, an efficient algorithm for R_n will be one which finds a suitable o while making at most $\mathsf{polylog}(n)$-many queries to the input. We will not charge the algorithm for other computational steps, and therefore an efficient algorithm corresponds to a shallow decision tree. Total query search problems which can be computed by decision trees of depth $\mathsf{polylog}(n)$ belong to the class FP^{dt}, where dt indicates that it is a black-box class. While search problems are formally defined as sequences $R = (R_n)$, we will often want to speak about individual elements in the sequence. For readability, we will abuse notation and refer to elements R_n in the sequence as total search problems; furthermore, we will often drop the subscript n, and rely on context to differentiate.

In this paper we will be considering total query search problems in the polynomial hierarchy TFPH^{dt}.

Definition 2.2. We say that a total search problem $R = (R_n)$, where $R_n \subseteq \{0,1\}^n \times \mathcal{O}_n$, belongs to the d^{th} level of the query *total function polynomial hierarchy* $\mathsf{TF\Sigma}_d^{dt}$ if for every $o \in \mathcal{O}_n$

$$(x,o) \in R \iff \forall z_1 \in \{0,1\}^{\ell_1}\ \exists z_2 \in \{0,1\}^{\ell_2}\ \ldots Qz_{d-1} \in \{0,1\}^{\ell_{d-1}}\ V_{o,(z_1,\ldots,z_{d-1})}(x) = 1,$$

where $Q \in \{\exists, \forall\}$, $V_{o,\vec{z}} = V_{o,(z_1,\ldots,z_{d-1})}$ is a decision tree of $\mathsf{polylog}(n)$-depth, and each $\ell_i \in \mathsf{polylog}(n)$.

Note that $\mathsf{FP}^{dt} = \mathsf{TF\Sigma}_0^{dt}$ and $\mathsf{TFNP} = \mathsf{TF\Sigma}_1^{dt}$. At this point one may ask about $\mathsf{TF\Pi}_d^{dt}$. Kleinberg et al. [KKMP21] showed that $\mathsf{TF\Pi}_d$ is efficiently reducible to $\mathsf{TF\Sigma}_{d-1}$, and vice versa. Hence, it does not offer a new perspective.

We can compare the complexity of total search problems by taking reductions between them. The following defines decision tree reductions between total search problems, the query analogue of polynomial-time reductions.

Definition 2.3. For total search problems $R \subseteq \{0,1\}^n \times \mathcal{O}_n$, $S \subseteq \{0,1\}^m \times \mathcal{O}'_m$, there is an *$S$-formulation of R* if, for every $i \in [m]$ and $o \in \mathcal{O'}_m$, there are functions $f_i : \{0,1\}^n \to \{0,1\}$ and $g_o : \{0,1\}^n \to \mathcal{O}_n$ such that

$$(f(x), o) \in S \implies (x, g_o(x)) \in R,$$

where $f(x) = (f_1(x) \ldots f_n(x))$. The *depth* of the reduction is

$$d := \max\left(\{\mathsf{depth}(f_i) : i \in [m]\} \cup \{\mathsf{depth}(g_o) : o \in \mathcal{O}'_m\}\right),$$

where $\mathsf{depth}(f)$ denotes the minimum depth of any decision tree which computes f. The *size* of the reduction is m, the number of input bits to S. The *complexity* of the reduction is $\log m + d$, and the complexity of reducing R to S is the minimum S-formulation of R.

We extend this definition to sequences in the natural way. If $S = (S_n)$ is a sequence and R_n is a single search problem, then the complexity of reducing R_n to S is the minimum over m of the complexity of reducing R_n to S_m. For two sequences of search problems $S = (S_n)$ and $R = (R_n)$, the complexity of reducing R to S is the complexity of reducing R_n to S for each n. A reduction from R to S is efficient if its complexity is $\mathsf{polylog}(n)$; we denote this by $R \leq_{dt} S$.

Syntactic and Uniform Classes. We say that a class of total search problems $\mathcal{C} \subseteq \mathsf{TF}\Sigma_d^{dt}$ has $R \in \mathcal{C}$ as its *complete problem* if for every $S \in \mathcal{C}$, $S \leq_{dt} R$. We call subclasses with complete problems *syntactic*. Further, we say that a syntactic class is *uniform* if it has a complete problem R which is uniformly generated —- there is a polynomial-time Turing machine which on input n outputs the n^{th} instance of a complete problem for that class.

3 Proof Systems for TFPH

Search problems in the black-box model are intimately tied to the complexity of propositional theorem proving. A proof is a procedure for convincing a verifier that a statement is correct. In the propositional setting, a proof convinces the verifier that a propositional formula is unsatisfiable (equivalently, its negation is a tautology).

3.1 Recap: Proof Systems for TFNP

We begin by recalling how characterizations of proof systems by TFNP^{dt} subclasses occur. We will then generalize this to TFPH. Let UNSAT be the language of all unsatisfiable propositional formulas.

Definition 3.1. A *Cook-Reckhow proof system* is a polynomial-time function $P : \{0,1\}^* \to \{0,1\}$ such that for every propositional formula $F \in \{0,1\}^*$,

$$F \in \mathsf{UNSAT} \iff \exists \Pi \in \{0,1\}^*, P(\Pi, F) = 1.$$

The *size* of proving an unsatisfiable formula F in P is $\min\{|\Pi| : P(\Pi, F) = 1\}$.

For many proof systems there is an associated width/degree measure. For example, in resolution it is the maximum number of literals in any clause appearing in a proof, and in algebraic systems such as Sherali-Adams and Sum-of-Squares it is the maximum degree of the polynomials appearing in the proof. Characterizations of TFNP^{dt} subclasses are in terms of a *complexity* parameter of the proof system, denoted

$$P(F) \ := \ \min \big\{\mathsf{width}(\Pi) + \log \mathsf{size}(\Pi) : \Pi \text{ is a } P\text{-proof of } F\big\},$$

where width is some associated width measure particular to that system. For example, for example for resolution the width measure is the number of literals in a clause, while for algebraic proof systems the width measure is typically the proof degree. This complexity measure is studied in order to account for the fact that in the black-box setting our reductions are performed by decision trees and we would like the complexity of a proof to be closely related to the complexity of a formulation; width will correspond to the depth of the decision trees in the formulation.

Typically one studies the complexity of proving the unsatisfiability of *CNF* formulas. As a CNF formula $F = C_1 \wedge \ldots \wedge C_m$ is falsified only when one of its clauses is falsified, a proof convinces the verifier that for every assignment $x \in \{0,1\}^n$ there is some clause C_i of F such that $C_i(x) = 0$. Hence, the complexity of proving that F is unsatisfiable is intimately related to the complexity of exhibiting a falsified clause, given an assignment. This is known as the *false clause search problem* $\textsc{Search}_F \subseteq \{0,1\}^n \times [m]$, defined as

$$(x, i) \in \textsc{Search}_F \iff C_i(x) = 0.$$

As F is unsatisfiable, this search problem is total, and if each clause of F contains at most $\mathsf{polylog}(n)$-many variables, it belongs to TFNP^{dt}.

The above intuition suggests that understanding TFNP^{dt} (or at least the false clause search problem) is important for understanding proof complexity. Remarkably, proof complexity is also crucial for understanding TFNP^{dt}. It turns out that TFNP^{dt} is *equivalent* to a large sub-area of proof complexity! The intuition is the following: A reduction between two total search problems is a *proof* that the first is total, assuming the totality of the second. By employing this lens, works have shown that many common proof systems are *characterized* by certain well-studied tautologies in the sense that they can prove a tautology iff there is a short reduction of that tautology to the characterizing one.

The heart of this connection is the following claim, which shows that TFNP^{dt} is *exactly* the study of the false clause search problem. The proof proceeds by expressing the totality of any problem R in TFNP^{dt} as a tautology and then taking its negation.

Claim 3.2. If $R \in \mathsf{TFNP}^{dt}$ then there is an unsatisfiable $\mathsf{polylog}(n)$-width CNF formula F_R such that $\textsc{Search}_{F_R} \in \mathsf{TFNP}^{dt}$ and $R =_{dt} \textsc{Search}_{F_R}$.

From this, characterizations of TFNP^{dt} subclasses by proof systems have been derived. We say that a syntactic subclass $\mathcal{C} \subseteq \mathsf{TFNP}^{dt}$ is *characterized* by a proof system P if for every $\textsc{Search}_F \in \mathsf{TFNP}^{dt}$, $\textsc{Search}_F \in \mathcal{C}$ iff $P(F) = \mathsf{polylog}(n)$. Note that as a proof system is a polynomial-time Turing machine, any TFNP^{dt} class must be which characterizes that proof system must be *uniform*.

3.2 Proof Systems and TFPH

The aim of this paper is to explore characterizations of classes of problems belonging to higher levels of TFPH^{dt}. These will correspond to the provability of quantified formulas.

Definition 3.3. A $\Sigma_{d.5}$ formula F is the propositional translation of any quantified formula of the form

$$\exists z_1 \in \{0,1\}^{\ell_1}\, \forall z_2 \in \{0,1\}^{\ell_2} \ldots Q z_d \in \{0,1\}^{\ell_d}\, L(x, z_1, \ldots, z_d),$$

where $\ell_i \in \mathsf{polylog}(n)$, $Q \in \{\exists, \forall\}$, and L is a formula which depends on at most $\mathsf{polylog}(n)$-many free variables (x). That is, a $\Sigma_{d.5}$ formula is of the form

$$F(x) = \bigvee_{z_1 \in \{0,1\}^{\ell_1}} \bigwedge_{z_2 \in \{0,1\}^{\ell_2}} \cdots \bigcirc_{z_d \in \{0,1\}^{\ell_d}} L_{z_1,\ldots,z_d}(x),$$

where $\bigcirc \in \{\wedge, \vee\}$, and $L_{z_1,\ldots,z_d}(x) := L(x, z_1, \ldots, z_d)$. Similarly, $\Pi_{d.5}$ formulas are negations of $\Sigma_{d.5}$ formulas.

Note that because L_z depends on $\mathsf{polylog}(n)$-many variables, we may assume without loss of generality (with a quasi-polynomial blow-up in size) that L_z is a CNF/DNF formula with clauses/terms of width $\mathsf{polylog}(n)$. Hence, a $\Sigma_{d.5}$-formula is a layered circuit of depth d where the gates at each layer are the same, and the gates at the first d layers are allowed $2^{\mathsf{polylog}(n)}$ fanin, while the final layer is restricted to have $\mathsf{polylog}(n)$ fanin. Observe that a $\Pi_{1.5}$-formula is a low-width CNF formula.

Our aim is to characterize subclasses of the higher levels of the total function polynomial hierarchy. Towards this, we generalize the false clause search problem to $\Sigma_{d.5}$ formulas.

False Formula Search. For a formula $F := \bigwedge_{o \in [m]} H_o$ where each H_o is a $\Sigma_{d.5}$-formula, the *False Formula* search problem $\mathrm{FF}_F \subseteq \{0,1\}^n \times [m]$ is defined as

$$(x, o) \in \mathrm{FF}_F \iff H_o(x) = 0.$$

Observe that if F is unsatisfiable then FF_F is total and $\mathrm{FF}_F \in \mathsf{TF}\Sigma_{d+1}^{dt}$. The following lemma generalizes Claim 3.2 to say that $\mathsf{TF}\Sigma_d^{dt}$ is *exactly* the study of the false formula search problem.

Lemma 3.4. *For every $d \geq 1$ and $R \in \mathsf{TF}\Sigma_d$ there is an unsatisfiable $\Pi_{d.5}$-formula F_R such that $(x, o) \in R$ iff $(x, o) \in \mathrm{FF}_{F_R}$.*

Proof. Let $R \subseteq \{0,1\}^n \times [m] \in \mathsf{TF}\Sigma_d$. Then there are $\mathsf{polylog}(n)$-depth decision trees $V_{o,(z_1,\ldots,z_{d-1})}$ such that

$$(x, o) \in R \iff \forall z_1 \in \{0,1\}^{\ell_1}\, \exists z_2 \in \{0,1\}^{\ell_2} \ldots Q z_{d-1} \in \{0,1\}^{\ell_{d-1}}\, V_{o,(z_1,\ldots,z_{d-1})}(x) = 1,$$

where $Q \in \{\exists, \forall\}$, $V_{o,\bar{z}} = V_{o,(z_1,\ldots,z_{i-1})}$ is a decision tree of $\mathsf{polylog}(n)$-depth, and each $\ell_j \in \mathsf{polylog}(n)$. Slightly abusing notation, let V_o be a propositional translation of the verifier as a $\Pi_{(d-1).5}$-formula:

$$V_o(x) := \bigwedge_{z_1 \in \{0,1\}^{\ell_1}} \bigvee_{z_2 \in \{0,1\}^{\ell_2}} \cdots \bigcirc_{z_{d-1} \in \{0,1\}^{\ell_{d-1}}} V_{o,\bar{z}}(x),$$

where $\bigcirc \in \{\wedge, \vee\}$, and $V_{o,\bar{z}}(x)$ is computable by a $\mathsf{polylog}(n)$-depth decision tree, and hence propositionalized as a $\mathsf{polylog}(n)$-width CNF formula if $\bigcirc = \wedge$ or a $\mathsf{polylog}(n)$-width DNF if $\bigcirc = \vee$, collapsing the top gate into $\bigcirc$. This is done as follows: Say that a root-to-leaf path in $V_{o,\bar{z}}$ is a b-path if it ends at a leaf labeled $b \in \{0,1\}$. Then, $V_{o,\bar{z}}$ is propositionalized as

- If $d-1$ is even: $\bigvee_{\text{1-path } p \in V_{o,\bar{z}}} p$,
- If $d-1$ is odd: $\bigwedge_{\text{0-path } p \in V_{o,\bar{z}}} \neg p$,

where p is the conjunction of literals queried along p (if a variable x is queried and we take branch-0, then we consider this as literal $\neg x$ and otherwise as x). Note that in this case the outer gate of $V_{o,\vec{z}}$ matches $\bigcirc$, and the depth collapses by 1. Consider the following $\Pi_{d.5}$-formula, which states that R is not total:

$$F(x) := \bigwedge_{o \in \mathcal{O}} \neg V_o(x).$$

Observe that if $(x, o) \in R$, then there is some $z_1, \ldots, z_d$ such that $V_{o,z}(x) = 1$, and hence $(x, o) \in \mathrm{FF}_F$. Conversely, if $(x, o) \in \mathrm{FF}_F$, then $(x, o) \in R$. $\square$

We will call the formula F_R the *propositionalization* of R. This lemma allows us to relate the complexity of total search problems to the provability of propositional formulas. In the remainder we will develop what provability means in this context. In particular, what are the properties of a proof system which proves the formulas that result from TFPH^{dt} search problems.

A characterization of a TFPH^{dt} class by a proof system proceeds by showing that the proof system can prove the correctness of reductions to the class. To discuss this, we will need to propositionalize reductions.

Reduced Formula. Let $R \subseteq \{0,1\}^n \times \mathcal{O}$ be a problem in $\mathsf{TF}\Sigma_d^{dt}$ and let $V_{\vec{z},o}$, $o \in \mathcal{O}$ be its verifiers. Let (f, g) be an R-formulation where $f : \{0,1\}^m \to \{0,1\}^n$, $g : \{0,1\}^m \to \mathcal{O}$, then the *reduced formula* $F_R(f, g)$ is the $\Pi_{d.5}$-formula defined as

$$F_R(f,g) \;:=\; \bigwedge_{o \in \mathcal{O}} \; \bigwedge_{\text{path } p \in g_o} \neg V_{o,p}(f(x)),$$

where $V_{o,p}(f(x)) = \bigwedge_{z_1 \in \{0,1\}^{\ell_1}} \bigvee_{z_2 \in \{0,1\}^{\ell_2}} \cdots \bigcirc_{z_{d-1} \in \{0,1\}^{\ell_{d-1}}} (V_{o,\vec{z}}(f(x)) \wedge p)$ and $V_{o,\vec{z}}(f(x))$ can be represented as a $\mathsf{polylog}(n)$-width CNF/DNF as in Lemma 3.4, using that both $V_{o,\vec{z}}$ and f are computable by $\mathsf{polylog}(n)$-depth decision trees.

Reduced formulas capture formulations in the following sense. Let $H := \bigwedge_{o \in \mathcal{O}_H} H_o$ and (f, g) be an FF_F-formulation of FF_H, where $F = \bigwedge_{o \in \mathcal{O}_F} F_o$. Then for any $o \in \mathcal{O}_F$ and any path p in g_o labelled with some $o^* \in \mathcal{O}_H$ we have that

$$\neg V_{o,p}(f(x)) = 0 \implies H_{o^*}(x) = 0. \tag{1}$$

That is, $H_{o^*} \implies \neg V_{o,p}(f)$, and we say that $\neg V_{o,p}(f)$ is a *weakening* of H_{o^*}.

A proof system P is characterized by a TFPH^{dt} class $\mathcal{C}$ with complete problem FF_F if efficient provability of F in that proof system implies low-complexity reductions to the complete problem FF_F for that class, and membership in the class $\mathcal{C}$ implies that P can prove the correctness of the reduction to P. The latter takes the following form: if (f, g) is a FF_F-formulation of a $\mathrm{FF}_H \in \mathcal{C}$ then

i) From H, P can efficiently derive the reduced formula $F(f, g)$.
ii) P has an efficient proof of $F(f, g)$.

What properties must a proof system possess in order to perform (i) and (ii) for a subclass $\mathcal{C} \subseteq \mathsf{TFPH}^{dt}$? If $\mathsf{TFNP}^{dt} \subseteq \mathcal{C}$ then a Cook-Reckhow proof system (an NP-verifier) does not suffice unless $\mathsf{NP} = \mathsf{coNP}$[1]. Interestingly, what fails is step (i) — Theorem 5.1 shows that step (ii) can always be carried out by a Cook-Reckhow system. We will need to augment Cook-Reckhow proof systems in order to perform step (i). The issue is that Cook-Reckhow systems cannot always perform the weakening from (1). That is, if $F(f,g) = \bigwedge_{o \in \mathcal{O}_{F(f,g)}} F_o$ and $H = \bigwedge_{o' \in \mathcal{O}_H} H_{o'}$ then by correctness of the reduction, we know that for every $o \in \mathcal{O}_{F(f,g)}$, F_o is a weakening of some $H_{o'}$. However, Cook-Reckhow proof systems cannot necessarily derive F_o efficiently given H. For example, if $F_o = \top$, the trivial tautology, then this is tantamount to proving that F_o is a tautology, which is a coNP-complete task. It will suffice to augment our proof systems to be able to do so.

Definition 3.5. Let P be a Cook-Reckhow proof system. A proof of a Π_{d+1} formula $F = \bigwedge_{i \in [m]} F_i$ in the proof system Σ_d-P is a pair (H, Π) such that

[1] Indeed, for any unsatisfiable 3-CNF formula F, $\mathrm{FF}_F \in \mathsf{TFNP}^{dt}$.

1. Π is a P-proof that the Π_{d+1}-formula $H = \bigwedge_{j\in[k]} H_j$ is unsatisfiable.
2. Each H_j is a Σ_d-formula such that there is some $i \in [m]$ for which $F_i \implies H_j$. That is, H_j is a Σ_d-weakening of F_i.

The *complexity* of the proof (H, Π) is $\log|H| + \log s + d$, where $\log s + d$ is the complexity of the proof Π.

Clearly such proofs are verifiable in Σ_d. As we will see, they suffice to characterize subclasses of $\mathsf{TF}\Sigma_d^{dt}$. Note as well that the definition of a Σ_1-proof system agrees with the standard definition of any proof system which corresponds to a TFNP^{dt} subclass. Indeed, Σ_1-weakening is simply weakening over clauses, which can be performed in tree resolution, which characterizes FP^{dt}.

Comparison with Proof Systems for QBF. A line of research has explored proof systems for proving the unsatisfiability of quantified boolean formulas (QBF) (see [Bey22] for a survey). These QBF proof systems appear to be weaker than Σ_d-proof systems when restricted to Σ_d-formulas. Indeed, these proof systems are typically Cook-Reckhow systems augmented with a $\forall$Reduction rule, and hence lower bounds for CNF formulas from the propositional case readily apply to these proof systems. This is not the case for Σ_d-proof systems, and further Σ_d weakening is able to simulate $\forall$Reduction.

4 Sherali-Adams and Strong Range Avoidance

We begin with an example of a characterization by showing that STRONGRANGEAVOIDANCE is characterized by Σ_2-Sherali-Adams. A full treatment of the Sherali-Adams proof system is given in the monograph [FKP19].

For any boolean formula F, we will assume without loss of generality that all negations occur at the leaves and let $\mathsf{Vars}^+(F)$ be the positive literals in F and $\mathsf{Vars}^-(F)$ be the negative literals. For any conjunct $t = \bigwedge_{x\in \mathsf{Vars}^+(t)} x \wedge \bigwedge_{x\in\mathsf{Vars}^-(t)} \neg x$, we associate the polynomial $\prod_{x\in\mathsf{Vars}^+(t)} x \prod_{x\in\mathsf{Vars}^-(t)}(1-x)$, and refer to them also as conjuncts. A *conical junta* is a sum of conjuncts $\mathcal{J} := \sum t$.

Let $D = \bigvee_t t$ be any DNF. We can express D as a degree $\deg(D) := \max_{t\in D}\deg(t)$ polynomial

$$\sum_{t\in D} t - 1.$$

Observe that for any $x \in \{0,1\}^n$, $D(x) = 1$ iff $\sum_{t\in D} t(x) - 1 \geq 0$. Henceforth we will abuse notation and refer to D as both the DNF and the associated polynomial.

Throughout this section we will work with *multi-linear arithmetic*, associating $x_i^2 = x_i$ for every variable x. This has the effect of restricting the underlying linear program to $\{0,1\}$-points.

Definition 4.1. Let $F = \{D_i\}_{i\in[m]}$ be an unsatisfiable collection of DNFs. A Σ_2-*Unary Sherali-Adams* (which we denote by uSA) proof Π of F is a weakening $F' = \{D'_i\}_{i\in[m']}$ of F together with a list of conical juntas $\mathcal{J}_i, \mathcal{J}$, such that

$$\sum_{i\in[m']} D'_i\mathcal{J}_i + \mathcal{J} = -1.$$

The *degree* $\deg(\Pi)$ is the maximum degree among $D_i, D'_i\mathcal{J}_i$, and $\mathcal{J}$, and the *size* $\mathsf{size}(\Pi)$ is the number of monomials (counted with multiplicity)[2] in $D_i, D'_i\mathcal{J}_i, \mathcal{J}$. The *complexity* of the proof is $\mathsf{uSA}(\Pi) := \deg(\Pi) + \log\mathsf{size}(\Pi)$, and the complexity of proving F is $\mathsf{uSA}(F) := \min_\Pi \mathsf{uSA}(\Pi)$, where the minimum is taken over all uSA proofs of F.

Note also that weakening subsumes the need to explicitly allow the additional conical junta in a uSA proof; we could instead defined uSA as a Nullstellensatz proof $\sum D'_i\mathcal{J}_i = -1$. This is because the additional junta $\mathcal{J}$ may be introduced using weakening: for each conjunct t of $\mathcal{J}$, weaken some D_i in F to *true* or t. For example, D_i can be weakened to $x_i \vee \neg x_i \vee t$, the polynomial encoding of which is $x_i + (1-x_i) + t - 1 = t$.

Claim 4.2. uSA is sound and complete.

[2]This is a *unary* proof system as the number of monomials are counted with multiplicity, akin to writing coefficients in unary, rather than allowing for monomials with coefficients written in binary

Proof. Suppose that uSA is not sound; then there exists a uSA refutation of a satisfiable DNF $F = \{D_i\}_{i\in[m]}$,

$$\sum_{i\in[m']} D'_i \mathcal{J}_i + \mathcal{J} = -1.$$

Let $x \in \{0,1\}^n$ be a satisfying assignment to F, meaning that for every i, $D'_i(x) = 1$ for any weakening D'_i of D_i, and in particular the polynomial representation of $D'_i(x) \geq 0$. As juntas are non-negative over $\{0,1\}^n$, we have that

$$\sum_{i\in[m']} D'_i(x)\mathcal{J}_i(x) + \mathcal{J}(x) \geq 0,$$

which is a contradiction.

For completeness, let $F = \{D_i\}_{i\in[m]}$ be an unsatisfiable formula. Each assignment $x \in \{0,1\}^n$ must falsify some DNF of F, which we will denote by D_x. Let I_x be the indicator polynomial $I_x := \prod_{i:x_i=1} x_i \prod_{i:x_i=0}(1-x_i)$ of the assignment x. We claim that the polynomial

$$\sum_{x\in\{0,1\}^n} I_x D_x = -1,$$

and is therefore a uSA proof. To see this, since we are working over the ideal $\langle x_i - x_i^2\rangle$, it suffices to show that the polynomial evaluates to -1 on every $x \in \{0,1\}^n$. Observe that if $y \in \{0,1\}^n$ falsifies D_x, then $D_x(y) = -1$; additionally, if $x \neq y$, then $I_x(y) = 0$. Hence, for every $y \in \{0,1\}^n$,

$$\sum_{x\in\{0,1\}^n} I_x(y)D_x(y) = I_y(y)D_y(y) = D_y(y) = -1.$$

□

In the rest of this section, we show that uSA is closely related to STRONGRANGEAVOIDANCE. We restate an equivalent definition next.

Definition 4.3. An instance of STRONGRANGEAVOIDANCE (STRONGAVOID) is given by a map $f : [n] \to [n+1]$. A solution is any $h \in [n+1]$ such that for every $p \in [n]$, $f(p) \neq h$.

STRONGAVOID can be encoded as a CNF formula by introducing, for every $p \in [n]$, $\log(n+1)$-many binary variables $p_1, \ldots, p_{\log n+1}$ naming in binary the hole $h \in [n+1]$ to which pigeon p flies. For exposition, it will be convenient to think of p as an $(n+1)$-ary variable and we will denote by $[\![p = h]\!]$ the indicator conjunct that is satisfied iff p maps to $h \in [n+1]$ under the given assignment

$$[\![p = h]\!] \;:=\; p_1^{h_1} \wedge \ldots \wedge p_{\log(n+1)}^{h_{\log(n+1)}},$$

where $p_i^{h_i} = p_i$ if the i^{th} bit of h is 1 and $\neg p_i$ otherwise. Note that $\sum_{h\in[n+1]} [\![p = h]\!] = 1$ as polynomials.

We can express STRONGAVOID as the unsatisfiable family of DNFs,

$$\bigvee_{p\in[n]} [\![p = h]\!] \qquad\qquad \forall h \in [n+1].$$

The main theorem of this section is the following.

Theorem 4.4. *For any* $\mathsf{FF}_F \in \mathsf{TF}\Sigma_2^{dt}$ *there is a complexity-c* STRONGAVOID*-formulation of* FF_F *iff there is a* Σ_2-uSA *proof of F of complexity* $\Theta(c)$.

We break the proof of this theorem into Lemma 4.5 and Lemma 4.8, which are proven over the following two subsections. This theorem gives necessary and sufficient conditions for separating other $\mathsf{TF}\Sigma_2$ classes $\mathcal{C}$ from STRONGAVOID: Exhibit a pseudo-expectation (see e.g., [FKP19]) against any polylog(n)-width Σ_2-weakening of the propositionalization of STRONGAVOID.

4.1 SA Proofs Imply sRA Reductions

Lemma 4.5. *If there is a size s and degree d Σ_2-*uSA *proof of F, then there is a depth-d reduction from* FF_F *to an instance of* StrongAvoid *of size $O(s)$.*

To prove this lemma, it will be convenient to work with the following problem. We show that it is equivalent to StrongAvoid.

Definition 4.6. The *Unmetered Source of Dag* (USoD) problem is defined as follows. The input is a "successor" function $S : [n] \to [n]$ which defines a graph in which each vertex has fan-out ≤ 1 but arbitrary fan-in. There is an edge from i to j if $S(i) = j$. To make the problem total, we enforce that the vertex 1 is a sink; it will have fan-out 0 but fan-in at least 1. The goal is to find a source; the solutions are:

- 1 is a solution if either $S(1) \neq 1$ or $\forall v \neq 1 \in [n]$, $S(v) \neq 1$ (1 is not a sink).
- $v \in [n]$ is a solution if $S(v) \neq v$ but $\forall u \in [n]$, $S(u) \neq v$ (v is a source).

Lemma 4.7. USoD $=_{dt}$ StrongAvoid. *Furthermore, this reduction is by depth-*1 *decision trees.*

Proof. From an instance $S : [n] \to [n]$ of USoD, we construct an instance $f : [n] \to [n+1]$ of StrongAvoid as follows. For $v \neq 1 \in [n]$, let $f(v) := S(v)$ and let $f(1) := n+1$. We claim that any solution u to this StrongAvoid instance is a source in S. First observe that $u \neq n+1$ as $f(1) = n+1$. Hence, by construction, we have that $\forall v \in [n], S(v) \neq u$, and in particular $S(u) \neq u$, so u is a source.
For the converse direction, from an instance $f : [n] \to [n+1]$ of StrongAvoid we construct an instance $S : [n+1] \to [n+1]$ of USoD by defining $S(v+1) := f(v)$ for all $v \in [n]$ and letting $S(1) = 1$. Let v be a solution to this instance of USoD, if $v = 1$, then, since $S(1) = 1$, for all $u \in [n]$, $f(u) \neq 1$. Otherwise, $v \neq S(u)$ for all $u \in [n+1]$, and so $v \neq f(u)$ for all $u \in [n]$. $\square$

Proof of Lemma 4.5. Let $F = \bigwedge_{o \in \mathcal{O}} D_o$, and let Π be a size s and degree d uSA proof of F over n variables, where

$$\Pi := \sum_{i \in [m]} \sum_{j \in I_i} D'_i J_j + \sum_{k \in K} J_k + 1 = 0,$$

for sets of indices I_i, K, each D'_i is a weakening of some $D_o \in F$ and each J_j, J_k is a conjunct. We construct an instance of USoD with one node per occurrence of a (signed) monomial in Π. Therefore, for simplicity, we will refer to monomials as nodes and vice versa. The constant 1 will be our distinguished sink, and we will set $S(1) = 1$. We will define the remaining successor pointers as follows:

Negative Monomials. Since $\Pi = 0$, there is a positive and negative copy of every monomial occurring in the proof; construct a pairing of the monomials in this way. Furthermore, under any assignment $x \in \{0,1\}^n$, the number of monomials which evaluate to 1 and to -1 is equal. For each negative monomial $-m$ in Π, the decision tree $S(-m)$ queries the variables of m and outputs as follows:

i) If $m(x) = 0$, then $S(-m) = -m$.
ii) Otherwise, let m be the positive copy of $-m$ that $-m$ is paired with, and set $S(-m) = m$.

This completes the description of the successor pointer for negative monomials.

Positive Monomials. For any positive monomial m, the decision tree for $S(m)$ first queries the (at most d-many) variables of m to determine the value of $m(x)$. If $m(x) = 0$, then $S(m) = m$. Otherwise, we will define S as follows.

We define the successor pointer for the positive monomials which belong to each $D'_i J_j$ first, and handle the monomials from the conjuncts J_k later. Fix some $D'_i J_j$ in Π, where $D'_i = \sum_{k \in [\ell]} t_k - 1$, and consider the monomials within it. We would like to satisfy the following property: there is a source within the monomials $D'_i J_j$ iff $D'_i(x) = -1$ (i.e., the DNF $D'_i(x) = 0$). To get some intuition, first suppose that $J_j = 1$ and that all monomials m in D'_i are positive — that is, $D'_i J_j = \sum_{k \in [\ell]} m_k - 1$. Then, the current assignment to S affects $D'_i J_j$ as follows:

- Each monomial m_k such that $m_k(x) = 0$ is an isolated vertex for which $S(m_k) = m_k$.
- Each monomial m_k for which $m_k(x) \neq 0$ has a single incoming edge (from $-m_k$).
- The monomial -1 has an outgoing edge.

If at least one of the monomials m_k is non-zero, we can send it to -1, and otherwise -1 becomes a source (see Figure 2). Therefore, the only sources will come from the "-1 nodes" of falsified DNFs. To handle the general case, we use the fact that in every conjunct, under any assignment, there are at least as many non-zero positive monomials as non-zero negative monomials.

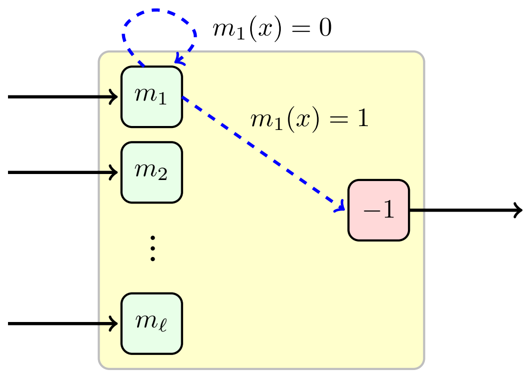

Figure 2: The "gadget" for a $D_i'J_j$ where $J_j = 1$ and D_i' contains only positive literals (each conjunct is a monomial).

We now describe the construction in general. Consider a $D_i'J_j$ in Π. For each positive monomial m in $D_i'J_j = (\sum_{k\in[\ell]} t_k - 1)J_j$, belonging to some conjunct t_kJ_j, the pointer $S(m)$ will query the (at most d-many) variables in t_kJ_j. Let $\alpha \in \{0,1\}^{\mathsf{Vars}(J_j)}$ be the assignment to the variables of J_j that was discovered.

If $J_j \restriction \alpha = 0$: Then $D_i'J_j \restriction \alpha = 0$. Hence, for every positive monomial m in $D_i'J_j$, either $m \restriction \alpha = 0$, in which case we have already set $S(m) = m$, or m must cancel with another monomial $-m'$ in $D_i'J_j$ under α. That is, $m \restriction \alpha = -m' \restriction \alpha$, and so we define $S(m) = -m'$. Note that in this case there are no sources within $D_i'J_j$: every monomial m $D_i'J_j$ either evaluates to 0 and nothing points to it, or has exactly one incoming and one outgoing edge.

If $J_j \restriction \alpha \neq 0$: We define the successor pointer for the monomials in $D_i'J_j$ so that there is a source iff every for every term t_k of D_i', $t_k(x) = 0$. Let $\mathsf{Mons}(J_j)^+, \mathsf{Mons}(J_j)^-$ be the (non-zero) positive and negative monomials in J_j respectively. Let

$$\delta \;:=\; |\mathsf{Mons}^+(J_j \restriction \alpha)| - |\mathsf{Mons}^-(J_j \restriction \alpha)|$$

be the difference between the number of positive and negative monomials, and note that $\delta > 0$ as J_j is a conjunct and $J_j \restriction \alpha \neq 0$. Recall that $D_i'J_j = \sum_{k\in[\ell]} t_kJ_j - J_j$. For each term, we will define a matching so that $-J_j$ has only δ-many negative monomials without incoming edges, and every negative monomial in t_kJ_j has an incoming edge.

- For $-J_j$: Define an arbitrary pairing $P \;:=\; \{(m, -m')\} \subseteq \mathsf{Mons}^+(J_j \restriction \alpha) \times \mathsf{Mons}^-(J_j \restriction \alpha)$ such that each positive monomial occurs in exactly one pair and each negative monomial occurs in at most one pair. Hence we have δ-many negative monomials that are not paired. For each pair $(m, -m') \in P$, define $S(m) = -m'$.

  Note that we have now defined the successor of every positive monomial in J_j.
- For each t_kJ_j: Observe that as t_k is a conjunct, under any assignment it contains at least as many positive monomials as negative monomials. Define an arbitrary pairing $P := \{(m, -m')\} \subseteq \mathsf{Mons}^+(t_kJ_j \restriction \alpha) \times \mathsf{Mons}^-(t_kJ_j \restriction \alpha)$ such that each negative monomial occurs in exactly one pair and each positive monomial occurs in at most one pair. For each pair $(m, -m') \in P$, define $S(m) = -m'$. Let $\beta \in \{0,1\}^{\mathsf{Vars}(t_kJ_j)}$ be the assignment to the variables of t_kJ_j that was discovered by the trees made by the decision tree of any of the monomials m in t_kJ_j. Let

  $$c \;:=\; |\mathsf{Mons}^+(t_k \restriction \beta)| - |\mathsf{Mons}^-(t_k \restriction \beta)|$$

  be the difference between the number of non-zero positive and negative monomials in t_k under β.

  If $c = 0$, then $t_kJ_j \restriction \beta = 0$, and so the number of non-zero positive and negative monomials is equal. In this case, each negative monomial has an incoming edge, which is provided by this pairing.

  Otherwise, if $t_kJ_j \restriction \beta \neq 0$, then there are $c\delta$-many non-zero positive monomials whose successor is still undefined, and partition them into c-many groups of $C_1, \ldots C_\delta$ of δ-many monomials each. Recall that $-J_j$

has exactly δ-many negative monomials with no incoming edge, $-m_1, \ldots, -m_\delta$. For each $m \in C_i$, define $S(m) = -m_i$. In this case, each monomial in $t_k J_j$ and $-J_j$ has an incoming edge.

Finally, we define the successor for each positive monomial in the conical junta $\sum_{k \in K} J_k$ in Π. We will do this individually for each J_k. To do so, we use the fact that J_k contains at least as many positive monomials as negative monomials in order to ensure that there is never any source among the monomials of J_k. The successor for each positive monomial m of J_k queries the (at most d-many) variables in J_k for an assignment $\alpha \in \{0,1\}^{\mathsf{Vars}(J_k)}$. Define an arbitrary pairing $P := \{(m, -m')\} \subseteq \mathsf{Mons}^+(J_k \restriction \alpha) \times \mathsf{Mons}^-(J_k \restriction \alpha)$ such that each negative monomial occurs in exactly one pair and each positive monomial occurs in at most one pair. For each pair $(m, -m') \in P$, define $S(m) = -m'$. For the remaining positive monomials m in J_k whose successor is not defined, set $S(m) = 1$ (this choice is somewhat arbitrary).

This completes the description of the successor function S (the f-part of the formulation). It remains to define the output function g of the formulation. For each potential solution m,

- If m is a monomial from some $D'_i J_j$, then D'_i is the weakening of some D_o of F, and we output o.
- Otherwise, we output an arbitrary index $o \in [m]$.

Finally, we prove that this formulation is correct. To do so, we show that the only monomials which do not have incoming edges belong to some $D'_i J_j$ for which $D'_i(x)$ is falsified. This suffices, as if m belongs to $D'_i J_j$ where $D'_i(x) = 0$, then $g_m(x) = o$ for some D_o of F of which D'_i is a weakening of. Hence, $D_o(x)$ is falsified, and we have found a solution to FF_F. By the *negative monomial case* in the formulation, every positive monomial has an incoming edge. By the pairings constructed in the formulation, every negative monomial in each J_k in the conical junta also has an incoming edge. As well, for each $D'_i J_j = \sum_k t_k J_j - J_j$, each negative monomial in each $t_k J_j$ has an incoming edge. Hence, the only potential sources belong to the $-J_j$ terms of each $D'_i J_j$. As we argued before, if $J_j(x) = 0$, then there is no source in the monomials of $D'_i J_j$, so suppose that this is not the case. As we have paired off positive and negative monomials in $-J_j$, the only incoming edge to each of the δ-many remaining negative monomials of J_j must come from some $t_k J_j$. If there is a t_k such that $t_k(x) \neq 0$ (and hence $D'_i(x)$ is satisfied), then $t_k J_j$ has $c\delta$-many monomials which map to to the δ-many remaining negative monomials of J_j, meaning that there is no source in $D'_i J_j$. Thus, $D'_i J_j$ becomes a source only if $J_j(x) \neq 0$ and $D'_i(x)$ is falsified. $\square$

4.2 sRA Reductions Imply SA Proofs

We begin by observing that there is a trivial unary Sherali-Adams refutation of Range Avoidance:

$$\sum_{h \in [n+1]} \Big(\sum_{p \in [n]} [\![p = h]\!] - 1 \Big) = \sum_{p \in [n]} \sum_{h \in [n+1]} [\![p = h]\!] - (n+1) = n - (n+1) = -1,$$

where the third equality follows as we $\sum_{h \in [n+1]} [\![p = h]\!] = 1$.

In the remainder of this section, we will show that unary Sherali-Adams can prove reductions to StrongAvoid.

Lemma 4.8. *If f, g is a* StrongAvoid*-formulation of* FF_F *of depth d and size s, then there is a degree-$O(d \log n)$ and size* $\mathrm{poly}(s \cdot n^d)$ Σ_2-uSA *proof of F.*

If (f, g) is a StrongAvoid-formulation of FF_F for some formula $F = \bigwedge_{i \in [m]} D_i$, let $P(g_h), P(f_p)$ be the set of all root-to-leaf paths in the decision trees g_h and f_p, respectively. As well, for any hole $h \in [n+1]$, let $P_h(f_p)$ be the set of paths in f_p whose leaf is labeled by hole h.

We can express the reduction from FF_F to StrongAvoid as the unsatisfiable formula StrongAvoid(f, g) defined as

$$\begin{aligned} &\forall h \in [n+1],\ \forall \sigma^* \in P(g_h), \bigvee_{\sigma \neq \sigma^* \in P(g_h)} \bigvee_{p \in [n]} [\![p = h]\!] \wedge \sigma \\ &= \forall h \in [n+1],\ \forall \sigma^* \in P(g_h), \bigvee_{\sigma \neq \sigma^* \in P(g_h)} \bigvee_{p \in [n]} \bigvee_{\rho \in P_h(f_p)} \rho \wedge \sigma \end{aligned}$$

Letting $D_{h,\sigma^*} := \bigvee_{\sigma \neq \sigma^* \in P(g_h)} \bigvee_{p \in [n]} \bigvee_{\rho \in P_h(f_p)} \rho \wedge \sigma$, this becomes the unsatisfiable family of DNFs

$$\textsc{StrongAvoid}(f, g) := \{D_{h,\sigma^*}\}_{h \in [n+1], \sigma^* \in P(g_h)}.$$

The following lemma shows that uSA can deduce StrongAvoid(f, g) from F.

Proof of Lemma 4.8. We will abuse notation and let $[\![p = h]\!] := \sum_{\delta \in P_h(f_p)} \delta$ denote the decision-tree substitution of the indicator $[\![p = h]\!]$. To begin, we will weaken F to $\textsc{StrongAvoid}(f, g)$, the polynomials of which are

$$D_{h,\sigma^*} := \sum_{\sigma \neq \sigma^* \in P(g_h)} \sum_{p \in [n]} [\![p = h]\!] \cdot \sigma - 1$$

for $h \in [n+1]$ and $\sigma^* \in P(g_h)$. As each $[\![p = h]\!]$ contains $O(\log n)$-many Boolean variables, and we are replacing each one by a depth-d decision tree, the degree of $\textsc{StrongAvoid}(f, g)$ is $O(d \log n)$. Similarly, the size blows up by a factor of n^d.

For any $h \in [n+1]$,

$$\begin{aligned}
\sum_{\sigma^* \in P(g_h)} D_{h,\sigma^*} &= \sum_{\sigma^* \in P(g_h)} \Big(\sum_{\sigma \neq \sigma^* \in P(g_h)} \sum_{p \in [n]} [\![p = h]\!]\sigma - 1 \Big) \\
&= \sum_{p \in [n]} \sum_{\sigma^* \in P(g_h)} \sum_{\sigma \neq \sigma^* \in P(g_h)} [\![p = h]\!]\sigma - |P(g_h)| \\
&= \sum_{p \in [n]} [\![p = h]\!] \sum_{\sigma^* \in P(g_h)} \sum_{\sigma \neq \sigma^* \in P(g_h)} \sigma - |P(g_h)| \\
&= \sum_{p \in [n]} [\![p = h]\!] \Big(|P(g_h)| - 1 \Big) \sum_{\sigma \in P(g_h)} \sigma - |P(g_h)| \\
&= \sum_{p \in [n]} [\![p = h]\!] \Big(|P(g_h)| - 1 \Big) - |P(g_h)| && \text{(Summing all paths in the DT } g(h)\text{)} \\
&= \big(|P(g_h)| - 1 \big) \Big(\sum_{p \in [n]} [\![p = h]\!] - 1 \Big)
\end{aligned}$$

By padding, we can assume without loss of generality that all decision trees g_h have the same number of paths; that is, $|P(g_h)| = |P(g_{h'})| = \alpha$ for all $h, h' \in [n+1]$, and some $\alpha \in \mathbb{N}$ with $\alpha > 1$. Then,

$$\begin{aligned}
\sum_{h \in [n+1]} \sum_{\sigma^* \in P(g_h)} D_{h,\sigma^*} &= \sum_{h \in [n+1]} (\alpha - 1) \Big(\sum_{p \in [n]} [\![p = h]\!] - 1 \Big) \\
&= (\alpha - 1) \Big(\sum_{p \in [n]} \sum_{h \in [n+1]} [\![p = h]\!] - (n+1) \Big) \\
&= (\alpha - 1) \Big(\sum_{p \in [n]} \sum_{h \in [n+1]} \sum_{\delta \in P_h(p)} \delta - (n+1) \Big) \\
&= (\alpha - 1) \Big(\sum_{p \in [n]} \sum_{\delta \in P(p)} \delta - (n+1) \Big) \\
&= (\alpha - 1) \Big(\sum_{p \in [n]} 1 - (n+1) \Big) && \text{(Summing all paths in the DT } f(p)\text{)} \\
&= (\alpha - 1) \big(n - (n+1) \big) = -(\alpha - 1) \leq -1. && \square
\end{aligned}$$

5 A Generic Correspondence

In this section we establish a general correspondence between uniform subclasses of total search problems in the polynomial hierarchy and proof systems. Our characterizations will rely on the following two properties of a Σ_d-proof system:

- *Reduction-Closed.* For unsatisfiable $\Pi_{d.5}$ formulas F, H, if P has a complexity-s proof of F and there is a complexity-c FF_F-formulation of FF_H, then $P(H) = \mathrm{poly}(cs)$.
- *Reflective.* P has $\mathrm{polylog}(n)$-complexity proofs of a *reflection principle* about itself—a formula encoding the soundness of this proof system; we expand on the meaning of this below.

We show the following, generalizing [BFI23].

Theorem 5.1. *The following hold:*

i) Every uniform subclass of $\mathsf{TF}\Sigma_d$ *is characterized by a* Σ_d*-proof system.*

ii) Every Σ_d*-proof system which is reduction-closed and reflective is characterized by a subclass of* $\mathsf{TF}\Sigma_d$*.*

We prove (i) in subsection 5.1 and (ii) in subsection 5.2.

5.1 A Proof System for any $\mathsf{TF}\Sigma_d$ Problem

In this section we show how to construct a proof system from any total search problem $R \subseteq \{0,1\}^n \times \mathcal{O}$, which we think of as the complete problem for some uniform subclass. The key insight is that one can view a decision tree reduction from a total search problem $Q \subseteq \{0,1\}^m \times \mathcal{O}_Q$ to R as a *proof* that Q is total, if we take the totality of R as an axiom. In what follows, we formalize this intuition. We define proofs in the *canonical proof system* for a $\mathsf{TF}\Sigma_d^{dt}$ subclass as reductions to one of its complete problems.

Definition 5.2. Let $\mathrm{FF}_F \in \mathsf{TF}\Sigma_d^{dt}$ where $F_n = \bigwedge_{o\in[m]} F_o$. The *canonical proof system* for FF_F, denoted P_F, is defined as follows: A proof Π in P_F consists of a triple $(n, f, g, F_n(f, g))$, where

- (f, g) is a FF_F-formulation (i.e., a set of decision trees) to an instance of FF_F on n variables, and
- $F_n(f, g) = \bigwedge_{o\in[m^*]} L_o$ is the reduced formula associated with this formulation.

Π is a P_F proof of an unsatisfiable formula $H = \bigwedge_{t\in[m']} H_t$, where each H_t is a Σ_d-formula, if for every L_o in $F_n(f, g)$ there exists some $t \in [m']$ such that L_o is a Σ_d-weakening of L_o; that is,

$$H_t \implies L_o.$$

The *size* of the proof Π is the number of bits needed to write down Π, and the width of Π is the maximum depth among the decision trees in the formulation,

$$\mathsf{depth}(\Pi) := \max_{i\in[n],o\in[m]} \big\{\mathsf{depth}(f_i), \mathsf{depth}(g_o)\big\}.$$

The *complexity* of proving H in P_F is the minimum over all P_F-proofs of H,

$$P_F(H) := \min \big\{\mathsf{width}(\Pi) + \log \mathsf{size}(\Pi) : \Pi \text{ is a } P_F\text{-proof of } H\big\}.$$

This proof system is sound, since any substitution of an unsatisfiable formula remains unsatisfiable. As well, it is complete for unsatisfiable Π_{d+1} formulas, as depth-n decision trees suffice to solve any total search problem. It is also verifiable by a polynomial-time Π^P_{d-1} machine which generates F_n, checks that (f, g) is a valid FF_F-formulation which agrees with the reduced formula $F_n(f, g)$, and checks that $H_t \implies L_o$ Note that this proof system agrees with the definition of [BFI23] when $d = 1$.

We will show that P_F characterizes the subclass with complete problem FF_F, proving the first direction of Theorem 5.1.

Lemma 5.3. *If* $\mathrm{FF}_F, \mathrm{FF}_H \in \mathsf{TF}\Sigma_d^{dt}$*, then there is a complexity-*c FF_F*-formulation of* FF_H *iff* $P_F(H) \le c\cdot\mathsf{polylog}(n)$*.*

Proof. Let (f, g) be a complexity-c FF_F-formulation of FF_H. We claim that $(n, f, g, \mathrm{FF}_F(f, g))$ is a P_F proof of H. As $\mathrm{FF}_F \in \mathsf{TF}\Sigma_d^{dt}$, F is a $\Pi_{d.5}$ formula, and so the reduced formula $\mathrm{FF}_F(f, g)$ is a Π_{d+1}-formula ($\Pi_{d.5}$ if $c = \mathsf{polylog}(n)$). As well, the size of $\mathrm{FF}_F(f, g)$ is at most $\mathsf{size}(\mathrm{FF}_F) \cdot \exp(O(c))$, as each clause/term on the bottom layer of F has width at most $\mathsf{polylog}(n)$ and we replace it by the CNF/DNF representation of a depth-$O(c)$ decision tree, which has width $O(c)$ and size at most $\exp(O(c))$. Finally, for $F(f, g) := \bigwedge_{o\in[m^*]} L_o$ and $H := \bigwedge_{t\in[m]} H_t$, by the correctness of the formulation, we can conclude that for every $o \in [m^*]$ there exists some $t \in [m']$ such that $H_t \implies L_o$, and so L_o is a Σ_d-weakening of H_t.

For the converse direction, suppose that $(n, f, g, F(f, g))$ is a P_F proof of an unsatisfiable formula $H := \bigwedge_{i\in[m]} H_i$, where each H_i is a Σ_d-formula. By definition, (f, g) constitutes a complexity-c FF_F-formulation of FF_H. Indeed, each decision tree of (f, g) has depth at most c and there are at most 2^c-many of them, and so this is a complexity-c formulation. ☐

5.2 A $\mathsf{TF}\Sigma_d$ Problem for any Proof System which Reflects

In this section we show that a Σ_d-proof system P corresponds to a $\mathsf{TF}\Sigma_d$-problem if that proof system is *reduction closed* and *reflective*.

A reflection principle states that P-proofs are sound; we will restrict ourselves to proofs of $\Sigma_{d.5}$ formulas. Typically, the provability of a proof system's reflection principle is sufficient in order to simulate that system. In our setting, a reflection principle will falsely assert that there is a complexity-c P-proof Π of a $\Sigma_{d.5}$-formula H and that H is satisfied by a truth assignment α:

$$\textsc{Ref}_P \ := \ \textsc{Proof}(H, \Pi) \wedge \textsc{Sat}(H, \alpha).$$

This formula will be parameterized by n_H, the number of variables of H, as well as c the complexity of the proof Π.

SAT. The formula $\textsc{Sat}(H, \alpha)$ states that $\alpha \in \{0,1\}^{n_H}$ is a satisfying assignment to H, where $\alpha \in \{0,1\}^n$ and H are given as input. A generic $\Pi_{d.5}$-formula has the following structure:

$$H = \bigwedge_{o \in \mathcal{O}} \bigvee_{z_1 \in \{0,1\}^{\ell_1}} \bigwedge_{z_2 \in \{0,1\}^{\ell_2}} \cdots \bigcirc_{z_{d-1} \in \{0,1\}^{\ell_{d-1}}} H_{o, z_1, \ldots, z_{d-1}}$$

where $\bigcirc \in \{\wedge, \vee\}$ and $H_{o,z_1,\ldots,z_{d-1}}$, is a width $w \in \mathsf{polylog}(n)$ clause if $\bigcirc = \wedge$ or conjunct if $\bigcirc = \vee$. Each $H_{o,\vec{z}} := H_{o,z_1,\ldots,z_{d-1}}$ is specified by w-many $(2n+1)$-ary variables $v_{o,z,1}, \ldots, v_{o,z,w} \in [2(n+1)]$, where $v_{o,z,i} = j$ denotes the variable

- x_j if $i \in [n]$,
- $\neg x_{j-n}$ if $j \in \{n+1, \ldots, 2n\}$,
- constant 1 if $j = 2n+1$,
- constant 0 if $j = 2n+2$.

We could allow the formula $\textsc{Ref}_P$ to be parameterized by $|\mathcal{O}|, \ell_1, \ldots, \ell_{d-1}$. However, for simplicity, since we are considering complexity-c proofs, it suffices to simply set all of these parameters to 2^c and $w = c$. In this case, the number of $H_{o,\vec{z}}$ is 2^{cd}, and hence the number of Boolean variables of H is $c \log(2n_H + 2) \cdot 2^{cd}$. Then the $\Pi_{d.5}$ formula $\textsc{Sat}$ can be written as

$$\textsc{Sat}(H, \alpha) \ := \ \bigwedge_{o \in \mathcal{O}} H_o(\alpha) := \bigwedge_{o \in \mathcal{O}} \bigvee_{z_1 \in \{0,1\}^{\ell_1}} \bigwedge_{z_2 \in \{0,1\}^{\ell_2}} \cdots \bigcirc_{z_{d-1} \in \{0,1\}^{\ell_{d-1}}} [\![H_{o,\vec{z}}(\alpha) = 1]\!],$$

where $[\![H_{o,\vec{z}}(\alpha) = 1]\!]$ is the width-$O(w \log n_H)$ DNF (if $\bigcirc = \vee$) or CNF (if $\bigcirc = \wedge$) defined by the following decision tree $T_{o,\vec{z}}$: First query the $w \log(2n_H + 2)$-many Boolean variables $H_{o,\vec{z},1}, \ldots, H_{o,\vec{z},w}$ to determine the literals $\ell_1, \ldots, \ell_w$ of $H_{o,\vec{z}}$. Then, query the corresponding bits of α to determine if $H_{o,\vec{z}}$ is satisfied. If it is, then $T_{o,\vec{z}}$ outputs 1, and otherwise it outputs 0. This can be converted into a DNF or CNF in the usual way.

Proof. The formula $\textsc{Proof}(H, \Pi)$ states that Π is a P-proof of H. A complication is that there are many different ways by which one could encode a P-proof as a formula, some of which may change the difficulty of proving the reflection principle drastically. Following [BFI23], we define one reflection principle for each encoding of a P-proof; we call such an encoding a *verification procedure*.

Definition 5.4. A *verification procedure* V for a Σ_d-proof system P, parameterized by n_H, c, is a $\Pi_{d.5}$-formula which generically encodes a complexity-c P-proof Π of an n_H-variate formula H. Specifically, the formula $V_{n_H,c}(\Pi, H)$ has two sets of variables H, Π, where:

- An assignment to the variables $H = \{H_{o,\vec{z},i} | i \in [n_H]\}$ specifies a $\Pi_{d.5}$ formula as before.
- An assignment to the variables Π specifies a purported P-proof of H of complexity c, such that any error in Π can be verified by an efficient Σ_{d-1}-algorithm (placing $\textsc{Ref} \in \mathsf{TF}\Sigma_d$).
- V has $2^{\Theta(c)}$-many variables.

As c bounds the logarithm of the size of the proof, and the number of variables is exponential in $\Theta(c)$, the second condition ensures that a violated sub-formula of V can be verified by a Σ_{d-1}-algorithm making $\mathsf{polylog}(c)$-many queries.

A *reflection principle* for a proof system P and verification procedure V is

$$\textsc{Ref}_{P,V} := \textsc{Proof}_{n_H,c}(H,\Pi) \wedge \textsc{Sat}_{n_H,c}(H,\alpha),$$

where $\textsc{Proof}_{n_H,c}(H,\Pi) := V_{n_H,c}(H,\Pi)$. Often, we will suppress the subscripts P, V.

We now prove point (ii) of Theorem 5.1.

Lemma 5.5. *Let P be a Σ_d-proof system that is reduction closed and reflective for some $\textsc{Ref} := \textsc{Ref}_{P,V}$. Then for any $\mathsf{FF}_H \in \mathsf{TF}\Sigma_d$,*

i) If there is a complexity-c $\mathrm{FF}_{\textsc{Ref}}$-formulation of FF_H, then $P(H) = \mathrm{poly}(c \cdot P(\textsc{Ref}))$.
ii) There is a complexity $O(P(H))$ $\mathrm{FF}_{\textsc{Ref}}$-formulation of FF_H.

Proof. To prove (i), suppose that there is a complexity-c $\mathrm{FF}_{\textsc{Ref}}$-formulation of H. By the definition of being reduction closed, there is a P proof of H of complexity $\mathrm{poly}(c \cdot P(\textsc{Ref}))$.

For (ii), let Π be a complexity-c proof of H in P. We construct a $\mathrm{FF}_{\textsc{Ref}}$-formulation (f, g) of FF_H as follows. f will hard-wire (Π, H) as the input to $\textsc{Ref}$, and map the input variables of FF_H to the variables $\alpha_1, \ldots, \alpha_{n_H}$ of $\textsc{Ref}$. Since Π is a valid proof of H, $\textsc{Proof}(\Pi, H)$ is always satisfied, and we can set g_o arbitrarily for any solution o corresponding to a subformula of $\textsc{Proof}(\Pi, H)$. As $\textsc{Proof}(\Pi, H)$ is always satisfied under this reduction, the only solutions which may occur belong to $\textsc{Sat}(H, \alpha)$. In particular, as we have mapped the input variables of H to the bits $\alpha_1, \ldots, \alpha_{n_H}$, for any assignment $x \in \{0,1\}^n$, $H_o(x) = 0 \iff H_o(\alpha) = 0$. Hence, we define $g_o = o$. $\square$

6 Characterizations in $\mathsf{TF}\Sigma_2$

In this section we uncover $\mathsf{TF}\Sigma_2$ characterizations of several well-studied proof systems — DNF Resolution, DNF Circular Resolution [AL23, DR23], and DNF Reversible Resolution [GHJ+22b, DR23]. Along the way we introduce several new $\mathsf{TF}\Sigma_2$ classes, which are inspired by TFNP classes. These are analogs to the *coloured* TFNP classes introduced in [KST07, DR23]. In subsection 6.3 we explore the relationships between these and prominent $\mathsf{TF}\Sigma_2$ subclasses.

The *DNF resolution* proof systems are extensions of the resolution proof system (and restrictions of) to allow them to operate with DNF formulas, rather than only clauses. Davis and Robere [DR23] gave characterizations of these systems by coloured TFNP classes. We introduce several classes which characterize the Σ_2-variants of these proof systems; we believe these $\mathsf{TF}\Sigma_2$ classes *herbrandize* to the coloured classes.

Definition 6.1. A $\mathsf{Res}(\mathsf{polylog})$ *refutation* of a Π_2-unsatisfiable formula $F = \bigwedge_{i=1}^m A_i$ is a sequence of $\mathsf{polylog}(n)$-width DNF formulas $\Pi = (D_1, \ldots, D_s = \bot)$ where each D_i is deduced from previous DNFs by one of the following rules:

- *Axiom Introduction.* Introduce A_i for some $i \in [m]$.
- *Symmetric Cut.* From $D \vee t$ and $D \vee \bar{t}$ derive D, where t is any term.
- *Reverse Cut.* From D derive $D_i = D \vee t$ and $D_{i+1} = D \vee \bar{t}$, for some term t.

The *size* s of Π is the sum of the sizes of DNFs involved in Π, and the *width* w is the maximum width of any DNF in Π. The *complexity* of Π is $\log s + w$.

A $\mathsf{RevRes}(\mathsf{polylog})$ proof is a $\mathsf{Res}(\mathsf{polylog})$ proof in which every DNF in the sequence is used as the premise to a derivation rule at most once.

A $\mathsf{uCircRes}(\mathsf{polylog})$ proof has access to the additional rule

- *DNF Creation.* $S_i = S_{i-1} \cup \{D\}$, where D is any DNF formula.

provided that each copy of D that is created in this way is derived at least as many times as it is used as the premise to a derivation rule.

The following technical lemma will be key to our characterizations.

Lemma 6.2. *(Theorem 3.6 in [DR23])* $\mathsf{Res}(\mathsf{polylog})$, $\mathsf{RevRes}(\mathsf{polylog})$, *and* $\mathsf{uCircRes}(\mathsf{polylog})$ *are reduction closed.*

Davis and Robere proved Lemma 6.2 for DNF resolution proofs of $\Pi_{1.5}$-formulas (that is, when the axioms are clauses). It is straightforward to see that it holds by exactly the same argument (Claim 1) when the axioms are DNF formulas. In section 7 we prove this theorem for depth-$d.5$ Frege, for every d, of which $\mathsf{Res}(k)$ is $d = 1$.

In the following subsections we will prove Theorem 1.5, characterizing each of these proof systems by new $\mathsf{TF}\Sigma_2^{dt}$ subclasses. To define each of these classes, it will be convenient to use the following notion of a *meta-pointer*.

Definition 6.3. Given a function $S : [m] \times [t] \to [m]$, the *meta-pointer* $\tilde{S} : [m] \to [m] \cup \{undefined\}$ is defined as

$$\tilde{S}(u) = \begin{cases} v & \text{if for every } i \in [t],\ S(u,i) = v, \\ u & \text{if there is } i \in [t] \text{ such that } S(u,i) = u \\ undefined & \text{otherwise if there is } i, j \in [t] \text{ such that } u \neq S(u,i) \neq S(u,j). \end{cases}$$

Note that, if $u \neq v$, $\tilde{S}(u) = v$ is Π_1-verifiable: For all $i \in [t]$, we need to verify that $S(u,i) = v$, which takes $\log(m)$ queries . Moreover, $\tilde{S}(u) = u$ and $\tilde{S}(u) = undefined$ are Σ_1 verifiable: We can non-deterministically guess $i \in [t]$ such that $S(u,i) = u$, or $i \neq j \in [t]$ such that $u \neq S(u,i) \neq S(u,j)$; in other words, they are efficiently computable if we are given i (and j) as witnesses. The inclusion in $\mathsf{TF}\Sigma_2$ of the problems presented in this chapter follows directly from this fact.

6.1 DNF Resolution

$\mathsf{polylog}(n)$-width resolution was characterized by the TFNP^{dt} subclass PLS [BKT14]. In this section we introduce a $\mathsf{TF}\Sigma_2$-variant of the PLS-complete problem *iteration* and show that it characterizes Σ_2-$\mathsf{Res}(\mathsf{polylog})$. The *iteration* problem encodes the principle that *every DAG has a sink*. The input is given by a pointer function $S : [n] \to [n]$ giving the *successor* of a node $u \in [n]$, thought of as the next node on a root-to-leaf walk in the DAG. A solution is (i) an invalid source $S(1) = 1$, (ii) a u which points backwards $S(u) < u$, (iii) a sink: $u \in [n]$ such that $S(u) \neq u$ but $S(S(u)) = S(u)$, or (iv) a node u with an undefined pointer $S(u) = undefined$. Our $\mathsf{TF}\Sigma_2$ variant obfuscates the successor function. Similar ideas were used to define the RWPHP_2 problem in [KT21].

Definition 6.4. An instance of ITER_2 is given by a function $S : [m] \times [t] \to [m]$. A solution is a *witness* of a solution to the iteration instance defined by the meta-pointer $\tilde{S}$:

- (u, i, i') such that $S(u,i), S(u,i') \neq u$ and $S(u,i) \neq S(u,i')$, ($\tilde{S}(u)$ is undefined.)
- (u, i) such that $S(u,i) < u$. (A pointer which points backwards)
- $(1, i)$ if $S(1,i) = 1$. (1 is not a source)
- (u, v, i) such that $\tilde{S}(u) = v$ and $S(v,i) = v$. (v is a proper sink)

The class PLS_2^{dt} is the set of $R \in \mathsf{TF}\Sigma_2^{dt}$ such that $R \leq_{dt} \text{ITER}_2$.

Theorem 6.5. *For any* $\text{FF}_F \in \mathsf{TF}\Sigma_2^{dt}$*, there is a complexity-*c ITER_2*-formulation of* FF_F *iff there is a complexity* $O(c)$ Σ_2-$\mathsf{Res}(\mathsf{polylog})$ *proof of* F.

We prove this theorem in the following two lemmas, each giving one direction.

Lemma 6.6. *For* $\text{FF}_F \in \mathsf{TF}\Sigma_2$*, if* Σ_2-$\mathsf{Res}(\mathsf{polylog})(F) = c$*, then there is a complexity-*$O(c)$ ITER_2*-formulation of* FF_F.

Proof. Let (Π, H) be a Σ_2-$\mathsf{Res}(\mathsf{polylog})(F)$ proof of $F = \bigwedge_{i \in [\ell]} F_i$, where $H = \bigwedge_{i \in [k]} A_i$ and each A_i is a Σ_2-weakening of a DNF of F. Up to padding, we may assume that each DNF in the proof has the same number of terms t. Consider the proof $\Pi = D_1, \ldots, D_m$ in reverse order so that $D_1 = \bot$; this will be our designated source.

Let $t_{u,i}$ be the i^{th} term of D_u. Given an assignment $\alpha \in \{0,1\}^n$ to the variables of F, we construct a function $S_\alpha : [m] \times [t] \to [m]$ by setting $S_\alpha(u,i)$ to be:

- u if D_u is an axiom, or if $t_{u,i}(\alpha) = 1$;
- v if $t_{u,i}(\alpha) = 0$ and D_u was derived from D_v by the reverse cut rule or semantic weakening of an axiom;
- v if $t_{u,i}(\alpha) = 0$ and D_u was derived from $D_v = D_u \vee t$ and $D_w = D_u \vee \bar{t}$ via symmetric cut and $t(\alpha) = 0$ and w if $\bar{t}(\alpha) = 0$;

Finally, for each solution o to the instance S_α, we define the output of the reduction $g_o(\alpha)$ to be arbitrary if o does not correspond to an axiom A_i of H, and otherwise this axiom A_i is a weakening of a DNF F_j of F, and we set $g_o(\alpha) = j$. Note that in this case $A_i(\alpha) = 0 \implies F_j(\alpha) = 0$. Observe that computing $S_\alpha(u, i)$ involves evaluating at most two terms, and hence the depth of the reduction is at most twice the width of the proof. It remains to argue that the reduction is correct.

Claim. The function $\tilde{S}_\alpha$ satisfies the following properties:

i) $\tilde{S}_\alpha$ is defined everywhere.
ii) If D_u is not an axiom of H, then $D_u(\alpha) = 0$ iff $\tilde{S}_\alpha(u) \neq u$.
iii) If $\tilde{S}_\alpha(u) = v \neq u$, then $D_v(\alpha) = 0$.

Assuming the claim, we see that the only type of solution to this ITER_2 instance S_α are proper sinks corresponding to falsified axioms of H, which are weakenings of (falsified) axioms of F. Hence, g returns a correct solution to $\text{FF}_F(\alpha)$.

Proof of Claim. We prove each item, beginning with (i). Clearly $\tilde{S}_\alpha$ is well defined for any u that was not derived using the cut rule since $S_\alpha(u, i)$ only has one choice of value other than u. So now consider u such that D_u was derived from $D_v = D_u \vee t$ and $D_w = D_u \vee \bar{t}$. For $i \in [t]$, we see that $S_\alpha(u, i)$ depends on two values: $t_{u,i}(\alpha)$, and $t(\alpha)$ in the case where $t_{u,i}(\alpha) = 0$. Thus, $t(\alpha)$ being independent of i, $S_\alpha(u, i)$ is always identical when not equal to u.

(ii) follows from the fact that $D_u(\alpha) = 0$ iff $t_{u,i}(\alpha) = 0$ for all i, and $\tilde{S}_\alpha(u) = u$ iff $t_{u,i}(\alpha) = 1$ for at least one i. Finally, (iii) follows by definition.

□

We will now prove the converse. First, we describe the encoding of ITER_2 as an unsatisfiable formula. For each $(u, i) \in [m] \times [t]$, the m-ary value of $S_{u,i}$ will be described by $\log m$-many boolean variables $S_{u,i,b}$, where the indicator function

$$[\![S_{u,i} = v]\!] := \bigwedge_{b \in [\log m]} S_{u,i,b}^{v_b},$$

where we think of v as being written in its binary encoding, v_b is its b^{th} bit, and $S^1_{u,i,b} = S_{u,i,b}$ and $S^0_{u,i,b} = \neg S_{u,i,b}$. As well, $[\![S_{u,i} \neq v]\!] = \neg [\![S_{u,i} = v]\!]$, and

$$[\![\tilde{S}_u \neq v]\!] := \bigvee_{i \in [t]} [\![S_{u,i} \neq v]\!].$$

Then ITER_2 is the conjunction of the following subformulas:

- $[\![S_{1,i} \neq 1]\!]$ for each $i \in [n]$. (1 is not a source)
- $[\![S_{u,i} \neq v]\!] \vee [\![S_{u,i'} \neq v']\!]$ for all $v \neq v'$ and $i \neq i'$ such that $u \neq v, v'$ ($\tilde{S}$ is defined everywhere)
- $[\![S_{u,i} \neq v]\!]$ for all $v < u$ and $i \in [n]$. (Nothing points backwards)
- $[\![\tilde{S}_u \neq v]\!] \vee [\![S_{v,j} \neq v]\!]$ for all $u < v$ and $j \in [n]$. (v is not a proper sink)

Note that the subformulas of the ITER_2 formula are clauses making the formula a CNF. We may then question what makes ITER_2 a $\mathsf{TF}\Sigma_2^{dt}$ problem and not a TFNP^{dt} one. The key to understanding this resides in the size of said clauses. Indeed, for a false formula problem corresponding to a CNF to be in TFNP^{dt}, we need to be able to verify if a given clause is falsified by an assignment by only querying a $\mathsf{polylog}(n)$ amount of bits. This in turn directly implies that we would need each clause to be of $\mathsf{polylog}(n)$-width. This is not the case here because of the fourth type of axioms, which are of poly(n)-width. On the other hand, considering clauses as 1-width DNFs, we see that this false formula problem corresponding to this formula lands indeed in $\mathsf{TF}\Sigma_2^{dt}$. We now state the converse.

Lemma 6.7. *For* $\text{FF}_F \in \mathsf{TF}\Sigma_2$*, if there is a complexity-*c ITER_2*-formulation of* FF_F*, then there is a complexity-*$O(c)$ $\mathsf{Res}(\mathsf{polylog})$ *proof of* F*.*

Observe that the set of formulas $\{[\![S_{u,i} \neq v]\!]\}_{v\in[m]}$ contains all clauses containing all of the variables $S_{u,i,b}$. Hence they can be cut in $O(m \log m)$-many steps to obtain $\bot$. Throughout the proof we will write the

$$\frac{D_v \vee [\![S_{u,i} \neq v]\!],\ \forall v}{D}$$

as a shorthand for this derivation with $D = \bigvee_{v\in[m]} D_v$.

Proof of Lemma 6.7. By Lemma 6.2 it suffices to show that $\mathsf{Res}(\mathsf{polylog})$ can prove ITER_2. By induction from $u = m$ to $u = 1$, we will derive a set of formulas that state that does not point forward in $\tilde{S}$. Combining this with the fact that the image of u by $\tilde{S}$ cannot be undefined and u may not point backwards, this is semantically equivalent to stating that u points to itself. We then reach a contradiction when reaching $u = 1$, since 1 must be a proper source of our graph. This will be achieved by deducing

$$L_u \ := \ \{[\![\tilde{S}_u \neq v]\!] : u < v\},$$

which can be combined with axioms stating that no node points backwards for the desired statement.

The base case is trivial, as $L_m = \emptyset$. Consider some $u \in [m]$ and suppose that we have derived L_v for all $v > u$. We derive the formula $[\![\tilde{S}_u \neq v]\!] \in L_u$ as follows: Consider some $w > v > u$ and apply the reverse cut rule to $[\![\tilde{S}_v \neq w]\!]$ in order to obtain $[\![\tilde{S}_v \neq w]\!] \vee [\![\tilde{S}_u \neq v]\!]$. Now consider the cuts from $a = t$ to $a = 2$,

$$\frac{[\![\tilde{S}_u \neq w]\!] \vee \bigvee_{i<a+1}[\![S_{v,i} \neq v]\!] \qquad [\![\tilde{S}_u \neq v]\!] \vee [\![S_{v,a} \neq v]\!] \qquad [\![S_{v,1} \neq w]\!] \vee [\![S_{v,a} \neq w']\!], \forall w' \neq v, w}{[\![\tilde{S}_u \neq v]\!] \vee \bigvee_{i<a}[\![S_{v,i} \neq v]\!]}$$

to the set of formulas $[\![\tilde{S}_u \neq v]\!] \vee [\![S_{v,1} \neq w]\!]$. Finally, we do one last cut:

$$\frac{[\![\tilde{S}_u \neq v]\!] \vee [\![S_{v,1} \neq w]\!],\ \forall w > v \qquad [\![\tilde{S}_u \neq v]\!] \vee [\![S_{v,1} \neq v]\!] \qquad [\![S_{v,1} \neq w]\!],\ \forall w < v}{[\![\tilde{S}_u \neq v]\!]},$$

which derives the formula $[\![\tilde{S}_u \neq v]\!] \in L_u$.

Finally, once we have derived L_1, we can derive $\bot$ as follows. For a fixed $v > 1$, starting from $a = t$ down to $a = 2$, we operate the cuts:

$$\frac{\bigvee_{i<a+1}[\![S_{1,i} \neq v]\!] \qquad [\![S_{1,a} \neq 1]\!] \qquad [\![S_{1,1} \neq v]\!] \vee [\![S_{1,n} \neq v']\!],\ \forall v' \neq v, 1}{\bigvee_{i<a}[\![S_{1,i} \neq v]\!]}.$$

Once we have derived $[\![S_{1,1} \neq v]\!]$, we do one final cut:

$$\frac{[\![S_{1,1} \neq v]\!],\ \forall v \neq 1 \qquad [\![S_{1,1} \neq 1]\!]}{\bot}$$

□

6.2 Circular and Reversible DNF Resolution

In this section we characterize the Σ_2-$\mathsf{uCircRes}(\mathsf{polylog})$ proof system by a $\mathsf{TF}\Sigma_2$-variant of the *Sink-of-Line* problem. An instance of Sink-of-Line is given by functions $S, P : [m] \times [t] \to [m] \cup \{undefined\}$ which define a graph G as follows: there is a directed edge (u, v) if $\tilde{S}(u) = v$ and $\tilde{P}(v) = u$. A solution to this instance is either i) 1 if 1 is not a source in G, ii) a sink u in G, iii) a vertex u for which $\tilde{P}(u)$ or $\tilde{S}(u)$ is undefined. We now describe the $\mathsf{TF}\Sigma_2$ variant.

Definition 6.8. An instance of SOL_2 is given by functions $S, P : [m] \times [t] \to [m]$. A solution is a witness to a solution to the SOL instance defined by the meta-pointers $(\tilde{S}, \tilde{P})$:

- (u, i, i') if $S(u, i), S(u, i') \neq u$ and $S(u, i) \neq S(u, i')$;
  or $P(u, i), P(u, i') \neq u$ and $P(u, i) \neq P(u, i')$. (Predecessor or Successor of u is undefined)
- $(1, i)$ if $S(1, i) = 1$ or $\tilde{S}(1) = v \neq 1$ and $P(v, i) \neq 1$. (1 is not a source)
- (u, i) if $u \neq 1$ and $S(u, i) = 1$. (u has a pointer to 1)
- (u, v, i) for $u \neq v$ if $\tilde{S}(u) = v$, $\tilde{P}(v) = u$ and $S(v, i) = v$;
  or $\tilde{S}(u) = v$, $\tilde{P}(v) = u$, $\tilde{S}(v) = w$ and $P(w, i) \neq v$. (v is a proper sink)

Theorem 6.9. *For any* $\mathsf{FF}_F \in \mathsf{TF}\Sigma_2$*, there is a complexity-*c SOL_2*-formulation of* FF_F *iff there is a complexity* $O(c)$ Σ_2-$\mathsf{uCircRes}(\mathsf{polylog})$ *proof of* F*.*

This theorem follows by combining Lemma 6.10 and Lemma 6.14. We begin with the backwards direction, showing that $\mathsf{uCircRes}(\mathsf{polylog})$ can prove SOL_2 formulations. SOL_2 is encoded as an unsatisfiable formula which is the conjunction of the following

- $[\![S_{u,i} \neq 1]\!]$ for $u \in [m], i \in [t]$, and $[\![\tilde{S}_1 \neq v]\!] \vee [\![P_{u,i} \neq v]\!]$ for all $u, v \neq 1, i \in [t]$. (1 is a source)
- $[\![S_{u,i} \neq v]\!] \vee [\![S_{u,i'} \neq v']\!]$ for all $i \neq i'$, $v \neq v'$. ($\tilde{S}$ is not undefined)
- $[\![P_{u,i} \neq v]\!] \vee [\![P_{u,i'} \neq v']\!]$ for all $i \neq i'$, $v \neq v'$. ($\tilde{P}$ is not undefined)
- $[\![S_{u,i} \neq 1]\!]$ for all $i \in [t]$ and $u \neq 1$. (Nothing points to 1)
- Let $\bar{E}_{u,v} := [\![\tilde{S}_u \neq v]\!] \vee [\![\tilde{P}_v \neq u]\!]$, we include (No proper sinks)
  - $\bar{E}_{u,v} \vee [\![S_{v,i} \neq v]\!]$ for each $u \neq v$ and $i \in [m]$, and
  - $\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee [\![P_{w,k} \neq w']\!]$ for $u \neq v \neq w \neq w'$ and $k \in [t]$.

Lemma 6.10. *For* $\mathsf{FF}_F \in \mathsf{TF}\Sigma_2$*, if there is a* SOL_2*-formulation of* FF_F *of complexity* c *then there is a complexity* $O(c)$ Σ_2-$\mathsf{uCircRes}(\mathsf{polylog})(F)$ *proof of* F*.*

Proof. By Lemma 6.2, it suffices to show that $\mathsf{uCircRes}(\mathsf{polylog})$ can prove SOPL_2. For each $u \in [m]$, we would like to derive the set of formulas

$$L_u = \{\bar{E}_{u,v} : v \neq u, 1\},$$

stating that u has no outgoing edges. Our proof will proceed by the following three steps:

1. Assume L_u for each $u \neq 1$;
2. From L_v for $v \neq u$, deduce L_u. Since L_v is semantically equivalent to saying that node v points to itself, if u were to point to any other node, then said node would be a proper sink. Hence L_u follows.
3. L_1 is in direct contradiction with axioms stating that 1 is a source.

For step 1, we use the *DNF creation* rule,

$$\frac{}{\bar{E}_{u,v}}$$

For step 2 and $u \in [m]$, we perform the following. For $w \neq v \neq u$ with $w, v \neq 1$, consider $\bar{E}_{v,w} \in L_v$ and weaken it successively to get

$$\frac{}{\bar{E}_{u,v} \vee \bar{E}_{v,w}},$$

then we cut as follows: starting with $c = n$ down to $c = 1$,

$$\frac{\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee \bigvee_{k<c+1} [\![P_{w,k} \neq v]\!] \qquad \bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee [\![P_{w,c} \neq w']\!],\ \forall w' \neq w}{\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee \bigvee_{k<c} [\![P_{w,k} \neq v]\!]}$$

to get $\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!]$. Next, starting from $b = n$ down to $b = 2$,

$$\frac{\bar{E}_{u,v} \vee \bigvee_{j<b+1} [\![S_{v,j} \neq w]\!] \qquad \bar{E}_{u,v} \vee [\![S_{v,b} \neq v]\!] \qquad [\![S_{v,1} \neq w]\!] \vee [\![S_{v,b} \neq w']\!],\ \forall w' \neq v, w}{\bar{E}_{u,v} \vee \bigvee_{j<b} [\![S_{v,j} \neq w]\!]}$$

and end up with the formulas $\bar{E}_{u,v} \vee [\![S_{v,1} \neq w]\!]$. Finally,

$$\frac{\bar{E}_{u,v} \vee [\![S_{v,1} \neq w]\!],\ \forall w \neq v, 1 \qquad \bar{E}_{u,v} \vee [\![S_{v,1} \neq v]\!] \qquad [\![S_{v,1} \neq 1]\!]}{\bar{E}_{u,v}}$$

derives $\tilde{E}_{u,v} \in L_u$. Having derived L_1 allows us to take $\bar{E}_{1,v} \in L_1$ and, starting with $b = n$ down to $b = 1$, we may cut:

$$\frac{[\![\tilde{S}_1 \neq v]\!] \vee \bigvee_{j<b+1} [\![P_{v,j} \neq 1]\!] \qquad [\![\tilde{S}_1 \neq v]\!] \vee [\![P_{v,b} \neq w]\!],\ \forall w \neq 1}{[\![\tilde{S}_1 \neq v]\!] \vee \bigvee_{j<b} [\![P_{v,j} \neq 1]\!]}$$

to get $[\![\tilde{S}_1 \neq v]\!]$ for each $v \neq 1$. Next, starting from $a = n$ down to $a = 2$, we cut:

$$\frac{\bigvee_{i<a+1}[\![S_{1,i} \neq v]\!] \qquad [\![S_{1,a} \neq 1]\!] \qquad [\![S_{1,1} \neq v]\!] \vee [\![S_{1,a} \neq v']\!],\ \forall v' \neq v}{\bigvee_{i<a}[\![S_{1,a} \neq v]\!]}$$

to get $[\![S_{1,1} \neq v]\!]$ for $v \neq 1$. We may then cut one final time,

$$\frac{[\![S_{1,1} \neq v]\!],\ \forall v \neq 1 \qquad [\![S_{1,1} \neq 1]\!]}{\bot}. \qquad \square$$

We delay the proof of the other direction until the end of this section and complete it together with the proof of the same direction RevRes(polylog) as they are similar.

We characterize the RevRes(polylog) by a $\mathsf{TF}\Sigma_2$ variant of the *Sink-of-Potential-Line* (SOPL) problem. This is a *metered* variant of SOL, meaning that edges must always point towards larger numbers. An instance of SOPL is given by functions $S, P : [m] \to [m] \cup \{undefined\}$ that define a graph G with edges (u, v) iff $S(u) = v$ and $P(v) = u$. A solution is either i) 1 if 1 is not a source in G, ii) a sink u in G, iii) a vertex which points backwards $S(u) < u$, or iv) a vertex u if $S(u)$ or $P(u)$ is undefined.

Definition 6.11. An instance of SOPL_2 is given by functions $S, P : [m] \times [t] \to [m]$. A solution is a witness to a solution to the SOPL instance defined by the meta-pointers $(\tilde{S}, \tilde{P})$:

- (u, i, i') if $S(u, i), S(u, i') \neq u$ and $S(u, i) \neq S(u, i')$;
  or $P(u, i), P(u, i') \neq u$ and $P(u, i) \neq P(u, i')$.
  (Predecessor or Successor of u is undefined)
- $(1, i)$ if $S(1, i) = 1$ or $\tilde{S}(1) = v \neq 1$ and $P(v, i) \neq 1$. (1 is not a source)
- (u, i) if $S(u, i) < u$. (u points backwards)
- (u, v, i) for $u < v$ if $\tilde{S}(u) = v$, $\tilde{P}(v) = u$ and $S(v, i) = v$; or $\tilde{S}(u) = v$, $\tilde{P}(v) = u$, $\tilde{S}(v) = w$ and $P(w, i) \neq v$.
  (v is a proper sink)

Theorem 6.12. *For any* $\text{FF}_F \in \mathsf{TF}\Sigma_2$*, there is a complexity-c* SOPL_2*-formulation of* FF_F *iff there is a complexity* $O(c)$ Σ_2-RevRes(polylog) *proof of* F.

This theorem follows by combining Lemma 6.13 and Lemma 6.14. We begin with the backwards direction, showing that RevRes(polylog) can prove SOPL_2 formulations. SOPL_2 is encoded as an unsatisfiable formula, which is the conjunction of the following:

- $[\![S_{u,i} \neq 1]\!]$ for $u \in [m], i \in [t]$, and $[\![\tilde{S}_1 \neq v]\!] \vee [\![P_{u,i} \neq v]\!]$ for all $u, v \neq 1, i \in [t]$. (1 is a source)
- $[\![S_{u,i} \neq v]\!] \vee [\![S_{u,i'} \neq v']\!]$ for all $i \neq i'$, $v \neq v'$. ($\tilde{S}$ is not undefined)
- $[\![P_{u,i} \neq v]\!] \vee [\![P_{u,i'} \neq v']\!]$ for all $i \neq i'$, $v \neq v'$. ($\tilde{P}$ is not undefined)
- $[\![S_{u,i} \neq v]\!]$ for all $i \in [t]$ and $v < u$. (No backwards edges)
- Let $\bar{E}_{u,v} := [\![\tilde{S}_u \neq v]\!] \vee [\![\tilde{P}_v \neq u]\!]$, we include (No proper sinks)
  i) $\bar{E}_{u,v} \vee [\![S_{v,i} \neq v]\!]$ for each $u < v$ and $j \in [m]$, and
  ii) $\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee [\![P_{w,k} \neq w']\!]$ for $u < v < w$ and $w \neq w'$ and $k \in [t]$.

Lemma 6.13. *For* $\text{FF}_F \in \mathsf{TF}\Sigma_2$*, if there is a* SOPL_2*-formulation of* FF_F *of complexity c, then there is a complexity* $O(c)$ Σ_2-RevRes(polylog)(F) *proof of* F.

Proof. By Lemma 6.2, it suffices to show that RevRes(polylog) can prove SOPL_2. We will prove by induction on $u = m \dots 1$ that u does not have any outgoing edges. That is, we will derive the set of formulas:

$$L_u := \{\bar{E}_{u,v} : u > v\}.$$

First observe that the base case is given by the *no backwards edges* axioms. Assuming that we can derive L_1, we show how to complete the proof. For $v > 1$, starting with $b = n$ down to $b = 1$, we cut

$$\frac{[\![\tilde{S}_1 \neq v]\!] \vee \bigvee_{j<b+1}[\![P_{v,j} \neq 1]\!] \qquad [\![\tilde{S}_1 \neq v]\!] \vee [\![P_{v,b} \neq w]\!],\ \forall w \neq 1}{[\![\tilde{S}_1 \neq v]\!] \vee \bigvee_{j<b}[\![P_{v,j} \neq 1]\!]}.$$

Next, starting from $a = n$ down to $a = 2$, we successively cut

$$\frac{\bigvee_{i<a+1}[\![S_{1,i} \neq v]\!] \qquad [\![S_{1,a} \neq 1]\!] \qquad [\![S_{1,1} \neq v]\!] \vee [\![S_{1,a} \neq w]\!],\ \forall w \neq 1, v}{\bigvee_{i<a}[\![S_{1,i} \neq v]\!]}.$$

Once all those formulas are derived, we cut one final time to finish the proof

$$\frac{[\![S_{1,1} \neq 1]\!] \qquad [\![S_{1,1} \neq v]\!],\ \forall v > 1}{\bot}.$$

We now describe how to derive L_u from all L_v with $v > u$. For a given v and $\bar{E}_{v,w} \in L_v$, we start by weakening it to get $[\![\tilde{P}_v \neq u]\!] \vee \bar{E}_{v,w}$ and again to get $\bar{E}_{u,v} \vee \bar{E}_{v,w}$. Once this is done, starting at $c = n$ down to $k = 1$, we cut

$$\frac{\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee \bigvee_{k<c+1}[\![P_{w,k} \neq v]\!] \qquad \bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee [\![P_{v,c} \neq w']\!],\ \forall w' \neq w}{\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!] \vee \bigvee_{k<c}[\![P_{w,k} \neq v]\!]}$$

to get $\bar{E}_{u,v} \vee [\![\tilde{S}_v \neq w]\!]$. Finally, from $b = n$ down to $b = 2$, we cut

$$\frac{\bar{E}_{u,v} \vee \bigvee_{j<b+1}[\![S_{v,j} \neq w]\!] \qquad \bar{E}_{u,v} \vee [\![S_{v,b} \neq v]\!] \qquad [\![S_{v,1} \neq w]\!] \vee [\![S_{v,c} \neq w]\!],\ \forall w' \neq v, w}{\bar{E}_{u,v} \vee \bigvee_{j<b}[\![S_{v,j} \neq w]\!]}.$$

Once we derived $\bar{E}_{u,v} \vee [\![S_{v,1} \neq w]\!]$ for each $w > v$, we have one final cut

$$\frac{\bar{E}_{u,v} \vee [\![S_{v,1} \neq w]\!],\ \forall w > v \qquad \bar{E}_{u,v} \vee [\![S_{v,1} \neq v]\!] \qquad [\![S_{v,1} \neq w]\!],\ \forall w < v}{\bar{E}_{u,v}}$$

to get $\bar{E}_{u,v} \in L_u$. □

Finally, we prove the other direction of Theorem 6.12 and Theorem 6.9.

Lemma 6.14. *Let* $\mathsf{FF}_F \in \mathsf{TF}\Sigma_2$. *Suppose that* F *admits a complexity-*c Σ_2-$\mathsf{uCircRes}(\mathsf{polylog})$ *(*Σ_2-$\mathsf{RevRes}(\mathsf{polylog})$*) proof. Then there is a complexity-*$O(c)$ SoL_2*-(*SoPL_2*-)formulation of* FF_F.

Proof. We first handle Circular DNF Resolution and discuss what needs to be changed in order to handle Reversible DNF Resolution at the end of the proof. The idea for the transformation of a $\mathsf{uCircRes}(\mathsf{polylog})$ proof into an SoL_2 formulation is the same as the transformation of a $\mathsf{Res}(\mathsf{polylog})$ into an Iter_2 formulation (Lemma 6.6) with the addition of defining a predecessor function. Let $\Pi = (D_1, \ldots, D_m)$ be such a proof. By padding, we may assume that each DNF in the proof has the same number of terms. Let us consider the proof in reverse order such that $D_1 = \bigvee_{i\in[t]} \bot$.

Let $t_{u,i}$ be the i^{th} term of D_u. Given an assignment $\alpha \in \{0,1\}^n$ to the variables of F, we construct a function $S_\alpha : [m] \times [t] \to [m]$ by setting $S_\alpha(u,i)$ to be:

- u if D_u is an axiom, or if $t_{u,i}(\alpha) = 1$;
- v if $t_{u,i}(\alpha) = 0$ and D_u was derived from D_v by the reverse cut rule or semantic weakening of an axiom;
- v if $t_{u,i}(\alpha) = 0$ and D_u was derived from $D_v = D_u \vee t$ and $D_w = D_u \vee \bar{t}$ via symmetric cut and $t(\alpha) = 0$ and w if $\bar{t}(\alpha) = 0$;

As well, define the predecessor function $P_\alpha : [m] \times [t] \to [m]$, as $P_\alpha(u,i)$:

- u if either $u = 1$, or the formula D_u was deduced but never used as the premise of a rule, or if $t_{u,i}(\alpha) = 1$;
- v if $t_{u,i}(\alpha) = 0$ and u is used as a premise to derive D_v via any of the rules but the reverse cut;
- v or w if $t_{u,i}(\alpha) = 0$ and D_u was used as the premise of the reverse cut rule to derive $D_v = D_u \vee t$ and $D_w = D_u \vee \bar{t}$. If $t(\alpha) = 0$, then $P_\alpha(u,i) = v$ and $P_\alpha(u,i) = w$ otherwise.

Finally, for each solution o to the instance S_α, we define the output of the reduction $g_o(\alpha)$ to be arbitrary if o does not correspond to an axiom A_i of H, and otherwise this axiom A_i is a weakening of a DNF F_j of F, and we set $g_o(\alpha) = j$. Note that in this case $A_i(\alpha) = 0 \implies F_j(\alpha) = 0$. Observe that computing $S_\alpha(u,i)$ and $P_\alpha(u,i)$ involves evaluating at most two terms, and hence the reduction is efficient.

It remains to argue that the reduction is correct.

Claim. The following hold:

i) $\tilde{P}_\alpha$ and $\tilde{S}_\alpha$ are defined everywhere;
ii) If D_u was used as the premise of a rule, $D_u(\alpha) = 0$ if and only if $\tilde{P}_\alpha(u) \neq u$ and $\tilde{S}_\alpha(u) \neq u$;
iii) If $\tilde{P}_\alpha(u) = v \neq u$, then $D_v(\alpha) = 0$;
iv) For a pair $u \neq v$, $\tilde{S}_\alpha(u) = v$ if and only if $\tilde{P}_\alpha(v) = u$.

Assuming the claim, the only solutions are proper sinks corresponding to falsified axioms of H, which are weakenings of (falsified) axioms of F. Hence, g returns a correct solution to $\mathrm{FF}_F(\alpha)$.

Proof of Claim. The proof of this claim is, at heart, the same as the proof of the claim in Lemma 6.6. The behavior of both functions implies that the only solutions one might get in the instance are proper sinks and that these proper sinks can only be falsified axioms. Finally, when Π is a $\mathsf{RevRes}(\mathsf{polylog})$ proof, $S_\alpha(u,i) \geq u$ and $P_\alpha(v,j) \leq v$ for any u and v since the graph representation of Π does not include cycles, and thus we would not have *fake solutions* corresponding to edges pointing backwards, making our formulation a valid SoPL$_2$-formulation. ☐

6.3 Relationships in $\mathsf{TF}\Sigma_2$

In this subsection, we use the characterizations that we have constructed in order to prove all of the new inclusions in Figure 1. We begin by giving a $\mathsf{uCircRes}$ proof of the unmetered source-of-DAG problem, which is equivalent to StrongAvoid. USoD is encoded propositionally by the conjunction of the following formulas:

– $[\![S_1 = 1]\!]$ and $\bigvee_{t\in[n]}[\![S_t = 1]\!]$. (1 is a sink);
– $\bigvee_{t\in[n]}[\![S_t = u]\!]$ for all $u \neq 1$. (u is not a source).

Proposition 6.15. USoD *has a* $\mathsf{polylog}(n)$*-complexity* $\mathsf{uCircRes}(\mathsf{polylog}(n))$*-proof and so* StrongAvoid $\in \mathsf{SOL}_2$.

Proof. The strategy for the proof is:

i) Assume that $S(u) = u$ for any $u \neq 1$;
ii) From the fact that $S(v) = v$ for all $v \neq u$, deduce that $S(u) = u$. Indeed if all other nodes point to themselves, u can not point to anything but itself since otherwise it would qualify as a source. We also derive $S(u) \neq 1$ during this process;
iii) Once this is done, we will be left with the fact that $S(u) \neq 1$ for each $u \neq 1$ which is in direct contradiction with the second axiom.

For step (i), we introduce $[\![S_u = u]\!]$ for each $u \neq 1$ via the DNF creation rule. Now, fixing some such u, for $t \neq u$, we weaken $[\![S_t = t]\!]$ to $[\![S_t = t]\!] \vee [\![S_u \neq w]\!]$ for each $w \in [n]$ and consider the case $w = t$. Since $u \neq t$, the formula $[\![S_t \neq t]\!] \vee [\![S_u \neq t]\!]$ is a tautology, and therefore we can introduce it. For each $t \neq u$ we cut

$$\frac{[\![S_t = t]\!] \vee [\![S_t \neq u]\!] \qquad [\![S_t \neq t]\!] \vee [\![S_t \neq u]\!]}{[\![S_t \neq u]\!]}.$$

Then, cutting

$$\frac{\bigvee_t [\![S_t = u]\!] \qquad [\![S_t \neq u]\!],\ \forall t \neq u}{[\![S_u = u]\!]}$$

From $[\![S_u = u]\!]$ we can deduce $[\![S_u \neq 1]\!]$, completing step (ii).

Finally, we can perform step (iii) by cutting

$$\frac{\bigvee_{u\neq 1}[\![S_u = 1]\!] \qquad [\![S_u \neq 1]\!],\ \forall u \neq 1}{\bot}$$

The size of the proof and the characterization theorem shows that USoD $\in \mathsf{SOL}_2$. Also, the equivalence USoD $=_{dt}$ StrongAvoid gives us StrongAvoid $\in \mathsf{SOL}_2$ ☐

The *sink-of-DAG* problem is the canonical PLS-complete problem in which one is given a source of a DAG and one wants to find a sink. Our characterization of unary Sherali-Adams by StrongAvoid proceeded via the equivalent *unmetered source-of-DAG* problem. Hence, it is natural to also consider a metered version of these problem, where one is given a sink of a DAG and one wants to find a source.

Definition 6.16. The *Source of DAG* (SoD) problem is defined as follows. The input is a "successor" function $S : [n] \to [n]$ which defines a graph in which each vertex has fan-out ≤ 1 but arbitrary fan-in. There is an edge from i to j if $S(i) = j$. A solution to the instance S is:

i) i if $S(i) < i$; (i has a backward edge)
ii) n if for all $i < n$, $S(i) \neq n$; (n is not a sink)
iii) i if for all $j \in [n]$, $S(j) \neq i$. (A source)

We can encode SoD propositionally as the conjunction of the following formulas:

i) $\bigvee_{t \neq n} [\![S_t = n]\!]$; ($n$ is a proper sink)
ii) $\bigvee_t [\![S_t = u]\!]$ for each $u \neq 1$; (no sources)
iii) $[\![S_u \neq v]\!]$ for any pair of nodes $v < u$. (no edges pointing backwards)

Proposition 6.17. *There is a* $\mathsf{polylog}(n)$*-complexity* $\mathsf{RevRes}(\mathsf{polylog}(n))$ *proof of* SoD*, and hence* SoD $\in \mathsf{SOPL}_2$.

Proof. The strategy of the proof is as follows:

i) Given that $S(t) = t$ for each $t < u$, deduce that $S(u) = u$. This must be true since otherwise u is a source.
ii) Use the fact that the derived formulas directly contradicts the first axiom.

For step (i), by induction assume that we have derived $[\![S_t = t]\!]$ for each $t < u$. Weaken these formulas to get $[\![S_t = t]\!] \vee [\![S_t \neq v]\!]$ and consider the case when $v = u$. Since $t \neq u$, the formula $[\![S_t \neq t]\!] \vee [\![S_t \neq u]\!]$ is a tautology that we introduce, and we cut

$$\frac{[\![S_t \neq t]\!] \vee [\![S_t \neq u]\!] \qquad [\![S_t = t]\!] \vee [\![S_t \neq u]\!]}{[\![S_t \neq u]\!]}$$

to obtain $[\![S_t \neq u]\!]$. Next, we cut

$$\frac{\bigvee_t [\![S_t = u]\!] \qquad [\![S_t \neq u]\!],\ \forall t < u \qquad [\![S_t \neq u]\!], \forall t > u}{[\![S_u = u]\!]}$$

to derive $[\![S_u = u]\!]$. Fianlly, cut

$$\frac{\bigvee_{t \neq n} [\![S_t = n]\!] \qquad [\![S_t \neq n]\!],\ \forall t \neq n}{\perp}$$

hence SoD $\in \mathsf{SOPL}_2$. ☐

Observe that these proofs indicate that up to complexifying a function, it is possible to build an inverse that is also hard to compute with an efficient reduction. Also, since we know how to transform $\mathsf{uCircRes}(\mathsf{polylog})$ refutations (resp. $\mathsf{RevRes}(\mathsf{polylog})$ refutations) into SOL_2-instances (resp. SOPL_2-instances), following the instructions lets us concretely build those inverses.

We end this section by proving several inclusions which do not rely on our characterizations.

Proposition 6.18. LOP $\leq_{dt}$ ITER$_2$.

Proof. Let $\prec$ be an LOP instance on $[n]$. By encoding it with $\binom{n}{2}$ variables such that, for $i < j \in [n]$, $x_{i,j} = 1$ means $i \prec j$, and $x_{i,j} = 0$ means $j \prec i$, we can force the purported order to always be total. An output to the LOP instance would thus either be a $\prec$-minimal element or a proof that $\prec$ is not an order, i.e., that the transitivity does not hold. Consider the ITER$_2$ instance on $\binom{n}{2} + n$ meta-nodes with a meta-node for each $(i, j) \in [n]^2$ with $i \geq j$. Let $(1, 1)$ be the source. It helps to think of the meta-nodes as arranged in n levels, with the first element in the label being the level a meta-node is at.

The idea is that (i, j) is valid (i.e., has an outgoing edge) iff $\prec$ is transitive and j is the $\prec$-minimal value in $[i]$. If $i < n$, it will point to $(i + 1, j')$, where $j' = j$ if j is still $\prec$-minimal in $[i + 1]$, and $j' = i + 1$ otherwise. We now formally define the nodes with index (i, j). If $i = n$, then it contains a single node that points to itself. Otherwise, there are two kinds of nodes:

Transitivity nodes. $\binom{n}{3}$-many nodes verifying the transitivity of $\prec$. Each of those nodes is associated with 3 distinct elements $(a, b, c) \in [n]^3$. We define $S((i, j), (a, b, c))$ as follows:

– Query $a \prec b$, $b \prec c$ and $a \prec c$. If the answers show that $\prec$ is not transitive on (a, b, c), point to (i, j).
– Query $j \prec i + 1$. If it holds, point to $(i + 1, j)$. Otherwise, point to $(i + 1, i + 1)$.

Validity nodes. $(i-1)$-many nodes verifying the validity of (i, j). Each of those nodes is associated with a value $k \in [i] \setminus \{j\}$. We define $S((i, j), k)$ as follows:

– Query $j \prec k$. If it does not hold, point to (i, j).
– Query $j \prec i + 1$. If it holds, point to $(i + 1, j)$; otherwise, point to $(i + 1, i + 1)$.

Since every node that points out of its index does the same query to decide where to point, the meta successor is well-defined. If $\prec$ is not transitive, every meta-node will point to itself. The solution can thus only be of type $((1, 1), i)$, with this node being of the transitive type. This immediately gives us a triple in $[n]$, proving $\prec$ is not transitive. If $\prec$ is indeed a total order, then it is clear that every level has a single active node; the only proper sink on level n indicates the $\prec$-minimal value in $[n]$. □

Proposition 6.19. SOD $\leq_{dt}$ LOP.

Proof. Let S be an SOD instance on n vertices. Consider an LOP instance $\prec$ on $2n$ values split into two groups $C = [n]$ and $L = [n]$. We denote elements of C by i_C and elements of L by i_L, for $i \in [n]$. The group C's goal is to "check for backward pointers"; if the $\prec$-minimal element is i_C, then i points backwards. The group L checks for loops: if the $\prec$-minimal element is i_L, then there are no backward edges. Moreover, if $i = n$, then n is not a proper sink. Otherwise, i is the first node (in regular order) to not point to itself in S, i.e., i is a source.
Formally, for $i, j \in [n]$, we define $\prec$ as follows:

– $i_C \prec j_C$ iff $i < j$;
– $i_C \prec j_L$ iff $S(i) < i$;
– $i_L \prec j_L$ iff either one of the following holds:
  i) $S(i_L) = i$, $S(j_L) = j$ and $i > j$;
  ii) $S(i_L) \neq i$ and $S(j_L) = j$;
  iii) $S(i_L) \neq i$, $S(j_L) \neq j$ and $i < j$.

Observe that $\prec$ is total. If it is transitive, then the minimal element is either the first source in S, or n if it is not a proper sink. If it is not transitive, the minimal element allows us to find a backward pointer. □

Theorem 1 in [KKMP21] proves that FNP $\subseteq$ PEPP; we prove that actually FNP $\subseteq$ SOD. As it is straightforward that SOURCEOFDAG reduces to UNMETEREDSOURCEOFDAG, which is equivalent to empty, this implies that every TFΣ_2 class studied in this paper, apart from APEPP, contains FNP.

Proposition 6.20. FNP $\subseteq$ SOD.

Proof. Let x be an instance of R_n, an FNP problem, and let $\mathcal{O}$ be its set of solutions. By definition of FNP, this set is of size at most quasipolynomial in n. Consider the SOD instance with $|\mathcal{O}| + 1$ nodes. Consider the extra node as n. To define $S(o)$, run the verifier $T_o(x)$. If it accepts, point to n; otherwise, point $S(o)$ to itself.

Any solution o for the input x will then point to n, making it a source. The case where n is not a proper sink may occur only if x does not admit an output. □

7 Characterizing Bounded-Depth Frege

In this section we prove Theorem 1.6, introducing a hierarchy of classes in the polynomial hierarchy which characterize bounded-depth Frege systems. Depth-d Frege generalizes resolution to allow one to cut (resolve) over depth d formulas of unbounded fanin. Recall that the *depth* of a formula is the length of the longest root-to-leaf path, and the *size* is the number of wires in the formula. The *width* of a Σ_d-formula is defined as the maximum fanin among the gates at depth d. From now on and for the remainder of this section, we assume $d \geq 3$.

Definition 7.1. A *Frege* proof of an unsatisfiable formula $F = \bigwedge_{i=1}^{m} A_i$ is a sequence $\Pi = (\pi_1, \ldots, \pi_\ell = \bot)$ of formulas, where each π_i is deduced from the previously derived formulas by one of the following rules:

– *Axiom Introduction.* Introduce $\pi_i = A_i$ for some $i \in [m]$.

– *Cut*. From $C \vee D$ and $\overline{D} \vee H$ derive $C \vee H$ for any formula D.
– *Weakening.* From C derive $C \vee D$ for any formula D.

The *depth* (resp. *width*) of a Frege proof is the maximum depth (resp width) among any of the formulas $\pi_i \in \Pi$. In particular, we say that Π is a depth-d Frege (which we denote Frege$_d$) proof if each π_i is a Σ_d-formula. The *size* $|\Pi|$ of the proof is $\sum_{i=1}^{\ell} |\pi|$. The *complexity* of a Frege$_d$ proof Π is $\mathsf{width}(\Pi) + \log |\Pi|$, and the complexity of proving an unsatisfiable formula in Frege$_d$ is the minimum complexity of any Frege$_d$ proof of F.

Our characterization will generalize our characterization of Σ_2-Res(polylog) by $\textsc{Iter}_2$ (Theorem 1.5). The high-level idea is to obfuscate the successor function of the TFNP problem $\textsc{Iter}$ so that it is efficiently computable with access to a Π_{d-1} oracle, but not obviously efficiently computable with any weaker oracle. For $\textsc{Iter}_2$, this was accomplished by replacing each node $v \in [m]$ of $\textsc{Iter}$ with a group of nodes $v_1, \ldots, v_t$, each with their own successor function, pointing to a "meta-node" in $[m]$. We then treated v as pointing to a meta-node $u \in [m]$ iff all of $v_1, \ldots, v_t$ pointed to u. To generalize this to a problem in the d^{th} layer of the polynomial hierarchy, which we call $\textsc{Iter}_d$, we will take a slightly different approach. We will still replace the "meta-nodes" of $\textsc{Iter}$ with groups of nodes — in fact, this is recursively repeated $d-1$ times in order to simulate d alternating quantifiers — however, we will no longer insist that they all point to the same node. Instead, the successor will be defined by alternatively taking the minimum (corresponding to universal quantifiers) or maximum (corresponding to existential quantifiers) of the pointed-to nodes. The intuition is that the evaluation of an existentially quantified relation $\exists x F(x)$ is true if some value of x makes F true, hence we should take the maximum value of F over all x. Similarly, the evaluation of universally quantified relation $\forall x F(x)$ should be false unless every assignment makes x output true, and so this corresponds to a minimum value of F over all x. This is inspired by the problems GPLS$_d$ and PE$_d$ from [PT12], which characterize the $\forall \Sigma_1^b$ consequences of T_2^d in bounded arithmetic.

For an integer $d \geq 1$, a product set $\mathbf{r} = [r_1] \times \cdots \times [r_d]$, and a function $S : [r_1] \times \ldots \times [r_d]$ we will denote by

$$\mathsf{MAX}(S, \mathbf{r}) := \max_{i_1 \in [r_1]} \min_{i_2 \in [r_2]} \cdots \max_{i_d \in [r_d]} S(i_1, \ldots, i_d),$$
$$\mathsf{MIN}(S, \mathbf{r}) := \min_{i_1 \in [r_1]} \max_{i_2 \in [r_2]} \cdots \min_{i_d \in [r_d]} S(i_1, \ldots, i_d),$$

if d is odd, and if d is even, we change the final min or max to its opposite. We now formalize the aforementioned intuition about the connection between $\exists/\forall$ quantifiers (that is, $\vee/\wedge$ gates) and $\max / \min$ which will allow us to connect our Frege proofs to

Observation 7.2. *Let $F = \bigvee_{i_1 \in [r_1]} \bigwedge_{i_2 \in [r_2]} \cdots \bigcirc_{i_d \in [r_d]} F_{\mathbf{i}}$, where $\mathbf{i} = (i_1, \ldots, i_d)$ and each $F_{\mathbf{i}}$ is a formula. Then, for any assignment $x \in \{0,1\}^n$, $F(x) = \mathsf{MAX}(F_x, \mathbf{r})$, where $F_x(\mathbf{i}) := F_{\mathbf{i}}(x)$. Similarly, if instead F begins with $\wedge$, then $F(x) = \mathsf{MIN}(F_x, \mathbf{r})$.*

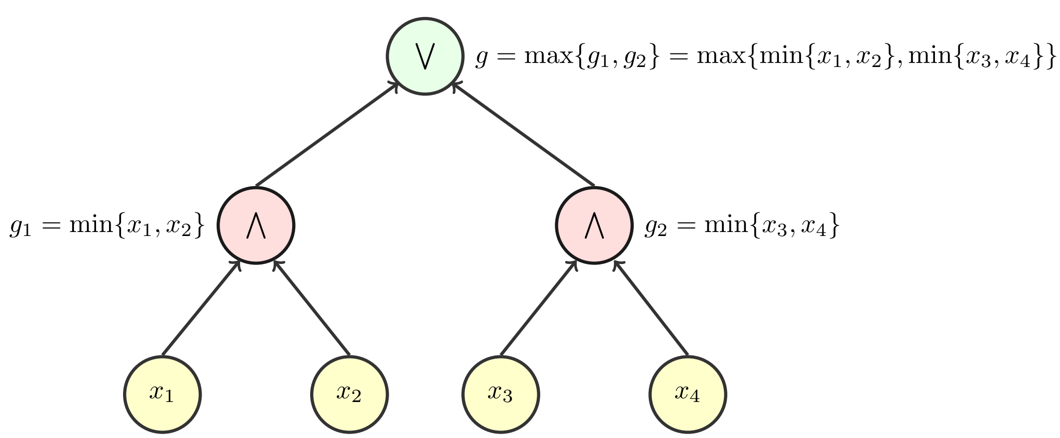

Figure 3: An example of how a formula is converted into a sequence of minimums/maximums in Observation 7.2.

Proof. The main idea is depicted Figure 3. The proof is by induction on the depth d, observing that a disjunction returns the maximum value of its subformulas, while a conjunction returns the minimum value. $\square$

Definition 7.3. An instance of ITER_d is given by a successor function $S : [m] \times [r_1] \times \cdots \times [r_{d-1}] \to [m]$ that describes a directed graph on m vertices as follows. For $u \in [m]$, let S_u denote the function S where the first input is fixed to u, let $\mathbf{r} = [r_1] \times \ldots \times [r_{d-1}]$, and define the *meta-pointer* $\tilde{S} : [m] \to [m]$ as

$$\tilde{S}(u) := \mathsf{MIN}(S_u, \mathbf{r}).$$

There is an edge from u to v in this graph if $\tilde{S}(u) = v$. A solution to ITER_d is then a solution to the ITER instance defined by $\tilde{S}$. In particular, a solution is a quadruple (u, i_1^*, v, j_1^*) such that $\tilde{S}(u) = v$ and either

i) $u = v = 1$ (1 is not a source);
ii) $v < u$ (u admits a backward pointer);
iii) $u < v$ and $\tilde{S}(u) = \tilde{S}(v) = v$ (v is a proper sink);

and i_1^* and j_1^* witness the outermost minimums for u and v: if d is even

$$i_1^* = \arg\min_{i_1 \in [r_1]} \left\{ \max_{i_2 \in [r_2]} \cdots \min_{i_{d-1} \in [r_{d-1}]} S(u, \mathbf{i}) \right\};$$
$$j_1^* = \arg\min_{j_1 \in [r_1]} \left\{ \max_{j_2 \in [r_2]} \cdots \min_{j_{d-1} \in [r_{d-1}]} S(v, \mathbf{j}) \right\};$$

and if d is odd the last $\min$ is replaced by a $\max$.

The class $\mathsf{PLS}_d \subseteq \mathsf{TF}\Sigma_d$ is the class of problems that admit an efficient reduction to ITER_d.

One should think of the indices i_1^* and j_1^* in a solution to ITER_d as the outer-most existential in Σ_d-certificates of the computation of the successor functions for u and v. One reason that this problem is hard is that for the solutions where $u < v$, the verifier must be able check that $\tilde{S}(u) = v$ and $\tilde{S}(v) = v$ or, in other words, it must be able to verify that the certificates i^* and j^* indeed witness a correct computation for their respective input nodes.

Proposition 7.4. $\text{ITER}_d \in \mathsf{TF}\Sigma_d$ *for all* $d \geq 1$.

Proof. Let us assume d is even; the case when d is odd is identical up to changing the final $\min$ into a $\max$. Let S be an instance of ITER_d and $o = (u, i_1^*, v, j_1^*)$ be a solution. Writing $\mathbf{i} = (i_1, \ldots, i_{d-1})$, checking that $\tilde{S}(u) = v$ is equivalent to checking that

$$\tilde{S}(u) \geq v \equiv \forall i_1 \exists i_2 \cdots \forall i_{d-1} S(u, \mathbf{i}) \ \geq v;$$
$$\tilde{S}(u) \leq v \equiv \exists i_1 \forall i_2 \cdots \exists i_{d-1} S(u, \mathbf{i}) \leq v.$$

Our polynomial-time verifier V_o, given witnesses $\mathbf{i} = (i_2, \ldots, i_{d-1})$, $\mathbf{j} = (j_2, \ldots, j_{d-1})$, $\mathbf{i}' = (i'_1, \ldots, i'_{d-1})$, $\mathbf{j}' = (j'_1, \ldots, j'_{d-1})$, V_o, behaves as follows:

i) It checks that $S(u, \mathbf{i}') \geq v$ and $S(u, i_1^* \mathbf{i}) \leq v$; if not, it outputs 0[3];
ii) It outputs 1 if $u < v$ or $u = v = 1$;
iii) Otherwise, it checks whether $S(v, \mathbf{j}') \geq v$ and $S(v, j_1^* \mathbf{j}) \leq v$; if this is the case then it outputs 1, and otherwise it outputs 0.

Observe that the expression

$$\forall (i'_1, i_2, j'_1, j_2) \exists (i'_2, i_3, j'_2, j_3) \cdots \exists (i'_{k-2}, i_{d-1}, j'_{k-2}, j_{d-1}) \forall (i'_{d-1}, j'_{d-1}) \ V_o(S, \mathbf{i}, \mathbf{i}', \mathbf{j}, \mathbf{j}') = 1$$

is true iff o is a solution to S. $\square$

[3]Here, $i_1^* \mathbf{i}$ denotes the concatonation of i_1^* with $\mathbf{i}$.

7.1 Proofs as Games

To establish the correspondence between bounded-depth Frege and $\textsc{Iter}_d$ it will be useful to view proofs as games. The *depth-d Prover-Delayer game* (essentially also known as the Buss-Pudlák game [PB94]) for an unsatisfiable formula F consists of two players, Prover and Delayer. Intuitively, the Prover is attempting to convince itself that F is unsatisfiable, while the Delayer is trying to postpone this. The game proceeds in rounds, where in each round the Prover asks for the value of an arbitrary formula C and the delayer responds with an answer — either $C = 1$ (in which case the Prover remembers C) or $\neg C = 1$ (and the Prover remembers $\neg C$). Finally, at the end of each round the Prover may forget any number of formulas from its memory. The game ends when the set of $\{0, 1\}$-assignments consistent with the Prover's memory all falsify some axiom A_i. That is, when the conjunction of the formulas in the Prover's memory logically imply $\overline{A}_i$.

Definition 7.5. Let $F = \bigwedge_{i=1}^m A_i$ be an unsatisfiable formula. A *Prover strategy* is a rooted fan-out ≤ 2 DAG G in which every node v is labeled with a set of boolean formulas M_v, which we call the memory at node v. Let $\mathsf{False}(M_n) := \{x \in \{0,1\}^n : C(x) = 0, \forall C \in M_v\}$ be the set of assignments which falsify all of the formulas. The labels M_v satisfy the following:

- *root.* If v is the root then $M_v = \emptyset$;
- *single child.* If v has one child c then $M_c = M_v \setminus \{C\}$ for some formula C;
- *two children.* If v has two children c, c', then $M_c = M_v \cup \{\neg C\}$ and $M_{c'} = M_v \cup \{C\}$ for some formula C;
- *leaf.* If v has no children, then there is some A_i for $i \in [m]$ such that $A_i(x) = 0$ for all $x \in \mathsf{False}(M_v)$.

The *width* of the strategy is $\max_{v \in V} \max_{C \in M_v} \{\mathsf{width}(C)\}$ and its *depth* is $\max_{v \in V} \max_{C \in M_v} \{\mathsf{depth}(C)\} + 1$ — the off-by-one is to account for the fact that, as we will see, the conjunction of the formulas in memory will correspond to a line in Frege proof. The *size* of the strategy is $\sum_{v \in V(G)} \sum_{C \in M_v} |C|$.

As for the original Prover-Delayer game for the resolution proof system and the Buss-Pudlák game for bounded-depth Frege proofs, finding a strategy for a formula closely relates to finding a refutation.

Lemma 7.6. *Let $F = \bigwedge_{i \in [m]} A_i$ be an unsatisfiable Π_{d+1}-formula. There exists a width-w and size-s Frege$_d$ refutation of F iff there is a width-w, depth-d, and size-s Prover strategy for F.*

Proof. Let Π be a Frege$_d$ proof of F. The graph of the Prover strategy will be the same as that of the proof. Beginning at the root r, where $M_r = \emptyset$, the Prover's memory is constructed as follows: let v be a node with memory M_v; we have several cases based on the rule used to derive the corresponding line π_v.

- If π_v was obtained by weakening π_u with a formula D, then $M_u := M_v \setminus \{\neg D\}$. That is, the Prover forgets $\neg D$.
- If π_v was obtained by cutting π_u and π_w on a formula D, then $M_u := M_v \cup \{D\}$ and $M_w := M_v \cup \{\neg D\}$. That is, the Prover queries the Delayer for the value of D.
- If π_v was obtained by axiom introduction, then v is a leaf.

By induction, observe that the conjunction of the formulas in M_v logically implies $\overline{\pi}_v$, and hence the leaf case is satisfied. The width and size of the strategy are the same as the proof. The fact that Π is a Frege$_d$ refutation (meaning that D is always a $\Pi_{d-1} \cup \Sigma_{d-1}$-formula), gives us the depth of the strategy to be d.

For the converse direction, a Prover strategy can be converted into a Frege proof by replacing each memory $M_v = \{C_1, \dots, C_k\}$ with the line $\neg C_1 \vee \ldots \vee \neg C_k$. As each formula in memory is a Π_{d-1} or a Σ_{d-1}-formula, the lines of this proof are Σ_d-formulas and the proof is a Frege$_d$ proof of with the same width and size as the strategy. $\square$

As seen in section 5, for a proof system to correspond to a TFPH class it must be *reduction closed*. We verify that Frege$_d$ satisfies this property.

Lemma 7.7. *Let $F = \bigwedge_{i \in [m]} F_i$ and $G = \bigwedge_{j \in \ell} G_j$ be unsatisfiable Π_{d+1}-formulas on n variables, and suppose that there is a width-w and size-s Frege$_d$ proof of F. If (f, g) is an FF_F-formulation of FF_G of depth $\mathsf{polylog}(n)$, then G has a Σ_d-Frege$_d$ refutation of size $s \cdot 2^{\mathsf{polylog}(n)}$ and width w.*

Proof. Let Π be a size-s, Frege$_d$ proof of F. First, we modify Π to be a proof of $F(f) = \bigwedge_{i \in [m]} F_i(f)$, where $F_i(f)$ is the Σ_d obtained by replacing each variable x_j with the propositionalization of decision tree f_j as defined in the reduced formula. To do so, we will view Π as a Prover strategy and replace each memory $M_v = \{C_1, \dots, C_k\}$ by $\{C_1(f), \dots, C_k(f)\}$. That is, instead of querying C, the prover will now query the formula $C(f)$.

We now transform this into a Prover strategy, and hence proof, of $F(f, g)$, which is a Σ_d-weakening of G. To do so, consider any leaf of the Prover strategy labeled by some $F_i(f)$ (corresponding to the Prover learning that $F_i(f)$ is falsified). At this leaf the Prover queries the decision tree g_i, one variable at a time. Each leaf is labeled with $F_i(f) \vee \overline{p}$ for some path $p \in g_i$; an axiom of the formula of $F(f, g)$. $\square$

7.2 Chracterizing PLS$_d$

Now we are ready to prove our characterization Theorem 1.6, which we state formally next.

Theorem 7.8. *For any* $\mathrm{FF}_F \in \mathsf{TF}\Sigma_d$, *there is a complexity-*c $\textsc{Iter}_d$*-formulation of* FF_F *iff there is a complexity-*$\Theta(c)$ Σ_d*-Frege*$_d$ *refutation of* F.

We will break the theorem into two lemmas, Lemma 7.9 transforming Frege proofs into $\textsc{Iter}_d$-formulations, and Lemma 7.10 together with Lemma 7.7 providing the converse.

Lemma 7.9. *Let* F *be an unsatisfiable* Π_{d+1} *formula on* n *variables. Suppose that there is a width-*w *and size-*s Σ_d*-Frege*$_d$ *proof of* F. *Then* FF_F *admits an* $\textsc{Iter}_d$*-formulation of size* s *and depth* w.

Proof. Let $\Pi' = (\pi'_1, \dots, \pi'_m)$ be a Frege$_d$ refutation of size s and width w, and suppose that it is ordered in reverse topological order so that $\pi_1 = \bot$. At a high level the meta-pointer $\tilde{S} : [m] \to [m]$ of the $\textsc{Iter}_d$ formula will trace a path from the root to a falsified axiom of the proof by always pointing a node u, for which $\pi'_u(x)$ is false under the given assignment x, to a falsified child which is guaranteed to exist by the soundness of the proof. If $\pi'_u(x)$ is false and was derived by cutting on $\pi'_v = A \vee C$ and $\pi'_w = B \vee \neg C$ then $\tilde{S}$ should point u to the child v or w that is falsified. To determine which of π'_v, π'_w is falsified, we need to evaluate A, B, and C. Hence, we need to ensure that the size of the domain of the successor S is large enough. A simple way to do so is to pre-process our proof Π' into a proof $\Pi = (\pi_1, \dots, p_m)$ as follows: if $\pi'_u = A \vee B$ was derived by cutting $A \vee \neg C$ and $B \vee C$, then replace $\pi_u = A \vee B \vee C$. Now, let r_i be the maximum fanin at layer i of any line π_i in the proof Π. By padding (with, for example, $\top = x \vee \neg x$ or $\bot = x \wedge \neg x$) we may assume that every line π_u in Π has the same fanin r_i at each layer. That is,

$$\pi_u = \bigvee_{i_1 \in [r_1]} \bigwedge_{i_2 \in [r_2]} \cdots \bigcirc_{i_{d-1} \in [r_{d-1}]} G_{u,\mathbf{i}},$$

where $G_{u,\mathbf{i}}$ is a fanin-w clause if $\bigcirc = \wedge$ and a fanin-w term if $\bigcirc = \vee$. Let $\mathbf{r} = [r_1] \times \cdots \times [r_{d-1}]$.

For any assignment $x \in \{0,1\}^n$ the successor function S_x of our $\textsc{Iter}_d$ instance is defined as:

- *Axiom Introduction.* If π_u is an axiom then $S_x(u, \mathbf{i}) = u$;
- *Weakening.* If π_u is a weakening of π_v then $S_x(u, \mathbf{i}) = u$ if $G_{u,\mathbf{i}}(x) = 1$ and v otherwise.
- *Cut.* If $\pi_u = A \vee B \vee C$ was derived by cutting $\pi_v = A \vee \neg C$ and $\pi_w = B \vee C$ then
  i) If $G_{u,\mathbf{i}}$ is a subformula belonging to A or B then $S_x(u, \mathbf{i}) = u$ if $G_{u,\mathbf{i}}(x) = 1$ and v otherwise.
  ii) $G_{u,\mathbf{i}}$ is a subformula belonging to C then $S_x(u, \mathbf{i}) = v$ if $G_{u,\mathbf{i}}(x) = 1$ and w otherwise.

Observe that $G_{u,\mathbf{i}}(x)$ have width w and so $S_x(u, \mathbf{i})$ can be evaluated by a depth-w decision tree $f_{u,\mathbf{i}}$ querying the variables x. Finally, for each solution $o = (u, i_1^*, v, j_1^*)$, the output decision tree g_o is the constant function which returns v.

The following claim asserts the correctness of the formula, completing the proof.

Claim. The meta-pointer $\tilde{S}_x$, defined from S_x, satisfies the following properties:

i) $\tilde{S}_x(u) \geq u$ for any $u \in [m]$;
ii) If $u \in [m]$ is such that π_u is not an axiom, then $\tilde{S}_x(u) = u$ iff $\pi'_u(x) = 1$ (recall that π'_u belongs to the proof Π' before pre-processing);
iii) For any $u \in [m]$, if $\tilde{S}_x(u) = v \neq u$ then $\pi_v(x) = 0$.

Proof of Claim. We will consider cases based on how π_u was derived. If π_u was deduced by *axiom introduction* then the claim holds by definition. If π_u was derived by *weakening* π_v, then re-parameterize the formula $G_{u,x}(\mathbf{i}) := G_{u,\mathbf{i}}(x)$ and let $S_{x,u}$ be S_x with the first input fixed to u. Then, noting that the definition of the successor in the case of weakening is equivalent to $S_{x,\mathbf{i}}(u) = G_{u,\mathbf{i}} \cdot u + (1 - G_{u,\mathbf{i}}) \cdot v$, we have

$$\begin{aligned} \tilde{S}_x(u) &= \mathsf{MIN}\big(S_{x,u},\ \mathbf{r}\big) = \mathsf{MIN}\big(G_{u,x} \cdot u + (1 - G_{u,x}) \cdot v,\ \mathbf{r}\big) \\ &= \mathsf{MAX}(G_{u,x}, \mathbf{r})(u - v) + v \\ &= \pi_u(x)(u - v) + v && \text{(Observation 7.2)} \\ &= \pi_u(x) \cdot u + (1 - \pi_u(x)) \cdot v, \end{aligned}$$

where the MIN switched to a MAX because $u - v < 0$. The final equality is equivalent to

$$\tilde{S}_x(u) = \begin{cases} u & \text{if } \pi_u(x) = 1; \\ v & \text{if } \pi_u(x) = 0. \end{cases}$$

Thus, the claim holds when π_u was derived by weakening.

Finally, consider the case when $\pi_u = A \vee B \vee C$ was derived by cutting $\pi_v = A \vee \neg C$ and $\pi_w = B \vee C$. Denote by $\mathbf{r}_{A\vee B} = [r_1 - 1] \times [r_2] \times \cdots \times [r_{d-1}]$ and $\mathbf{r}_C = \{r_1\} \times [r_2] \times \cdots \times [r_{d-1}]$. By partitioning the indices in this way, we will enforce that the subformulas with indices in $\mathbf{r}_{A\vee B}$ belong to C, and the remaining belong to $A \vee B$; that is, we take the convention that all subformulas of C have index r_1. Then,

$$\begin{aligned} \tilde{S}_x(u) &= \mathsf{MIN}\big(S_{x,u},\ \mathbf{r}\big) \\ &= \min\Big\{\mathsf{MIN}\Big(G_{u,x} \cdot u + (1 - G_{u,x})v,\ \mathbf{r}_{A\vee B}\Big),\ \mathsf{MAX}\Big(G_{u,x} \cdot v + (1 - G_{u,x})w,\ \mathbf{r}_C\Big)\Big\} \\ &= \min\Big\{\mathsf{MAX}\big(G_{u,x},\ \mathbf{r}_{A\vee B}\big)(u - v) + v,\ \mathsf{MIN}\big(G_{u,x},\ \mathbf{r}_C\big)(v - w)\Big\} \\ &= \min\Big\{(A(x) \vee B(x))(u - v) + v,\ C(x)(v - w) + w\Big\} && \text{(Observation 7.2)} \\ &= \big(A(x) \vee B(x)\big) \cdot u + \big(1 - (A(x) \vee B(x)\big) \cdot \big(C(x) \cdot v + (1 - C(x)) \cdot w\big), \end{aligned}$$

where we have swapped MINs and MAXs using that $u < v, w$. That is,

$$\tilde{S}_x(u) = \begin{cases} u & \text{if } (A \vee B)(x) = 1; \\ v & \text{if } (A \vee B)(x) = 0 \text{ and } C(x) = 0; \\ w & \text{if } (A \vee B)(x) = 0 \text{ and } C(x) = 1. \end{cases}$$

It follows that the claim holds in the case of the cut rule. ☐

We now turn to establishing the forward direction of Theorem 7.8, converting $\textsc{Iter}_d$-formulations into Σ_d-Frege$_d$ proofs. As observed in Lemma 7.7, Σ_d-Frege$_d$ is reduction closed. Hence, in order to establish the forward direction of Theorem 7.8, it suffices to show that Frege$_d$ has efficient proofs of the propositional encoding of $\textsc{Iter}_d$, which we describe next.

The $\textsc{Iter}_d$ Formula. For simplicity (by padding) we may assume without loss of generality that in the definition of $\textsc{Iter}_d$, $m = r_1 = \ldots = r_{d-1}$. Our formula will be over m-ary variables $S_{u,\mathbf{i}} \in [m]$ for $u \in [m]$ and $\mathbf{i} \in [m]^{d-1}$; this may be encoded by $\log n$-many binary variables $S_{u,\mathbf{i},k}$ which spell out the binary encoding of the value of $S_{u,\mathbf{i}}$. In particular, if $v \in [m]$ has binary expansion $v = \sum_{k\in[\log m]} b_k 2^{k-1}$, then the formula $[\![S_{u,\mathbf{i}} = v]\!] := \bigwedge_{k\in[\log m]} S_{u,\mathbf{i},k}^{b_k}$ with the notation that $x^1 = x$ and $x^0 = \neg x$ for any variable x. Similarly, we write $[\![S_{u,\mathbf{i}} \neq v]\!] = \neg[\![S_{u,\mathbf{i}} = v]\!]$, where the negation is propagated to the literals. As well, for $u, v, i^* \in [m]$ denote by $[\![\mathsf{bad}(i^*) \vee \tilde{S}(u) \neq v]\!]$ the following formula:

$$[\![\mathsf{bad}(i_1^*) \vee \tilde{S}(u) \neq v]\!] := \Big(\bigvee_{i_1}\bigwedge_{i_2}\cdots\bigwedge_{i_{d-1}}\bigwedge_{v\le v'}[\![S_{u,\mathbf{i}} \neq v']\!]\Big) \vee \Big(\bigvee_{i_2'}\bigwedge_{i_3'}\cdots\bigvee_{i_{d-1}'}\bigvee_{v<v'}[\![S_{u,i_1^*\mathbf{i}'} = v']\!]\Big) \quad \text{if } d \text{ is odd;}$$

$$[\![\mathsf{bad}(i_1^*) \vee \tilde{S}(u) \neq v]\!] := \Big(\bigvee_{i_1}\bigwedge_{i_2}\cdots\bigvee_{i_{d-1}}\bigvee_{v'<v}[\![S_{u,\mathbf{i}} = v']\!]\Big) \vee \Big(\bigvee_{i_2'}\bigwedge_{i_3'}\cdots\bigwedge_{i_{d-1}'}\bigwedge_{v'\ge v}[\![S_{u,i_1^*\mathbf{i}'} \neq v']\!]\Big) \quad \text{if } d \text{ is even;}$$

It states that $\tilde{S}(u) \neq v$ by asserting that either i_1^* is not a certificate of computation for u (left-hand side of the disjunction) or that the minimum is greater than v (right-hand side of the disjunction). These are $\Sigma_{(d-1).5}$-formulas.

The $\textsc{Iter}_d$ formula is the conjunction of the following:

i) $[\![\mathrm{bad}(i_1^*) \vee \tilde{S}(1) \neq 1]\!]$ for each i_1^*, (1 is a source)

ii) $[\![\mathrm{bad}(i_1^*) \vee \tilde{S}(u) \neq v]\!]$ for each pair of nodes $v < u$ and index i_1^*, (No backwards pointer)

iii) $[\![\mathrm{bad}(i_1^*) \vee \tilde{S}(u) \neq v]\!] \vee [\![\mathrm{bad}(i_1^*) \vee \tilde{S}(v) \neq v]\!]$. for each $u, v, i_1^*, j_1^* \in [m]$ with $u < v$. (No proper sinks)

Lemma 7.10. *There is a size-$O(n^2)$ and width-$O(\log n)$ $Frege_d$ proof of* $\textsc{Iter}_d$.

For nodes u, v and an index i_1, it will be convenient to denote by $[\![S_{i_1}(u) \leq v]\!]$ the formula

$$[\![S_{i_1}(u) \leq v]\!] := \bigwedge_{i_2} \bigvee_{i_3} \cdots \bigwedge_{i_{d-1}} \bigwedge_{v < v'} [\![S_{u,i_1\mathbf{i}} \neq v']\!] \quad \text{if } d \text{ is odd,}$$

$$[\![S_{i_1}(u) \leq v]\!] := \bigwedge_{i_2} \bigvee_{i_3} \cdots \bigvee_{i_{d-1}} \bigvee_{v' \leq v} [\![S_{u,i_1\mathbf{i}} = v']\!] \quad \text{if } d \text{ is even,}$$

which encodes that $\max_{i_2} \min_{i_3} \cdots \bigcirc_{i_d} S(u, i_1\mathbf{i})$ is less than or equal to v. These are $\Pi_{(d-2).5}$-formulas that will be used as cuts in the proof (or queries in the Prover's strategy).

Proof. We give a Prover strategy for $\textsc{Iter}_d$. The idea of the strategy is as follows: the Prover will begin at the root node 1 and will traverse the successor $\tilde{S}$ until they reach a solution (either 1 points to itself, u points to v with $v < u$, or u is a proper sink). To achieve this, at each node u, the Prover tries to determine the value of $\tilde{S}(u)$. They are unable to do so directly, as this would require the Prover to query a formula of depth d. Instead, they determine the value of $\tilde{S}(u)$ via an *auction procedure*, which we describe below. Once the Prover has determined v such that $\tilde{S}(u) = v$, then either the Prover has found a solution, or otherwise the Prover forgets everything except the information necessary to infer $\tilde{S}(u) = v$, and queries the Delayer via the auction procedure to learn the value of $\tilde{S}(v)$. At each step, the Prover has in memory the value of $\tilde{S}(w)$ for at most two nodes $w \in [m]$.

The Auction Procedure. The procedure determines a value $v \in [m]$ so that $\tilde{S}(u) = v$ is the only value compatible with the answers given by the Delayer. The procedure is in rounds $v = m-1, \ldots, 1$. At each round the Prover queries the formula $[\![S_{i_1}(u) \leq v]\!]$ for $i_1 = 1, \ldots, m$ and reacts in the following way to the answers of the Delayer:

Round $v = m-1$: As soon as the Delayer answers 1 for some $i_1^* \in [m]$, the Prover forgets all of the previously-learned formulas of the form $\neg[\![S_j(u) \leq m-1]\!]$ for $j < i_1^*$, retains in memory the formula $[\![S_{i_1^*}(u) \leq m-1]\!]$, and moves to the round $v = m-2$. If the Delayer answers 0 to all $i_1^* \in [m]$, then the Prover knows that $\tilde{S}(u) = m$.

Round $v < m-1$: By induction the Prover's memory consists only of $[\![S_{j_1^*}(u) \leq v+1]\!]$ for some $j_1^* \in [m]$. Hence, the Prover knows that $\tilde{S}(u) \leq v+1$, and it would like to determine whether $\tilde{S}(u) = v+1$ or $\tilde{S}(u) \leq v$. As soon as the Delayer answers 1 to the queries made during this round, for some $i_1^* \in [m]$, the Prover retains in memory only the formula $[\![S_{i_1^*}(u) \leq v]\!]$, forgetting the formulas $\neg[\![S_k(u) \leq v]\!]$ for all $k < i_1^*$ and $[\![S_{j_1^*}(u) \leq v+1]\!]$. They then move to round $v-1$ if $v > 1$ or halt if $v = 1$, as this implies that $\tilde{S}(u) = 1$.

If the Delayer answers 0 to all $i_1^* \in [m]$, then the Prover's memory contains $[\![S_{j_1^*}(u) \leq v+1]\!]$ and $\neg[\![S_k(u) \leq v]\!] = 0$ for all $j \in [m]$, which implies that $\tilde{S}(u) = v$. The Prover then halts the auction procedure, keeping its memory as is.

Observe that there are at most $O(m^4)$-many possible states in any auction procedure. The memory of each state contains at most $O(m)$-many formulas, and each formula has size $O(m^d)$ and width $O(\log m)$. Hence the size of each auction phase is at most $O(m^{d+5})$.

With the auction procedure in place, we are ready to describe the Prover strategy in detail. Beginning with the node $u = 1$, the Prover determines a node v such that $\tilde{S}(u) = v$ via the auction procedure.

- If $u = v = 1$: the Prover halts;
- If $v < u$: the Prover forgets everything in memory except for the formulas determining that $\tilde{S}(u) = v$ and halts.
- If $v = u \neq 1$, then prior to computing $\tilde{S}(u)$, the Prover's memory already contains formulas which enforce that $\tilde{S}(w) = u$ for some $u < w$, hence they have found a solution and they halt.

– If $u < v$ then the Prover had in its memory the formulas determining that $\tilde{S}(w) = u$ for some $w < u$. They forget these equalities which are not relevant to enforcing that $\tilde{S}(u) = v$, and the Prover moves to node v.

This process terminates as the current node u increases by at least 1 at each step. We now calculate the number of possible states (and hence the number of lines in the corresponding Frege$_d$ proof). As the auction procedure is performed once per round, and there are at most m^2-many choices for the memory at each round (corresponding to the value of i_1^* and the node pointing to v), the total size of the strategy is at most $O(m^{d+8})$, which is $O(n^4)$ where n is the number of variables of the formula.

It remains to argue that this is a valid strategy for the Prover; that is, when the Prover halts, at least one axiom of the formula is violated by all assignments which satisfy their memory. There are several cases based on the reason for halting:

– If $u = v = 1$: then the memory contains $[\![S_{i_1^*}(1) \leq 1]\!]$ for some $i_1^* \in [m]$. This is incompatible with the axiom $[\![\text{bad}(i_1^*) \vee \tilde{S}(1) \neq 1]\!]$.
– If $v < u$: there are two possibilities based on the value of v. If $v = 1$, then the memory contains $[\![S_{i_1^*}(u) \leq 1]\!]$ for some $i_1^* \in [m]$, contradicting the axiom $[\![\text{bad}(i_1^*) \vee \tilde{S}(u) \neq 1]\!]$. Otherwise, if $v \neq 1$ then the memory contains $\neg[\![S_i(u) \leq v-1]\!]$ for all $i \in [m]$ and $[\![S_{i_1^*}(u) \leq v]\!]$ for some i_1^*, contradicting the axiom $[\![\text{bad}(i_1^*) \vee \tilde{S}(u) \neq v]\!]$.
– If $\tilde{S}(u) = \tilde{S}(v) = v$: then there are two cases based on whether $v = m$. If $v \neq m$ then the memory contains $[\![S_{i_1^*}(u) \leq v]\!]$ and $[\![S_{j_1^*}(v) \leq v]\!]$ for some $i_1^*, j_1^* \in [m]$ along with the formulas $\neg[\![S_i(u) \leq v-1]\!]$ and $\neg[\![S_j(v) \leq v-1]\!]$ for all $i, j \in [m]$. This contradicts the axiom $[\![\text{bad}(i_1^*) \vee \tilde{S}(u) \neq v]\!] \vee [\![\text{bad}(j_1^*) \vee \tilde{S}(v) \neq v]\!]$.
  Otherwise, the memory contains $\neg[\![S_i(u) \leq m-1]\!]$ and $\neg[\![S_i(v) \leq m-1]\!]$ for all $i, j \in [m]$ which contradicts the axiom $[\![\text{bad}(i_1^*) \vee \tilde{S}(u) \neq m]\!] \vee [\![\text{bad}(j_1^*) \vee \tilde{S}(m) \neq m]\!]$ for all values of i_1^* and j_1^*. Indeed, for any node u, if $\tilde{S}(u) = m$ then i_1^* will always be an argument of the minimum, regardless of its value.

Hence, the Prover strategy is correct. □

8 Acknowledgments

Noah Fleming was supported by NSERC. The authors thank Robert Robere, Toniann Pitassi, and Oliver Korten for helpful discussions.

References

[AL23] Albert Atserias and Massimo Lauria. Circular (yet sound) proofs in propositional logic. *ACM Trans. Comput. Log.*, 24(3):20:1–20:26, 2023.

[BB09] Arnold Beckmann and Samuel R. Buss. Polynomial local search in the polynomial hierarchy and witnessing in fragments of bounded arithmetic. *Journal of Mathematical Logic*, 09(01):103–138, 2009.

[BCE+98] Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998.

[Bey22] Olaf Beyersdorff. Proof complexity of quantified boolean logic–a survey. *Mathematics for Computation (M4C)*, pages 353–391, 2022.

[BFI23] Sam Buss, Noah Fleming, and Russell Impagliazzo. TFNP characterizations of proof systems and monotone circuits. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 30:1–30:40. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[BKT14] Samuel R. Buss, Leszek Aleksander Kolodziejczyk, and Neil Thapen. Fragments of approximate counting. *J. Symb. Log.*, 79(2):496–525, 2014.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1990–1999. ACM, 2024.

[DR23] Ben Davis and Robert Robere. Colourful TFNP and propositional proofs. In Amnon Ta-Shma, editor, *38th Computational Complexity Conference, CCC 2023, July 17-20, 2023, Warwick, UK*, volume 264 of *LIPIcs*, pages 36:1–36:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[FGPR24] Noah Fleming, Stefan Grosser, Toniann Pitassi, and Robert Robere. Black-box PPP is not turing-closed. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1405–1414. ACM, 2024.

[FKP19] Noah Fleming, Pravesh Kothari, and Toniann Pitassi. Semialgebraic proofs and efficient algorithm design. *Found. Trends Theor. Comput. Sci.*, 14(1-2):1–221, 2019.

[GHJ+22a] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. *Electron. Colloquium Comput. Complex.*, TR22-018, 2022.

[GHJ+22b] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. *CoRR*, abs/2205.02168, 2022.

[GKRS19] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 38:1–38:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Kor21] Oliver Korten. The hardest explicit construction. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 433–444. IEEE, 2021.

[KP24] Oliver Korten and Toniann Pitassi. Strong vs. weak range avoidance and the linear ordering principle. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 1388–1407. IEEE, 2024.

[KST07] Jan Krajícek, Alan Skelley, and Neil Thapen. NP search problems in low fragments of bounded arithmetic. *J. Symb. Log.*, 72(2):649–672, 2007.

[KT21] Leszek Aleksander Kolodziejczyk and Neil Thapen. Approximate counting and NP search problems. 2021.

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 2000–2007. ACM, 2024.

[LPR24] Yuhao Li, William Pires, and Robert Robere. Intersection classes in TFNP and proof complexity. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 74:1–74:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[PB94] Pavel Pudlák and Samuel R. Buss. How to lie without being (easily) convicted and the length of proofs in propositional calculus. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 1994.

[PT12] Pavel Pudlák and Neil Thapen. Alternating minima and maxima, nash equilibira and bounded arithmetic. *Annals of Pure and Applied Logic*, 72:604–614, 2012.

[Pud15] Pavel Pudlák. On the complexity of finding falsifying assignments for herbrand disjunctions. *Arch. Math. Log.*, 54(7-8):769–783, 2015.

[ST11] Alan Skelley and Neil Thapen. The provably total search problems of bounded arithmetic. *Proceedings of the London Mathematical Society*, 103(1):106–138, 2011.

[Tha24] Neil Thapen. How to fit large complexity classes into TFNP. 2024.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il